Worksheet 2. Hello Fortran

This worksheet gives a quick introduction to Fortran, in particular the fortran 95 dialect. We will use the Salford compiler and the Plato IDE (Integrated Development Environemnt). Find the Plato IDE from the start menu and open a new file.

Hello World

Type the following program

```
program hello
print *,"Hello world"
end
```

Every program must begin with a `program` and end with an `end` directive. `print` prints to the screen, the first agrument, `*`, is a format code. Compile and run the program using the run option in the project menu.

Data Types

Fortran supports the same data types as matlab (and virtually every other programming language or environment). Unlike matlab variables must be declared in advance and the type of a variable cannot be changed during the course of a program. Open a new file and type in the following program

```
program types
implicit none
integer i
real(kind=kind(0d0)) x
complex(kind=kind(0d0)) c
i=10
x=3e0
c=(3e0,7e0)
print '(A,I8)', "i=",i
print '(A,F18.6)', "x=",x
print '(A,2F18.6,a)',"c=(",c,")"
end
```

Compile and run the program. The result should be

```
i=      10
x=          3.000000
c=          3.000000          7.000000)
```

The Mandelbrot Set

We revisit the Mandelbrot set, this time using fortran to calculate the set of points within the set and using matlab to visualise the results. One way to persuade matlab to accept output from a fortran program is to have that program write an m-file.

```
PROGRAM mandelbrot
IMPLICIT NONE
! Declare variables
INTEGER, PARAMETER :: n=100, itmax=50
COMPLEX(kind=kind(1d0)) :: z(n,n),c(n,n)
REAL(kind=kind(1d0)) :: xx(n),yy(n),x(n,n),y(n,n)
REAL(kind=kind(1d0)), PARAMETER :: xmin=-2.1, xmax=0.6
REAL(kind=kind(1d0)), PARAMETER :: ymin=-1.1, ymax=1.1, zmax=1d6
INTEGER :: i,j
! Vectors of x any y coordinates
DO i=1,n
   xx(i)=xmin+(xmax-xmin)*real(i-1)/(n-1)
   yy(i)=ymin+(ymax-ymin)*real(i-1)/(n-1)
END DO
! Convert to matrices of x and y coordinates
DO i=1,n
   x(:,i)=xx
   y(i,:)=yy
END DO
! c=x+i*y
c=cmplx(x,y)
z=c
! Perform iterations
DO i=1,itmax
   WHERE(abs(z)<zmax)
      z=z**2+c
   END WHERE
END DO
! Write m file
open(unit=1,file="mand.m")
write(1,*) "z=zeros(",n,",",n,");"
DO i=1,n
   DO j=1,n
      IF(abs(z(i,j))<zmax) THEN
         write(1,*) "z(",i,",",j,")=255e0;"
      ELSE
         write(1,*) "z(",i,",",j,")=0e0;"
      END IF
   END DO
END DO
write(1,*) "image(z')"
close(unit=1)
END
```

Compiling and running this file generates the m-file `mand.m`. Open matlab and run the file (type `mand` at the command line having set the correct directory) to produce an image of the Mandelbrot set.

The Lorenz Equations

The Lorenz equations are

$$\frac{dy_1}{dt} = 10\left(y_2 - y_1\right) \tag{1}$$

$$\frac{dy_2}{dt} = 28y_1 - y_2 - y_1y_3 \tag{2}$$

$$\frac{dy_3}{dt} = y_1y_2 - \frac{8y_3}{3} \tag{3}$$

These are solved by the program

```
PROGRAM lorenz
IMPLICIT NONE
REAL(kind=kind(1d0)) y(3)
REAL(kind=kind(1d0)) t
REAL(kind=kind(1d0)), PARAMETER :: dt=1e-2, tmax=50e0
INTEGER i

! initialise
y(1)=0e0; y(2)=1e0; y(3)=0e0
t=0; i=0
! open data file
open(unit=1,file="lorenz.dat")
! Forward euler method (bad but quick to code)
do while(t<tmax)
   y(1)=y(1)+10*(y(2)-y(1))*dt
   y(2)=y(2)+(28*y(1)-y(2)-y(1)*y(3))*dt
   y(3)=y(3)+(y(1)*y(2)-8*y(3)/3)*dt
   t=t+dt; i=i+1
   write(1,'(4E18.6)') t,y(1),y(2),y(3)
end do
close(1)
end
```

Compile and run the code. To visualise the data in matlab type

```
octave:2> dat=load("lorenz.dat");
octave:3> plot(dat(:,2),dat(:,4))
```