CMSC660 Homework 1

Daniel Allen Schug

August 2024

1. **(4 pts)** (Problem 7 from Section 2.10 in D.Bindel's and J.Goodman's book "Principles of Scientific Computing") Starting with the declarations

```
float x, y, z, w;
const float oneThird = 1/ (float) 3;
const float oneHalf = 1/ (float) 2;
// const means these never are reassigned
```

we do lots of arithmetic on the variables `x, y, z, w`. In each case below, determine whether the two arithmetic expressions result in the same floating point number (down to the last bit) as long as no NaN or inf values or denormalized numbers are produced.

To clarify, we write in verbatim when referring to code, where operations of the form `a op b` are assumed to be equivalent to float$(a \text{ op } b) = (a \text{ op } b)(1+\epsilon_i)$ where ops in this case are restricted to addition and multiplication. In each case, we expand out the **TOP** and **BOTTOM** expressions to show equivalence, where we use $\{\epsilon_i\}_{i=0}^{N}$ as needed to express the error, where each $|\epsilon_i| < \epsilon_M$. ϵ_M is defined to be machine epsilon. In most cases we omit $\epsilon_i \cdot \epsilon_j$, stating them as lower order terms since these terms are insignificant for error analysis.
The code for generating figures, and working problem 2 can be found here under the `hw1` section.

(a)
```
( x * y ) + ( z - w )
( z - w ) + ( y * x )
```

Solution:

TOP `( x * y ) + ( z - w )`
$= ((xy)(1+\epsilon_1) + (z-w)(1+\epsilon_2))(1+\epsilon_3)$
$= xy + (z-w) + xy\epsilon_1 + xy\epsilon_3 + (z-w)\epsilon_2 + (z-w)\epsilon_3 + xy\epsilon_3$
$= xy + z - w + xy\epsilon_1 + xy\epsilon_3 + z\epsilon_2 + z\epsilon_3 - w\epsilon_2 - w\epsilon_3$

BOTTOM `( z - w ) + ( y * x )`
$= ((z-w)(1+\epsilon_2) + (yx)(1+\epsilon_1))(1+\epsilon_3)$
$= (z-w) + (yx) + yx)\epsilon_1 + yx\epsilon_3 + (z-w)\epsilon_2 + (z-w)\epsilon_3$
$= xy + z - w + xy\epsilon_1 + xy\epsilon_3 + z\epsilon_2 + z\epsilon_3 - w\epsilon_2 - w\epsilon_3$

in this case, the associated error is identical, therefore the **TOP** expression produces the *same* result as the **BOTTOM** expression

(b)
```
( x + y ) + z
x + ( y + z )
```

Solution:

TOP `( x + y ) + z`
$= ((x+y)(1+\epsilon_1)+z)(1+\epsilon_2)$
$= x+y+z+(x+y)\epsilon_1+(x+y)\epsilon_2+z\epsilon_2$
$= x+y+z+x\epsilon_1+x\epsilon_2+y\epsilon_1+y\epsilon_2+z\epsilon_2$

BOTTOM `x + ( y + z )`
$= (x+(y+z)(1+\epsilon_3))(1+\epsilon_4)$
$= x+y+z+(y+z)\epsilon_3+(y+z)\epsilon_4+x\epsilon_4$
$= x+y+z+y\epsilon_3+y\epsilon_4+z\epsilon_3+z\epsilon_4+x\epsilon_4$

for **TOP**, we have an absolute error of $x(\epsilon_1+\epsilon_2)+y(\epsilon_1+\epsilon_2)+z\epsilon_2$, whereas for **BOTTOM** we have an absolute error of $y(\epsilon_3+\epsilon_4)+z(\epsilon_3+\epsilon_4)+x\epsilon_4$.
These will clearly produce different results if x and z differ dramatically in order of magnitude. We can demonstrate this using $x = 1.0e32$, $y = -1.0e32$ and $z = 1.0$. If we first compute $(x+y)+z$, we obtain $(0)+1 = 1$, whereas if we compute $x+(y+z)$ we obtain $1.0e32+(-1.0e32) = 0$.
therefore the **TOP** expression produces a *different* result from the **BOTTOM** expression

