Homework 1 MA580

Lindsay Eddy

September 12, 2018

1. Let $x = 0.d_1 d_2 \dots d_n d_{n+1} \dots \times \beta^b$, $d_1 \neq 0$, $0 \leq d_i \leq \beta - 1$. Let $fl_c(x) = 0.d_1 d_2 \dots d_n \times \beta^b$, where $fl_c(x)$ is obtained using the chopping method.

   Then the absolute error of $fl_c(x)$ has the upper bound

   $$|x - fl_c(x)| \leq \beta^{b-n}.$$

   The relative error of $fl_c(x)$ has the upper bound

   $$\left| \frac{x - fl_c(x)}{x} \right| \leq \frac{\beta^{b-n}}{0.1 \times \beta^b} = \beta^{1-n}.$$

   The relative and absolute errors bounds of $fl_c(x)$ are twice the respective relative and absolute error bounds obtained using the roundng-off approach. The rounding-off approach cuts the maximum error in half by being sensitive to d_{n+1}.

2. Let F be a computer system with 64 bits.

   (a) **The largest/smallest number:**

   Since 11 bits are used for the exponent (1 for the sign and 10 for the magnitude), the largest exponent is +1111111111. This is equal to $2^0 + 2^1 + \dots + 2^9 = 2^{10} - 1 = 1023$.

   Since 52 bits are used for the magnitude of the fraction, the largest fraction is $0.111\dots1$. (There are 52 1s). This is equal to $1 - 2^{-52}$.

   Thus the largest number is $+(1 - 2^{-52}) \times 2^{1023} \approx +8.98846567 \times 10^{307}$, and the smallest number is $-(1 - 2^{-52}) \times 2^{1023} \approx -8.98846567 \times 10^{307}$.

   (b) **The smallest normalized positive number:**

   The largest magnitude of the exponent is 1023 (see part (a)), so the smallest exponent is -1023. Since normalized numbers require the first bit of the fraction to be greater than or equal to 1, the smallest normalized positive number is $+0.100\dots0 \times 2^{-1023} = 2^{-1024} \approx 5.56268465 \times 10^{-309}$.

   (c) **The smallest positive number:**

   The smallest positive number is $+0.000\dots1 \times 2^{-1023} = 2^{-52} \times 2^{-1023} = 2^{-1075}$.

   (d) **Underflow and overflow:**

   An example of underflow is 10^{-2000} – it is smaller than the smallest number a computer can store. An example of overflow is 10^400 – it is larger than the largest number a computer can store.

   (e) **Machine precision:**

   Using the rounding-off approach, machine precision is $\frac{1}{2}\beta^{-n+1}$. In this case, $\beta = 2$ and $n = 52$, so machine precision is

   $$\epsilon = \frac{1}{2} 2^{-52+1} = 2^{-1-52+1} = 2^{-52} = 2.22044604 \times 10^{-16}.$$

(f) **Error bounds:**
The upper bound of the **absolute** error of $fl(x)$ using the rounding approch:

$$|fl(x) - x| \leq \frac{1}{2} \times 2^{-52} \times 2^b = 2^{-53} \times 2^b$$

The upper bound of the **relative** error of $fl(x)$ using the rounding approch:
The relative error is maximized when x is small, i.e., at $x = 0.100\ldots0 \times 2^b$.

$$\left|\frac{fl(x) - x}{x}\right| \leq \left|\frac{2^{-53} \times 2^b}{2^{-1} \times 2^b}\right| = 2^{-52} \approx 2 \times 10^{-16}$$

3. Let us denote machine precision as ϵ_{mach}.

(a) $p = xyz$

$$fl(x) = x(1+\epsilon_1) \qquad fl(y) = y(1+\epsilon_2) \qquad fl(z) = z(1+\epsilon_3)$$

where $|\epsilon_1|$, $|\epsilon_2|$, $|\epsilon_3| \leq \epsilon_{\text{mach}}$.

