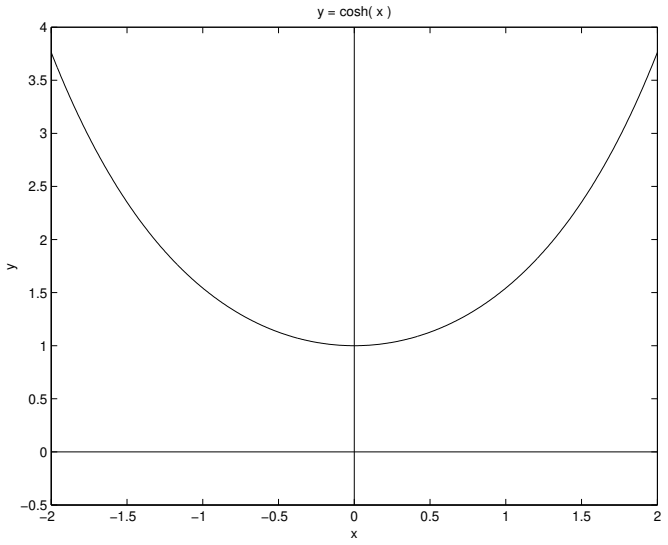
This week in lecture and section 1.3.9 of Heath, we discussed that subtraction of similar values causes cancellation (loss of precision). We also mentioned that floating point calculations may overflow (intermediate results become too large). Here we are going to look at an example of these problems and explore possible ways to fix them. Some of the parts don't give a lot of guidance, so please post on Piazza and collaborate with each other as you work through this worksheet. I'll be posting some extra materials about similar examples later in the week as needed.

Computing $x = cosh^{-1}(y)$

In first year calculus, you may have seen the hyperbolic cosine function

$$y = \cosh(x).$$

It's graph looks like:



$\cosh(x)$ can be expressed in terms of the exponential function and we can derive:

$$y = \cosh(x) = \frac{e^x + e^{-x}}{2}.$$

We will examine the inverse cosh function, $x = \cosh^{-1}(y)$, and derive an algorithm for calculating it. You will see that this process is not as simple as the mathematical expressions might suggest.

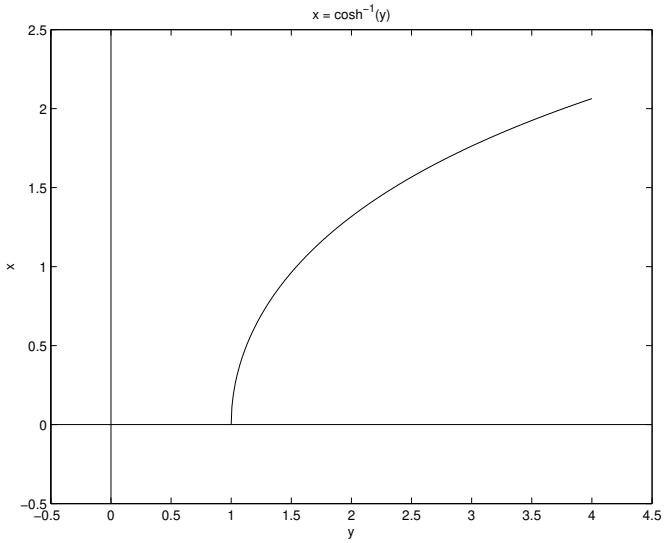
It can be shown that if you solve for x in terms of y, you will get:

$$x = \cosh^{-1}(y) = \pm \ln(y - \sqrt{y^2 - 1}).$$

The $\pm$ should not be surprising, since $\cosh(-x) = \cosh(x)$. If we choose the negative branch, then x will be ≥ 0, (since the argument to the ln function is between 0 and 1, which makes the ln value negative). We will therefore take

$$x = \cosh^{-1}(y) = -\ln(y - \sqrt{y^2 - 1})$$

as a formula for calculating the inverse function $\cosh^{-1}(y)$, with $y \geq 1$. It's graph looks like:



We are interested in how to accurately evaluate this formula

$$\cosh^{-1}(y) = -\ln(y - \sqrt{y^2 - 1}), \qquad y \geq 1.$$

(a) What sort of difficulties might we encounter when we implement this formula directly on a computer? (think about potential cancellation and overflow) Spend some time thinking about this before going to the next page.

