Lecture 2

Taylor Series, Rate of Convergence, Condition Number, Stability

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

January 25, 2011

What we'll do:

- Section 1: Refresher on Taylor Series
- Section 2: Measuring Error and Counting the Cost of the Method
  - big-$\mathcal{O}$(continuous function)
  - big-$\mathcal{O}$ (discrete function)
  - Order of convergence
- Section 3: Taylor Series in Higher Dimensions
- Section 4: Condition Number of a Mathematical Model of a Problem

Section 1: Taylor Series

- 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

$$
\begin{aligned}
f(x) &= f(c) + f^{(1)}(c)(x-c) + \frac{f^{(2)}(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n + \ldots \\
&= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k
\end{aligned}
$$

Taylor Example

Taylor Series

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

$$f(x) = f(c) + f^{(1)}(c)(x-c) + \frac{f^{(2)}(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n + \ldots$$

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

Example (e^x)

We know $e^0 = 1$, so expand about $c = 0$ to get

$$f(x) = e^x = 1 + 1 \cdot (x-0) + \frac{1}{2!} \cdot 1 \cdot (x-0)^2 + \ldots$$

$$= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

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

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

Example (e^x)

In the case of the exponential

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

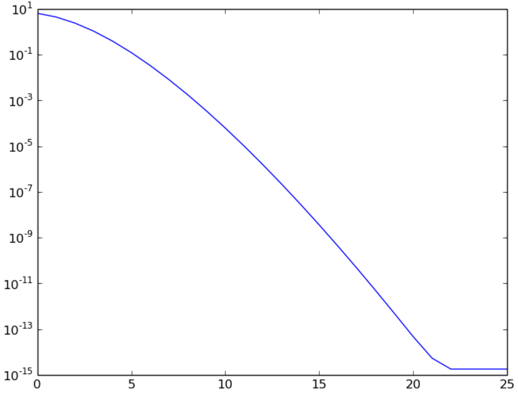
Taylor Approximation

Evaluate e^2:

```
import math
import matplotlib.pyplot as plt
import numpy
x=2.0
pn =0.0
error=[]
for j in range(0,26):
    pn = pn + (x**j)/math.factorial(j)
    error.append(math.exp(2.0)-pn)

j = numpy.arange(0,26)
plt.semilogy(j,error)
```

Taylor Approximation

Evaluate e^2:



Taylor Approximation Recap

Infinite Taylor Series Expansion (exact)

$$f(x) = f(c) + f^{(1)}(c)(x-c) + \frac{f^{(2)}(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n + \ldots$$

Finite Taylor Series Expansion (exact)

$$\begin{aligned} f(x) &= f(c) + f^{(1)}(c)(x-c) + \frac{f^{(2)}(c)}{2!}(x-c)^2 + \ldots \\ &+ \frac{f^{(n)}(c)}{n!}(x-c)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{(n+1)} \end{aligned}$$

where ξ lies between x and c but we don't know exactly where.

Finite Taylor Series Approximation

$$f(x) \approx f(c) + (x-c)f^{(1)}(c) + \frac{f^{(2)}(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n$$

Taylor Approximation Error

- How accurate is the Taylor series polynomial approximation?
- The n terms of the approximation are simply the first n 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

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

Section 2: Measuring Error and Counting the Cost of the Method

- Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to x near c (for fixed f and n).
- Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to n (for a fixed f and x).
- Goal: Determine how the cost of computing $p_n(x)$ behaves relative to n (for a fixed f and x).

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to x near c (for fixed f and n)

Big "$\mathcal{O}$" (continuous functions)

We write the error as

$$e_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}$$
$$= \mathcal{O}\left((x-c)^{n+1}\right)$$

since we assume the $(n+1)^{th}$ derivative is bounded on the interval $[a, b]$.

