

Numerical Analysis and Computing

Lecture Notes #2 — Calculus Review; Computer Arithmetic and Finite Precision; Algorithms and Convergence; Solutions of Equations of One Variable

Peter Blomgren,
(blomgren.peter@gmail.com)

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720
<http://terminus.sdsu.edu/>

Fall 2014

Lecture Notes #2

— (1/66)

Outline

- 1 Calculus Review
 - Limits, Continuity, and Convergence
 - Differentiability, Rolle's, and the Mean Value Theorem
 - Extreme Value, Intermediate Value, and Taylor's Theorem
- 2 Computer Arithmetic & Finite Precision
 - Binary Representation, IEEE 754-1985
 - Something's Missing...
 - Roundoff and Truncation, Errors, Digits
 - Cancellation
- 3 Algorithms
 - Algorithms, Pseudo-Code
 - Fundamental Concepts
- 4 Solutions of Equations of One Variable
 - $f(x) = 0$, "Root Finding"
 - The Bisection Method
 - When do we stop?!
 - *** Homework #1 ***

Lecture Notes #2

— (2/66)

Why Review Calculus???

It's a good warm-up for our brains!

When developing numerical schemes we will use theorems from calculus to guarantee that our algorithms make sense.

If the theory is sound, when our programs fail we look for bugs in the code!

Calculus Review

— (3/66)

Background Material — A Crash Course in Calculus

Key concepts from Calculus

- Limits
- Continuity
- Convergence
- Differentiability
- Rolle's Theorem
- Mean Value Theorem
- Extreme Value Theorem
- Intermediate Value Theorem
- **Taylor's Theorem...**

Calculus Review

— (4/66)

Limit / Continuity

Definition (Limit)

A function f defined on a set X of real numbers $X \subset \mathbb{R}$ has the limit L at x_0 , written

$$\lim_{x \rightarrow x_0} f(x) = L,$$

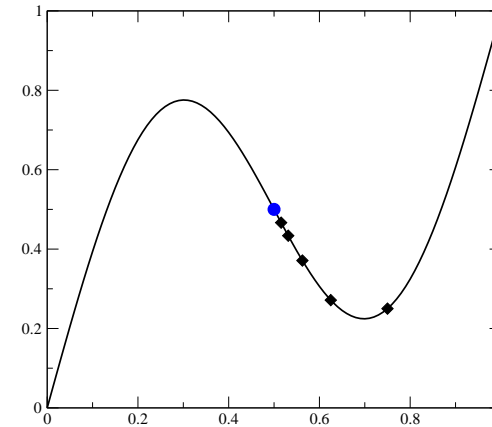
if given any real number $\epsilon > 0$ ($\forall \epsilon > 0$), there exists a real number $\delta > 0$ ($\exists \delta > 0$) such that $|f(x) - L| < \epsilon$, whenever $x \in X$ and $0 < |x - x_0| < \delta$.

Definition (Continuity (at a point))

Let f be a function defined on a set X of real numbers, and $x_0 \in X$. Then f is continuous at x_0 if

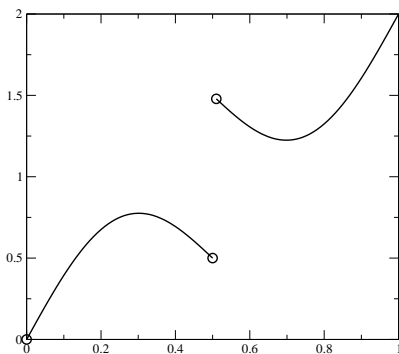
$$\lim_{x \rightarrow x_0} f(x) = f(x_0).$$

Example: Continuity at x_0



Here we see how the limit $x \rightarrow x_0$ (where $x_0 = 0.5$) exists for the function $f(x) = x + \frac{1}{2} \sin(2\pi x)$.

Examples: Jump Discontinuity

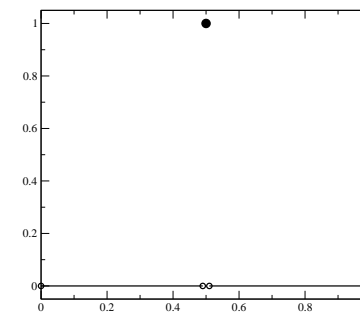


The function

$$f(x) = \begin{cases} x + \frac{1}{2} \sin(2\pi x) & x < 0.5 \\ x + \frac{1}{2} \sin(2\pi x) + 1 & x > 0.5 \end{cases}$$

has a jump discontinuity at $x_0 = 0.5$.

Examples: “Spike” Discontinuity



The function

$$f(x) = \begin{cases} 1 & x = 0.5 \\ 0 & x \neq 0.5 \end{cases}$$

has a discontinuity at $x_0 = 0.5$.

