Numerical Methods - 2013/14 - Example Sheet 1. (Second edition.)

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

S1.2. Here are two programs for long multiplication in a Java. They are online in the file JPrograms/TwoMultiplies.java

```
static void naive_method()
{
    for (int x=0; x<xd.length; x++)
        for (int y=0; y<yd.length; y++)
            {
                int b = x+y;
                int s = xd[x] * yd[y];
                while (s > 0)
                    {
                        s += rd[b];
                        rd[b++] = s % base;
                        s = s / base;
                    }
            }
}   // Do we need to check most-significant result bit is less than base?

static void vehdic_method()
{
    int b, c_in = 0;
    for (b=0; b<xd.length+yd.length; b++)
        {
            int s = c_in;
            for (int x=0; x<=b; x++)
                {
                    int y = b-x;
                    if (x >= xd.length || y >= yd.length) continue;
                    s += xd[x] * yd[y];
                }
            rd[b] = s % base;
            c_in = s / base;
        }
}
```

Ignoring implementation language, what is the essential difference between these long multiplications and the ML versions in the lecture notes?

Is long multiplication linear or quadratic in word length?

What is the maximum value of s in each method and can s overflow ?

Which method makes the fewest memory operations (operations on arrays) ?

S1.3. IBM System 360 used (uses!) floating point with basis 16. Was (is) this a good idea? Consider a 32-bit word partitioned into exponent and mantissa where the value represented jumps by a factor of 16 for each increment of the exponent. Can a hidden bit still be used?

S1.4. Give the code for base/radix 4 multiplication without Booth's optimisation. This will need to multiply by 3 on occasions.

S1.5. Exercise: Do these 'long' multiplication algorithms work with two's complement numbers? In one or both arguments?

S1.6. a) Where does the hidden bit get re-inserted in the floating point to ASCII base conversion code? (Note an ML version is on the course site but did not get included in the printed notes.) Describe the details of this step. Does this step also involve shifting the mantissa? Does it round correctly when only a limited number of significant digts are requested?

b) Is the provided range in the tables, $\pm 10^{63}$ sufficient?

c) Describe how the fourth base conversion program (ASCII to floating point) not given in the printed slides should work. Give either a program or else a full description of the procedure.

S1.7. Does the provided integer to ASCII function have a bug ? How can one modify that code to avoid suppressing all leading zeros in its special case?

S1.8. 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.

S1.9. 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 ?

S1.10. 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).

S1.11. 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

S1.12. 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.

S1.13. 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); }`'.

S1.14. Define machine epsilon and sketch a program to determine its value experimentally.

S1.15. 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 Tripos exams.

S1.16. 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);
  }
```

Numerical Methods - 2013/14 - Examples Sheet 2 of 3.

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? (This is given in the slide pack but derive it from first principles please.)

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 is 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 ? [*Hint:* This is a bit like an iteration to find the reciprocal. Concentrate first on the denominator's order of convergence to unity.]

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 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. Additional exercise if you have time: Using the poly.m Octave/Matlab program provided, or otherwise, compare the roughness in the plot to your own estimate of the worst-case absolute error. [*Hint:* First Consider whether the standard technique of adding absolute errors it relevant or whether you need to think about loss of precision owing to cancelling.]

S2.7. a) Show all the Chebyshev polynomials pass through (1,1). What other point do half of them pass through ?

