Lecture 4

Floating Point Arithmetic & Calc Review (Taylor Series)

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

August 29, 2024

Outline

The next topic will be numerical differentiation (next week):

- Floating Point Arithmetic
- Introduce Catastrophic Cancellation
- Review Taylor Series
- (Maybe) Try to approximate the derivative of $f'(x)$

Floating Point Errors

How big is this error? Suppose (x is closer to x_-)

$$\begin{aligned}
x &= (1.b_2b_3\dots b_{24}b_{25}b_{26})_2 \times 2^m \\
x_- &= (1.b_2b_3\dots b_{24})_2 \times 2^m \\
x_+ &= \left((1.b_2b_3\dots b_{24})_2 + 2^{-24}\right) \times 2^m \\
|x - x_-| &\leqslant \frac{|x_+ - x_-|}{2} = 2^{m-25} \quad \text{: absolute error} \\
\left|\frac{x - x_-}{x}\right| &\leqslant \frac{2^{m-25}}{1/2 \times 2^m} \leqslant 2^{-24} = \epsilon_m/2 \quad \text{: relative error}
\end{aligned}$$

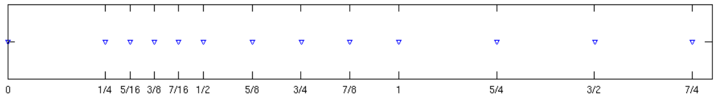
Notes

Floating Point Arithmetic

- Problem: The set of representable machine numbers is FINITE.
- So not all math operations are well defined!
- Basic algebra breaks down in floating point arithmetic

Example

$$a + (b + c) \neq (a + b) + c$$



Floating Point Arithmetic

Rule 1.

$$fl(x) = x(1+\epsilon), \quad \text{where} \quad |\epsilon| < \epsilon_m$$

Rule 2.

For all operations $\odot$ (one of $+, -, *, /$)

$$fl(x \odot y) = (x \odot y)(1+\epsilon_\odot), \quad \text{where} \quad |\epsilon_\odot| < \epsilon_m$$

Rule 3.

For $+, *$ operations

$$fl(a \odot b) = fl(b \odot a)$$

There were many discussions on what conditions/ rules should be satisfied by floating point arithmetic. The IEEE standard is a set of standards adopted by many CPU manufacturers.

Floating Point Arithmetic

Consider the sum of 3 numbers: $y = a + b + c$.

Done as $fl(fl(a+b)+c)$

$$\begin{aligned}
\eta &= fl(a+b) = (a+b)(1+\epsilon_1) \\
y_1 &= fl(\eta + c) = (\eta + c)(1+\epsilon_2) \\
&= \left[(a+b)(1+\epsilon_1) + c\right](1+\epsilon_2) \\
&= \left[(a+b+c) + (a+b)\epsilon_1)\right](1+\epsilon_2) \\
&= (a+b+c)\left[1 + \frac{a+b}{a+b+c}\epsilon_1(1+\epsilon_2) + \epsilon_2\right]
\end{aligned}$$

So disregarding the high order term $\epsilon_1\epsilon_2$

$$fl(fl(a+b)+c) = (a+b+c)(1+\epsilon_3) \quad \text{with} \quad \epsilon_3 \approx \frac{a+b}{a+b+c}\epsilon_1 + \epsilon_2$$

Floating Point Arithmetic

If we redid the computation as $y_2 = fl(a + fl(b + c))$ we would find

$$fl(a + fl(b + c)) = (a + b + c)(1 + \epsilon_4) \quad \text{with} \quad \epsilon_4 \approx \frac{b + c}{a + b + c}\epsilon_1 + \epsilon_2$$

Main conclusion:

The first error is $\underline{\text{amplified}}$ by the factor $(a + b)/y$ in the first case and $(b + c)/y$ in the $\overline{\text{second}}$ case.

In order to sum n numbers more accurately, it is better to start with the small numbers first. [However, sorting before adding is usually not worth the cost!]

Review

We can't store real numbers x perfectly, we can only store a finite-precision floating point version $fl(x)$.

- We have bounds on *relative* error:
- $fl(x) = x(1+\epsilon)$ where $\epsilon < \epsilon_m$.
- Equivalently

$$\left|\frac{fl(x) - x}{x}\right| < \epsilon_m.$$

Review

Machine epsilon is the “resolution” of our number system (in relative, not absolute terms)

- ϵ_m depends only on the length of the mantissa (in bits).
- The number of bits for the exponent falls out of the calculation.
- For single precision, $\epsilon_m = 2^{-23} \approx 1.19 \times 10^{-7}$
- For double precision, $\epsilon_m = 2^{-52} \approx 2.22 \times 10^{-16}$.
- Each floating point calculation can introduce errors. But their size is controlled by ϵ_m.