$$\begin{aligned}
fl(xyz) &= fl(xy)fl(z)(1+\delta_1) && \text{where } |\delta_1| \leq \epsilon_{\text{mach}}. \\
&= fl(x)fl(y)(1+\delta_2)fl(z)(1+\delta_1) && \text{where } |\delta_2| \leq \epsilon_{\text{mach}}. \\
&= x(1+\epsilon_1)y(1+\epsilon_2)z(1+\epsilon_3)(1+\delta_1)(1+\delta_2) \\
&= xyz(1+\epsilon_1+\epsilon_2+\epsilon_3+\delta_1+\delta_2+O(\epsilon_{\text{mach}}^2)) \\
&= xyz(1+\delta) && \text{where } |\delta| \leq 5\epsilon_{\text{mach}}.
\end{aligned}$$

Absolute error bound:

$$|fl(xyz) - xyz| = |xyz(1+\delta) - xyz| = |\delta xyz| \leq 5\epsilon_{\text{mach}}|xyz|$$

Relative error bound:

$$\left|\frac{fl(xyz) - xyz}{xyz}\right| \leq \frac{5\epsilon_{\text{mach}}|xyz|}{|xyz|} = 5\epsilon_{\text{mach}}$$

(b) $s = x + y + z$

$$fl(x) = x(1+\delta_x) \qquad fl(y) = y(1+\delta_y) \qquad fl(z) = z(1+\delta_z)$$

where $|\delta_x|$, $|\delta_y|$, $|\delta_z| \leq \epsilon_{\text{mach}}$.

$$\begin{aligned}
fl(x+y+z) &= \big(fl(x+y) + fl(z)\big)(1+\delta_1) && \text{where } |\delta_1| \leq \epsilon_{\text{mach.}} \\
&= \big[\big(fl(x) + fl(y)\big)(1+\delta_2) + fl(z)\big](1+\delta_1) && \text{where } |\delta_2| \leq \epsilon_{\text{mach.}} \\
&= \big[\big(x(1+\delta_x) + y(1+\delta_y)\big)(1+\delta_2) + z(1+\delta_z)\big](1+\delta_1) \\
&= x + y + z + x\delta_x + y\delta_y + z\delta_z + \delta_2(x+y) + \delta_1(x+y+z) + O(\epsilon_{\text{mach}}^2)
\end{aligned}$$

Absolute error bound:

$$\begin{aligned}
|fl(x+y+z) - (x+y+z)| &= |x\delta_x + y\delta_y + z\delta_z + \delta_2(x+y) + \delta_1(x+y+z)| \\
&\leq |3x\epsilon_{\text{mach}} + 3y\epsilon_{\text{mach}} + 2z\epsilon_{\text{mach}}|
\end{aligned}$$

Relative error bound:

$$\begin{aligned}
\left|\frac{fl(x+y+z) - (x+y+z)}{x+y+z}\right| &\leq \left|\frac{x\delta_x + y\delta_y + z\delta_z + \delta_2(x+y)}{x+y+z}\right| + |\delta_1| \\
&\leq \left|\frac{2x\epsilon_{\text{mach}} + 2y\epsilon_{\text{mach}} + z\epsilon_{\text{mach}}}{x+y+z}\right| + \epsilon_{\text{mach}}
\end{aligned}$$

4. Design an algorithm (in pseudo-code form) to evaulate the following:

   (a) $\log(1+x)/x$ in the interval $[-0.5, 0.5]$
   (We are assuming here that log is log base e.)
   Note that for sufficiently small x,

$$\frac{\log(1+x)}{x} \approx \frac{1}{x}\left[\log(1) + x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4\right]$$
$$= 1 - \frac{1}{2}x + \frac{1}{3}x^2 - \frac{1}{4}x^3$$

   Pseudo-code algorithm:

```
if abs(x) < epsilon
    result = 1 - 1/2*x + 1/3*x^2 - 1/4*x^3;
else
    result = log(1+x)/x;
end
```