(c)
```
x * oneHalf + y * oneHalf
( x + y ) * oneHalf
```

Solution:

We begin by noting that `oneHalf` has no error in its floating point representation, meaning that there is no error associated with this computation, eg. dividing 1 by 2 yields no rounding error. Proceeding with this assumption,

TOP `x * oneHalf + y * oneHalf`
$= ((x * \frac{1}{2})(1+\epsilon_1) + (y * \frac{1}{2})(1+\epsilon_2))(1+\epsilon_3)$
Expanding this out, we obtain
$= \frac{x}{2} + \frac{y}{2} + \frac{x\epsilon_1}{2} + \frac{y\epsilon_2}{2} + \frac{x\epsilon_3}{2} + \frac{y\epsilon_3}{2} +$ lower order terms

BOTTOM `(x + y) * oneHalf` $= ((x+y)(1+\epsilon_4) * \frac{1}{2})(1+\epsilon_5)$
Expanding this out, we obtain
$= \frac{x}{2} + \frac{y}{2} + \frac{x\epsilon_4}{2} + \frac{y\epsilon_4}{2} + \frac{x\epsilon_5}{2} + \frac{y\epsilon_5}{2} +$ lower order terms

The absolute error for **TOP** is $\frac{x(\epsilon_1+\epsilon_3)}{2} + \frac{y(\epsilon_2+\epsilon_3)}{2}$, and the absolute error for **BOTTOM** is $\frac{x(\epsilon_4+\epsilon_5)}{2} + \frac{y(\epsilon_4+\epsilon_5)}{2}$. Since each error term $\frac{\epsilon_i+\epsilon_j}{2} < \epsilon_M$, **TOP** and **BOTTOM** are each less than $2\epsilon_M$.

The only place where the two statements diverge (I checked this) is when we near the maximum number representable by the floating point arithmetic, like 2^{1023}. At this point, **BOTTOM** becomes inf, while **TOP** is one bit away, however this does not count.

Therefore the **TOP** expression produces the *same* result as the **BOTTOM** expression (provided the absence of NaN, inf, or denormalized values)

(d)
```
x * oneThird + y * oneThird
( x + y ) * oneThird
```

Solution:

TOP `x * oneThird + y * oneThird`
$= (x\frac{1}{3}(1+\epsilon_1)(1+\epsilon_2) + y\frac{1}{3}(1+\epsilon_1)(1+\epsilon_3))(1+\epsilon_4)$
Expanding this out, we obtain
$= \frac{x}{3} + \frac{y}{3} + \frac{y\epsilon_1}{3} + \frac{x\epsilon_1}{2} + \frac{y\epsilon_3}{3} + \frac{x\epsilon_4}{3} + \frac{y\epsilon_4}{3} +$ lower order terms

BOTTOM `(x+y) * oneThird`
$= ((x+y)(1+\epsilon_5)/3)(1+\epsilon_1)(1+\epsilon_6)$
Expanding this out, we obtain
$= \frac{x}{3} + \frac{y}{3} + \frac{x\epsilon_1}{3} + \frac{y\epsilon_1}{3} + \frac{x\epsilon_5}{3} + \frac{y\epsilon_5}{3} + \frac{x\epsilon_6}{3} + \frac{y\epsilon_6}{3} +$ lower order terms

it is easy to see that the errors are different, and using knowledge of the non-terminating nature of $\frac{1}{3}$. we can construct an example where the two are not the same. Take $x = 2.0 \times 10^{16}$, and $y = 8.0 \times 10^{16}$. In the top case, we incur 2 possible errors in dividing by 3, since neither x nor y cleanly divide. In the bottom case we incur only 1 possible error in dividing by 3, and confirming this numerically indicates that indeed they differ by a single bit in the last place.

Therefore the **TOP** expression produces a *different* result from the **BOTTOM** expression

2. **(10 pts)** The *tent map* of the interval $[0, 1]$ onto itself is defined as

$$f(x) = \begin{cases} 2x, & x \in [0, \frac{1}{2}) \\ 2 - 2x, & x \in [\frac{1}{2}, 1] \end{cases}$$

Consider the iteration $x_{n+1} = f(x_n), n = 0, 1, 2, ...$

(a) What are the fixed points of this iteration, i.e. the points x^* such that $x^* = f(x^*)$? Show that these fixed points are unstable, i.e., if you start iteration at $x^* + \delta$ for any δ small enough then the next iterate will be farther away from x^* than $x^* + \delta$.

