Round-off Errors and Floating Point Arithmetic

- **Binary Machine Numbers:** any double precision non-zero *floating point number* has form

$$x = (-1)^s \, 2^{c-1023} \, (1+f), \quad \text{with 64 bits.}$$

  - s = SIGN BIT: 0 for $x > 0$ and 1 for $x < 0$.
  - c = CHARACTERISTIC, with 11 bits:

$$c = c_1 \cdot 2^{10} + c_2 \cdot 2^9 + c_3 \cdot 2^8 + c_4 \cdot 2^7 + c_5 \cdot 2^6 + c_6 \cdot 2^5 + c_7 \cdot 2^4 + c_8 \cdot 2^3 + c_9 \cdot 2^2 + c_{10} \cdot 2^1 + c_{11} \cdot 2^0,$$

    with each $c_j = 0$ or 1.
  - f = MANTISSA with 52 bits

$$f = f_1 \cdot \left(\frac{1}{2}\right) + \cdots + f_{52} \cdot \left(\frac{1}{2}\right)^{52} = \sum_{j=1}^{52} f_j \cdot \left(\frac{1}{2}\right)^j, \quad \text{each } f_j = 0 \text{ or } 1.$$

  - **Floating Point**: Binary point always comes after 1, independent of c.
- $\boxed{\text{Special cases for special numbers}}$

Round-off Errors and Floating Point Arithmetic

- **Binary Machine Numbers:** Example binary string

  0 10000000011 1011100100010000000000000000000000000000000000000000

- $s = 0$, $c = (10000000011)_2 = 1024 + 2 + 1 = 1027$, and

$$f = 1\cdot\left(\frac{1}{2}\right)^1 + 1\cdot\left(\frac{1}{2}\right)^3 + 1\cdot\left(\frac{1}{2}\right)^4 + 1\cdot\left(\frac{1}{2}\right)^5 + 1\cdot\left(\frac{1}{2}\right)^8 + 1\cdot\left(\frac{1}{2}\right)^{12}.$$

$$(-1)^s 2^{c-1023}(1+f) = (-1)^0 \cdot 2^{1027-1023}\left(1 + \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{4096}\right)\right)$$

$$= 27.56640625.$$

$\boxed{\text{binary representation: ineffective for humans, magical for machines}}$

Street Numbers in Binary? (City of machines)

Round-off Errors and Floating Point Arithmetic

- *k*-**digit <u>Decimal</u> Machine Numbers:**

$$x = \pm 0.d_1 d_2 \cdots d_k \times 10^n, \quad \text{where} \quad 1 \le d_1 \le 9, \quad 0 \le d_i \le 9, i \ge 2.$$

- Any positive real number

$$\begin{aligned} y &= 0.d_1 d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \times 10^n, \\ &\approx 0.d_1 d_2 \cdots d_k \times 10^n \overset{def}{=} fl(y) \quad (\textbf{chopping}) \\ &\approx 0.\delta_1 \delta_2 \cdots \delta_k \times 10^n \overset{def}{=} fl(y) \quad (\textbf{rounding}), \end{aligned}$$

where

$$\textbf{rounding} = \textbf{chopping on} \quad y + 5 \times 10^{n-(k+1)}.$$

  - If $d_{k+1} < 5$: **rounding** $=$ **chopping.**
  - If $d_{k+1} \ge 5$: cut off d_{k+1} and below, then add 1 to d_k.

Round-off Errors and Floating Point Arithmetic

- 5-**digit Decimal Machine Numbers for** π**:**

$$
\begin{aligned}
\pi &= 0.314159265\cdots \times 10^1 \\
&\approx 0.31415 \times 10^1 = 3.1415 \quad (\textbf{chopping}) \\
&\approx (0.31415 + 0.00001) \times 10^1 = 3.1416 \quad (\textbf{rounding}).
\end{aligned}
$$

Absolute error vs. relative error

Suppose that p^* is an approximation to $p \neq 0$.