Significant Digits

Significant Digits

Digits in a number justified by the accuracy of the means of measurement

Example: Measurement from a ruler which goes from 0.0 to 9.0 (in cm):

- $x = .23 \times 10^1 cm$
- two significant digits
- 2 is the most significant digit
- 3 is the least significant digit

- $x = .40 \times 10^1 cm$
- still two significant digits

- $x = .400 \times 10^1 cm$
- The last zero is meaningless. From now on, we will never write it

To Notes

Lets solve a polynomial equation

$$x^2 + 2px - q = 0$$

for user input p and q.

Floating Point Arithmetic

Roundoff errors and floating-point arithmetic

One of the most serious problems in floating point arithmetic is that of cancellation. If two large and close-by numbers are subtracted the result (a small number) carries very few accurate digits (why?). This is fine if the result is not reused. If the result is part of another calculation, then there may be a serious problem

Example

Roots of the equation

$$x^2 + 2px - q = 0$$

Assume we want the root with smallest absolute value (and $p > 0$):

$$y = -p + \sqrt{p^2 + q} = \frac{q}{p + \sqrt{p^2 + q}} \quad \left(\text{mult. by } \frac{p + \sqrt{p^2 + q}}{p + \sqrt{p^2 + q}}\right)$$

Loss of Significance

Listing 1: alg 1

```
$s$ := $p^2$
$t$ := $s + q$
$u$ := $\sqrt{t}$
$y1$ := $-p + u$
```

Listing 2: alg 2

```
$s$ := $p^2$
$t$ := $s + q$
$u$ := $\sqrt{t}$
$v$ := $p + u$
$y2$ := $q/v$
```

Step 4 of Algorithm 1 can have severe cancellation when $p >> q$ and when p is positive (and other times).

```
p = 10000;
q = 1;
y1 = -p+sqrt(p^2 + q)
y1 = 5.000000055588316e-05

y2 = q/(p+sqrt(p^2+q))
y2 = 4.999999987500000e-05

x = y1;
x^2 + 2 * p * x -q
ans = 1.361766321927860e-08

x = y2;
x^2 + 2 * p * x -q
ans = -1.110223024625157e-16
```

Notes

Now what if p and q are really close together?
$p = -1 - \epsilon/2$, and $q = -1 - \epsilon$ (for small ϵ).

Even Worse

Consider now the case when

$$p = -(1 + \epsilon/2) \qquad \text{and} \qquad q = -(1 + \epsilon)$$

Exact Roots? Take $\epsilon = 1.E - 08$ and use Matlab:

```
   x = - p - sqrt(p^2 + q)     --->       1.000000000500000
   y = - p + sqrt(p^2 + q)     --->       1.000000000500000
```

Catastrophic Cancellation

Adding $c = a + b$ will result in a large error if

- $a \gg b$
- $a \ll b$

Let

$$a = x.xxx \cdots \times 10^0$$
$$b = y.yyy \cdots \times 10^{-8}$$

Then

$$\begin{array}{rll} & \overbrace{x.xxx\ xxxx\ xxxx\ xxxx}^{\text{finite precision}} & \\ + & 0.000\,0000\ yyyy\ yyyy & yyyy\ yyyy \\ \hline = & x.xxx\ xxxx\ zzzz\ zzzz & \underbrace{????\ ????}_{\text{lost precision}} \end{array}$$

Catastrophic Cancellation

Subtracting $c = a - b$ will result in large error if $a \approx b$. For example

$$a = x.xxxx\,xxxx\,xxx1\,\overbrace{ssss\ldots}^{\text{lost}}$$

$$b = x.xxxx\,xxxx\,xxx0\,\overbrace{tttt\ldots}^{\text{lost}}$$

Then
$$\begin{array}{cl} & \overbrace{x.xxx\,xxxx\,xxx1}^{\text{finite precision}} \\ - & x.xxx\,xxxx\,xxx0 \\ \hline = & 0.000\,0000\,0001 \quad \underbrace{????\,????}_{\text{lost precision}} \end{array}$$

Summary

- addition (suffers cancellation): $c = a + b$ if $a \gg b$ or $a \ll b$
- subtraction (suffers cancellation): $c = a - b$ if $a \approx b$
- catastrophic: caused by a single operation, not by an accumulation of errors
- can often be fixed by mathematical rearrangement

