Homework 3 (Solution)

Math 3607

Summer 2021

Tae Eun Kim

Table of Contents

Problem 1 (LM 9.3--3a)

```
format long g
```

Successor of 8:

```
(8 + 4*eps) - 8
```

```
ans =
     0
```

```
(8 + 4.01*eps) - 8
```

```
ans =
       1.77635683940025e-15
```

Observe that the gap between `8 + 4.01*eps` and `8` is not `4.01*eps`, but rather `8*eps`.

```
8*eps
```

```
ans =
       1.77635683940025e-15
```

Predecessor of 16:

```
16 - (16 - 4.01*eps)
```

```
ans =
       1.77635683940025e-15
```

```
16 - (16 - 4*eps)
```

```
ans =
     0
```

Note that `16 - 4*eps` is registered to be the same as `16` in MATLAB while `16 - 4.01*eps` is rounded down to `16 - 8*eps`. This is how we know that `16 - 8*eps` comes immediately before `16` on the floating-point number system.

Neighbors of 2^{10}:

The gap between 2^{10} and the next floating-point number is $2^{10} \cdot \texttt{eps} = 2^{-42}$.

```
(2^10 + 2^9*eps) - 2^10
```

```
ans =
     0
```

```
(2^10 + (2^9+1)*eps) - 2^10
```

```
ans =
       2.27373675443232e-13
```

```
2^(-42)
```

```
ans =
       2.27373675443232e-13
```

As a bonus, the gap between 2^{10} and the one before is $2^{9} \cdot \texttt{eps} = 2^{-43}$.

```
2^10 - (2^10 - 2^8*eps)
```

```
ans =
     0
```

```
2^10 - (2^10 - (2^8+1)*eps)
```

```
ans =
       1.13686837721616e-13
```

```
2^(-43)
```

```
ans =
       1.13686837721616e-13
```

Problem 2 (LM 9.3--10)

(a) Using the Taylor series $\log(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \cdots$ for x near 0, we can write and simply $f(x)$ for x near 0 (but not equal to zero) as

$$f(x) = \frac{\log(1+x)}{x} = \frac{x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \cdots}{x} = 1 - \frac{1}{2}x + \frac{1}{3}x^2 - \cdots.$$

Therefore, in the limit as x tends to 0, $f(x)$ tends to 1, that is, $\lim_{x\to 0} f(x) = 1$.

(b) Similar to the script provided in the hint:

```
k = [1:20]';
x = 10.^(-k);
fx = log(1+x)./x;
```

```
f1x = log(1+x)./((1+x)-1);
f2x = log1p(x)./x;
format long e
disp([x fx f1x f2x])
```

```
     1.000000000000000e-01     9.531017980432493e-01     9.531017980432485e-01     9.531017980432486e-01
     1.000000000000000e-02     9.950330853168092e-01     9.950330853168083e-01     9.950330853168083e-01
     1.000000000000000e-03     9.995003330834232e-01     9.995003330835333e-01     9.995003330835331e-01
     1.000000000000000e-04     9.999500033329731e-01     9.999500033330834e-01     9.999500033330834e-01
     9.999999999999999e-06     9.999950000398842e-01     9.999950000333330e-01     9.999950000333331e-01
     1.000000000000000e-06     9.999994999180668e-01     9.999995000003333e-01     9.999995000003334e-01
     1.000000000000000e-07     9.999999505838705e-01     9.999999500000033e-01     9.999999500000034e-01
     1.000000000000000e-08     9.999999889225291e-01     9.999999950000000e-01     9.999999950000000e-01
     1.000000000000000e-09     1.000000082240371e+00     9.999999995000000e-01     9.999999995000000e-01
     1.000000000000000e-10     1.000000082690371e+00     9.999999999500000e-01     9.999999999500000e-01
     1.000000000000000e-11     1.000000082735371e+00     9.999999999949999e-01     9.999999999949999e-01
     1.000000000000000e-12     1.000088900581841e+00     9.999999999995001e-01     9.999999999995000e-01
     1.000000000000000e-13     9.992007221625909e-01     9.999999999999499e-01     9.999999999999500e-01
     1.000000000000000e-14     9.992007221626359e-01     9.999999999999949e-01     9.999999999999950e-01
     1.000000000000000e-15     1.110223024625156e+00     9.999999999999994e-01     9.999999999999994e-01
     1.000000000000000e-16                         0                       NaN     1.000000000000000e+00
     9.999999999999999e-18                         0                       NaN     1.000000000000000e+00
     1.000000000000000e-18                         0                       NaN     1.000000000000000e+00
     1.000000000000000e-19                         0                       NaN     1.000000000000000e+00
     1.000000000000000e-20                         0                       NaN     1.000000000000000e+00
```

Explanation. The evaluation of $f(x)$ is severely affected by catastrophic cancellation for small x because of the what is written at the beginning of the problem. Though identical to $f(x)$ mathematically, the function $f_1(x)$ does a better job, which can be reasoned in a manner analogous to the one presented in the hint. To give you the gist of the argument: let $\hat{x} = \widehat{(1+x)-1} = fl((1+x)-1)$, the floating-point representation of the expression $(1+x)-1$. Note that the subtraction undergoes *catastrophic cancellation* for small x. Also note that when $\log(1+x)$ is evaluated in the computer, the input $(1+x)$ is formed first and then 1 is subtracted off from it before it is fed into an algorithm based on the Taylor series

$$\log \zeta = (\zeta - 1) - \frac{1}{2}(\zeta - 1)^2 + \frac{1}{3}(\zeta - 1)^3 - \cdots.$$ (To compute $\log(1+x)$, set $\zeta = 1 + x$.)