Often, we let $h = x - c$ and we have

$$f(x) = p_n(x) + \mathcal{O}(h^{n+1})$$

Big "$\mathcal{O}$" (continuous functions)

We write that $g(h) \in O(h^r)$ when

$$|g(h)| \leqslant C|h^r| \text{ for some } C \text{ as } h \to 0$$

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to x near c (for fixed f and n)

For the Taylor series of $f(x) = \frac{1}{1-x}$ about $c = 0$ we note that since the function can be written as a geometric series,

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

we can (in this specific problem) obtain an explicit formula for the error function,

$$\begin{aligned} |e_n(x)| &= \sum_{k=n+1}^{\infty} x^k = \sum_{\tilde{k}=0}^{\infty} x^{\tilde{k}+n+1} = x^{n+1} \sum_{\tilde{k}=0}^{\infty} x^{\tilde{k}} \\ &= \frac{x^{n+1}}{1-x} \text{ for a fixed } x \in (-1, 1) \\ &= O(h^{n+1}) \text{ where } h = x - c = x - 0 = x \end{aligned}$$

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to n (for a fixed f and x)

Taylor Series for $f(x) = \frac{1}{1-x}$

From the previous slide we computed the error exactly as,

$$\frac{x^{n+1}}{1-x} \text{ for a fixed } x \in (-1, 1)$$

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

$$|e_n(x)| = 2 \cdot (1/2)^{n+1} < 2 \times 10^{-8}$$

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

$$n > 26$$

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to n (for a fixed f and x)

If we use another method for computing $f(x)$ how can we compare the methods order of convergence for a fixed value of x?

Order of Convergence

Definition

If

$$\lim_{n\to\infty} a_n = L$$

then the Order of Convergence of the sequence $\{a_n\}$ is the largest positive number r such that

$$\lim_{n\to\infty} \frac{|a_{n+1} - L|}{|a_n - L|^r} = C < \infty$$

- For $r = 1$ and $C = 1$ the convergence is said to be sub-linear.
- For $r = 1$ and $0 < C < 1$ the convergence is said to be linear.
- For $r = 1$ and $C = 0$ the convergence is said to be super
- For $r > 1$ the convergence is said to be superlinear.
- For $r = 2$ the convergence is said to be quadratic.

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to n (for a fixed f and x)

Taylor Series for $f(x) = \frac{1}{1-x}$

From the previous slide we computed the error exactly as,

$$\frac{x^{n+1}}{1-x} \text{ for a fixed } x \in (-1, 1)$$

Order of convergence

We know that $\lim_{n\to\infty} p_n(x) = L = \frac{1}{1-x}$ for a fixed $x \in (-1, 1)$. To find the order of convergence we compute,

$$\begin{aligned} \frac{|p_{n+1} - L|}{|p_n - L|^r} &= \frac{|e_{n+1}(x)|}{|e_n(x)|^r} \\ &= \frac{|\frac{x^{n+2}}{1-x}|}{|\frac{x^{n+1}}{1-x}|^r} = |(1-x)^{(r-1)} x^{((n+1)(1-r)+1)}| \end{aligned}$$

Goal: Determine how the error $e_n(x) = |f(x) - p_n(x)|$ behaves relative to n (for a fixed f and x)

Order of convergence of the Taylor Series for $f(x) = \frac{1}{1-x}$

Using the result from the previous slide, we need to find the largest value of r such that the following limit is finite.

$$\lim_{n\to+\infty} |(1-x)^{(r-1)} x^{((n+1)(1-r)+1)}|$$

Since $x \in (-1, 1)$ if $r > 1$ then $|x^{((n+1)(1-r)+1)}| \to +\infty$ as $n \to +\infty$.
When $r = 1$ we have the result that,

$$\lim_{n\to+\infty} |(1-x)^{(r-1)} x^{((n+1)(1-r)+1)}| = \lim_{n\to+\infty} |x| = |x|$$

Therefore, for $x \in (-1, 1) and X \neq 0$ the order of convergence is 1 and the convergence is linear.

Goal: Determine how the cost of computing $p_n(x)$ behaves relative to n (for a fixed f and x)

- For example, how do we evaluate

$$f(x) = 5x^3 + 3x^2 + 10x + 8$$

  at the point $1/3$?
- This would require 5 multiplications and 3 additions.
- If we regroup as

$$f(x) = 8 + x(10 + x(3 + x(5)))$$

  then we have 3 multiplications and 3 additions.
- This is Nested Multiplication or Synthetic Division or Horner's Method

Nested Multiplication

- To evaluate

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

rewrite as

$$p_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x(a_n)) \ldots))$$

- A polynomial of degree n requires no more than n multiplications and n additions. That is, the number of floating point operations is $\mathcal{O}(n)$.