   (b) $b - \sqrt{b^2 - \delta}$, where b and δ are two parameters with $b^2 - \delta \geq 0$.
   Note that

$$b - \sqrt{b^2-\delta} = b - \sqrt{b^2-\delta}\left(\frac{b+\sqrt{b^2-\delta}}{b+\sqrt{b^2-\delta}}\right) = \frac{b^2 - b^2 + \delta}{b+\sqrt{b^2-\delta}} = \frac{\delta}{b+\sqrt{b^2-\delta}}$$

   Pseudo-code algorithm:

```
result = delta / ( b + sqrt(b^2 - delta) );
```

   (c) $\nabla\phi(x)/|\nabla\phi(x)|$
   Pseudo-code algorithm:

```
result = ( $\nabla\phi(x)$ ) / ( abs( $\nabla\phi(x)$ ) + epsilon )
```

5. The second formula for computing π seems better, because nearly equal numbers are frequently subtracted, resulting in loss of accuracy. However, I do not know the theoretical accuracy of each method. Also, on the other hand, the first formula looks like it might be faster, since there are fewer operations per term. Therefore, I wrote a program to test the speed and accuracy of both methods.

   In the first method for computing π, 10^6 terms are needed in order to make the error less than 10^{-6}. In the second method, only 7 terms are needed to make the error less than 10^{-6}.

   I used the `timeit` function to measure the amount of time it took each method to compute the first ten terms, as well as how long it took each method to compute π to within $\pi \pm 10^{-6}$ (ie, to get the error less than 10^{-6}). To compute the first 10 terms, it took the first method 5.8985×10^{-7} seconds, and it took the second method 1.8710×10^{-5} seconds. Thus the first method has a faster average computational time per term. I would expect the first method to have a faster average computational time per term even if more than the first ten terms were calculated, becuase the terms in the first method always involve just one operation (a division), and the terms in the second method involve more and more operations the further along in the sequence you go. To compute π with an error of less than 10^{-6}, it took method one 0.0320 seconds, while it took method two only 2.5402×10^{-5} seconds. Thus the second method can calculate π accurately more quickly than the first method can.

6.

(a)

Method 1:

$$\begin{aligned}
f'(x_0)_{\text{approx}} &= \frac{f(x_0+h)-f(x_0)}{h} \\
&= \frac{1}{h}\big[f(x_0)+f'(x_0)h+f''(x_0)h^2/2+f'''(x_0)h^3/6+O(h^4)-f(x_0)\big] \\
&= f'(x_0)+f''(x_0)h/2+f'''(x_0)h^2/6+O(h^3)
\end{aligned}$$

$$\Rightarrow \big|f'(x_0)_{\text{approx}}-f'(x_0)\big| = \big|f''(x_0)h/2+f'''(x_0)h^2/6+O(h^3)\big|$$

Method 2:

$$\begin{aligned}
f'(x_0)_{\text{approx}} &= \frac{f(x_0+h)-f(x_0-h)}{2h} \\
&= \frac{1}{2h}\big[f(x_0)+f'(x_0)h+f''(x_0)h^2/2+f'''(x_0)h^3/6+O(h^4) \\
&\quad -f(x_0)+f'(x_0)h-f''(x_0)h^2/2+f'''(x_0)h^3/6+O(h^4)\big] \\
&= f'(x_0)+f'''(x_0)h^2/6+O(h^3)
\end{aligned}$$

$$\Rightarrow \big|f'(x_0)_{\text{approx}}-f'(x_0)\big| = \big|f'''(x_0)h^2/6+O(h^3)\big|$$

Method 3:

$$\begin{aligned}
f'(x_0)_{\text{approx}} &= \frac{f(x_0)-f(x_0-h)}{h} \\
&= \frac{1}{h}\big[f(x_0)-f(x_0)+f'(x_0)h-f''(x_0)h^2/2+f'''(x_0)h^3/6+O(h^4)\big] \\
&= f'(x_0)-f''(x_0)h/2+f'''(x_0)h^2/6+O(h^3)
\end{aligned}$$

$$\Rightarrow \big|f'(x_0)_{\text{approx}}-f'(x_0)\big| = \big|-f''(x_0)h/2+f'''(x_0)h^2/6+O(h^3)\big|$$

Method 2 is second-order accurate in h, while methods 1 and 3 are first-order accurate in h. Thus method 2 is more accurate in theory.