The **limit exists**, but

$$\lim_{x \rightarrow 0.5} f(x) = 0 \neq 1$$

Continuity / Convergence

Definition (Continuity (in an interval))

The function f is continuous on the set X ($f \in C(X)$) if it is continuous at each point x in X .

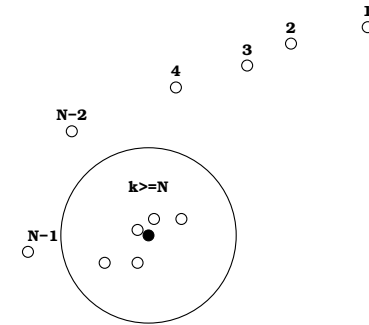
Definition (Convergence of a sequence)

Let $\underline{x} = \{x_n\}_{n=1}^{\infty}$ be an infinite sequence of real (or complex numbers). The sequence \underline{x} converges to x (has the limit x) if $\forall \epsilon > 0, \exists N(\epsilon) \in \mathbb{Z}^+ : |x_n - x| < \epsilon \forall n > N(\epsilon)$. The notation

$$\lim_{n \rightarrow \infty} x_n = x$$

means that the sequence $\{x_n\}_{n=1}^{\infty}$ converges to x .

Illustration: Convergence of a Complex Sequence



A sequence in $\underline{z} = \{z_k\}_{k=1}^{\infty}$ converges to $z_0 \in \mathbb{C}$ (the black dot) if for any ϵ (the radius of the circle), there is a value N (which depends on ϵ) so that the “tail” of the sequence $\underline{z}_t = \{z_k\}_{k=N}^{\infty}$ is inside the circle.

Differentiability

Theorem

If f is a function defined on a set X of real numbers and $x_0 \in X$, the the following statements are **equivalent**:

- (a) f is continuous at x_0
- (b) If $\{x_n\}_{n=1}^{\infty}$ is any sequence in X converging to x_0 , then $\lim_{n \rightarrow \infty} f(x_n) = f(x_0)$.

Definition (Differentiability (at a point))

Let f be a function defined on an open interval containing x_0 ($a < x_0 < b$). f is differentiable at x_0 if

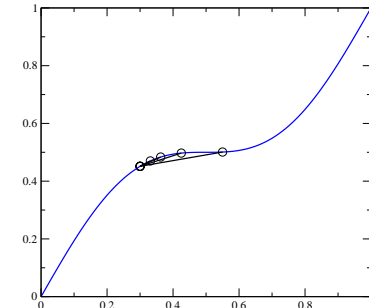
$$f'(x_0) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \text{ exists.}$$

If the limit exists, $f'(x_0)$ is the derivative at x_0 .

Definition (Differentiability (in an interval))

If $f'(x_0)$ exists $\forall x_0 \in X$, then f is differentiable on X .

Illustration: Differentiability



Here we see that the limit

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

exists — and approaches the slope / derivative at x_0 , $f'(x_0)$.

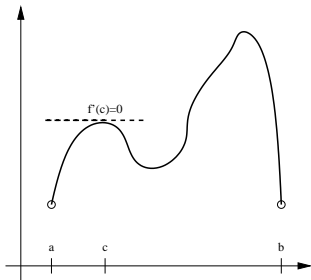
Continuity / Rolle's Theorem

Theorem (Differentiability \Rightarrow Continuity)

If f is differentiable at x_0 , then f is continuous at x_0 .

Theorem (Rolle's Theorem [Wiki-Link](#))

Suppose $f \in C[a, b]$ and that f is differentiable on (a, b) . If $f(a) = f(b)$, then $\exists c \in (a, b): f'(c) = 0$.

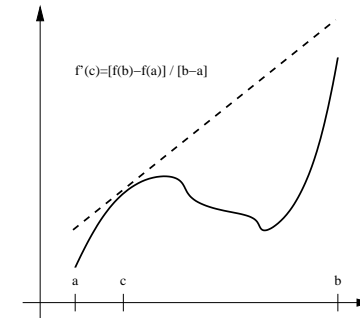


Mean Value Theorem

Theorem (Mean Value Theorem [Wiki-Link](#))

If $f \in C[a, b]$ and f is differentiable on (a, b) , then $\exists c \in (a, b):$

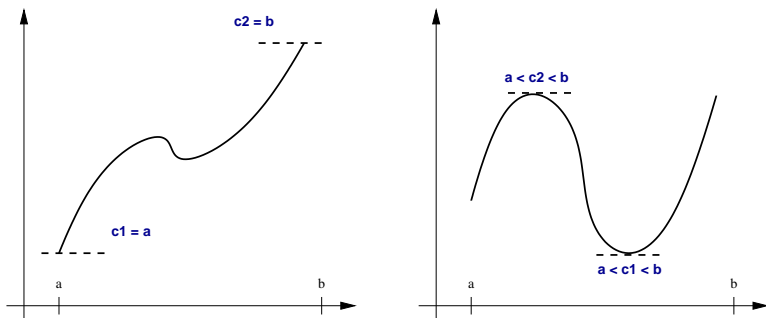
$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$