Solution

Overflow may occur in y^2 - note that we end up taking the square root soon after, so this intermediate overflow could be avoided by rearranging the equation.

Both subtractions may lead to cancellation - when y is much larger than 1 or when y is close to 1. This is explored in more detail in the following questions.

(b) First we will consider what happens when y is a large value. What happens when y is much larger than 1?

Solution

When y is large, the squaring y^2 might overflow and cause difficulties in the calculation. But this is a "spurious" overflow because soon after squaring y, we take a square-root. Therefore we need to determine a way to avoid the overflow difficulty.

If y is large, but not so large that y^2 overflows, we have to calculate

$$y^2 - 1$$

but if, for example, $y = 1.200 \times 10^{10}$, and we are working with 4 decimal digit arithmetic, then

$$\begin{aligned} y^2 &= 1.440 \times 10^{20} \\ 1 &= 1.000 \times 10^{0} \end{aligned}$$

and the subtraction yields $1.4399\cdots99 \times 10^{20}$, which is represented in 4 digit floating-point as 1.440×10^{20}.

Therefore, for y large, $\text{fl}(y^2 - 1) = \text{fl}(y^2)$. (i.e., the "$-1$" drops-off the end.) (Recall $\text{fl}(\cdot)$ means "the floating-point representation of ...".)

This is still a relatively accurate result. The square-root that follows will give back y, which is also a relatively accurate approximation to $\sqrt{y^2 - 1}$, when y is large.

But then we compute

$$\begin{aligned} & -\ln(y - \sqrt{y^2 - 1}) \\ \approx\ & -\ln(y - y) \\ =\ & -\ln(0) \qquad \text{which is undefined!} \end{aligned}$$

There is a catastrophic cancellation in the evaluation of the expression $y - \sqrt{y^2 - 1}$ when y is large. This cancellation reveals the importance of the -1 term that gets dropped off the end. We will have to repair this flaw.

(c) Once you have identified the problem of large y, how can we fix it? The idea will be to use mathematical identities to remove the subtractions. Hint 1: $-\ln x = \ln \dfrac{1}{x}$. Hint 2: It is often useful to cleverly multiply by $1 = \dfrac{a+b}{a+b}$, for appropriate choice of a and b. Think of using the conjugate (a+b is the conjugate of a-b).

Solution

Note that $-\ln(y - \sqrt{y^2 - 1}) = \ln\left(\dfrac{1}{y - \sqrt{y^2 - 1}}\right)$ (a property of ln) and

$$\begin{aligned} \frac{1}{y - \sqrt{y^2 - 1}} &= \frac{1}{y - \sqrt{y^2 - 1}} \times \frac{y + \sqrt{y^2 - 1}}{y + \sqrt{y^2 - 1}} \\ &= \frac{y + \sqrt{y^2 - 1}}{y^2 - (y^2 - 1)} \\ &= y + \sqrt{y^2 - 1} \end{aligned}$$

Therefore, by using the equivalent formula

$$\cosh^{-1}(y) = \ln(y + \sqrt{y^2 - 1}), \qquad y \geq 1,$$

we can avoid the catastrophic cancellation difficulty that appeared in the first formula.

(d) If y is very large, y^2 might overflow. How can you manipulate the $\sqrt{y^2-1}$ term to avoid computing y^2? (Exercise 1.13 in Heath is similar.)

Solution

We can also write

$$\begin{aligned} & \sqrt{y^2-1} \\ = \; & |y|\sqrt{1-\frac{1}{y^2}} \\ = \; & y\sqrt{1-\frac{1}{y^2}} \qquad \text{we can drop the } |\cdot| \text{ since } y \geq 1. \\ = \; & y\sqrt{1-(\frac{1}{y})^2} \end{aligned}$$

If y is large, $(\frac{1}{y})^2$ could still underflow, but we can safely set the underflowed value to 0 without increasing the error in the calculation. This is because $(\frac{1}{y})^2$ is subtracted from 1, a much large number.

