Please write your family and given names and **underline** or **capitalize** your family name on the front page of your paper.

Answers should be highly preferably typed (and preferably in latex), and compiled to a single pdf. If an assignment is *very cleanly* handwritten and scanned *on a proper scanner* as a single pdf file, and *not photographed*, then it is also acceptable. Code, output and plots of Q2 should be *embedded* in latex/pdf. Code and output should be embedded with `fixed-width fonts`, e.g. Courier. Font size of all fonts must be 12. Linespacing set to 1.1 or close. Do **not** use dark backgrounds at any point of the pdf file.

Submit the single pdf file 00A1.pdf (with embedded code, output and plots), as well as your code. Thus, the code will be available within latex/pdf, *as well as* separately. Do **not** submit zip and similar files. Only pdf and code (.m, etc) are accepted.

See course website for example of latex, embedding plots, code, using fixed width fonts, etc.

Some points will be given for the quality of presentation.

1. [*15 points*] Assume there are two computer systems:

(i) One that uses fixed-point decimal arithmetic with 2 digits after the decimal point and 3 digits before the decimal point (i.e. ddd.dd). The digit required for the sign is separate and you don't need to worry about it.

(ii) Another that uses floating-point decimal arithmetic with 3 digits mantissa (after the decimal point) and 1 digit for the exponent (i.e. .ddd $\times 10^d$). The digits required for the sign of the mantissa and the sign of the exponent are separate and you don't need to worry about them.

Compute the results of the following five arithmetic operations on each computer system. Assume that all computer operations return the correctly rounded result (i.e. the number closest to the correct answer in the representation being used). Indicate if rounding, overflow or underflow occurs.

$(260 + 1.27) + 0.1$ $\quad 260 + (1.27 + 0.1)$ $\quad 3.08 \times 0.1$ $\quad 3.08 \times 0.01$ $\quad -900 - 100$

What is the range of representable numbers (i.e. the range of numbers for which **no** overflow happens) in each of the two systems? Which one if larger?

What are the ranges of numbers for which underflow happens in each of the two systems? Which one if smaller? (Assume that, in the fixed-point system, the leading decimal digit can be zero, while, in the floating-point system, only normalized mantissae are allowed.)

2.

(a) [*15 points*] Find the condition number of $f(x) = \dfrac{1}{1+2x} - \dfrac{1-x}{1+x}$, simplify as much as you can, and study for what values of x in $\mathbb{R}$ the function $f(x)$ is ill-conditioned. (You may need to use de l' Hospital's rule.)

More specifically, study for what values of x the condition number of $f(x)$ becomes $\pm$infinity. For those values, consider an x which is δ away, where $|\delta|$ small, and find (an approximation) to the condition number in terms of δ. For what ranges of x does the condition number become greater than $1/\varepsilon_{\text{mach}}$?

Also compute the condition number of $f(x)$ for $x = 0$.

(b) [*10 points*] Consider the (numerical) stability of the computation of the expression $\dfrac{1}{1+2x} - \dfrac{1-x}{1+x}$ (as it is given) for some x close to 0, either positive or negative. Explain what problems the computation of the expression $\dfrac{1}{1+2x} - \dfrac{1-x}{1+x}$ may give rise to. Propose a mathematically equivalent expression that is likely to be more stable for x close to 0, and explain. Specify (possibly in terms of $\varepsilon_{\text{mach}}$) a value of x close to 0 that demonstrates the advantage of the expression in (b) over that in (a) and explain.

Note: In (b), we are interested in the stability of an expression that computes $f(x)$ and not in the conditioning of $f(x)$.

(c) [*15 points*] Write a matlab script that, for $i = 1, 2, 3, \cdots, 15$, and $x = -10^{-i}$, computes $f(x)$ as is given in (a) and as you proposed in (b). Also compute the relative error assuming the expression you proposed in (b) is the exact result. Comment on the results.