Extreme Value Theorem

Theorem (Extreme Value Theorem [Wiki-Link](#))

If $f \in C[a, b]$ then $\exists c_1, c_2 \in [a, b]: f(c_1) \leq f(x) \leq f(c_2)$
 $\forall x \in [a, b]$. If f is differentiable on (a, b) then the numbers c_1, c_2 occur either at the endpoints of $[a, b]$ or where $f'(x) = 0$.



Intermediate Value Theorem

Theorem (Intermediate Value Theorem [Wiki-Link](#))

if $f \in C[a, b]$ and K is any number between $f(a)$ and $f(b)$, then there exists a number c in (a, b) for which $f(c) = K$.

Taylor's Theorem

Theorem (Taylor's Theorem [Wiki-Link](#))

Suppose $f \in C^n[a, b]$, $f^{(n+1)} \exists$ on $[a, b]$, and $x_0 \in [a, b]$. Then $\forall x \in (a, b)$, $\exists \xi(x) \in (x_0, x)$ with $f(x) = P_n(x) + R_n(x)$ where

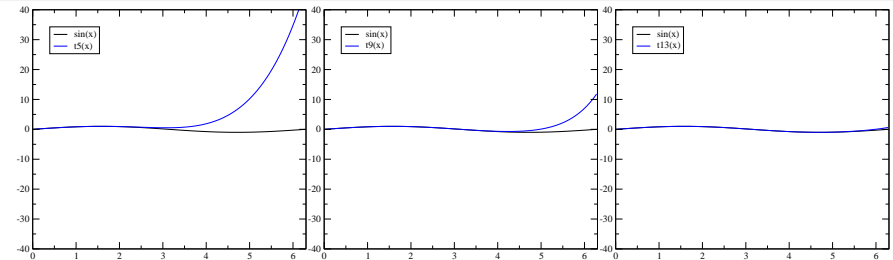
$$P_n(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x-x_0)^k, \quad R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} (x-x_0)^{(n+1)}.$$

$P_n(x)$ is called the **Taylor polynomial of degree n** , and $R_n(x)$ is the **remainder term (truncation error)**.

This theorem is **extremely important** for numerical analysis; Taylor expansion is a fundamental step in the derivation of many of the algorithms we see in this class (and in Math 542 & 693ab).

Illustration: Taylor's Theorem

$f(x) = \sin(x)$



$P_5(x)$

$P_9(x)$

$P_{13}(x)$

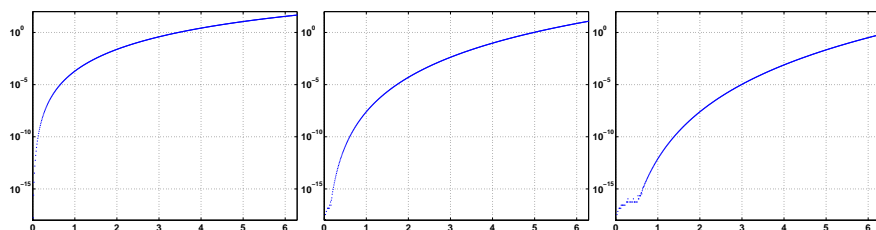
$$P_{13}(x) = x - \underbrace{\frac{1}{3!}x^3 + \frac{1}{5!}x^5}_{P_5(x)} - \frac{1}{7!}x^7 + \frac{1}{9!}x^9 - \frac{1}{11!}x^{11} + \frac{1}{13!}x^{13}$$

$\underbrace{\hspace{10em}}_{P_9(x)}$

Illustration: Taylor's Theorem

Errors

$f(x) = \sin(x)$



$E_5(x)$

$E_9(x)$

$E_{13}(x)$

$$P_{13}(x) = x - \underbrace{\frac{1}{3!}x^3 + \frac{1}{5!}x^5}_{P_5(x)} - \frac{1}{7!}x^7 + \frac{1}{9!}x^9 - \frac{1}{11!}x^{11} + \frac{1}{13!}x^{13}$$

$\underbrace{\hspace{10em}}_{P_9(x)}$

Taylor Expansions — Matlab

- A **Taylor polynomial of degree n** requires all derivatives up to order n , and order $n + 1$ for the **remainder**.
- Derivatives may be [more] complicated expression [than the original function].
- **Matlab** can compute derivatives for you:

Matlab: Symbolic Computations

Try this!!!

```
>> syms x
>> diff(sin(2*x))
>> diff(sin(2*x),3)
>> taylor(exp(x),x,0,'order',5)
>> taylor(exp(x),x,1,'order',5)
```

Computer Arithmetic and Finite Precision

Computers use a finite number of bits (0's and 1's) to represent numbers.

