Floating Point Computation - 2012/13 - Example Sheet 1.

(First edition.)

S1.1. What is BCD (binary-coded decimal) ? Why do pocket calculators tend to work internally in BCD?

S1.2. a) Perform round-to-even on the following decimal numbers 2.2, 3.5, 4.5, 5.6, 10.9. Answers should be natural numbers.

b) Perform round-to-even applied to the following binary numbers 111.11, 111.101, 101.10, 110.1. Answers should be natural numbers expressed in binary.

c) Apply round-to-even for 3 significant digits to the following 1.2345e5, 2.255e-10.

S1.3. Given that most computer languages today support 32-bit signed integers and double-precision floating point, can it be argued that having the integers is silly since they are a subset of the doubles ?

S1.4. Sketch a proof that integer comparison predicates can be applied to the bit patterns of IEEE unsigned floating point and mention any exceptions. (Or do a structured proof if feeling ambitious).

S1.5. What do the following single precision IEEE bit patterns represent, where the msb of the first-listed byte is the sign bit ?

a) 00 00 00 00

b) 80 00 00 00

c) BF 01 00 00

d) 3F C0 00 00

e) 04 04 04 00

S1.6. What is $\log_{10}(2)$? How does this relate the number of bits in a binary integer to the number of digits in a decimal integer ? Give an example or two.

S1.7. From the slides: 'For the example where `a, b & c` all have type float, give an example where `f(a*b+c)` generally gives a different answer to `{ float t=a*b; f(t+c); }`'.

S1.8. Define machine epsilon and sketch a program to determine its value experimentally.

S1.9. F1) (3.4522 * 11.233) + 13.966

F2) (3.4522 - 3.4166) / 17.822

a) Round the numbers in F1 and F2 to three significant figures. Compute the exact results of the expressions before and after rounding.

b) Assume we did only know the numbers in their rounded form (so did not know the accurate versions). Work out the relative **and** absolute error bounds for the two expressions using the lectured rules (abs. errors sum for add and sub, rel. errors sum for times and divide). Compare the errors predicted by the rules with the actual errors if you work to 3sf throughout.

c) What difference does it make if you do not truncate and round to 3sf after each step but only at the end?

d) What is the expected value of the initial quantisation errors and final result assuming fair rounding is used?

Note: questions that are easier to answer with a calculator than without will not be set for Floating Point Tripos.

S1.10. Give hand-crafted code in ML or Java that divides a floating point number by the constant 3. (Note: the slides provide code that divides a 32-bit integer by the constant 10). The following Java fragment might help:

```
public class MyDiv3
{
  public static float div3(final float f)
  { int i = Float.floatToRawIntBits(f);
    //TODO: divide f by 3 by performing operations on i
    // ...

    float fDiv3 = Float.intBitsToFloat(i);
    return fDiv3;
  }

  void testDiv3()
  { //TODO: call div3 with a set of test inputs and check it works
    //...
    //For example:
    int fpuResult = Float.floatToRawIntBits(div3(5.0));
    int myResult = Float.floatToRawIntBits(5.0/3.0);
  }
```

Floating Point Computation - 2012/13 - Examples Sheet 2 of 2.

S2.1. Give four candidate control criteria that might be used to control the number of steps in an iteration and explain when they might be a good or bad choice ?

S2.2. a) What is the formula for finding the square root of a number using Newton Raphson?

b) When do we say that a numerical method is a second order method and when do we say an iteration has order of convergence 2 ?

c) What is the order of convergence of Newton Raphson?

d) Suppose the derivative of the target function is expensive to compute or not available in computable form: suggest a cheaper iteration based on Newton's method (makes the same graphical construction) and estimate/find/state its order of convergence.

S2.3. a) Give the Taylor series expansion for cosine ignoring cubic and higher powers.

b) For what input range does this approximation to cosine completely accurate for a single precision range and domain ? (Completely accurate means its result is always rounded to the closest representable result to the true answer).

c) Is this implementation semi-monotonic ? Why?

d) Precisely why should range reduction be used for periodic trigonometric functions like cosine? What is the best range reduction approach for cosine.

Note: Answers which illustrate the correct approach will be given full marks, even if the final answer is output is wrong by, say, a factor of two.