Listing 1: nested mult

```
p = a[n]
for i in range(n-1,-1,-1):
    p = a[i] + x * p
```

Big "$\mathcal{O}$" (discrete functions)

How to measure the impact of n on algorithmic cost?

$\mathcal{O}(\cdot)$

Let $g(n)$ be a function of n. Then define

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c, n_0 > 0 \,:\, 0 \leqslant f(n) \leqslant cg(n),\ \forall n \geqslant n_0\}$$

That is, $f(n) \in \mathcal{O}(g(n))$ if there is a constant c such that $0 \leqslant f(n) \leqslant cg(n)$ is satisfied.

- assume non-negative functions (otherwise add $|\cdot|$) to the definitions
- $f(n) \in \mathcal{O}(g(n))$ represents an asymptotic upper bound on $f(n)$ up to a constant
- example: $f(n) = 3\sqrt{n} + 2\log n + 8n + 85n^2 \in \mathcal{O}(n^2)$

Big-O (Omicron)

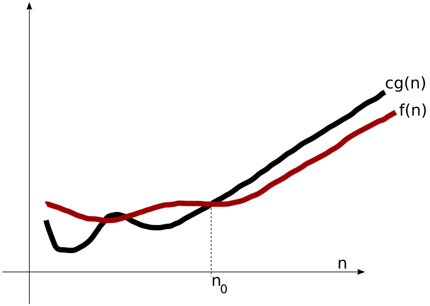
asymptotic upper bound

$\mathcal{O}(\cdot)$

Let $g(n)$ be a function of n. Then define

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c, n_0 > 0 \,:\, 0 \leqslant f(n) \leqslant cg(n),\ \forall n \geqslant n_0\}$$

That is, $f(n) \in \mathcal{O}(g(n))$ if there is a constant c such that $0 \leqslant f(n) \leqslant cg(n)$ is satisfied.



Definition Multi-Index Notation

Denote $\mathbf{k} = (k_1, k_2, \cdots, k_n)$ and $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ then we will use the following notation,

- $|\mathbf{k}| = k_1 + k_2 + \cdots + k_n$
- $\mathbf{k}! = k_1!k_2! \cdots k_n!$
- $\mathbf{x}^{\mathbf{k}} = x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$
- $\dfrac{\partial^{\mathbf{k}}}{\partial \mathbf{x}^{\mathbf{k}}} = \dfrac{\partial^{k_1}}{\partial x_1^{k_1}} \dfrac{\partial^{k_2}}{\partial x_2^{k_2}} \cdots \dfrac{\partial^{k_n}}{\partial x_n^{k_n}}$

Further Classification of Functions

Definition

Given a function,

$$f = \mathbb{R}^n \to \mathbb{R}$$

then f is called $C^m(\mathbb{R}^n)$ if $\dfrac{\partial^k f(x_1, x_2, \cdots, x_n)}{\partial x_1^{k_1} \partial x_2^{k_2} \cdots \partial x_n^{k_n}}$ (where $k_1 + k_2 + \cdots + k_n = k$) is a continuous function for all values $m \geqslant k \geqslant 0$. For $m = 0$ we write $C(\mathbb{R}^n)$ which denotes the set of all continuous functions.
If f is $C^m(\mathbb{R}^n)$ for all $m \geqslant 0$ then f is called $C^\infty(\mathbb{R}^n)$.

Example

- $\dfrac{\partial^2(x^2 y)}{\partial x \partial y} = \dfrac{\partial^2(x^2 y)}{\partial y \partial x} = 2x$.

Taylor Series in Higher Dimensions

Taylor Series (using multi-index notation)

If $f : \mathbb{R}^n \to \mathbb{R}$, f is $C^{m+1}(\mathbb{R}^n)$ and $\mathbf{x}, \mathbf{x} \in \mathbb{R}^n$ then we can approximate the function f by the formula:

$$f(\mathbf{x}) = \sum_{|\mathbf{k}|=0}^{m} \frac{1}{\mathbf{k}!} \frac{\partial^{\mathbf{k}} f(\mathbf{c})}{\partial \mathbf{x}^{\mathbf{k}}} (\mathbf{x} - \mathbf{c})^{\mathbf{k}} + R_{m+1}(\mathbf{x}, \mathbf{c})$$

where $R_{m+1}(\mathbf{x}, \mathbf{c})$ is the remainder.