- **absolute error** $= |p - p^*|$,
- **relative error** $= \frac{|p-p^*|}{|p|}$.

$$\pi \approx 0.31415 \times 10^1 = 3.1415(_{\text{chopping}}), \quad \pi \approx 0.31416 \times 10^1 = 3.1416\ (_{\text{rounding}})$$

- **absolute errors:**

$$|\pi - 3.1415| \approx 9 \times 10^{-5}, \quad |\pi - 3.1416| \approx 7 \times 10^{-6}.$$

- **relative errors:**

$$\frac{|\pi - 3.1415|}{\pi} \approx 3 \times 10^{-5}, \quad \frac{|\pi - 3.1416|}{\pi} \approx 2 \times 10^{-6}.$$

Cool $\$200,000$ wager by LeSean McCoy, 2017

Cool $\$200,000$ wager by LeSean McCoy, 2017

Cool $\$200,000$ wager by LeSean McCoy, 2017

- Wager: Warriors to win NBA Finals
- McCoy made $\$6M$ in 2017. $\dfrac{\textbf{wager}}{\textbf{salary}} \approx 3\%$
- If lost, wager would be a **huge** absolute error, but **small** relative error, to his salary. He won $\$62,500$

Relative error for k-digit **chopping**

Suppose that $y = 0.d_1 d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \times 10^n$, with $d_1 \geq 1$.

$$\left|\frac{y - fl(y)}{y}\right| = \left|\frac{0.d_1 d_2 \ldots d_k d_{k+1} \ldots \times 10^n - 0.d_1 d_2 \ldots d_k \times 10^n}{0.d_1 d_2 \ldots \times 10^n}\right|$$

$$= \left|\frac{0.d_{k+1} d_{k+2} \ldots \times 10^{n-k}}{0.d_1 d_2 \ldots \times 10^n}\right| = \left|\frac{0.d_{k+1} d_{k+2} \ldots}{0.d_1 d_2 \ldots}\right| \times 10^{-k}.$$

But $0.d_1 d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \geq 0.1$,

$$\left|\frac{y - fl(y)}{y}\right| \leq \frac{1}{0.1} \times 10^{-k} = 10^{-k+1}.$$

Relative error for k-digit **rounding**

Suppose that $y = 0.d_1 d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \times 10^n$, with $d_1 \geq 1$.

$$\left| \frac{y - fl(y)}{y} \right| \leq 0.5 \times 10^{-k+1}.$$

Proof: Exercise in text.

Floating Point Arithmetic Magic:

$$\boxed{\text{RELATIVE ERROR} \approx 10^{-k+1} \text{ INDEPENDENT OF } n.}$$

Machine addition, subtraction, multiplication, and division

$$x \oplus y = fl(fl(x) + fl(y)), \quad x \otimes y = fl(fl(x) \times fl(y)),$$

$$x \ominus y = fl(fl(x) - fl(y)), \quad x \oslash y = fl(fl(x) \div fl(y)).$$

Some computations involve millions of these operations, the result could be very different from expected.

<u>Sometimes</u> it takes numerical analysis to make it right

Cancellation of significant digits, k digit arithmetic, $p < k$

Cancellation of significant digits, k digit arithmetic, $p < k$

Suppose that x and y do not differ much:

$$\begin{aligned} x &= 0.d_1 \cdots d_p \alpha_{p+1} \cdots \times 10^n \\ &= 0.d_1 \cdots d_p \alpha_{p+1} \cdots \alpha_k \times 10^n + \epsilon_x = fl(x) + \epsilon_x, \\ y &= 0.d_1 \cdots d_p \beta_{p+1} \cdots \times 10^n \\ &= 0.d_1 \cdots d_p \beta_{p+1} \cdots \beta_k \times 10^n + \epsilon_y = fl(y) + \epsilon_y, \end{aligned}$$

with $\epsilon_x, \epsilon_y \approx 10^{n-k}, \quad k > p.$

Cancellation of significant digits, k digit arithmetic, $p < k$

Suppose that x and y do not differ much:

$$\begin{aligned}
x &= 0.d_1 \cdots d_p \alpha_{p+1} \cdots \times 10^n \\
&= 0.d_1 \cdots d_p \alpha_{p+1} \cdots \alpha_k \times 10^n + \epsilon_x = fl(x) + \epsilon_x, \\
y &= 0.d_1 \cdots d_p \beta_{p+1} \cdots \times 10^n \\
&= 0.d_1 \cdots d_p \beta_{p+1} \cdots \beta_k \times 10^n + \epsilon_y = fl(y) + \epsilon_y,
\end{aligned}$$

