PHYS-4007/5007: Computational Physics

Tutorial #5

Determining Machine Precision of Real Numbers

1 Prep Work

Log into your account on the Linux side of the computers in Brown Hall 264. Open a terminal window, then check to make sure that you have a subdirectory called fortran in your login directory using

```
> ls
```

— remember that the '>' above represents the Linux system prompt. **If you do not have such a subdirectory**, make one using

```
> mkdir fortran
```

then change to this subdirectory using

```
> cd fortran
```

Now create a new file called 'limits.f' with the command

```
> emacs limits.f &
```

(remember that the '&' sign put this job in background so that you can still access Linux in the terminal window).

Now open your web browser and go to the 'Computer Programming Tutorials' page on the course home page by clicking the appropriate link. This will now place you at the following web page:

https://faculty.etsu.edu/lutter/courses/phys4007/tutorial_exercises/tutorials.htm

Scroll down to the Programs and Supplemental Files for the Tutorials table and click on the limits.f link in the data table on the Tutorial 5 row. Now, copy and paste this Fortran code from your web browser to your emacs GUI. Save this file after the 'paste' is complete.

2 Determining the Precision of Single- and Double-Precision Numbers with Fortran

For this tutorial, you will be running this limits.f code to determine the machine precision for single- and double-precision numbers using the Fortran programming language in Linux. The limits.f code that you downloaded is based on pseudocode from *Computational Physics: Problem Solving with Python, Third Edition* (2015) of Landau, Páez, and Bordeiana on Page 50.

In the emacs GUI (Graphic User Interface) window, you have the following limits.f code:

```
      PROGRAM LIMITS
C
C This code will determine the machine precision for the machine on which it
C runs.  It is based on pseudocode from Computational Physics: Problem
C Solving with Python, Third Edition} (2015) of Landau, Paez, and Bordeiana
C on Page 50.
C
C At the Unix prompt, use the command:
C     gfortran -o limits.exe limits.f        (Linux Workstations)
C to create the executable version of this code.
C
C Below we define some of the variables that will be used in this code:
C     I:      An integer counter variable for the current iteration.
C     NITER:  The maximum number of iterations allowed.
C     SEPS:   The machine error value for single-precision numbers.
C     SONE:   The current value of 1. + SEPS.
C     DEPS:   The machine error value for double-precision numbers.
C     DONE:   The current value of 1. + DEPS.
C     ASK:    A character string of one character containing either 'y' (yes)
C               or 'n' (no).
C
      INTEGER I, NITER
      REAL SEPS, SONE
      DOUBLE PRECISION DEPS, DONE
      CHARACTER ASK
C
C Set an initial guess for how many iterations this will take.
C
      DATA NITER / 200 /
C
```

```
C Determine the machine precision.  Ask the user to continue every 100th
C iteration.  Define initial values for SEPS and DEPS.
C
      SEPS = 1.0
      DEPS = 1.0D0
C
C Single precision calculation.
C
      PRINT *, 'In the tabel below:'
      PRINT *, '  ITER:  The iteration number.'
      PRINT *, '  ONE:   The value for the REAL variable ONE.'
      PRINT *, '  EPS:   The difference of ONE and the integer 1.'
      PRINT *, 'When the value printed for ONE is exactly 1, the'
      PRINT *, 'value for EPS will be the machine precision for a'
      PRINT *, 'single precision variable.'
      PRINT *, '  '
      PRINT *, ' ITER       ONE             EPS'
      DO 100 I = 1, NITER
         SEPS = SEPS / 2.0
         SONE = 1.0 + SEPS
         WRITE (*,50) I, SONE, SEPS
 50      FORMAT(I4, 2X, F15.12, 2X, 1PE14.7)
         IF (MOD(I, 100) .EQ. 0) THEN
            PRINT *, 'Continue with the single precision ',
     1         'calculations (y/n)? [y]'
            READ (*, '(A)') ASK
            IF ((ASK .EQ. 'N') .OR. (ASK .EQ. 'n')) GOTO 200
         ENDIF
 100  CONTINUE
C
C Double precision calculation.
C
 200  PRINT *, ' '
      PRINT *, 'In the tabel below:'
      PRINT *, '  ITER:  The iteration number.'
      PRINT *, '  ONE:   The value for the REAL*8 variable ONE.'
      PRINT *, '  EPS:   The difference of ONE and the integer 1.'
      PRINT *, 'When the value printed for ONE is exactly 1, the'
      PRINT *, 'value for EPS will be the machine precision for a'
      PRINT *, 'double precision variable.'
      PRINT *, '  '
      PRINT *, ' ITER          ONE                       EPS'
      DO 300 I = 1, NITER
         DEPS = DEPS / 2.0D0
         DONE = 1.0D0 + DEPS
         WRITE (*,250) I, DONE, DEPS
 250     FORMAT(I4, 2X, F23.20, 2X, 1PE23.15)
         IF (MOD(I, 100) .EQ. 0) THEN
            PRINT *, 'Continue with the double precision ',
     1         'calculations (y/n)? [y]'
            READ (*, '(A)') ASK
            IF ((ASK .EQ. 'N') .OR. (ASK .EQ. 'n')) STOP
         ENDIF
 300  CONTINUE
C
      STOP
      END
```

Now go to the terminal window and enter the following command to compile this code:

```
> gfortran -o limits.exe limits.f
```

Once you have successfully compiled this code, run it with

```
> ./limits.exe
```

You will get a line-by-line output for the first hundred iterations. Look at the 'ONE' column (*i.e.*, the second column of numbers) of data for the first line that is exactly equal to '1' (*i.e.*, 1.000000000000) — this should occur around iteration 24. Now, look at the line just above this line (*i.e.*, iteration 23). Write down the value you get for 'EPS' (*i.e.*, the 3rd column of numbers) here (six significant digits is sufficient for this number):

Fortran Single-Precision Machine Error: ________________

This represents the machine error for single-precision numbers. Now answer 'n' (minus the quotes) to the question *Continue with the single precision calculations (y/n)? [y]*.

You will now see another 3-column tabular list of data for the double precision machine error. Once again, look at the 'ONE' column (*i.e.*, the second column of numbers) of data for the first line that is exactly equal to '1' (*i.e.*, 1.00000000000000000000) — this should occur around iteration 53. Now, look at the line just above this line (*i.e.*, iteration 52). Write down the value you get for 'EPS' (*i.e.*, the 3rd column of numbers) here (ten significant digits is sufficient for this number):

Fortran Double-Precision Machine Error: ________________

This represents the machine error for double-precision numbers. Now answer ‘n’ (minus the quotes) to the question *Continue with the double precision calculations (y/n)? [y]*.

Save this information so that you can make use of these values when writing future codes for this course, or any future code you write for your own research.

3 Writing this Code in Python

Using the above Fortran code as a guide, now write your own code in Python 3 (limits.py) and run it with:

```
> python3 limits.py
```

Do not use iPython which brings up a GUI to help you write a Python code. Write your code using the emacs editor.

Record your values here from your Python code:

Python Single-Precision Machine Error: ____________________

Python Double-Precision Machine Error: ____________________

Do you get the same single- and double-precision machine errors in Python that you got with the Fortran compiler? If different, why do you think this occurred?

Once you have completed your Python run, save your work to a USB drive, log off your Linux account and reboot to the MS Windows Operating System.