For instance, an 8-bit unsigned integer (a.k.a a "char") is stored:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	1	1	0	1

Here, $2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77$, which represents the upper-case character "M" (US-ASCII).

The *Binary Floating Point Arithmetic Standard 754-1985* (IEEE — The Institute for Electrical and Electronics Engineers) standard specified the following layout for a 64-bit real number:

$$s \ c_{10} \ c_9 \ \dots \ c_1 \ c_0 \ m_{51} \ m_{50} \ \dots \ m_1 \ m_0$$

Where

Symbol	Bits	Description
s	1	The sign bit — 0=positive, 1=negative
c	11	The characteristic (exponent)
m	52	The mantissa

$$r = (-1)^s 2^{c-1023} (1 + m), \quad c = \sum_{k=0}^{10} c_k 2^k, \quad m = \sum_{k=0}^{51} \frac{m_k}{2^{52-k}}$$

As described in previous slide, we cannot represent zero!

There are some special signals in IEEE-754-1985:

Type	S (1 bit)	C (11 bits)	M (52 bits)
signaling NaN	u	2047 (max)	.0uuuuu—u (with at least one 1 bit)
quiet NaN	u	2047 (max)	.1uuuuu—u
negative infinity	1	2047 (max)	.000000—0
positive infinity	0	2047 (max)	.000000—0
negative zero	1	0	.000000—0
positive zero	0	0	.000000—0

From: <http://www.freesoft.org/CIE/RFC/1832/32.htm>

At the other extreme, the difference between

011111111011

and the previous value

0111111110110

is $\frac{2^{1023}}{2^{52}} = 2^{971} \approx 1.99 \cdot 10^{292}$.

That’s a “fairly significant” gap!!!

The number of atoms in the observable universe can be estimated to be no more than $\sim 10^{80}$.

The Relative Gap

It makes more sense to factor the exponent out of the discussion and talk about the relative gap:

Exponent	Gap	Relative Gap (Gap/Exponent)
2^{-1023}	2^{-1075}	2^{-52}
2^1	2^{-51}	2^{-52}
2^{1023}	2^{971}	2^{-52}

Any difference between numbers smaller than the local gap is not representable, e.g. any number in the interval

$$\left[3.0, 3.0 + \frac{1}{2^{51}}\right)$$

is represented by the value 3.0.

The Floating Point “Theorem”

“Theorem”

Floating point “numbers” represent intervals!

Since (most) humans find it hard to think in binary representation, from now on we will for **simplicity** and **without loss of generality** assume that floating point numbers are represented in the normalized floating point form as...

k-digit decimal machine numbers

$$\pm 0.d_1 d_2 \dots d_{k-1} d_k \cdot 10^n,$$

where

$$1 \leq d_1 \leq 9, \quad 0 \leq d_i \leq 9, \quad i \geq 2, \quad n \in \mathbb{Z}.$$

Finite Precision A 128-bit real number quadruple-precision

The *Binary Floating Point Arithmetic Standard 754-1985* (IEEE — The Institute for Electrical and Electronics Engineers) standard specified the following layout for a 128-bit real number:

$$s \ c_{14} \ c_{13} \ \dots \ c_1 \ c_0 \ m_{111} \ m_{110} \ \dots \ m_1 \ m_0$$

Where

Symbol	Bits	Description
s	1	The sign bit — 0=positive, 1=negative
c	15	The exponent
m	112	The fraction

$$r = (-1)^s 2^{c-16,383} (1 + m), \quad c = \sum_{k=0}^{14} c_k 2^k, \quad m = \sum_{k=0}^{111} \frac{m_k}{2^{52-k}}$$

Layout for a 256-bit real number:

$$s \ c_{17} \ c_{16} \ \dots \ c_1 \ c_0 \ m_{236} \ m_{235} \ \dots \ m_1 \ m_0$$

Where

Symbol	Bits	Description
s	1	The sign bit — 0=positive, 1=negative
c	18	The exponent
m	237	The fraction

$$r = (-1)^s 2^{c-131,071} (1 + m), \quad c = \sum_{k=0}^{17} c_k 2^k, \quad m = \sum_{k=0}^{236} \frac{m_k}{2^{52-k}}$$

Any real number can be written in the form

$$\pm 0.d_1 d_2 \dots d_\infty \cdot 10^n$$

given infinite patience and storage space.

