Assignment 1 - Solutions

1) a) Taylor series expansion of

$$f(x+h) = f(x) + f'(x)h + f''(x+\xi)\frac{h^2}{2}$$

for some $0 \le \xi \le h$. Substituting into $e(x,h)$,

$$e(x,h) = \frac{f(x)+f'(x)h+f''(x+\xi)h^2/2 - f(x)}{h} - f'(x)$$

$$= \frac{f''(x+\xi)h}{2}.$$

On a computer, it is impossible to exactly compute $f(x+h)$ and $f(x)$. Define,

$$\bar{d}(x,h) = \frac{\bar{f}(x+h) - \bar{f}(x)}{h}$$

and observe

$$\left| \frac{df}{dx} - \bar{d} \right| \le \frac{\varepsilon_{mach}}{h}.$$

By defining $Err(h) = \frac{Ch}{2} + \frac{\varepsilon_{mach}}{h}$, taking the derivative

$\frac{dErr}{dh} = \frac{c}{2} - \frac{\varepsilon_{mach}}{h^2}$ and setting it to zero, we find the local minimum for error as $h \approx 10^{-8}$.

1) b) Taylor series expansions are

$$f(x \pm h) = f(x) \pm hf'(x) + \frac{h^2}{2} f''(x) \pm \frac{h^3}{6} f'''(x) + \frac{h^4}{24} f^{(4)}(x + \xi_{1/2})$$

for some $-h \leq \xi_1 \leq 0$ and $0 \leq \xi_2 \leq h$. Define

$$e(x,h) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - f''(x)$$

and, it trivially follows,

$$= \frac{h^2}{12} f^{(4)}(x + \xi)$$

for some $\xi_1 \leq \xi \leq \xi_2$. Similar to part a), we can say that, floating point error is,

$$Err = \frac{\varepsilon_{mach}}{h^2} + \frac{ch^2}{12},$$

$$\frac{dErr}{dh} = -\frac{\varepsilon_{mach}}{h^3} + ch.$$

Setting the derivative to zero, we find $h \approx 10^{-4}$ is a local minimum.

2) Inexactness:

Consider $x = 3\left(\frac{4}{3} - 1\right) - 1$. We expect $x = 0$, but when run on MATLAB, we get $x = -2.2204e-16$.

Non-commutativeness:

Consider $3\left(\frac{1}{3} + \frac{1}{7}\right) - \left(1 + \frac{3}{7}\right)$. In exact arithmetic it is supposed to be zero, but in floating point we get; $-2.2204e-16$.

Concellation Error

Classical example to this is $10^{17} - 1$. Consider $10^{17} - (10^{17} - 1)$. In exact arithmetic we should get 1 as answer, but MATLAB returns 0 due to loss of significance.

3) Using the formula we get;

$$x_1 = 100\,000\,000, \quad x_2 = 7.4506e-9$$

whereas if we use roots function, they are

$$x_1 = 100\,000\,000, \quad x_2 = 0$$

which is not consistent with exact computation done with hand,

$$\underbrace{x_1 = 5\cdot 10^7 + \sqrt{2.5\cdot 10^{15} - 1}}_{\approx 10^8}, \quad \underbrace{x_2 = 5\cdot 10^7 - \sqrt{2.5\cdot 10^{15} - 1}}_{\approx 10^{-8}}.$$

Assuming we have found x_1 correctly, we can compute

$$x_2 = \frac{c}{x_1 a} = \frac{1}{x_1} = 10^{-8}$$

and this is pretty good as it is exactly what we found using exact computation.

4) 3.1 Using sign and modulus and 32-bit word

$$\underbrace{0}_{\text{sign bit}}\ \underbrace{0 \cdots\cdots\cdots 0}_{31 \text{ bits}},$$

$2^{32}-1$ unique numbers can be represented as -0 and $+0$ have different representations. Using 2s complement we can represent 2^{32} numbers and 0 is unique.

3.2 In 16-bit, range of integers in 2s complement is between -2^{15} and $2^{15}-1$, (or -32768 and 32767)

3.3

discard

1:	0000 0001			
−1:	1111 1111	1	0000	0000
10:	0000 1010			
−10:	1111 0110	1	0000	0000
100:	0110 0100			
−100:	1001 1100	1	0000	0000

3.4 Let's first consider positive integers, note zero is all zeros in 2s complement,

1: $0 \cdots\cdots 01$
2: $0 \cdots\cdots 010$
⋮
$2^{31}-1$: $01 \cdots\cdots 1$.

Now looking at negative numbers,

−1: $1111 \cdots 111$
−2: $1111 \cdots 110$
⋮
-2^{31}: $1000000 \cdots 0$.

(3.5) To visualize consider in 16-bit

$x = 71$ 0000 0000 0100 0111

$y = -71$ 1111 1111 1011 1001.

Let's go through steps, switching bits of x

$\sim y = -72$ 1111 1111 1011 1000

which is pretty much looks like y, but one less that necessary.

Adding 1; we get exactly

$y = -71$ 1111 1111 1011 1001.

This step is necessary because switching bits is equivalent to finding a number $\bar{x}$ for given number x s.t. $\bar{x} + x = 2^{32} - 1$ (or in this case $2^{16} - 1$) but we want a number y s.t. $y + x = 2^{32}$ (or 2^{16} in this case) so it overflows and becomes $y + x = 0$.

(3.6) 50: 0011 0010 -50: 1100 1110

100: 0110 0100 -100: 1001 1100

```
     11            carry
   50: 0011 0010
+ -100: 1001 1100
  -50: 1100 1110
```

```
      1   1  1          carry
  100:  0110 0100
+ -50:  1100 1110
   50: [1]0011 0010
        ↑
     overflow
     (discard)
```

```
     11     1   carry
  50: 0011 0010
  50: 0011 0010
 100: 0110 0100
```