This suggests that we use the formulation (and algorithm)

$$\cosh^{-1}(y) = \ln\left(y + y\sqrt{1-(\frac{1}{y})^2}\right)$$

to evaluate the inverse cosh function.

During the Wednesday night session, some of you also pointed out that we could use $\sqrt{y-1}\sqrt{y+1}$ instead. This fixes the overflow and, as is discussed later, will reduce the cancellation problem when $y \approx 1$.

(e) What about when $y \approx 1$? Describe the difficulties that we might encounter.

Solution

When $y \approx 1$, we have other difficulties.

There could be a catastrophic cancellation in the calculation of $\sqrt{1-(\frac{1}{y})^2}$ or in the equivalent original form $\sqrt{y^2-1}$, because $(\frac{1}{y})^2 \approx 1$ and $y^2 \approx 1$.

Furthermore, when $y \approx 1$, the argument to the ln function is close to 1, since

$$\begin{aligned} y + y\sqrt{1-(\frac{1}{y})^2} \; & \approx \; y + y(0) \\ & \approx \; 1 \qquad \text{since } y \approx 1. \end{aligned}$$

(f) Consider the conditioning of the ln function when its argument is close to 1. (This will be the case when $y \approx 1$). Is ln ill-conditioned in this case?

Solution

Let z represent the quantity $y + y\sqrt{1-(\frac{1}{y})^2}$. As you have seen in class, the condition number for the function $\ln(z)$ is

$$\kappa(\ln(z)) = \frac{1}{\ln(z)}$$

and therefore the calculation is ill-conditioned for $z \approx 1$ (which happens when $y \approx 1$.)

Note, we can also see the ill-conditioning in the graph of the function near 1, where it is nearly a vertical line.

(g) How can we repair these cancellation and ill-conditioning difficulties from the last two parts? Hint 1: Exercise 1.12 in Heath is the same idea for fixing the cancellation. Hint 2: For the ill-conditioning, consider the conditioning of the function $\ln(1+w)$, with respect to input w and consider what happens for small w.

Solution

You know that catastrophic cancellation can only occur when the two terms in the difference are within $1/\beta$ of each other. So, assuming that we compute using a binary machine, we know that catastrophic cancellation can only occur when 1 and $(1/y)^2$ are within 1/2 of each other. That is, $1 \geq (1/y)^2 > 1/2$, or $1 \leq y < \sqrt{2}$. But since $\sqrt{2}$ is not exactly representable (it is irrational), let us say that catastrophic cancellation can only occur if $1 \leq y < 2$.

When we are in danger of seeing catastrophic cancellation, we can go back to the original formulation (because y^2 won't overflow when $1 \leq y < 2$) and write:

$$\begin{aligned} y\sqrt{1-(\frac{1}{y})^2} &= \sqrt{y^2-1} \\ &= \sqrt{(y-1)*(y+1)}. \end{aligned}$$

There will be no catastrophic cancellation in the $y-1$ part of the last expression, for the following reasons. We can represent 1 exactly in a floating-point system. And the input y to our function is a floating-point number. Since there are no errors for cancellation to reveal, there can be no catastrophic cancellation. (The expression y^2-1 could have a catastrophic cancellation since y^2 usually isn't exactly representable in the floating-point system.) (You might balk at the assumption that y is a floating-point number. But it is a floating-point number that the "user" has given us to work with. We can't really make any assumptions about how much error might be in the data that the user gives us. Making the "y is a floating-point number" assumption is the best that we can do.)

What about the ill-conditioning of the ln function? We know that $\ln(z)$ is ill-conditioned for $z \approx 1$. If we know that $z \approx 1$, let us define the quantity w to be such that $z = 1+w$. Then $\ln(z) = \ln(1+w)$.