We can obtain the floating-point representation $\text{fl}(r)$ in two ways:

- (1) Truncating (chopping) — just keep the first k digits.
- (2) Rounding — if $d_{k+1} \geq 5$ then add 1 to d_k . Truncate.

Examples

$$\text{fl}_{t,5}(\pi) = 0.31415 \cdot 10^1, \quad \text{fl}_{r,5}(\pi) = 0.31416 \cdot 10^1$$

In both cases, the error introduced is called the **roundoff error**.

Let p^* be an approximation to p , then...

Definition (The Absolute Error)

$$|p - p^*|$$

Definition (The Relative Error)

$$\frac{|p - p^*|}{|p|}, \quad p \neq 0$$

Definition (Significant Digits)

The number of **significant digits** is the largest value of t for which

$$\frac{|p - p^*|}{|p|} < 5 \cdot 10^{-t}$$

1) Representation — Roundoff.

2) Cancellation — Consider:

$$\begin{aligned} & 0.12345678012345 \cdot 10^1 \\ - & 0.12345678012344 \cdot 10^1 \\ \hline & = 0.10000000000000 \cdot 10^{-13} \end{aligned}$$

this value has (at most) **1** significant digit!!!

If you assume a “canceled value” has more significant bits (the computer will happily give you some numbers) — I don't want you programming the autopilot for any airlines!!!

Rounding 5-digit arithmetic

$$\begin{aligned}(96384 + 26.678) - 96410 &= \\ (96384 + 00027) - 96410 &= \\ 96411 - 96410 &= 1.0000\end{aligned}$$

Truncating 5-digit arithmetic

$$\begin{aligned}(96384 + 26.678) - 96410 &= \\ (96384 + 00026) - 96410 &= \\ 96410 - 96410 &= 0.0000\end{aligned}$$

Rearrangement changes the result:

$$(96384 - 96410) + 26.678 = -26.000 + 26.678 = 0.67800$$

Numerically, order of computation matters! (This is a HARD problem)

Rounding 5-digit arithmetic

$$\begin{aligned}(0.96384 \cdot 10^5 + 0.26678 \cdot 10^2) - 0.96410 \cdot 10^5 &= \\ (0.96384 \cdot 10^5 + 0.00027 \cdot 10^5) - 0.96410 \cdot 10^5 &= \\ 0.96411 \cdot 10^5 - 0.96410 \cdot 10^5 &= 0.10000 \cdot 10^1\end{aligned}$$

Truncating 5-digit arithmetic

$$\begin{aligned}(0.96384 \cdot 10^5 + 0.26678 \cdot 10^2) - 0.96410 \cdot 10^5 &= \\ (0.96384 \cdot 10^5 + 0.00026 \cdot 10^5) - 0.96410 \cdot 10^5 &= \\ 0.96410 \cdot 10^5 - 0.96410 \cdot 10^5 &= 0.0000 \cdot 10^0\end{aligned}$$

Rearrangement changes the result:

$$\begin{aligned}(0.96384 \cdot 10^5 - 0.96410 \cdot 10^5) + 0.26678 \cdot 10^2 &= \\ -0.26000 \cdot 10^2 + 0.26678 \cdot 10^2 &= 0.67800 \cdot 10^0\end{aligned}$$

Consider the recursive relation

$$x_{n+1} = 1 - (n+1)x_n \quad \text{with} \quad x_0 = 1 - \frac{1}{e}.$$

This sequence can be shown to converge to **0** (in 2 slides).

Subtractive cancellation produces an error which is approximately equal to the machine precision times $n!$.

n	x_n	$n!$	n	x_n	$n!$
0	0.63212056	1	11	0.07735223	3.99e+007
1	0.36787944	1	12	0.07177325	4.79e+008
2	0.26424112	2	13	0.06694778	6.23e+009
3	0.20727665	6	14	0.06273108	8.72e+010
4	0.17089341	24	15	0.05903379	1.31e+012
5	0.14553294	120	16	0.05545930	2.09e+013
6	0.12680236	720	17	0.05719187	3.56e+014
7	0.11238350	5.04e+003	18	-0.02945367	6.4e+015
8	0.10093197	4.03e+004	19	1.55961974	1.22e+017
9	0.09161229	3.63e+005	20	-30.19239489	2.43e+018
10	0.08387707	3.63e+006			

Example: Proof of Convergence to 0

The recursive relation is

$$x_{n+1} = 1 - (n+1)x_n$$

with

$$x_0 = 1 - \frac{1}{e} = 1 - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \dots$$

From the recursive relation

$$x_1 = 1 - x_0 = \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} - \dots$$

$$x_2 = 1 - 2x_1 = \frac{1}{3} - \frac{2}{4!} + \frac{2}{5!} - \dots$$

$$x_3 = 1 - 3x_2 = \frac{3!}{4!} - \frac{3!}{5!} + \frac{3!}{6!} - \dots$$

⋮

$$x_n = 1 - nx_{n-1} = \frac{n!}{(n+1)!} - \frac{n!}{(n+2)!} + \frac{n!}{(n+3)!} - \dots$$