Solution:

We begin by finding $x^* = f(x^*)$. We proceed by splitting the domain into $x \in [0, \frac{1}{2})$ and $x \in [\frac{1}{2}, 1]$.
If $x \in [0, \frac{1}{2})$, then $f(x^*) = 2x^* = x^*$ iff. $x^* = 0$
and if $x \in [\frac{1}{2}, 1]$, then $f(x^*) = 2 - 2x^* = x^* \rightarrow 2 = 3x^*$ iff. $x^* = \frac{2}{3}$

As for their instability, we define instability around a fixed point x^* such that for some arbitrarily small δ, $|f(x^* + \delta) - x^*| > \delta$, where $|.|$ denotes the distance expressed by the inner quantity in absolute terms.

For $x^* = 0$, $f(x^* + \delta) = f(\delta) = 2 * \delta$, which is greater than δ, thus f is unstable at $x^* = 0$.

Similarly, for $x^* = \frac{2}{3}$, $f(x^* + \delta) = f(\frac{2}{3} + \delta) = 2 - 2(\frac{2}{3} + \delta) = \frac{2}{3} - 2\delta$

(b) Prove that if x_0 is rational, than the sequence of iterates generated starting from x_0 is periodic.

Solution:

Let $x_0 \in [0,1]$ be a rational number,

Then x_0 can be expressed as $\frac{p_0}{q_0}$ where $p_0, q_0 \in \mathbb{N}$

From properties of the rationals, if x is rational, then $2x$ is rational and so is $2-2x$, therefore $f(x)$ is also rational which implies x_i can be expressed as $\frac{p_n}{q_n}$ where $p_n, q_n \in \mathbb{N}$

Since $x_n \in [0,1]\ \forall n \in \mathbb{N}$, and $x_{n+1} = f(x_n) = \begin{cases} 2\frac{p_n}{q_n}, & \frac{p_n}{q_n} \in [0, \frac{1}{2}) \\ 2 - \frac{2p_n}{q_n} = \frac{2q_n - 2p_n}{q_n}, & \frac{p_n}{q_n} \in [\frac{1}{2}, 1] \end{cases}$

it follows that the denominator q_n is unchanged by the action of f, therefore x_n can be represented as $\frac{r_n}{q_n}$ for some $r_n \in \mathbb{N}$

Since $q_n = q$ remains fixed, and $\frac{r_n}{q} \in [0,1]$, we know that there will be at most q distinct numerators.

Thus, since there are finitely many numerators on $[0,1]$, precisely q-many numerators $r_i|_{i=0}^{q}$ (containers), and infinitely many iterates (items), we can invoke the pigeonhole principle which states that if there are more items than there are containers, at least one item must go in the same container as another. Therefore there must exist at least k, j such that r_k is attained by another iterate r_j

Therefore it follows that $f(\frac{r_k}{q}) = f(\frac{r_j}{q})$ for some $k, j \in \mathbb{N}$ and therefore f is periodic for rational x_0.

(c) Show that for any period length p, one can find a rational number x_0 such that the sequence of iterates generated starting from x_0 is periodic of period p.

Solution:

Consider the binary representation for some rational x_0, where x_0 is not finitely represented by powers of two (non terminating), eg.

$$x_0 = \sum_{i=1}^{\infty} \frac{b_i(x_0)}{2^i}$$

where $b_i(x_0) = b_{i+k}(x_0)$ for all $k \geq 1$

In this light, we can reinterpret f in its dyadic behavior by noting that each application of $f(x)$ either multiplies it by 2, shifting all bits to the left, or $2-2x = 2(1-x)$, complimenting all bits followed by shifting all bits to the left.

from this reasoning, it can be seen that if x_0 is a non terminating rational number, we can select non terminating sequences of b_i which retain their order under bit shifts and compliments.
For example, for period $p = 3$, we can construct x_0 with $b_i = \{0, 0, 1, 1, 0, 0, 1, 1, ...\}$ so that every 3 left bit shifts, b_0 will equal 1 (making $x_0 > \frac{1}{2}$), causing a complimenting and returning to $2x_0$, meaning that x_0 constructed in this way will be periodic with period 3.