Taylor Series Example

$f(x, y) = x^2 + y^2 - \cos(x)$

$f : \mathbb{R}^2 \to \mathbb{R}$ and we will put $\mathbf{c} = (0, 0)$ and $\mathbf{x} = (x, y)$. Note that $f \in C^\infty(\mathbb{R}^2)$.
Find the Taylor Series terms for $|k| = 0, 1, 2$.
The partial derivatives of f are:

- $\dfrac{\partial f}{\partial x} = 2x + \sin(x)$
- $\dfrac{\partial f}{\partial y} = 2y$
- $\dfrac{\partial^2 f}{\partial x^2} = 2 + \cos(x)$
- $\dfrac{\partial^2 f}{\partial x \partial y} = 0$
- $\dfrac{\partial^2 f}{\partial y^2} = 2$

Taylor Series Example (continued)

$f(x,y) = x^2 + y^2 - \cos(x)$

For $|k| = 0$ there is only one term in the series:

$$\frac{1}{0!0!}\frac{\partial^0}{\partial x^0}\left(\frac{\partial^0 f(\mathbf{c})}{\partial y^0}\right)(x-0)^0(y-0)^0 = f(\mathbf{c}) = -1$$

For $|k| = 1$ there are two terms in the series:

$$\frac{1}{1!0!}\frac{\partial^1 f(\mathbf{c})}{\partial x^1}(x-0)^1(y-0)^0 + \frac{1}{0!1!}\frac{\partial^1 f(\mathbf{c})}{\partial y^1}(x-0)^0(y-0)^1 = 0$$

For $|k| = 2$ there are three terms in the series:

$$\frac{1}{2!0!}\frac{\partial^2 f(\mathbf{c})}{\partial x^2}(x-0)^2(y-0)^0 + \frac{1}{1!1!}\frac{\partial^1}{\partial x^1}\left(\frac{\partial^1 f(\mathbf{c})}{\partial y^1}\right)(x-0)^1(y-0)^1 +$$

$$\frac{1}{0!2!}\frac{\partial^2 f(\mathbf{c})}{\partial y^2}(x-0)^0(y-0)^2 = \frac{3}{2}x^2 + y^2$$

Taylor Series Example (continued)

Thus we have the truncated approximation,

$f(x, y) = x^2 + y^2 - \cos(x)$

$$f(x, y) = x^2 + y^2 - \cos(x) \approx -1 + \frac{3}{2}x^2 + y^2$$

Taylor Series Example

The general formula for $f : \mathbb{R}^2 \rightarrow \mathbb{R}$

For $|k| = 0, 1, 2$ where $c = (x_0, y_0)$:

$$\begin{aligned} f(x,y) \quad \approx \quad & f(\mathbf{c}) + \frac{\partial f(\mathbf{c})}{\partial x}(x - x_0) + \frac{\partial f(\mathbf{c})}{\partial y}(y - y_0) + \\ & \frac{1}{2!}\frac{\partial^2 f(\mathbf{c})}{\partial x^2}(x - x_0)^2 + \left(\frac{\partial^2 f(\mathbf{c})}{\partial x \partial y}\right)(x - x_0)(y - y_0) + \frac{1}{2!}\frac{\partial^2 f(\mathbf{c})}{\partial y^2}(y - y_0)^2 \end{aligned}$$

Taylor Series Example

The vector form of the general formula

For $|k| = 0, 1, 2$ where $c = (x_0, y_0)$:

$$f \approx f(\mathbf{c}) + [\nabla f(\mathbf{c})]^T * (\mathbf{x} - \mathbf{c}) + \frac{1}{2!}(\mathbf{x} - \mathbf{c})^{\mathbf{T}} * H(f(\mathbf{c})) * (\mathbf{x} - \mathbf{c})$$

where $\mathbf{x}, \mathbf{c}, \mathbf{x} - \mathbf{c}$ are column vectors, the T represents the *tranpose* operator, the column vector $\nabla f(\mathbf{c})$ represents the *gradient* of $f(\mathbf{x})$ and finally the *Hessian* matrix,

$$H(f(\mathbf{c})) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{c})}{\partial x^2} & \frac{\partial^2 f(\mathbf{c})}{\partial x \partial y} \\ \frac{\partial^2 f(\mathbf{c})}{\partial y \partial x} & \frac{\partial^2 f(\mathbf{c})}{\partial y^2} \end{bmatrix}$$