```
(* ML coding of Tchebychev basis functions *)
fun compute_cheby n x =
    let fun tch 0 = (1.0, 123456.0) (*snd arbitrary*)
        |   tch 1 = (x, 1.0)
        |   tch n =
                let val (p, pp) = tch (n-1)
                in (2.0 * x * p - pp, p) end
    val (ans, _) = tch n
    in ans end
```

b) The ML program provided online directly uses the coefficients it has generated in its output phase. An alternative is to collate on each power of x to convert the answer back to a regular polynomial. Another possibility is to use the polynomial form of the basis polynomials rather than the recursive form illustrated above. (The expanded forms for the first four or so are in the notes and there is a longer list on Wikipedia.) It is interesting to compare the collated form coefficients with their Taylor equivalents: perhaps make such a comparison if you have time. You will probably need to implement the code to collate the coefficients. What form is likely to be used in practice ?

S2.8. a) The CORDIC algorithm is used to determine the cosine of 22.5 degrees. Give the result obtained by CORDIC limited to using at most three subdivision steps (in other words quite an approximate result). (*Hint:* The fact that 22.5 is a binary subdivision of the right angle is unimportant.)

b) Given that $\arctan(x) \approx x - x^3/3$ explain why we need at "least 30 and potentially 40 iterations" to achieve 10 significant figures as stated on Wikipedia.

c) Is it helpful that $\arctan(0.5) > 90°/4$ and is such a condition strictly needed for the algorithm to work?

d) How would you use CORDIC for angles greater than 45 degrees (or does it work anyway)?

S2.9. 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 a rough and approximate implementation of four-quadrant arctan that, instead of any sort of Taylor series, uses just 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 ?

S2.10. 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 ($\log_{10}$ 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. Are they *backwards stable* (everywhere) ?

S2.11. 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 multiplication 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.

Numerical Methods - 2013/14 - Examples Sheet 3 of 3.

S3.1. a) What condition is needed for a pair of matrices to be multipliable?

b) What is the complexity (in floating point operations (flops)) of converting to upper triangular form for a general set of coefficients ?

c) What is the complexity of converting to the triangular form generated using Cholesky's method ?

d) What is the complexity of forwards and backwards substitution ?

e) If we need to solve a set of equations for R right-hand sides, for what (if any) value of R does it become more efficient to use L/U (or L/D/U) decomposition?

f) What happens if Gaussian Elimination is attempted for non-square matrices?

g) Optional: Complete the Java program LUdemo so that the code to convert A to U also returns the corresponding L. (Hint: The L answer is a modified identity matrix needing just one simple operation to be added into the outer loop of the triangularisation code.) Once you have done that you can solve some equations and check that the product LU is indeed the original A.

h) Optional: Look at the McMaster's L/U lecture note on the course web page. Note that it performs the L/U decomposition in-place, putting both L and U in the same array that originally held A. What is the complexity of its code to find the matrix inverse?

S3.2. Consider the following transformation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1.111 & 1.112 \\ 2.222 & 2.222 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

a) What can you say about condition number ($\log_{10}$ thereof) for points near the origin?

S3.3. Consider the darts throwing exercise in the slides:

