Lab #4

Computational Physics I (**Phys381**)
R. Ouyed
Due March 11, 2014 (at the end of class)

[**Total: 100 marks including 20 marks for the report**]

- The latexed report is worth **20%** of the total mark. Only well documented and neatly presented reports will be worth that much! It is not sufficient to give only numerical results and show plots, you should also discuss your results. Include complete figure captions, introduction and conclusion sections.
- Your report must be in a *two-column format*.
- Your report should include your *Fortran code* and *Gnuplot scripts* in an Appendix using the *verbatim* command.
- If applicable, animations should be shown to the teacher and/or TA before handing in the lab report. The animation should be well documented and contains necessary information (student name, assignment number, run time etc ...). Basically, the information should be included in each frame before they are put together.
- You must name your report using the names (last names only): *student1-student2-phys381-lab#.pdf*.
- **Procedure for Handing in your lab report (see instructions on phys381 wbsite)**:

  1) Set permission to your PDF report as 644. It means:
  chmod 644 *student1-student2-phys381-lab#.pdf*
  2) cp -a *student1-student2-phys381-lab#.pdf* **/home/ambrish/phys381/labs/lab#**
  3) Copy a second time to ensure that your exam copied correctly. If you are prompted as to whether or not you would like to replace the existing file, then you report has been successfully submitted.

- **You must check with your TAs (Ambrish or Zach) that your report was received and is readable BEFORE you leave the lab.**

1 Know your machine: [45 Marks]

In this section you will write a main routine that tests your machine limitations. The main code should call two internal subroutines. One giving you the **precision** of your machine and the second subroutine estimates **underflow** and **overflow** limit in your machine.

1.1 Determine your machine precision

[25 marks]

Machine epsilon, ϵ_{M}, is the distance ϵ_{M} between 1.0 and the next largest floating point number; it is a measure of the precision of the floating point number system. An alternative, equivalent, definition of ϵ_{M} is that it is the smallest real number such that $1.0 + \epsilon_{\mathrm{M}}$ is distinguishable from 1.0 in the given floating point number system. The objective of this exercise is to determine ϵ_{M} for both **single and double precision** real numbers on your machine.

A straightforward way of determining ϵ_{M} is to set $\epsilon_{\mathrm{M}} = 0.5$ and then successively decrement ϵ_{M} (by dividing it by 2.0); ϵ_{M} is the smallest possible value for which $1.0 + \epsilon_{\mathrm{M}}$ is greater than 1.0 (i.e. is distinguishable from 1.0).

- **[10 Marks]** Write a Fortran 90 code to test your machine precision. Your final result should be something like the following:

```
! Calculate macheps (single)
 PRINT *, 'single precision machine epsilon = ', mach_eps_single
!
! Calculate macheps (double)
PRINT *, 'double precision machine epsilon = ', mach_eps_double
```

HINT 1: Adding a double-precision number to a single precision number gives a double-precision number! So be careful when using $1.0 + \epsilon_{\mathrm{M}}$.

HINT 2: You will need to use KIND values to specify the different precisions[1] and a WHILE loop to determine ϵ_{M} (for each precision).

[1] From the Portland Group Fortran Reference, the KIND parameter "specifies a precision for intrinsic data types." Thus, in the declaration

```
real(kind=4) :: float32
real(kind=8) :: float64
```

the variable float64 declared as an 8-byte real (the old Fortran DOUBLE PRECISION) and the variable float32 is declared as a 4-byte real (the old Fortran SINGLE PRECISION). This is nice because it allows you to fix the precision for your variables independent of the compiler and machine you are running on.

- **[5 Marks]** Once your code is written and is running, implement the following and comment on your findings:

  case (i):

```
real*4 eps
do while ((1.0+eps) > 1.0)
  temp = 1.0+eps
```

  case (ii):

```
real*4 eps, temp
do while (temp > 1.0)
  temp = 1.0+eps
```

- **[5 Marks]** Using gnuplot, plot ϵ_{M} versus number of iterations (i.e. number of divisions by 2). Use *open red circles* when plotting results from single precision calculations and *open blue circles* when plotting results from double precision calculations.

- **[5 Marks]** Comment on the number of iteration needed to reach the smallest (machine) number in single and double precision cases. Discuss the implication of the smallest (machine) number on calculations in general.

1.2 Determine underflow and overflow in your machine

[20 marks]

Here you will test for the underflow and overflow limits in your machine. An outline structure of the code (subroutine) is given below:

```
do N times
    under = under/2.
    over  = over*2.
    write out:  loop number, under, over
 enddo
```

Your code should have single- and double-precision incorporated using, again, the *kind* option.

- **[10 Marks]** Add a subroutine to your code to check for underflow and overflow.

- **[5 Marks]** Check **where** under- and over-flow occur for single-precision floating-point number (floats). Give your answers as decimals.
- **[5 Marks]** Check **where** under- and over-flow occur for double-precision floating-point number (floats). Give your answers as decimals.

  For (i) and (ii) above, if you want to be more precise regarding the limits of your machine, you may want to multiply and divide by a number smaller than 2 (have it as an input in your code).

2 Loss of accuracy: [30 Marks]

Here we consider the case discussed in section 1.4 in Ouyed&Dobler notes. One of the solutions to the quadratic equation $x^2 - 2x + \epsilon$ is $x_1 = 1 - \sqrt{1-\epsilon}$ where $\epsilon << 1$ is a small number.

(i) **[10 Marks]** Explain how the different expression for x_1, in the first column in Table 1.5, are derived?

(ii) **[10 Marks]** Write a Fortran code that reads in ϵ and writes out the value for x_1. Reproduce the results shown in Table 1.5. Your report should include the table exactly as displayed in Ouyed&Dobler notes.

(iii) **[10 Marks]** Discuss your results and comment on possible implications of the loss accuracy on numerical calculations of similar nature.