Properties of the above formula

- True for $f : \mathbb{R}^n \to \mathbb{R}$.
- For $f : \mathbb{R}^n \to \mathbb{R}$ the Hessian has size $nxn, H = [H_{ij}]$ where $H_{ij} = \dfrac{\partial^2 f(\mathbf{c})}{\partial x_i \partial x_j}$.

Test your understanding

$f(x,y) = x^2 + y^2 + \cos(x)$

Does $f(x,y)$ have a maxima or minima?
Set the "derivative" equal to zero to find critical points.

$$\mathbf{0} = \nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x - \sin(x) \\ 2y \end{bmatrix}$$

has the solution $x = 0, y = 0$ thus $\mathbf{c} = [0,0]^T$ is a critical point.

Check the sign of the "second derivative".

"Second derivative" test

Given that $\mathbf{c}$ is a critical point of f, then

- If $\mathbf{x}^T * H * \mathbf{x} < 0, \mathbf{x} \neq \mathbf{0}$ H is negative-definite(Maxima)
- If $\mathbf{x}^T * H * \mathbf{x} > 0, \mathbf{x} \neq \mathbf{0}$ H is positive-definite(Minima)

Test your understanding (continued)

"Second derivative" test (continued)

For $\mathbf{x} = [x, y]^T$,

$$\begin{aligned} \mathbf{x}^T * H * \mathbf{x} &= \mathbf{x}^{\mathbf{T}} * \begin{bmatrix} \frac{\partial^2 f(\mathbf{c})}{\partial x^2} & \frac{\partial^2 f(\mathbf{c})}{\partial x \partial y} \\ \frac{\partial^2 f(\mathbf{c})}{\partial y \partial x} & \frac{\partial^2 f(\mathbf{c})}{\partial y^2} \end{bmatrix} * \mathbf{x} && (2) \\ &= \mathbf{x}^{\mathbf{T}} * \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} * \mathbf{x} && (3) \\ &= x^2 + 2y^2 > 0 \text{ for } \mathbf{x} \neq \mathbf{0} && (4) \end{aligned}$$

Section 4: Condition Number of Mathematical Model of a Problem

Given a function $G : \mathbb{R} \rightarrow \mathbb{R}$ that represents a mathematical model where the computation $y = G(x)$ solves a specific problem we ask... How sensitive is the solution to changes in x? We can measure this sensitivity in two ways:

- Absolute Condition Number = $\lim_{h \to 0} \frac{|G(x+h) - G(x)|}{|h|}$
- Relative Condition Number = $\lim_{h \to 0} \frac{\frac{|G(x+h) - G(x)|}{|G(x)|}}{\frac{|h|}{|x|}}$

Condition numbers much greater than one mean that the problem is inherently sensitive. We call the problem/model ill-conditioned. Even using a "perfect" algorithm" (no truncation errors) and a "perfect" implementation (no-roundoff errors) can produced inexact results for an ill-conditioned problem/model (since slight errors in input data can produce huge errors in the results).

A specific problem may be modelled mathematically in different ways. The condition number for each model of the same problem may not be the same and may vary to a great degree

Inherent errors in computations

A problem with subtracting nearly equal values

Problem: Compute the sum $x+y$ for $x \in \mathbb{R}, y \in \mathbb{R}$. What is the condition number of this simple problem?
We can simplify this problem further by using the following model. Compute $G(x) = x + y$ for a fixed y.

$$\begin{aligned} \textit{Relative Condition Number} &= \left| \frac{x \frac{dG(x)}{dx}}{G(x)} \right| \\ &= \frac{|x|}{|x+y|} \end{aligned}$$

Problem with cancelation errors

If $|x+y|$ is small then the relative condition number in computing $x+y$ will be large. This happens when $x \approx -y$. In particular, when subtracting floating point numbers with finite precision **catastrophic cancelation** can occur. We will discuss this later.