(b) Write a program to compute the derivative with

- $f_1(x) = x^2$, $x_0 = 1.8$.
- $f_2(x) = e^x \sin x$, $x_0 = 0.55$.

Plot the errors versus h using log-log plot with labels and legends if necessary. In the plot, h should range from 0.1 to the order of machine constant (10^{-16}) with h being cut by half each time ($h = 0.1$, $h = 0.1/2$, $h = 0.1/2^2$, $h = 0.1/2^3$, $\cdots$, until $h \leq 10^{-16}$.)

Tabulate the absolute and relative errors corresponding to $h = 0.1$, $0.1/2$, $0.1/4$, $0.1/8$, and $0.1/16$ (that is, difference choices of h compared with that used in the plots). The ratio (should be around 2 or 4) is defined as the quotient of two consecutive errors.

$f_1 = x^2$, $x_0 = 1.8$:

Method 1:

$1/h$	Abs. Error	Rel. Error	ratio
10	1.0000×10^{-1}	2.7778×10^{-02}	–
20	5.0000×10^{-02}	1.3889×10^{-02}	2.0000
40	2.5000×10^{-02}	6.9444×10^{-03}	2.0000
80	1.2500×10^{-02}	3.4722×10^{-03}	2.0000
160	6.2500×10^{-03}	1.7361×10^{-03}	2.0000

Method 2:

$1/h$	Abs. Error	Rel. Error	ratio
10	3.1086×10^{-15}	8.6351×10^{-16}	–
20	3.1086×10^{-15}	8.6351×10^{-16}	1.0000
40	1.4655×10^{-14}	4.0708×10^{-15}	2.1212×10^{-01}
80	5.7732×10^{-15}	1.6037×10^{-15}	2.5385
160	6.5281×10^{-14}	1.8134×10^{-14}	8.8435×10^{-02}

Method 3:

$1/h$	Abs. Error	Rel. Error	ratio
10	1.0000×10^{01}	2.7778×10^{-02}	–
20	5.0000×10^{-02}	1.3889×10^{-02}	2.0000
40	2.5000×10^{-02}	6.9444×10^{-03}	2.0000
80	1.2500×10^{-02}	3.4722×10^{-03}	2.0000
160	6.2500×10^{-03}	1.7361×10^{-03}	2.0000

$f_2 = e^x \sin x$, $x_0 = 0.55$:

Method 1:

$1/h$	Abs. Error	Rel. Error	ratio
10	1.4950×10^{-01}	6.2720×10^{-02}	–
20	7.4329×10^{-02}	3.1183×10^{-02}	2.0113
40	3.7048×10^{-02}	1.5543×10^{-02}	2.0063
80	1.8490×10^{-02}	7.7571×10^{-03}	2.0037
160	9.2326×10^{-03}	3.8734×10^{-03}	2.0027

Method 2:

$1/h$	Abs. Error	Rel. Error	ratio
10	1.8876×10^{-03}	7.9192×10^{-04}	–
20	4.6583×10^{-04}	1.9543×10^{-04}	4.0521
40	1.0899×10^{-04}	4.5725×10^{-05}	4.2741
80	1.9691×10^{-05}	8.2611×10^{-06}	5.5349
160	2.6386×10^{-06}	1.1070×10^{-06}	7.4627

Method 3:

$1/h$	Abs. Error	Rel. Error	ratio
10	1.4573×10^{-01}	6.1137×10^{-02}	–
20	7.3397×10^{-02}	3.0793×10^{-02}	1.9854
40	3.6830×10^{-02}	1.5451×10^{-02}	1.9929
80	1.8451×10^{-02}	7.7406×10^{-03}	1.9961
160	9.2379×10^{-03}	3.8756×10^{-03}	1.9973

Analyze and explain your plots and tables. What is the best h for each case with and without round-off errors?