This shows that

$$x_n = \frac{1}{n+1} - \frac{1}{(n+1)(n+2)} + \dots \rightarrow 0 \text{ as } n \rightarrow \infty.$$

Example: Loss of Significant Digits

Matlab code

Matlab code: Loss of Significant Digits

```
clear
x(1) = 1-1/exp(1);
s(1) = 1;
f(1) = 1;
for i = 2:21
    x(i) = 1-(i-1)*x(i-1);
    s(i) = 1/i;
    f(i) = (i-1)*f(i-1);
end
n = 0:20;
z = [n; x; s; f];
fprintf(1, '\n\n n x(n) 1/(n+1) n!\n\n')
fprintf(1, '%2.0f %13.8f %10.8f %10.3g\n', z)
```

Algorithms

Definition (Algorithm)

An **algorithm** is a procedure that describes, in an **unambiguous manner**, a finite sequence of steps to be performed in a specific order.

In this class, the objective of an algorithm is to implement a procedure to solve a problem or approximate a solution to a problem.

Most homes have a collection of algorithms in printed form — we tend to call them “recipes.”

There is a collection of algorithms “out there” called **Numerical Recipes**, Google for it!

Pseudo-code

Definition (Pseudo-code)

Pseudo-code is an algorithm description which specifies the input/output formats.

Note that pseudo-code is **not** computer language specific, but should be easily translatable to any procedural computer language.

Examples of Pseudo-code statements:

for $i = 1, 2, \dots, n$

Set $x_i = a_i + i * h$

While $i < N$ do Steps 17 - 21

If ... then ... else

Definition (Stability)

An algorithm is said to be stable if small changes in the input, generates small changes in the output.

At some point we need to quantify what “small” means!

If an algorithm is stable for a certain **range** of initial data, then is it said to be **conditionally stable**.

Stability issues are discussed in great detail in **Math 543**.

Suppose $E_0 > 0$ denotes the initial error, and E_n represents the error after n operations.

If $E_n \approx CE_0 \cdot n$ (for a constant C which is independent of n), then the growth is **linear**.

If $E_n \approx C^n E_0$, $C > 1$, then the growth is **exponential** — in this case the error will dominate very fast (undesirable scenario).

Linear error growth is usually unavoidable, and in the case where C and E_0 are small the results are generally acceptable. — **Stable algorithm**.

Exponential error growth is unacceptable. Regardless of the size of E_0 the error grows rapidly. — **Unstable algorithm**.

Example BF-1.3.3

1 of 2

The recursive equation

$$p_n = \frac{10}{3}p_{n-1} - p_{n-2}, \quad n = 2, 3, \dots, \infty$$

has the exact solution

$$p_n = c_1 \left(\frac{1}{3}\right)^n + c_2 3^n$$

for any constants c_1 and c_2 . (Determined by starting values.)

In particular, if $p_0 = 1$ and $p_1 = \frac{1}{3}$, we get $c_1 = 1$ and $c_2 = 0$, so

$$p_n = \left(\frac{1}{3}\right)^n \text{ for all } n.$$

Now, consider what happens in 5-digit rounding arithmetic...

Example BF-1.3.3

2 of 2

Now, consider what happens in 5-digit rounding arithmetic...

$$p_0^* = 1.0000, \quad p_1^* = 0.33333$$

which modifies

$$c_1^* = 1.0000, \quad c_2^* = -0.12500 \cdot 10^{-5}$$

The generated sequence is

$$p_n^* = 1.0000 (0.33333)^n - \underbrace{0.12500 \cdot 10^{-5} (3.0000)^n}_{\text{Exponential Growth}}$$

p_n^* quickly becomes a very poor approximation to p_n due to the exponential growth of the initial roundoff error.

Reducing the Effects of Roundoff Error

The effects of roundoff error can be reduced by using higher-order-digit arithmetic such as the double or multiple-precision arithmetic available on most computers.

Disadvantages in using double precision arithmetic are that it takes more computation time and **the growth of the roundoff error is not eliminated but only postponed.**

Sometimes, but not always, it is possible to reduce the growth of the roundoff error by restructuring the calculations.

Key Concepts

Rate of Convergence

Definition (Rate of Convergence)

Suppose the sequence $\underline{\beta} = \{\beta_n\}_{n=1}^{\infty}$ converges to zero, and $\underline{\alpha} = \{\alpha_n\}_{n=1}^{\infty}$ converges to a number α .