with $\epsilon_x, \epsilon_y \approx 10^{n-k}, \quad k > p$. The floating-point form of $x - y$ is

$$fl\,(fl(x) \; - \; fl(y)) \;\approx\; x - y - \epsilon_x + \epsilon_y.$$

$\boxed{\text{if}}$ $\quad |x - y| \approx 10^{n-p}, \quad$ $\boxed{\text{then}}$ **relative error** is

$$\begin{aligned}
\left| \frac{\text{error in computed } x - y}{x - y} \right| &= \left| \frac{(x - y) - fl(\,fl(x) \; - \; fl(y))}{x - y} \right| \\
&\approx \left| \frac{|\epsilon_x| + |\epsilon_y|}{x - y} \right| \approx \frac{10^{n-k}}{10^{n-p}} = 10^{-(k-p)}.
\end{aligned}$$

Quadratic formula for $ax^2 + bx + c = 0$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

One of x_1, x_2 faces cancellation of significant digits if

$$|4ac| \ll b^2$$

Quadratic formula for $ax^2 + bx + c = 0$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

One of x_1, x_2 faces cancellation of significant digits if

$$|4ac| \ll b^2 \quad \Longrightarrow \sqrt{b^2 - 4ac} \approx |b|$$

- If $b > 0$, then x_1 is hard to calculate.
- If $b < 0$, then x_2 is hard to calculate.

Roots to Quadratic to Roots (I)

```
function xx = quadroot(x)
a = 1;
b =-(x(1)+x(2));
c = x(1)*x(2);
del = sqrt(b*b-4*a*c);
xx(1) = (-b+del)/(2*a);
xx(2) = (-b-del)/(2*a);
xx =xx(:);
```

$\boxed{b \text{ and } c\text{: Vieta's formulas}}$

Roots to Quadratic to Roots (II)

```
>> format long e
format long e
>> x = randn(2,1)
x = randn(2,1)

x =

     1.630235289164729e+00
     4.888937703117894e-01

>> xx = quadroot(x)
xx = quadroot(x)

xx =

     1.630235289164729e+00
     4.888937703117894e-01

>> x = [randn*1e5;randn*1e-12]
x = [randn*1e5;randn*1e-12]

x =

     1.034693009917860e+05
     7.268851333832379e-13

>> xx = quadroot(x)
xx = quadroot(x)

xx =

     1.034693009917860e+05
                         0
```

Numerical instability: complete loss of significant digits in smaller root

Solving $ax^2 + bx + c = 0$ the better way

- Compute $\delta = \sqrt{b^2 - 4 * a * c}$
- If $b > 0$ then
$$x_1 = \frac{-b - \delta}{2a} = -\frac{|b| + \delta}{2a};$$
if $b \leq 0$ then
$$x_1 = \frac{-b + \delta}{2a} = \frac{|b| + \delta}{2a}.$$
- Vieta's formula
$$x_2 = \frac{c}{a\,x_1}.$$

Roots to Quadratic to Roots (III)

```
>> a = randn*1e-5;b = 1; c = - randn*1e-12;
a = randn*1e-5;b = 1; c = - randn*1e-12;
>> roots([a b c])
roots([a b c])

ans =

     3.295534380226372e+05
     2.938714670966580e-13

>> del = sqrt(b*b-4*a*c)
del = sqrt(b*b-4*a*c)

del =

     1

>> x(1) = (-b+del)/(2*a);x(2) = (-b-del)/(2*a)
x(1) = (-b+del)/(2*a);x(2) = (-b-del)/(2*a)

x =

                         0
     3.295534380226372e+05

>> x(2) = (-b-del)/(2*a);x(1)=(c/a)/x(2)
x(2) = (-b-del)/(2*a);x(1)=(c/a)/x(2)

x =

     2.938714670966580e-13
     3.295534380226372e+05
```

Numerical stability: both roots accurately computed

§1.3 Horner's Method for Fibonacci's Problem in 1224

Solve

$$f(x) = x^3 + 2x^2 + 10x - 20 = 0.$$

Fibonacci's Solution

$$x = 1+22\left(\frac{1}{60}\right)+7\left(\frac{1}{60}\right)^2+42\left(\frac{1}{60}\right)^3+33\left(\frac{1}{60}\right)^4+4\left(\frac{1}{60}\right)^5+40\left(\frac{1}{60}\right)^6.$$