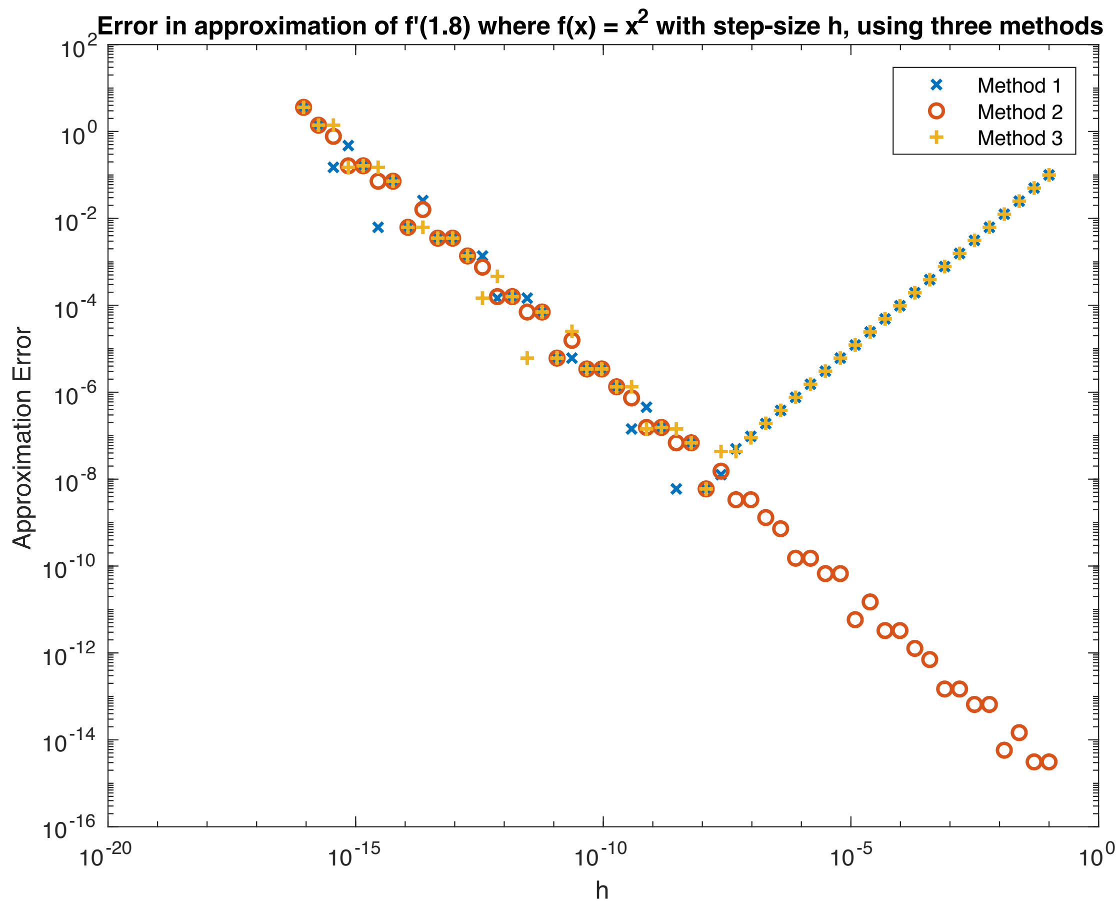
Since method 2 has higher order theoretical accuracy than methods 1 and 3, it achieves high accuracy even for relatively large h. In each method, as h decreases, the round-off error increases due to the subtraction of nearly equal numbers. In method 2, the round-off error dominates even for fairly small h, since the theoretical error is so low. Thus the best h for method 2 (with round-off error) is $h = 0.1$ for f_1 and $h = \frac{0.1}{2^4}$ for f_2.

In methods 1 and 3, the theoretical error dominates for large h, since methods 1 and 3 have only first-order theoretical accuracy. Only when h is small does round-off error dominate in these methods. The best h for methods 1 and 3 (with round-off error) is $h \approx 10^{-8}$ for f_1 and $10^{-11} \leq h \leq 10^{-5}$ for f_2.

