Numerical Recipes

2015/16 Dept Physics and Astronomy

Unit 1

Matrices

(and an introduction to χ^2)

Matrices

Aim:

- To “warm you back up” to OO coding
- To introduce you to the working environment
- To give you the first comparison between “home grown” and “package” code

Objectives

- Define Matrix class
- Implement the class with suitable methods for
  - Instantiation
  - Addition
  - Multiplication
- Perform same operations but using a matrix algebra package

- By way of a matrix exercise you will formulate a “χ^2” for later use in parameter estimation

Matrices :

Matrices will be objects in the code

Assume we have written a class called MyMatrix

It would be used like this

```
....

// Create some matrices
MyMatrix A
MyMatrix B

....some code to fill the elements of A and B...


// Add  C = A + B

MyMatrix C

C = A.add(B)


....


// Multiply  C = A * B

MyMatrix D

D = A.postMultiply(B)
```

Note: written in a symbolic way – i.e. not any particular language.
E.g. In Java all lines need to end in ;

Matrices : The methods

Here are some methods you might think of

```
class MyMatrix {

public methods:

     // Constructor with dimensions
     MyMatrix( int n,  int m )

     // Copy/Clone
     MyMatrix ( MyMatrix source )









}
```

Matrices : The methods

Here are some methods you might think of

```
class MyMatrix {

public methods:

    // Constructor with dimensions
    MyMatrix( int n,  int m )

    // Copy/Clone
    MyMatrix ( MyMatrix source )

    // To set elements
    void setElement( int i, int j, double val )

    // To get elements
    double getElement( int i, int j )









}
```

Matrices : The methods

Here are some methods you might think of

```
class MyMatrix {

public methods:

     // Constructor with dimensions
     MyMatrix( int n,  int m )

     // Copy/Clone
     MyMatrix ( MyMatrix source )

     // To set elements
     void setElement( int i, int j, double val )

     // To get elements
     double getElement( int i, int j )

     // Addition
     MyMatrix add( MyMatrix m )

     //Multiplication
     MyMatrix postMultiplyBy( MyMatrix m )


}
```

[Aside: practical issue]

Here is an issue which obviously arises: How are you going to know if the dimensions match for multiplication and addition ?

- Do we add getNumberOfRows(…) methods ?
- Or do we put the decision inside the class ?
- Or just return a NULL matrix if the operation fails
- Or print out a message and exit
- Or....

```
i.e to know if we can do this (M and N are two matricies)

    M.postMultiplyBy( N )

do we test on methods:

    int getNumberOfRows()
    int getNumberOfColumns()

or test on method:

    bool canMultiply (MyMatrix m)

    ie.
     if( M.canMultiply( N) ) { ............

or .....
```

Matrices : Implementation

- You should design a concrete class to implement MyMatrix
  - You need to decide what private internal member variables are needed ?
  - You need to think if you want to define some private internal methods ?
  - ...do it...
  - ...test it...
    - you may need to be reminded to how to provide a main() program
    - if so see the example blow

- Java worked examples are at:
  - Unit1-Matricies/java/MyMatrix.java
  - Unit1-Matricies/java/TestMyMatrix.java

- Python worked examples are at:
  - Unit1-Matricies/py/MyMatrix.py
  - Unit1-Matricies/py/TestMyMatrix.py

Matrices : Using a package

- After this perform the same tests, but instead of your own MyMatrix class, you should use a package for matrix algebra
  - We will use
    - Java: Jama [http://math.nist.gov/javanumerics/jama/]
    - C++ : Armadillo
    - Python: numpy matrix class
- Use WWW for the documentation -its far better than any that could be written in slides.

```
//To use the Jama matrix algebra package

import Jama.*;

public final class TestJama {

    public static final void main(String... aArgs){
        ......
```

Matrices : Using a package

- Java worked examples are at:
  - Unit1-Matricies/java/TestJama.java

- Python worked examples are at:
  - Unit1-Matricies/py/TestMatrix.py

Exercise 1:

- Solve the following simultaneous equations for quantities “a” to “e”

3a +	4b +	8c +	5d +	4e		170
5a +	6b +	9c +	10d +	1e		204
7a +	3b +	9c +	3d +	2e	=	138
12a +	15b +	20c +	5d +	9e		394
9a +	3b +	5c +	6d +	8e		223

- These numbers can be found in the files
  `Unit1-Matrices/java/SimultaneousMatrix.txt`
  `Unit1-Matrices/java/SimultaneouskVector.txt`

- Hint: the matrix algebra is completely trivial - you can write it down on one line.