With Horner's nested sum method, let $\tau = \frac{1}{60}$:

$$x = 1 + \tau \cdot (22 + \tau \cdot (7 + \tau \cdot (42 + \tau \cdot (33 + \tau \cdot (4 + 40\tau))))).$$

Pseudocode for Horner's Method (nested arithmetic)

Evaluate function $f(x)$ for given x:

$$f(x) \quad = \quad a_1 + a_2\,x + \cdots + a_n\,x^{n-1}$$

Pseudocode for Horner's Method (nested arithmetic)

Evaluate function $f(x)$ for given x:

$$\begin{aligned} f(x) &= a_1 + a_2 x + \cdots + a_n x^{n-1} \\ &= a_1 + x \cdot (a_2 + x \cdot (\cdots + x \cdot (a_{n-1} + x \cdot a_n) \cdots)) \end{aligned}$$

Pseudocode for Horner's Method (nested arithmetic)

Evaluate function $f(x)$ for given x:

$$\begin{aligned} f(x) &= a_1 + a_2 x + \cdots + a_n x^{n-1} \\ &= a_1 + x \cdot (a_2 + x \cdot (\cdots + x \cdot (a_{n-1} + x \cdot a_n) \cdots)) \end{aligned}$$

```
function SUM = horner(x,a)
%
% horner's method
%
n = length(a);
SUM = a(n)*ones(size(x));
for i=n-1:-1:1
    SUM = a(i) + x .* SUM;
end
return
```

Numerical stability: a second order recursion

For any constants c_1 and c_2,

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

is a solution to the recursive equation

$$p_n = \frac{10}{3} p_{n-1} - p_{n-2}, \quad \text{for } n = 2, 3, \ldots .$$

- $$\lim_{n\to\infty} |p_n| = \begin{cases} \infty & \text{if} \quad c_2 \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$
- $$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \frac{1}{8} \begin{pmatrix} 9p_0 - 3p_1 \\ 3p_1 - p_0 \end{pmatrix}, \quad \text{given } p_0, p_1.$$
- condition $c_2 = 3p_1 - p_0 = 0$ **hard to satisfy exactly in finite precision computations**.

Numerical values go crazy for $p_0 = 1, p_1 = 1/3$.

With five-digit rounding arithmetic,

n	Computed $\hat{p}_n$	Correct p_n	Relative Error
0	0.10000×10^1	0.10000×10^1	
1	0.33333×10^0	0.33333×10^0	
2	0.11110×10^0	0.11111×10^0	9×10^{-5}
3	0.37000×10^{-1}	0.37037×10^{-1}	1×10^{-3}
4	0.12230×10^{-1}	0.12346×10^{-1}	9×10^{-3}
5	0.37660×10^{-2}	0.41152×10^{-2}	8×10^{-2}
6	0.32300×10^{-3}	0.13717×10^{-2}	8×10^{-1}
7	-0.26893×10^{-2}	0.45725×10^{-3}	7×10^0
8	-0.92872×10^{-2}	0.15242×10^{-3}	6×10^1

Numerical instability: More details in Chapter 5

Rate of convergence: the Big O (I)

$\boxed{\text{Suppose}}$

- $\{\beta_n\}_{n=1}^{\infty}$ is a sequence known to converge to 0,
- $\{\alpha_n\}_{n=1}^{\infty}$ is a sequence known to converge to α.

$\boxed{\text{If}}$ there exists a positive constant K such that

$$|\alpha_n - \alpha| \leq K \, |\beta_n| \quad \text{for large } n,$$

$\boxed{\text{then}}$ we say that

$\{\alpha_n\}_{n=1}^{\infty}$ converges to α with **rate of convergence** $O\left(|\beta_n|\right)$:

$$\alpha_n = \alpha + O\left(|\beta_n|\right)$$

Rate of convergence: the Big O (II)

- **Example**: Suppose that for all $n \geq 1$,

$$\alpha_n = \cos\left(\frac{1 + n\cos\left(n^2+1\right)}{(1+n)^2}\right), \quad \beta_n = \frac{1}{n^2}.$$