Inherent errors in computations

Multiplication errors

Problem: Compute the product $x * y$ for $x \in \mathbb{R}, y \in \mathbb{R}$. What is the condition number of this simple problem?
We can simplify this problem further by using the following model. Compute $G(x) = x * y$ for a fixed y.

$$\begin{aligned} \textit{Relative Condition Number} &= \left| \frac{x \frac{dG(x)}{dx}}{G(x)} \right| \\ &= \frac{|x * y|}{|x * y|} \quad , x * y \neq 0 \\ &= 1 \end{aligned}$$

No problem with multiplication errors

The relative condition number is just one.

Inherent errors in computations

Division errors

Problem: Compute the product $\frac{y}{x}$ for $x \in \mathbb{R}, x \neq 0, y \in \mathbb{R}$. What is the condition number of this simple problem?
We can simplify this problem further by using the following model. Compute $G(x) = \frac{y}{x}$ for a fixed y.

$$\begin{aligned} Relative\ Condition\ Number &= \left| \frac{x \frac{dG(x)}{dx}}{G(x)} \right| \\ &= \frac{|x * \frac{-y}{x^2}|}{|\frac{y}{x}|} \quad , x * y \neq 0 \\ &= 1 \end{aligned}$$

No problem with division errors

The relative condition number is just one.

Computing e^{-20}, a well conditioned problem but unstable algorithm

Compute the condition number

Problem: Compute the value of e^{-20}. What is the condition number of this problem? Use $G(x) = e^x$.
The condition number can be computed as follows.

$$
\begin{aligned}
Relative\ Condition\ Number &= \left| \frac{x \frac{dG(x)}{dx}}{G(x)} \right| \\
&= \frac{|x * e^x|}{|e^x|} \\
&= |x| = 20 \text{ when } x = 20
\end{aligned}
$$

The value 20 is not large so the problem is well conditioned.

Computing e^{-20}, a well conditioned problem but unstable algorithm

Use a Taylor Series expansion of e^x as our method to solve the problem.

```
def myexp(x):
    y,newy,term,k = -1.,1.,1,0
    while newy != y:
        k = k+1
        term = (term * x)/k
        y = newy
        newy = y + term
    return newy
```

Large relative error

Python gives the value of $exp(-20) = 2.061153622438558e - 09$ but our code gives $myexp(-20) = 5.621884472130418e - 09$.
Why is the relative error not small when the problem is well conditioned? The answer is that the algorithm is not stable.

Stability

Suppose that we want to solve the problem $y = G(x)$ given both $G : \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$. However, our algorithm for this problem suffers from roundoff and/or truncation errors so we actually compute an approximation $y + \Delta y$.

Define the function $\widehat{G} : \mathbb{R} \to \mathbb{R}$ as $\widehat{G}(x) = y + \Delta y$.

Assuming that G is continuous then if Δy is small enough there will be a value $x + \Delta x$ near x such that $G(x + \Delta x) = y + \Delta y$ (Intermediate Value Theorem). Actually there may be more than one value of Δx that produces this equality so we will choose the one with the smallest value of $|\Delta x|$.

$$
\begin{tikzcd}
x \arrow[r, "G"] \arrow[d, dash, "\text{backward error}"'] \arrow[dr, "\widehat{G}"] & y = G(x) \arrow[d, dash, "\text{forward error}"] \\
x + \Delta x \arrow[r, "G"] & y + \Delta y = \widehat{G}(x) = G(x + \Delta x)
\end{tikzcd}
$$

Stability

Using the diagram below we can find a formula for the approximate value of the relative error.

$$\text{Relative Condition Number} \approx \frac{\left|\frac{(G(x+\Delta x)-G(x))}{G(x)}\right|}{\left|\frac{\Delta x}{x}\right|}$$

$$= \frac{\left|\frac{\Delta y}{y}\right|}{\left|\frac{\Delta x}{x}\right|} = \frac{\text{relative forward error}}{\text{relative backward error}}$$

$$\text{Relative Condition Number} * \left|\frac{\Delta x}{x}\right| \approx \left|\frac{\Delta y}{y}\right|$$

$$\begin{tikzcd}
x \arrow[r, "G"] \arrow[d, dash, "\text{backward error}"'] \arrow[dr, "\widehat{G}"] & y = G(x) \arrow[d, dash, "\text{forward error}"] \\
x + \Delta x \arrow[r, "G"] & y + \Delta y = \widehat{G}(x) = G(x + \Delta x)
\end{tikzcd}$$

Stability

Definition of Stability

An algorithm is stable (backward stable) if the relative backward error $\left|\frac{\Delta x}{x}\right|$ is small, otherwise the algorithm is unstable.