If $\exists K > 0: |\alpha_n - \alpha| < K\beta_n$, for n large enough, then we say that $\{\alpha_n\}_{n=1}^{\infty}$ converges to α with a **Rate of Convergence** $\mathcal{O}(\beta_n)$ ("Big Oh of β_n ").

We write

$$\alpha_n = \alpha + \mathcal{O}(\beta_n)$$

Note: The sequence $\underline{\beta} = \{\beta_n\}_{n=1}^{\infty}$ is usually chosen to be

$$\beta_n = \frac{1}{n^p}$$

for some positive value of p .

Examples: Rate of Convergence

Example #1: If

$$\alpha_n = \alpha + \frac{1}{\sqrt{n}}$$

then for any $\epsilon > 0$

$$|\alpha_n - \alpha| = \frac{1}{\sqrt{n}} \leq \underbrace{(1 + \epsilon)}_K \underbrace{\frac{1}{\sqrt{n}}}_{\beta_n}$$

hence

$$\alpha_n = \alpha + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

Examples: Rate of Convergence

Example #2: Consider the sequence (as $n \rightarrow \infty$)

$$\alpha_n = \sin\left(\frac{1}{n}\right) - \frac{1}{n}$$

We **Taylor expand** $\sin(x)$ about $x_0 = 0$:

$$\sin\left(\frac{1}{n}\right) \sim \frac{1}{n} - \frac{1}{6n^3} + \mathcal{O}\left(\frac{1}{n^5}\right)$$

Hence

$$|\alpha_n| = \left| \frac{1}{6n^3} + \mathcal{O}\left(\frac{1}{n^5}\right) \right|$$

It follows that

$$\alpha_n = \mathbf{0} + \mathcal{O}\left(\frac{1}{n^3}\right)$$

Note:

$$\mathcal{O}\left(\frac{1}{n^3}\right) + \mathcal{O}\left(\frac{1}{n^5}\right) = \mathcal{O}\left(\frac{1}{n^3}\right), \text{ since } \frac{1}{n^5} \ll \frac{1}{n^3}, \text{ as } n \rightarrow \infty$$

Generalizing to Continuous Limits

Definition (Rate of Convergence)

Suppose

$$\lim_{h \searrow 0} G(h) = 0, \quad \text{and} \quad \lim_{h \searrow 0} F(h) = L$$

If $\exists K > 0$:

$$|F(h) - L| \leq K |G(h)|$$

$\forall h < H$ (for some $H > 0$), then

$$F(h) = L + \mathcal{O}(G(h))$$

we say that $F(h)$ converges to L with a **Rate of Convergence** $\mathcal{O}(G(h))$.

Usually $G(h) = h^p$, $p > 0$.

Examples: Rate of Convergence

Example #2-b: Consider the function $\alpha(h)$ (as $h \rightarrow 0$)

$$\alpha(h) = \sin(h) - h$$

We **Taylor expand** $\sin(x)$ about $x_0 = 0$:

$$\sin(h) \sim h - \frac{h^3}{6} + \mathcal{O}(h^5)$$

Hence

$$|\alpha(h)| = \left| \frac{h^3}{6} + \mathcal{O}(h^5) \right|$$

It follows that

$$\lim_{h \rightarrow 0} \alpha(h) = \mathbf{0} + \mathcal{O}(h^3)$$

Note:

$$\mathcal{O}(h^3) + \mathcal{O}(h^5) = \mathcal{O}(h^3), \quad \text{since } h^5 \ll h^3, \quad \text{as } h \rightarrow 0$$

Solutions of Equations of One Variable

Our new favorite problem:

$$f(x) = 0.$$

Solutions of Equations of One Variable

Introduction

We are going to solve the equation $f(x) = 0$ (i.e. finding root to the equation), for functions f that are complicated enough that there is no closed form solution (and/or we are too lazy to find it?)

In a lot of cases we will solve problems to which we can find the closed form solutions — we do this as a training ground and to evaluate how good our numerical methods are.

Suppose f is continuous on the interval (a_0, b_0) and $f(a_0) \cdot f(b_0) < 0$ — This means the function changes sign at least once in the interval.

The **intermediate value theorem** guarantees the existence of $c \in (a_0, b_0)$ such that $f(c) = 0$.

Without loss of generality (just consider the function $-f(x)$), we can assume (for now) that $f(a_0) < 0$.

We will construct a sequence of intervals containing the root c :

$$(a_0, b_0) \supset (a_1, b_1) \supset \cdots \supset (a_{n-1}, b_{n-1}) \supset (a_n, b_n) \ni c$$

The sub-intervals are determined recursively:

Given (a_{k-1}, b_{k-1}) , let $m_{k-1} = \frac{a_{k-1} + b_{k-1}}{2}$ be the mid-point.

