Homework 0

Theory:

1. Give examples of floating-point numbers $x$, $y$, and $z$ for which addition is not associative. Do the same for multiplication. (Assume a 64 bit floating point representation, with 10 bits allocated to the power, 53 to the mantissa, and 1 to the sign.)

Solution:

• Let $x = 1$, $y = 2^{-54}(1 + 2^{-1})$, and $z = y$. Then since $y < \epsilon/2$,

  \[(x + y) + z = x + z = x = 1.\]

  On the other hand, $\epsilon/2 < y + z < \epsilon$, whence

  \[x + (y + z) = 1 + \epsilon.\]

• Let $x = 2$, $y = 2^{-1}$, and $z = 1 + 2^{-52}$. Then

  \[x \ast (y \ast z) = 2 \ast [1/2 \ast (1 + 2^{-52})] = 2 \ast [1/2] = 1,\]

  while

  \[(x \ast y) \ast z = (2 \ast 1/2) \ast (1 + 2^{-52}) = (1 + 2^{-52}).\]

2. (Challenge 1.6) Suppose $x$ and $y$ are true (non-zero) values and $\tilde{x}$ and $\tilde{y}$ are our approximations to them. Express the errors as

  \[
  \tilde{x} = x(1 - r) \\
  \tilde{y} = y(1 - s).
  \]

  Show that the relative error in $\tilde{x}$ is $|r|$ and the relative error in $\tilde{y}$ is $|s|$. Also show that the relative error in $\tilde{x}\tilde{y}$ (as an approximation to $xy$) is bounded by $|r| + |s| + |rs|$.

Solution:

• The relative error in $\tilde{x}$ is

  \[
  \frac{|\tilde{x} - x|}{|x|} = \frac{|xr|}{|x|} = r.
  \]

  The proof for $\tilde{y}$ is identical.
We calculate as follows:

\[
\frac{|\bar{y} - xy|}{xy} = \frac{|xy(1 - r)(1 - s) - xy|}{|xy|} = |rs - r - s| \\
\leq |rs| + |r| + |s|.
\]

Computation:

1. Write a Matlab program that determines the length of the mantissa on your machine. (Hint: use a simple while loop.)

**Solution:**

The key is to observe that machine epsilon is given by \(2^{-m}\), where \(m\) is the length of the mantissa. Thus it suffices to successively raise \(m\) until \(1 + 2^{-m} = 1\). This can be coded as follows:

```matlab
% Script: find_mantissa.m
% Description: calculates the length of the mantissa on this machine

x = 1;
m = 1;
while(1 + 2^{-m} > 1)       % while 2^{-m} > machine epsilon
    m = m+1;            % successively increase m
end
mantissa_length = m-1; % subtract the 1 that was added in the last % iteration of the loop
```

2. (Challenge 1.8) Write a Matlab function that computes the two roots of a quadratic polynomial with good relative precision.

**Solution:**

- The key is to avoid a subtraction that might lead to “catastrophic cancelation”. One way to do this is to use the quadratic formula to find one root (the one that involves adding two terms), and then use the fact that \(r_1r_2 = c/a\) to calculate the other. Here’s how you could code that in Matlab:

```matlab
function [r1 r2] = quadratic_root(a,b,c)
% function [r1 r2] = quadratic_root(a,b,c)
% Description: computes the two roots of a quadratic ax^2 + bx + c
% Inputs:
% a, b, c scalar coefficients of the quadratic
% Outputs:
% r1, r2 scalar roots

% error checking
if a == 0
    disp('Warning: not a quadratic function');
    if b == 0
        r1 = -c/b;
        r2 = nan;
    return;
end
```
elseif c == 0
    disp('Warning: this a constant function with no roots');
    r1 = nan;
    r2 = nan;
    return;
end

end

disc = sqrt(b^2 - 4*a*c); % calculate the discriminant

% case 0: imaginary roots
if ~isreal(disc)
    r1 = (-b + disc)/(2*a);
    r2 = r1'; % roots are complex conjugates
    return;
end

% next deal with real roots
sdisc = sign(real(disc)); % find the sign of the discriminant
sb = sign(b); % find the sign of b

% case 1: zero is a double root
if sdisc == 0 & sb == 0
    r1 = 0;
    r2 = 0;
% case 2: non-zero double root
elseif sdisc == 0
    r1 = -b/(2*a);
    r2 = r1;
% case 3: distint roots
elseif sdisc == sb
    r1 = (-b - disc)/(2*a); % this root cannot be zero
    r2 = c/(r1*a);
else
    r1 = (-b + disc)/(2*a); % ditto
    r2 = c/(r1*a);
end %if