Notes

Loss of Significance

Example

$x = 0.37214\,48693$ and $y = 0.37202\,14371$. What is the relative error in $x - y$ in a computer with 5 decimal digits of accuracy?

$$\frac{|x - y - (\bar{x} - \bar{y})|}{|x - y|} = \frac{|0.37214\,48693 - 0.37202\,14371 - 0.37214 + 0.37202|}{|0.37214\,48693 - 0.37202\,14371|}$$
$$\approx 3 \times 10^{-2}$$

Error is greater than 1%. I can't even trust the second significant digit!

Loss of Significance

Loss of Precision Theorem

Let x and y be (normalized) floating point machine numbers with $x > y > 0$.

If $2^{-p} \leqslant 1 - \frac{y}{x} \leqslant 2^{-q}$ for positive integers p and q, the significant binary digits lost in calculating $x - y$ is between q and p.

Loss of Significance

This works in base 10 as well

Let x and y be (normalized) floating point machine numbers with $x > y > 0$.

If $10^{-p} \leqslant 1 - \frac{y}{x} \leqslant 10^{-q}$ for positive integers p and q, the significant digits lost in calculating $x - y$ is between q and p.

Loss of Significance

Example

Consider $x = 37.593621$ and $y = 37.584216$.

$$2^{-11} < 1 - \frac{y}{x} = 0.00025\,01754 < 2^{-12}$$

So we lose 11 or 12 bits in the computation of $x - y$. yikes!

Example

Back to the other example (5 decimal digits): $x = 0.37214$ and $y = 0.37202$.

$$10^{-4} < 1 - \frac{y}{x} = 0.00032 < 10^{-3}$$

So we lose 3 or 4 (decimal) digits in the computation of $x - y$. Here,

$x - y = 0.00012$ which has only 1 significant digit that we can be sure about

Loss of Significance

So what to do? Mainly rearrangement.

$$f(x) = \sqrt{x^2 + 1} - 1$$

Loss of Significance

So what to do? Mainly rearrangement.

$$f(x) = \sqrt{x^2 + 1} - 1$$

Problem at $x \approx 0$.

Loss of Significance

So what to do? Mainly rearrangement.

$$f(x) = \sqrt{x^2+1} - 1$$

Problem at $x \approx 0$.

One type of fix:

$$\begin{aligned} f(x) &= \left(\sqrt{x^2+1} - 1\right)\left(\frac{\sqrt{x^2+1}+1}{\sqrt{x^2+1}+1}\right) \\ &= \frac{x^2}{\sqrt{x^2+1}+1} \end{aligned}$$

no subtraction!

Calculus Review

Changing Gears!

Taylor

- All we can ever do is add and multiply
- We can't directly evaluate e^x, $cos(x)$, $\sqrt{x}$
- What to do? Taylor Series *approximation*

Taylor

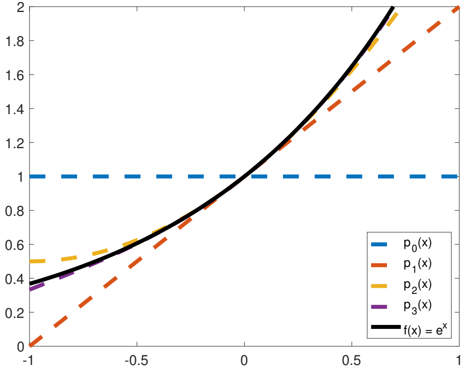
The Taylor series expansion of $f(x)$ at the point $x = c$ is given by

$$\begin{aligned} T(x) &= f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(c) + \ldots \\ &= \sum_{k=0}^{\infty} \frac{(x-c)^k}{k!} f^{(k)}(c) \end{aligned}$$

Note: Assumes perfect knowledge of the function $f(x)$ at the point c.

Taylor Idea

Better & better approximation



Taylor Example

Example (e^x)

For the function $f(x) = e^x$, we know $f^{(k)}(x) = e^x$ for all k.
If we let $c = 0$, then $e^0 = 1$ and

$$T(x) = f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(c) + \dots$$

becomes

$$\begin{aligned} T(x) &= 1 + (x-0)\cdot 1 + \frac{(x-0)^2}{2}\cdot 1 + \dots \\ &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \end{aligned}$$

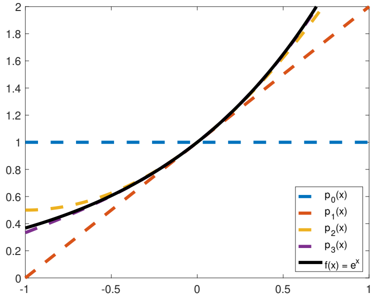
Taylor is Great!