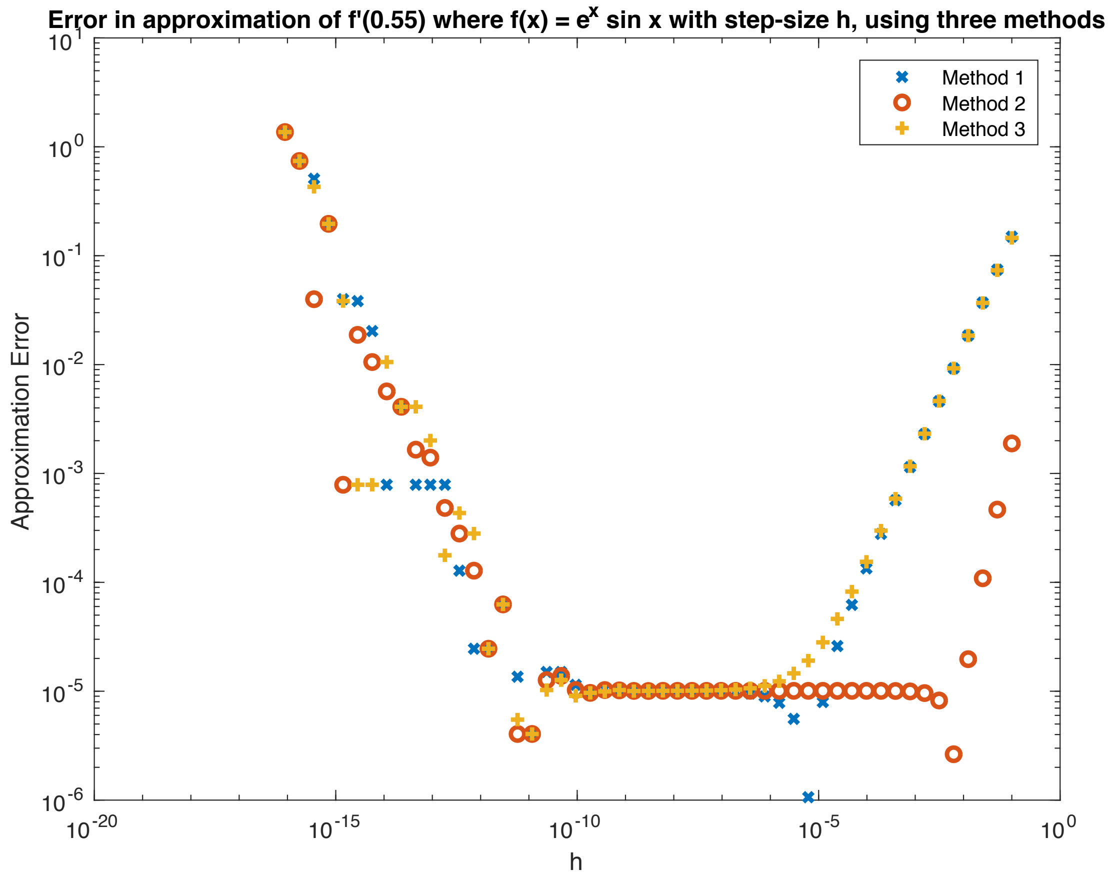
In all methods, if round-off error were not a factor, the best h would be the smallest h, because the theoretical error approaches zero as h approaches zero.

In methods 1 and 3, for the values of h shown in the tables, the ratio of consecutive errors is about 2. This means that the error decreases by half as h decreases by half. This is consistent with the first-order accuracy in h, which we derived in part (a).

In method 2, for f_1, the error is too noisy for the ratio of consecutive errors to be consistent. Note however, that in f_1, for the values of h represented in the table, the error increases as h decreases by half. This is due to round-off error. In method 2 for f_2, for the values of h represented in the table, the ratio of consecutive errors is about 4. This means that the error decreases by one fourth as h decreases by half. This is consistent with second-order accuracy in h, which we derived in part (a).



(a) Error plot for $f_1(x) = x^2$, $x_0 = 1.8$.



(b) Error plot for $f_2(x) = e^x \sin x$, $x_0 = 0.55$.