- The example code is in:
  `Unit1-Matrices/java/Simultaneous.java`

(but don’t look at it until you have tried)

How to read and write data to a file in Java

We have supplied a simple I/O facility for use in this course

- MyFileReader : reads a text file into a JAMA Matrix
- MyFileWriter : writes a JAMA Matrix to a text file
- Please look in the `MyUtilities/java/TestMyIO` for example to see how it works.
- Also the example `Unit1-Matrices/java/Simultaneous.java`

```
MyFileReader theMatrix = new MyFileReader("SimultaneousMatrix.txt") ;
Matrix matrix = theMatrix.getMatrix() ;
matrix.print(5,3);

MyFileReader theVector = new MyFileReader("SimultaneousVector.txt") ;
Matrix vector = theVector.getMatrix() ;
vector.print(5,3);
```

How to read and write data to a file in python

- Python already has simple I/O functions supplied
- Uses the numpy library → numpy.loadtext
  - This reads data in as a list or list of lists
- Easy to create a numpy.matrix from what gets read in

[Note it is useful for you to understand the difference between a "list of lists" and a matrix]

```
 # Read in the data
print("Reading in coefficients matrix")
coeffs = numpy.loadtxt("SimultaneousMatrix.txt")

print("Reading in vector")
vector = numpy.loadtxt("SimultaneousVector.txt")

# Create matrices
mcoeffs = numpy.matrix(coeffs)
mvector = numpy.matrix(vector).T
```

Matrices : Excercise 2 - χ^2

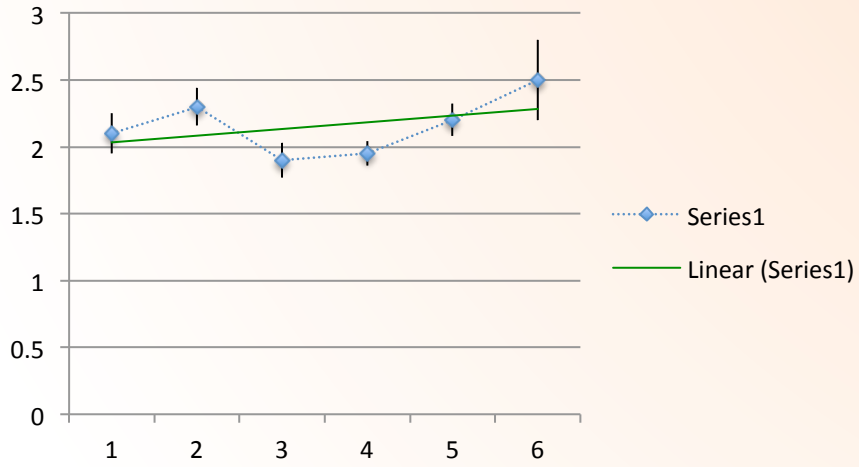
First I have to remind you what a χ^2 is

Fitting : forming a χ^2

We will now use your matrix class to formulate a χ^2 ("Chi-squared")

χ^2 is a quantity which arises in "fitting"
This is more properly called "parameter estimation"

The simplest example is fitting a straight line to some data:



Line is characterised by
$y = mx + c$

To find the best estimates for the parameters "m" and "c" you have to vary them to find the best fit

The most simple way to do this is *least squares fitting* – although this is very limited in its application as it does not take account measurement errors (so it does not give the best estimation)

In a real situation it is most likely that you would do a χ^2 minimisation.
We are going to formulate the χ^2 as an exercise in this Unit (for use in a later Unit).

Fitting : forming a χ^2

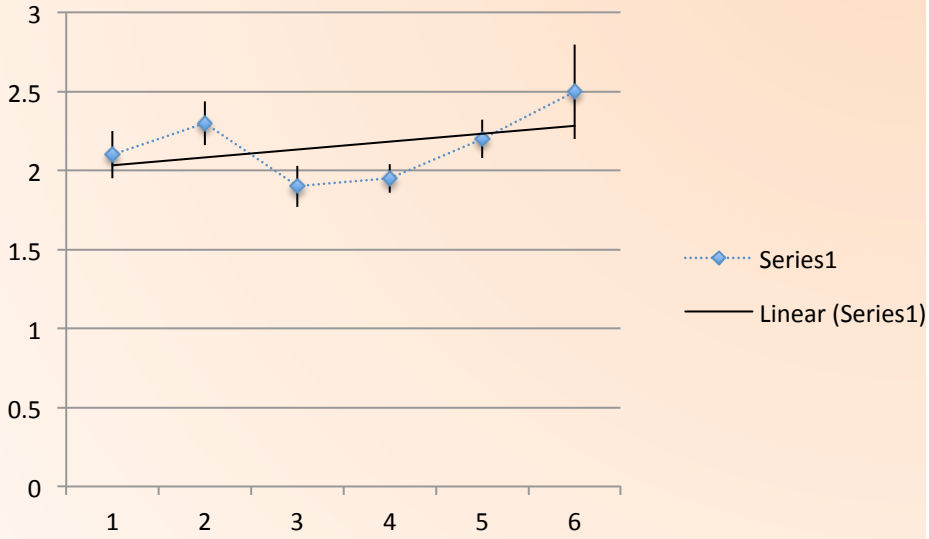
1. For each measured data point you form the quantity d^2:

$$d^2 = [(y_{meas} - y_{pred}) / \varepsilon_x]^2$$

where

y_{meas} is the value of the measured data point
ε_x is the error on the measured data point
y_{pred} is the prediction for that point from y=mx+c