S2.4. Consider an iteration for the division n/d expressed using the following code fragment:

```
float n = 3223.231;
float d = 0.342;
for(...)
  {
    printf("%f  %f  %f\n", n, d, n/d);
    double f = 2.0 - d;
    n *= f;
    d *= f;
  }
```

a) What happens? Does it work for a good range of n and d ?

b) What is the order of convergence and why ?

c) Is this a good method for division in general ?

S2.5. Consider the quadratic $x^2 + 5x + 2 = (x + 0.438)(x + 4.561) = 0$.

Two iterations can be considered (derive these):

$$\begin{aligned} x_{n+1} &= -2/(x_n + 5) \\ x_{n+1} &= -2/x_n - 5 \end{aligned}$$

Is it the case that one finds one root and the other finds the other ? What happens if the starting guess for one is set at the solution of the other? (You may find it helpful to code this in ML or Java and try a few experiments.)

[NB: You will not be expected to answer problems that are easier to solve using calculator/computer than by hand in Floating Point Tripos Examinations.]

```
Hint: here are the helpful runes for gnuplot:  gnuplot> plot -2/(x+5),x with lines
                                               gnuplot> plot -5-2/(x),x with lines
```

S2.6. Consider the function $\arctan(y/x)$ giving the angle subtended at the origin for the point (x, y).

a) The above function is 'two quadrant' only. Using ML or Java, provide a more-complex implementation of this function that involves '`if`' statement(s) and which gives a different answer in all four quadrants. The new implementation should have the same type as the old implementation.

b) Provide an approximate implementation of four-quadrant arctan that, instead of any sort of Taylor series, uses a few applications of the four basic arithmetic operators and also some '`if`' statements. You may return degrees instead of radians if you prefer.

c) If a computer game needs to rapidly determine whether one point subtends a greater or smaller angle than another, is your implementation in part *b* suitable ?

S2.7. Consider the functions:

F1: $f_1(x, y) = 10000x - 500y$.

F2: $f_2(x, y) = (x + y)/(x - y)$.

a) What are their partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$?

b) What are the condition numbers for the two expressions (log to base 10 of largest relative error amplification) ? Or do we need to qualify these?

c) If we are given simultaneous numeric values for $f_1(x, y)$ and $f_2(x, y)$ we can clearly work out the values of x and y. Is this *backwards stable* (everywhere) ?

S2.8. Consider fixed-point decimal arithmetic:

a) What are the following decimal numbers when rounded to even for a fixed-point decimal arithmetic system with eight places before the point and two after: 10.751, -100.755, -4032.382.

b) Give an example, possibly such as an addition, in this number system where round to minus infinity gives a different answer from the normal round to even rule. Or say why not possible.

c) Give an example of an interval arithmetic subtraction that illustrates the different rounding modes needed for the upper and lower bounds of the result.

NB: An interval arithmetic system might equally well use floating point or fixed point for each member of a max/min pair.

Additional Exercises

These additional exercises are perhaps not as useful as practicing on past Tripos Questions or other exercises set by your supervisor.

AD1. See if you understand the corner case described in (Java hangs when converting)

`http://www.exploringbinary.com/java-hangs-when-converting-2-2250738585072012e-308`

AD2. a) What is meant by a chaotic system ? What feature of $x_{n+1} = 4 * x_n * (1 - x_n)$ leads to chaos ?

```
NB: Verhulsts Logistic map: runes for gnuplot:  gnuplot> plot 4*x*(1-x), x  with lines
```

b) What would we expect and like to see in terms of the numerical values of, and analytic expressions for, the partial derivatives in

i) a well-behaved system,

ii) a chaotic system, and

iii) a system for suggesting re-balancing operations on an investment portfolio ?

AD3. Consider the matrix filled with Fibonacci numbers in the lecture notes:

$$\begin{pmatrix} 17711 & 10946 \\ 6765 & 4181 \end{pmatrix}$$

they were inserted from the bottom right. What undesirable effect does this lead to when considered as a transformer on 2D co-ordinate spaces? Is it backwards stable? An array based on Fib(1,1) always has determinant 1, but what general propery of a 2×2 array leads to this undesirable effect?