If $f(m_{k-1}) = 0$, we're done, otherwise

$$(a_k, b_k) = \begin{cases} (m_{k-1}, b_{k-1}) & \text{if } f(m_{k-1}) < 0 \\ (a_{k-1}, m_{k-1}) & \text{if } f(m_{k-1}) > 0 \end{cases}$$

This construction guarantees that $f(a_k) \cdot f(b_k) < 0$ and $c \in (a_k, b_k)$.

After n steps, the interval (a_n, b_n) has the length

$$|b_n - a_n| = \left(\frac{1}{2}\right)^n |b_0 - a_0|.$$

We can take

$$m_n = \frac{a_n + b_n}{2}$$

as the estimate for the root c and we have

$$c = m_n \pm d_n, \quad d_n = \left(\frac{1}{2}\right)^{n+1} |b_0 - a_0|.$$

Convergence is slow:

At each step we gain **one binary digit in accuracy**. Since $10^{-1} \approx 2^{-3.3}$, it takes on average 3.3 iterations to gain one decimal digit of accuracy.

Note: The rate of convergence is completely independent of the function f .

We are only using the **sign of f** at the endpoints of the interval(s) to make decisions on how to update. — By making more effective use of the values of f we can attain significantly faster convergence.

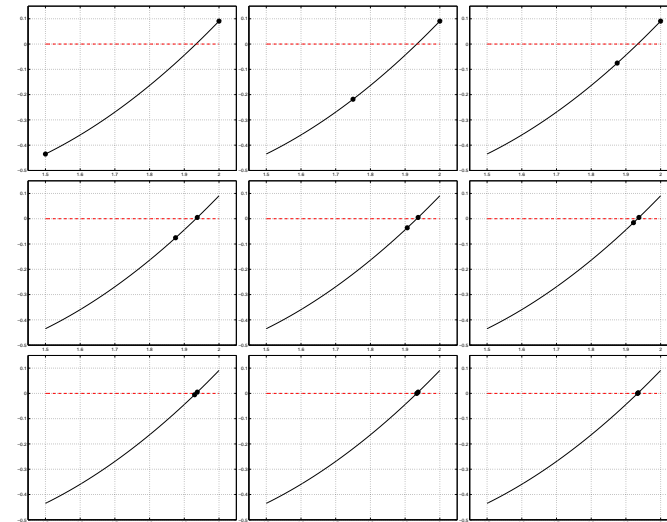
First an example...

The bisection method applied to

$$f(x) = \left(\frac{x}{2}\right)^2 - \sin(x) = 0$$

with $(a_0, b_0) = (1.5, 2.0)$, and $(f(a_0), f(b_0)) = (-0.4350, 0.0907)$ gives:

k	a_k	b_k	m_k	$f(m_k)$
0	1.5000	2.0000	1.7500	-0.2184
1	1.7500	2.0000	1.8750	-0.0752
2	1.8750	2.0000	1.9375	0.0050
3	1.8750	1.9375	1.9062	-0.0358
4	1.9062	1.9375	1.9219	-0.0156
5	1.9219	1.9375	1.9297	-0.0054
6	1.9297	1.9375	1.9336	-0.0002
7	1.9336	1.9375	1.9355	0.0024
8	1.9336	1.9355	1.9346	0.0011
9	1.9336	1.9346	1.9341	0.0004



Matlab code: The Bisection Method

```
% WARNING: This example ASSUMES that f(a)<0<f(b)...
x = 1.5:0.001:2;
f = inline('(x/2).^2-sin(x)', 'x');
a = 1.5;
b = 2.0;
for k = 0:9
    plot(x,f(x),'k-', 'linewidth',2)
    axis([1.45 2.05 -0.5 .15])
    grid on
    hold on
    plot([a b],f([a b]),'ko', 'linewidth',5)
    hold off
    m = (a+b)/2;
    if( f(m) < 0 )
        a = m;
    else
        b = m;
    end
    pause
    print('-depsc', ['bisec' int2str(k) '.eps'], '-f1');
end
```

When do we stop?

We can **(1)** keep going until successive iterates are close:

$$|m_k - m_{k-1}| < \epsilon$$

or **(2)** close in relative terms

$$\frac{|m_k - m_{k-1}|}{|m_k|} < \epsilon$$

or **(3)** the function value is small enough

$$|f(m_k)| < \epsilon$$

No choice is perfect. In general, where no additional information about f is known, the second criterion is the preferred one (since it comes the closest to testing the relative error).

help — Display help text in Command Window

matlab>> help, by itself, lists all primary help topics. [...]

matlab>> help help, gives help for the help command.

lookfor — Find all functions with

matlab>> lookfor function, will return a (long) list of things related to functions.

- Will open on 08/29/2014 at 09:30am PDT.
- Will close no earlier than 09/09/2014 at 09:00pm PDT.