Similarly, for any p, we can construct x_0 such that $b_i = \{0, ..., 0, 1, ..., 1, ...\}$ where there are $p-1$ 0-s followed by $p-1$ 1-s (repeating forever). By the above reasoning, f will shift left p times until $b_0 = 1$, where it will return to $2x_0$, thereby making f periodic with period p for x_0 constructed in this way.

(d) Generate several long enough sequences of iterates on a computer using any suitable language (Matlab, Python, C, etc.) starting from a pseudorandom x_0 uniformly distributed on $[0, 1)$ to observe a pattern. I checked in Python and C that 100 iterates are enough. Report what you observe. If possible, experiment with single precision and double precision.

Solution:

Under iteration using pseudorandom x_0 on $[0, 1)$ we observe that the sequence breaks down after around over 50 iterations on 64-bit floating point numbers, eventually flatlining to 0. When the same iteration is performed with 32-bit floating point numbers, the sequence breaks down sooner, flatlining after only around 25 iterations. Additionally, there is a kind of dead zone around $\frac{2}{3}$, which was predicted by our determination that this is a fixed point for f.

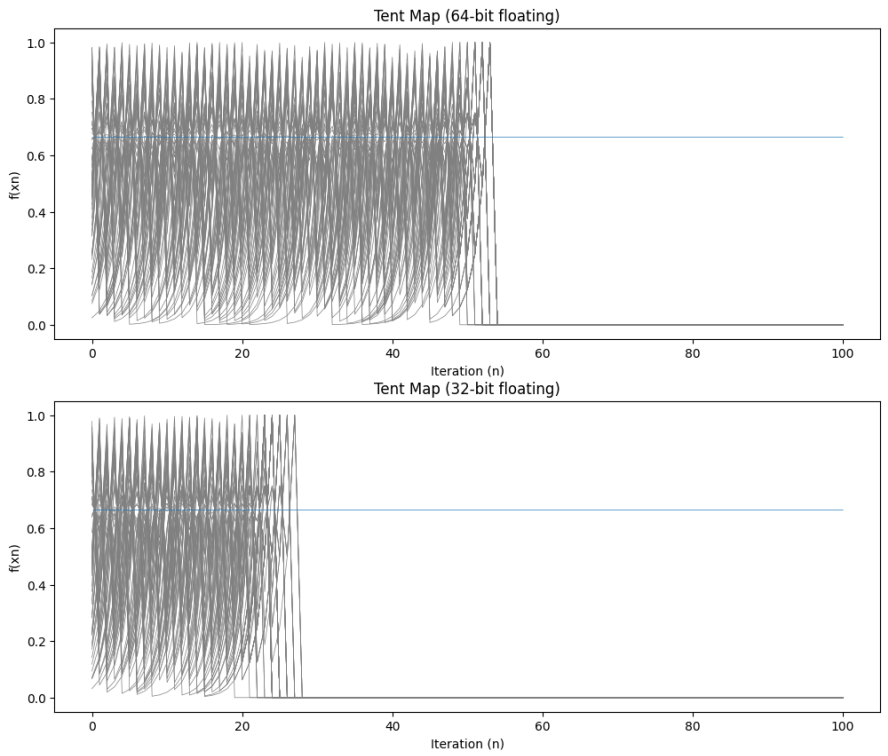

Figure 1: Tent map iterated over 100 iterations. Blue line is fixed point of y=2/3.

(e) Explain the observed behavior of the generated sequence of iterates.

Solution:

These numbers for termination are suspiciously close to the IEEE standard number of mantissa bits. This is no coincidence, and it can be explained using a similar reasoning to our response in c). With each iteration, we apply a left bit shift, which results in the last digit in the representation being zero. It is easy to see that if you do this for a maximum of 53 times on 64-bit floating point numbers, you will eventually reach 0, meaning that this is expected using standard floating point representations. One can proceed even more vaguely, by adding that each iterate involves imperfect floating point operations which accumulate, making the mapping not precisely in $[0, 1]$, causing it to degenerate.