The script should look like the following:

```
x = -10.^(-1:-1:-15);
f = 1./(1+2*x) - (1-x)./(1+x);
ff = ?; % fill-in your proposed expression
rerr = (ff-f)./ff;
for i = 1:15
    fprintf('%10.2e %22.15e %22.15e %12.4e\n', x(i), f(i), ff(i), rerr(i));
end
```

Do not alter the output format.
Plot in **log-log** scale, the first 8 values of `f` versus the first 8 values of `x` and (the whole) `ff` versus `x`, in **one** plot (two lines on plot). Add labels and a legend.
Plot in **log-log** scale, `abs(rerr)` versus `x`. Add labels.
Place the two plots side-by-side in your document. Add a caption under the plots (either for the two plots together, or for each individual plot).

3.

(a) [*3 points*] Give the Taylor's series for the function $f(x) = \cos x$ about the point 0, writing explicitly all terms of the Taylor polynomial $t_4(x)$ of degree 4 and the remainder R_6. Note that there are only 3 non-zero terms in $t_4(x)$. Indicate $t_4(x)$ and R_6. Note that the remainder involves x and an unknown c. Indicate the (smallest) interval where c lies.

(b) [*5 points*] Using $t_4(x)$ approximate $\cos 1$. Indicate the approximate value in 5 significant decimals. Using R_6, give an upper bound (as sharp as you can) for the (absolute value of the) error of the approximation to $\cos 1$. Explain how you got the bound.

(c) [*5 points*] Give the Taylor's series for $f(x) = \cos x$ about the point 0, in a form so that the $(2n)$th term (for a general n) of the Taylor polynomial $t_{2n}(x)$ of degree $2n$ and the remainder R_{2n+2} are shown explicitly. Using R_{2n+2}, give, in terms of n and x, an upper bound (as sharp as you can) for the (absolute value of the) error in the approximation to $\cos x$ arising from $t_{2n}(x)$.

(d) [*12 points*] Assume you have an accurate way of calculating the Taylor's series in (c), up to any n. How would you use the Taylor's series in (c) to obtain an efficient and accurate approximation to $\cos x$ for
(i) $x = 2$?
(ii) $x = 4$?
(iii) $x = 6$?
(iv) some large x (e.g. $x = 63$)?
(v) some negative x?
Generalize for any x and explain. This generalization can be given as pseudo-code of one or more if-then-else statements.
What is the maximum number of terms (or what is the maximum n) that should be used to keep the absolute value of the remainder $|R_{2n+2}|$ below 10^{-16} for any x? Explain.

(e) [*10 points*] In the form of pseudo-code (a for-loop), give a way of calculating the Taylor's polynomial in (c), without using powers (exponentiations) and without using factorials. (You can use additions, subtractions, multiplications and divisions.)

4. [*10 points*] Consider the Taylor expansions (assume $h > 0$, and h small)

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(\xi_+) \tag{1}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(\xi_-) \tag{2}$$

Substracting (2) from (1) and re-arranging, we get

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \tag{3}$$

where the $O(h^2)$ term involves f''', and is of the form $\frac{h^2}{3}(f'''(\xi_+) + f'''(\xi_-))$. Assuming $|f'''(x)| \le M$ for all $x \in [x-h, x+h]$, the approximation

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

is said to be of *second* order. Now consider taking a different stepsize left and right of x. Show that

$$f'(x) = \frac{-h_R^2 f(x-h_L) + (h_R^2 - h_L^2)f(x) + h_L^2 f(x+h_R)}{h_R(h_R+h_L)h_L} + O(h_R \cdot h_L) \tag{4}$$

and derive the form of the term $O(h_R \cdot h_L)$. (The arising approximation to $f'(x)$ can also be called of second order assuming h_L and h_R decrease at the same rate, and $|f'''(x)| \le M$ for all $x \in [x-h_L, x+h_R]$. But it is *nonuniform*, while (3) is *uniform*.)