Note that the "difference" is scaled by the error – which correctly accounts for the true significance of the difference: hence better than simple least squares



You may note that least squares fitting is just the same, but assuming the error is 1 (which is likely to be wrong)

2. You then sum d^2_i over all data points, i, to form χ^2

$$\chi^2 = \Sigma_i \; d^2_i$$

3. You then vary "m" and "c" until χ^2 takes its minimum value → later unit

Matrices : Excercise 2 - chisq

For the exercise I want you to:

- Write code to create the two vectors for 10 measured data points and the corresponding 10 measured predictions from y=mx+c
  (note a vector is 10x1 matrix)

- Take the difference vector D (i.e. the vector of [($y_{meas} - y_{pred}$))

- Create an instance of a 10x10 matrix where on each diagonal element you insert the error squared, ε_x^2, corresponding to each measurement. Call this E

- Invert this matrix to form E^{-1} (it would be quite hard to write your own code for this !)

- Form the matrix multiplication quantity $D^t\ E^{-1}\ D$ (where ł means transpose)

- Convince yourself that this is the χ^2 which you will need later to minimise (for m and c)

Once you have got all of this you will know almost everything you need to know to do the most complex parameter fitting in future

Example code is in

- Unit1-Matrices/java/ChisqTask.java
- Unit1-Matrices/py/Chisq.java

When m = 0 and c = 1 then the answer is $\chi^2 = 71.01388888888897$

Matrices : Excercise 2 – Class for y=mx+c

For the first part of the exercise I suggest you write a class to implement the y =mx +c theory *[you will see later why I want this in a class]*

```
//.....................................
//Simple class to calculate a linear function
class Linear {
      private double m = 0.;
      private double c= 1.;

      public Linear( ) {}

      // Set the m and parameters prior to evaluation
      public void setParameters( Matrix in ) {
            m = in.get(0,0) ;
            c = in.get(1,0) ;
            return ;
      }

      // Evaluate for a single value of x
      public double evaluate( double x ) {
            return m*x + c ;
      }

      // Evaluate for a vector of x values
      public Matrix evaluate( Matrix x ) {
            int length = x.getRowDimension();
            Matrix result = new Matrix(length,1) ;
            for(int ind=0;ind<length;++ind) {
                  result.set(ind,0, m*x.get(ind,0)+c ) ;
            }
            return result ;
      }
}
```

Matrices : Excercise : Where to get the Input Data

For data use:

x	y	err
1.0	1.05	0.03
2.0	0.93	0.04
3.0	0.99	0.02
4.0	1.06	0.02
5.0	0.94	0.01
6.0	1.08	0.03
7.0	1.01	0.02
8.0	0.89	0.04
9.0	0.95	0.025
10.0	1.03	0.03

These data are already written in a file called

`testData.txt`

in

`Unit1-Matrices/java`

```
MyFileReader theData = new MyFileReader("testdata.txt");

if(! theData.isValid() ) log("Could not open file
testdata.txt" ) ;

Matrix x = theData.getColumn(0) ;
Matrix y = theData.getColumn(1) ;
Matrix e = theData.getColumn(2) ;

x.print(5,3);
y.print(5,3);
e.print(5,3);
```

Fitting: correlations

Why did I get you to form the χ^2 in this apparently complex way (apart from it being a matrix exercise)

Because in many real situations measurements are correlated. This means that the errors on two measurements may vary in the same direction in a correlated way.

This is expressed by having not only the errors on any pair of measurements, ε_i^2 and ε_j^2 , but also the correlation given by:

$$\varepsilon_i \, \varepsilon_j \, \rho_{ij}$$

where ρ_{ij} is known as the correlation coefficient.

To include this in the fitting all you need to do is insert this into the matrix E on the off diagonal elements, i.e. all E_{ij} then are given by

$$E_{ij} = \varepsilon_i \, \varepsilon_j \, \rho_{ij}$$

where by definition $\rho=1$ when $i=j$

You can see that this gives the simple case we already coded up if there is no correlation ($\rho=0$ when i .ne. j)

This is the only change you have to make when forming the χ^2 and you can see its a lot easier to do using this matrix formulation. E^{-1} is a lot more complex, and you would not write it by hand easily. This way you get it for free having written the much simpler matrix E

Matrices - summary of this unit

Aim:

- To "warm you back up" to OO coding
- To introduce you to the working environment
- To give you the first comparison between "home grown" and "package" code

Objectives

- Define Matrix class
- Implement the class with suitable methods for
  - Instantiation
  - Addition
  - Multiplication
- Perform same operations but using a matrix algebra package
- Solve simultaneous equations
- Formulate a "χ^2" for later use in parameter estimation

All example code is in `Unit1-Matrices/java/.....`