$\left|\frac{\Delta y}{y}\right| \approx$ Relative Condition Number $* \left|\frac{\Delta x}{x}\right|$

The approximation above shows that if the condition number of the problem is small and the algorithm is stable then the solution the algorithm produces is accurate.

$$
\begin{tikzcd}
x \arrow[r, "G"] \arrow[d, dash, "\text{backward error}"'] \arrow[dr, "\widehat{G}"] & y = G(x) \arrow[d, dash, "\text{forward error}"] \\
x + \Delta x \arrow[r, "G"] & y + \Delta y = \widehat{G}(x) = G(x + \Delta x)
\end{tikzcd}
$$

Computing e^{-20} a well conditioned problem but with an unstable algorithm

Why then is the following algorithm unstable?

```
import math
def myexp(x):
    y = -1.
    newy = 1
    term = 1
    k = 0
    while newy != y:
        k = k+1
        term = (term * x)/k
        y = newy
        newy = y + term
    return newy
```

Since we showed the the problem was well conditioned, if the algorithm were stable then the relative error in the solution would have been small.
Note that we can find a stable algorithm to compute e^{-20} by rewriting this expression as $\frac{1}{e^{20}}$ and using a Taylor series for e^{20}. Why does this work when using a Taylor Series for e^{-20} does not?

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

floating point addition is not associative

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

Example

$$(1.0 + 2^{-53}) + 2^{-53} \neq 1.0 + (2^{-53} + 2^{-53})$$

Floating Point Arithmetic

Rule 1. $x \in \mathbb{R}$, $fl(x)$ not subnormal

$$fl(x) = x(1+\delta), \quad \text{where} \quad |\delta| \leqslant \mu$$

Rule 2. x, y are both IEEE floating point numbers

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

$$fl(x \odot y) = (x \odot y)(1+\delta), \quad \text{where} \quad |\delta| \leqslant \mu$$

Rule 3. x, y are both IEEE floating point numbers

For $+, *$ operations

$$fl(x \odot y) = fl(y \odot x)$$

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.

Errors in Floating Point Arithmetic

Consider the sum of 3 numbers: $y = a + b + c$ where a, b, c are machine (normalized) representable numbers.

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

$$
\begin{aligned}
\eta &= fl(a+b) = (a+b)(1+\delta_1) \\
y_1 &= fl(\eta + c) = (\eta + c)(1+\delta_2) \\
&= \left[(a+b)(1+\delta_1) + c\right](1+\delta_2) \\
&= \left[(a+b+c) + (a+b)\delta_1)\right](1+\delta_2) \\
&= (a+b+c)\left[1 + \frac{a+b}{a+b+c}\delta_1(1+\delta_2) + \delta_2\right]
\end{aligned}
$$

So disregarding the high order term $\delta_1\delta_2$

$$
fl(fl(a+b)+c) = (a+b+c)(1+\delta_3) \quad \text{with} \quad \delta_3 \approx \frac{a+b}{a+b+c}\delta_1 + \delta_2
$$

Floating Point Arithmetic

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

$$fl(a + fl(b + c)) = (a + b + c)(1 + \delta_4) \quad \text{with} \quad \delta_4 \approx \frac{b + c}{a + b + c}\delta_1 + \delta_2$$

Main conclusion:

The first error is <u>amplified</u> by the factor $(a + b)/y$ in the first case and $(b + c)/y$ in the 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!]

Loss of Significance

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

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

Summary

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

Cancellation

So what to do? Mainly rearrangement.

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

Cancellation

So what to do? Mainly rearrangement.

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

Problem at $x \approx 0$.

Cancellation

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!