Therefore, the numerical evaluation of $f_1(x)$ can be approximated by

$$\widehat{f_1(x)} \approx \frac{\hat{x} - \frac{1}{2}\hat{x}^2 + \frac{1}{3}\hat{x}^3 - \cdots}{\hat{x}} = 1 - \frac{1}{2}\hat{x} + \frac{1}{3}\hat{x}^2 - \cdots,$$

which resembles the series expansion used in part (a). This is why the results are much more tamed with this encoding. However, when x gets sufficiently small, $(1+x)$ gets very close to 1 to a point that they are not distinguishable on the floating-point number system. In our experiment, that happened when $k \geq 16$:

```
x_small = 1e-16;
(1+x_small)-1
```

```
ans =
     0
```

So both the numerator and the denominator are zero, resulting in `NaN`, for $16 \le k \le 20$.

The function `log1p` was designed to avoid catastrophic cancellation occurring in calculating $\log(1+x)$ for small x. See

```
help log1p
```

```
 log1p  Compute LOG(1+X) accurately.
    log1p(X) computes LOG(1+X), without computing 1+X for small X.
    Complex results are produced if X < -1.

    For small real X, log1p(X) should be approximately X, whereas the
    computed value of LOG(1+X) can be zero or have high relative error.

    See also log, expm1.

    Documentation for log1p
```

Problem 3 (Inverting hyperbolic cosine)

```
t = -4:-4:-16;
x = cosh(t);
```

(a) Let $f(x) = \log(x - \sqrt{x^2-1}) = \mathrm{acosh}(x)$. Calculation shows that

$$\kappa_f(x) = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x}{\sqrt{x^2-1}} \cdot \frac{1}{\log(x-\sqrt{x^2-1})}\right|.$$

We evaluate the condition number at the entries of `x`, all at once, by

```
f  = log(x - sqrt(x.^2-1));
fp = -1./sqrt(x.^2-1);
kappa = abs( x.*fp./f )
```

```
kappa = 1×4
    2.501677876004177e-01    1.250000281311236e-01    8.333333238773505e-02 ···
```

Note that the condition number itself is not bad at all. In fact, as $x \to \infty$, $\kappa_f(x) \to 0$.

Exercise. Confirm using calculus that $\lim_{x\to\infty} \kappa_f(x) = 0$.

(b) We have already evaluted $t = f(x)$ in part (a), saved as `f`. We compare against the original values stored in `t`;

```
absErr = abs(f - t)';
relErr = absErr./abs(t);
for j = 1:length(x)
    if j == 1
        fprintf(' %10s %16s %16s\n', 'x', 'abs error', 'rel error')
```

```
        fprintf(' %45s\n', repmat('-', 1, 45))
    end
    fprintf(' %10.4e %16.8e %16.8e\n', x(j), absErr(j), relErr(j))
end
```

```
         x        abs error         rel error
 ---------------------------------------------
 2.7308e+01   4.61852778e-14   1.15463195e-14
 1.4905e+03   1.71089809e-10   4.27724522e-11
 8.1377e+04   1.37072186e-07   3.42680466e-08
 4.4431e+06   1.37512880e-03   3.43782200e-04
```

Unlike what the condition number $\kappa_f(x)$ predicts, the numerical evaluation loses accuracy as x become large. Why would this be? See below.

(c,d) Let $g(x) = -2\log\left(\sqrt{\frac{x+1}{2}} + \sqrt{\frac{x-1}{2}}\right)$. Analytically, $g(x) = f(x)$. Unlike $f(x)$, however, numerical evaluation of $g(x)$ is done much more stably:

```
g = -2*log(sqrt((x+1)/2) + sqrt((x-1)/2));
absErr = abs(g - t)';
relErr = absErr./abs(t);
for j = 1:length(x)
    if j == 1
        fprintf(' %10s %16s %16s\n', 'x', 'abs error', 'rel error')
        fprintf(' %45s\n', repmat('-', 1, 45))
    end
    fprintf(' %10.4e %16.8e %16.8e\n', x(j), absErr(j), relErr(j))
end
```

```
         x        abs error         rel error
 ---------------------------------------------
 2.7308e+01   0.00000000e+00   0.00000000e+00
 1.4905e+03   0.00000000e+00   0.00000000e+00
 8.1377e+04   0.00000000e+00   0.00000000e+00
 4.4431e+06   0.00000000e+00   0.00000000e+00
```

The key difference is that the expression for $g(x)$ does not involve any ill-conditioned steps whereas $f(x)$ requires a subtraction which is prone to catastrophic cancellation for large x as seen in part (b).