```
double x = new_random(0.0, 1.0);
double y = new_random(0.0, 1.0);
if (x*x + y*y < 1.0) hit ++;
darts ++;
```

a) What is the expected change in accuracy if the $<$ were changed to $<=$?

b) Is this an ergodic simulation? Give examples of ergodic and non-ergodic simulations.

c) Make notes on or discuss with your supervisor whether weather forecasting needs to use Monte Carlo simulation or some other type.

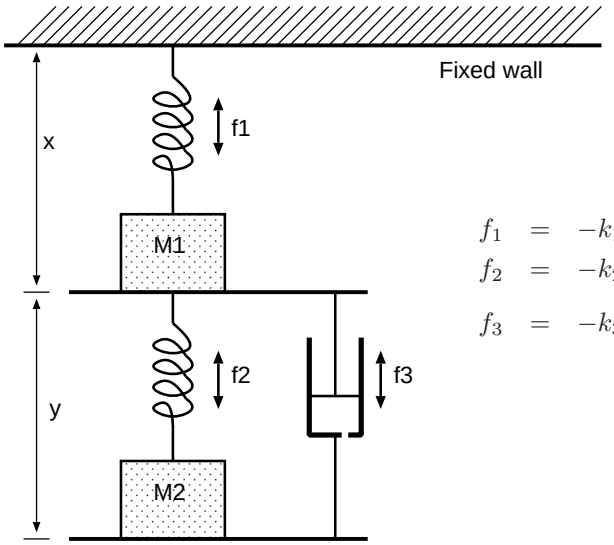
S3.4. *NB: The related material was not lectured owing to running out of time.*

a) What was the signal to noise ratio achieved by NiCam digital audio ?

b) An echo unit is implemented with the following block diagram. Assuming the feedback gain is set such that successive echos die away (i.e. don't get louder) what is the maximum number of discernible echos it could generate ?

Diagram missing - material not lectured.

S3.5. Consider the following 1-D system (with or without gravity). Two masses are connected with two springs and a dashpot. The springs apply restoring forces proportional to their extension above or below their nominal lengths. The dashpot provides a dampening force proportional its rate of operation.



$$
\begin{aligned}
f_1 &= -k_1(x - x_0) \\
f_2 &= -k_2(y - y_0) \\
f_3 &= -k_3 \frac{dy}{dt}
\end{aligned}
$$

a) Give a suitable **state vector**. This is the list of values that must be preserved from one iteration to the next. Write out suitable update equations for a FDTD simulation of this system using forward differences and fixed time step ΔT (Hint: There are four state variables).

b) What, roughly, is the worst error in computing each state variable in terms of ΔT? Does this error accumulate systematically as time progresses?

c) Assuming that the dashpot (shock absorber) is man enough to eliminate significant oscillations in the lower subsystem, what is the resonant frequency of the system and how does this indicate we should choose ΔT.

S3.6. a) Does an adaptive time step help when modelling non-linear components?

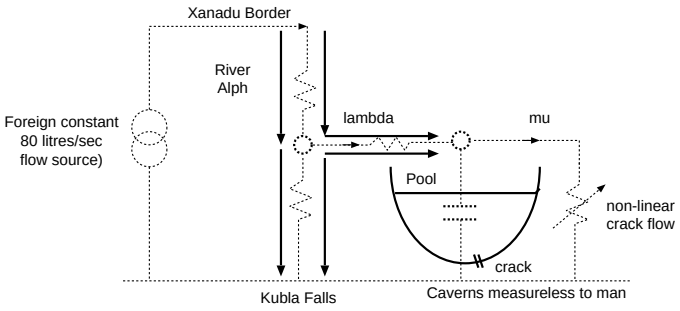
b) Does an adaptive time step help when modelling time-varying components?

S3.7. Relate the heat flow (thermal) and charge flow (electrical) systems, given in the slides, to a third: a water flow (hydraulic) system where each element is a bucket with a given depth of water in it and the buckets are interconnected by narrow pipes between their bottoms.

S3.8. **Killer Question: I now realise a full answer to this question might take a long time, but please at least consider how you would go about each part and discuss it with your supervisor.**

The sacred river Alph enters the mountainous country of Xanadu at flow rate of $F_a = 160$ litres/second. 2 km inside the border it dives over the Kubla Falls. The emperor builds a stately pleasure dome beside the river, half the way to the falls. The dome contains an enormous parabolic basin swimming pool which is fed from the river by a channel conveying λ liters/second from the river. The water extraction causes the river level at the point of extraction to fall a small amount, given by $\delta = \rho\lambda$ where ρ is the constant 1/32=0.03125.

The pool has a crack in the bottom that lets water out at a rate μ litres/second. The water flows through unmeasurable caverns inside the mountains and eventually rejoins the Alph in the lower plains (where there is a sunless sea). The shape of the pool causes the flow through the crack μ to be θd^2 where d is the depth of water in the pool and θ is the constant 0.5. The channel delivers water to the pool at a rate, λ, equal to $\alpha = 32/3 = 10.667$ times the height difference between the levels in the pool and the river. The bottom of the pool is $B = 5$ metres below river bank level, which would be the level of the river if no water were extracted.



This drawing may suggest the water from the channel spills into the pool. But such an arrangement would be like the Kubla Falls and the flow rate would be unaffected by the current depth. In reality the channel joins the pool below the waterline, and hence backpressure applies.

a) **Steady-state solution:** Once the water depth has stabilised we are in the steady state. Find the steady-state solution (also known as the operating point).

$$\begin{aligned} \delta &= \rho\lambda \quad = \lambda/32 \\ \lambda &= \alpha(B-\delta-d) \quad = 32/3 \times (5-\delta-d) \\ \mu &= \theta d^2 \quad = 4 \times d^2 \end{aligned}$$

The pool's radius at height h is $50h^{\frac{1}{2}}$: you need the volume of revolution given by this to find its (non-linear) capacity function.

$$\text{volume}(d) = \pi \int_0^d (50h^{\frac{1}{2}})^2 \mathrm{d}h = 2500\pi d^2/2$$

b) Phrase the simultaneous equations in a matrix form as though they were linear in preparation for being solved using Gaussian elimination. You will needa square conductance matrix multiplied by a potential/height vector to give a vector of node flow sums. Measure heights with respect to the bottom of the pool. Note: one equation

is not linear so you will have to make a linearisation construction. Since we have closed-form, analytic equations for each flow we can differentiate them and potentially use Newton-Raphson iterations. How many iterations are needed to solve the equations to four significant figures? (Note: you do not actually need to construct the iteration function to answer this.)

c) **Time-varying simulation:** The channel is closed every night and the water drains out.

Roughly determine a suitable time step so that the refilling each morning is modelled to four significant figures.

d) Sketch the code for an FDTD simulation of the system. You do not need to use the 'formal' nodal matrix style for this unles you wish to. Instead, just use free-form code. (Write a fully running program if you wish. There are just four simple key assignment statements.)

You may give answers by running your own program and adjusting the parameters or by using mathematical analysis.

Note, this question is a lot longer than any that would be set for Tripos.

Note: If you wish, you can assume the river has even fall between the border and the Kubla Falls. So, as the channel is taken off half way along the river course, the river altitude, when no water is extracted, with respect to the falls, will be half the altitude at the border. However, what happens upstream of the channel take off does not make any difference: a resistance connected in series with the output of a constant flow generator may make the flow generator have to work harder, but this had no effect on the system under study and such a resistance can always be ignored.

Additional Exercises

This is not sheet 3. These are additional exercises that are perhaps not as useful as practising on past Floating Point 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: Verhulst's 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?