Now consider the function $f(w) = \ln(1+w)$ and its condition number.

$$\begin{aligned} \kappa(f(w)) &= w \cdot \frac{f'(w)}{f(w)} \\ &= w \cdot \frac{\frac{1}{1+w}}{\ln(1+w)} \end{aligned}$$

The original function $\ln(z)$ is ill-conditioned when $z \to 1$, which corresponds to $w \to 0$. You can show (using l'Hôpital's rule) that

$$\lim_{w \to 0} \kappa(f(w)) = 1.$$

Hence the function $\ln(1+w)$ is well-conditioned when $w \to 0$. For this reason, many math libraries (including numpy) contain a `log1p` function which calculates $\text{log1p}(x) = \log(1+x)$. (Recall that many software libraries call ln `log` and $\log_{10}$ `log10`.) See `print(np.log1p.__doc__)` for details.

(h) Try implementing this function in python, so that it produces accurate results. (You'll hand in your code as a small part of the homework this week, but feel free to help each other.) Note, you can safely assume that np.arccosh(y) produces an accurate result when you test your function. Hint: You'll find the numpy function log1p(x) useful. Type `print(np.log1p.__doc__)` in python for its documentation (provided on the next page for reference).

Solution

Summary

Our identity becomes

$$\begin{aligned} x &= \cosh^{-1}(y), \qquad y \geq 1, \\ &= -\ln\left(y - \sqrt{y^2-1}\right), \qquad y \geq 1, \\ &= \ln\left(y + \sqrt{y^2-1}\right), \qquad y \geq 1, \\ &= \begin{cases} \ln\left(1 + (y-1) + \sqrt{(y-1)*(y+1)}\right), & \text{when } 1 \leq y < 2, \\ \ln\left(y + y * \sqrt{1 - (\frac{1}{y})^2}\right), & \text{when } y \geq 2. \end{cases} \end{aligned}$$

Altogether then, we could "code up" our function as

```
if ( y < 2 ) then
    x = np.log1p( (y-1) + np.sqrt( (y-1)*(y+1) ) )
else
    x = np.log( y + y*np.sqrt( 1 - (1/y)*(1/y) ) ) }
end if
```

and we will get an accurate result.

It turns out we don't actually need the two cases - just using the form in the first case works - if we do $\sqrt{y-1}\sqrt{y+1}$ instead. One of you also pointed out that you can pull out a factor of y instead of doing the $(y-1)$, so:

```
x = np.log(y) + np.log1p( np.sqrt(y-1) * np.sqrt(y+1) / y )
```

```
log1p(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, e

Return the natural logarithm of one plus the input array, element-wise.

Calculates ``log(1 + x)``.

Parameters
----------
x : array_like
    Input values.
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or `None`,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    Values of True indicate to calculate the ufunc at that position, values
    of False indicate to leave the value in the output alone.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
y : ndarray
    Natural logarithm of `1 + x`, element-wise.
    This is a scalar if `x` is a scalar.

See Also
--------
expm1 : ``exp(x) - 1``, the inverse of `log1p`.

Notes
-----
For real-valued input, `log1p` is accurate also for `x` so small
that `1 + x == 1` in floating-point accuracy.

Logarithm is a multivalued function: for each `x` there is an infinite
number of `z` such that `exp(z) = 1 + x`. The convention is to return
the `z` whose imaginary part lies in `[-pi, pi]`.

For real-valued input data types, `log1p` always returns real output.
For each value that cannot be expressed as a real number or infinity,
it yields ``nan`` and sets the `invalid` floating point error flag.

For complex-valued input, `log1p` is a complex analytical function that
has a branch cut `[-inf, -1]` and is continuous from above on it.
`log1p` handles the floating-point negative zero as an infinitesimal
negative number, conforming to the C99 standard.

References
----------
.. [1] M. Abramowitz and I.A. Stegun, "Handbook of Mathematical Functions",
       10th printing, 1964, pp. 67. http://www.math.sfu.ca/~cbm/aands/
.. [2] Wikipedia, "Logarithm". https://en.wikipedia.org/wiki/Logarithm
```

```
Examples
--------
>>> np.log1p(1e-99)
1e-99
>>> np.log(1 + 1e-99)
0.0
```