Question: Is $T(x)$ the same as $f(x)$?

Yes!

For "nice" functions $f(x)$

$$f(x) = T(x) = f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(c) + \ldots$$



Taylor Approximation

- So

$$e^2 = 1 + 2 + \frac{2^2}{2!} + \frac{2^3}{3!} + \dots$$

- But we can't evaluate an infinite series, so we truncate...

Taylor Series Polynomial Approximation

The Taylor Polynomial of degree n for the function $f(x)$ about the point c is

$$p_n(x) = \sum_{k=0}^{n} \frac{(x-c)^k}{k!} f^{(k)}(c)$$

Example (e^x)

In the case of the exponential

$$e^x \approx p_n(x) = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

Taylor Approximation

“Evaluate” e^2:

- Using 0^{th} order Taylor series: $e^x \approx 1$ does not give a good fit.
- Using 1^{st} order Taylor series: $e^x \approx 1 + x$ gives a better fit.
- Using 2^{nd} order Taylor series: $e^x \approx 1 + x + x^2/2$ gives a a really good fit.

```
x=2;
pn=0;
for j=0:15
  pn = pn + (x^j)/factorial(j);
  err = exp(2)-pn
  pause
end
```

Taylor Approximation Recap

Infinite Taylor Series Expansion (exact)

$$f(x) = f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(c) + \dots$$

Finite Taylor Series Approximation

$$f(x) \approx f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(x),$$

Finite Taylor Series Expansion (exact)

$$f(x) = f(c) + (x-c)f'(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(x) + \frac{(x-c)^{n+1}}{(n+1)!}f^{(n+1)}(\xi),$$

but we don't know ξ (it has to be somewhere between x and c) and it could be different for every x.

Taylor Approximation Error

- How accurate is the Taylor series polynomial approximation?
- The $n+1$ terms of the approximation are simply the first $n+1$ terms of the *exact* expansion:

$$e^x = \underbrace{1 + x + \frac{x^2}{2!}}_{p_2 \text{ approximation to } e^x} + \underbrace{\frac{x^3}{3!} + \dots}_{\text{truncation error}} \tag{1}$$

- So the function $f(x)$ can be written as the Taylor Series approximation plus an error (truncation) term:

$$f(x) = p_n(x) + E_n(x)$$

where

$$E_n(x) = \sum_{k=n+1}^{\infty} \frac{(x-c)^k}{k!} f^{(k)}(c) \quad \text{or} \quad \frac{(x-c)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$$

Note: Taylor's theorem guarantees some ξ exists, but doesn't tell us what it is.

Truncation Error

Example ($sin(x)$)

The Taylor series expansion of $\sin(x)$ (at the point $c = 0$) is

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

If $x \ll 1$, then the remaining terms are small. If we neglect these terms

$$\sin(x) = \underbrace{x - \frac{x^3}{3!} + \frac{x^5}{5!}}_{\text{approximation to sin}} \underbrace{- \frac{x^7}{7!} + \frac{x^9}{9!} - \dots}_{\text{truncation error}}$$

Taylor Series: Computations

- How do we evaluate $f(x) = \frac{1}{1-x}$ computationally?
- Taylor Series Expansion:

$$f(x) = f(c) + (x-c)f'(c) + \frac{(x-c)^2}{2!}f''(c) + \cdots + \frac{(x-c)^n}{n!}f^{(n)}(\xi),$$

- Thus with c = 0

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \dots$$

- Second order approximation:

$$\frac{1}{1-x} \approx 1 + x + x^2$$

Taylor Errors

- How many terms do I need to make sure my error is less than 2×10^{-8} for $x = 1/2$?

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots + x^n + \sum_{k=n+1}^{\infty} x^k$$

- so the error at $x = 1/2$ is

$$e_{x=1/2} = \sum_{k=n+1}^{\infty} \left(\frac{1}{2}\right)^k = \frac{(1/2)^{n+1}}{1-1/2}$$
$$= 2 \cdot (1/2)^{n+1} < 2 \times 10^{-8}$$

- then we need

$$n + 1 > \frac{-8}{\log_{10}(1/2)} \approx 26.6 \quad \text{or}$$
$$n > 26$$