Cancellation

Compute the following with $x = 1.2e-5$.

$$f(x) = \frac{(1-\cos(x))}{x^2}$$

Cancellation

Compute the following with $x = 1.2e - 5$.

$$f(x) = \frac{(1 - \cos(x))}{x^2}$$

At $x = 1.2e - 5$ we get $f(x) = 0.499999732974901$.

Compute the following with $x = 1.2e-5$.

$$f(x) = \frac{(1-\cos(x))}{x^2}$$

At $x = 1.2e-5$ we get $f(x) = 0.499999732974901$.

One type of fix:

$$f(x) = 0.5\left(\frac{\sin(x/2)}{x/2}\right)^2$$

which gives, $f(x) = 0.499999999994000$ again no subtraction!

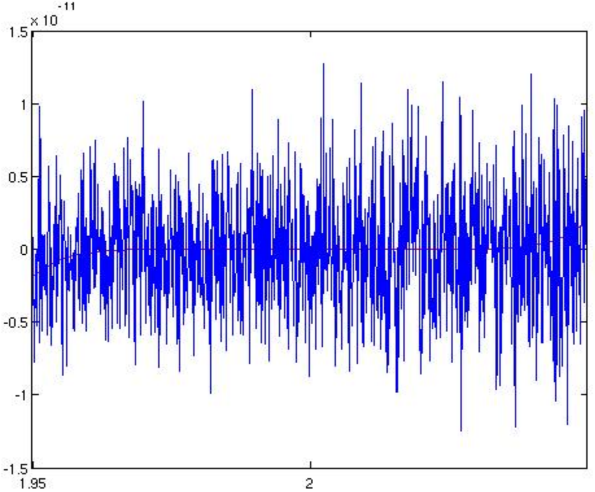
Cancellation Example

We want to plot the function $y = (x-2)^9, x \in [1.95, 2.05]$

```python
import numpy as np
import numpy.polynomial.polynomial as ply
import matplotlib.pyplot as plt

# plot y = (x-2)**9 in red
x = np.linspace(1.95,2.05, 1000)
plt.plot(x,(x-2)**9,'r')

# plot p = x**9-18x**8+144x**7-672x**6+2016x**5-4032x**4+5376x**3-4608x**2+2304x-512 in blue
roots = 2 * np.ones(9)
p = ply.polyfromroots(roots)
#coefficients are in reverse order for polyval
p = p[::-1]
plt.plot(x, np.polyval(p,x),'b')

plt.show()
```

(see plot on next slide)

Cancellation Example



Example

Roots of the equation

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

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

$$y = -p + \sqrt{p^2 + q} = \frac{q}{p + \sqrt{p^2 + q}}$$

Floating Point Arithmetic Example 1
Where is the cancellation error?

```
>>> import math
>>> p = 1000
>>> q = 1
>>> y = -p + math.sqrt(p**2+q)
>>> y
0.0004999998750463419
>>>
>>> y2 = q/(p+math.sqrt(p**2+q))
>>> y2
0.0004999998750000625
>>>
>>> x = y
>>> x**2+2*p*x-q
9.2558847378945e-11
>>> x = y2
>>> x**2+2*p*x-q
0.0
```

Floating Point Arithmetic Example 2
Where is the cancellation error?

Consider now the case when

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

The exact roots are $1 + \delta$ and 1. Take $\delta = 1.E - 08$ and use Python:

```
>>> d = 1.0e-8
>>> p = -(1+d/2)
>>> p
-1.000000005
>>> q = -(1+d)
>>> q
-1.00000001
>>> x = -p-math.sqrt(p**2+q)
>>> x
1.000000005
>>> y = -p+math.sqrt(p**2+q)
>>> y
1.000000005
```

Instructor Notes

- The Euler formula $e^{i\theta}$ needs to be included if DFT is to be included in future notes.
- A more general definition of a ”problem” (as opposed to computing $y = G(x)$) is to solve $G(x, d) = 0$ for $x \in \mathbb{R}^n$ where the data $d \in \mathbb{R}^m$ but this involves discussing the matrix norm and partial derivatives.
- Exponential Convergence

$$|e_n| \leqslant C^{-2^n} \text{ for a constant } C > 1$$