- **Then** $\alpha = 1$,

$$|\alpha_n - 1| \leq \frac{1}{2} \cdot \frac{1}{n^2}.$$

- **Therefore** $\{\alpha_n\}_{n=1}^{\infty}$ converges to $\alpha = 1$ with *rate of convergence* $O\left(\frac{1}{n^2}\right)$: $\alpha_n = 1 + O\left(\frac{1}{n^2}\right)$

- $\boxed{\text{Not to be confused with } \textit{order of convergence} \text{ later on.}}$

Rate of convergence: the Big O (III)

Definition: Suppose that $\lim_{h\to 0} G(h) = 0$ and $\lim_{h\to 0} F(h) = L$. If there exists a positive number K so that

$$|F(h) - L| \le K\,|G(h)| \quad \text{for all sufficiently small } h\text{, then}$$

$$F(h) = L + O(G(h)).$$

Rate of convergence: the Big O (III)

Definition: Suppose that $\lim_{h\to 0} G(h) = 0$ and $\lim_{h\to 0} F(h) = L$. If there exists a positive number K so that

$$|F(h) - L| \leq K\,|G(h)| \qquad \text{for all sufficiently small } h\text{, then}$$

$$F(h) = L + O(G(h)).$$

- **Example I**: Show that

$$\mathbf{sin}\ (h) = h + O\left(h^3\right).$$

- PROOF: By Taylor expansion,

$$\mathbf{sin}\ (h) = h - \frac{1}{6}\,h^3\,\mathbf{cos}\ \left(\overline{\xi}(h)\right),$$

for some number $\overline{\xi}(h)$ between 0 and h. Hence

$$\left|\mathbf{sin}\ (h) - h\right| \leq \frac{1}{6}\ |h|^3.$$

- **Therefore**

$$\mathbf{sin}\ (h) = h + O\left(h^3\right).$$

Rate of convergence: the Big O (IV)

Definition: Suppose that $\lim_{h\to 0} G(h) = 0$ and $\lim_{h\to 0} F(h) = L$. If there exists a positive number K so that

$$|F(h) - L| \leq K|G(h)| \quad \text{for all sufficiently small } h \text{, then}$$

$$F(h) = L + O(G(h)).$$

Rate of convergence: the Big O (IV)

Definition: Suppose that $\lim_{h\to 0} G(h) = 0$ and $\lim_{h\to 0} F(h) = L$. If there exists a positive number K so that

$$|F(h) - L| \leq K\,|G(h)| \qquad \text{for all sufficiently small } h \text{, then}$$

$$F(h) = L + O(G(h)).$$

- **Example II**: Taylor expand a function $f(x)$ at $x = x_0$:

$$f(x) = f(x_0) + f'(x_0)\,(x - x_0) + \frac{1}{2} f''(\xi)\,(x - x_0)^2,$$

  with ξ somewhere between x_0 and x.
- If $|f''(\xi)| \leq K$ for some constant K, then

$$f(x) = f(x_0) + f'(x_0)\,(x - x_0) + O\left((x - x_0)^2\right).$$

Class algorithms vs. Commercial software

For any vector $\mathbf{x} \in \mathbf{R}^n$, compute its norm

$$\|\mathbf{x}\|_2 = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{\frac{1}{2}} = \left(\sum_{k=1}^{n} x_k^2\right)^{\frac{1}{2}}.$$

- **INPUT:** $n, x_1, \cdots, x_n$.
- **OUTPUT: Norm**.
- **Step 1**: Set **SUM** $= 0$.
- **Step 2**: For $k = 1, \cdots, n$ set **SUM** $=$ **SUM** $+ x_k * x_k$.
- **Step 3**: Set **Norm** $= \sqrt{\textbf{SUM}}$.
- **Step 4**: Output **Norm**.
  STOP.

Class algorithms vs. Commercial software (I)

```
>>
>> n = 10;
>>
>> x = (1:n)';
>>
>> sum = 0;
>>
>> for k = 1:n

    sum = sum + x(k) * x(k);

  end
>>
>> x_norm = sqrt(sum);
>>
>> disp([x_norm,abs(x_norm-norm(x)), abs(x_norm-sqrt(n*(n+1)*(2*n+1)/6))]);
  19.62142    0.00000    0.00000
```

Class algorithms vs. Commercial software (II)

```
>>
>> x = 1e200 * (1:n)';
>>
>> sum = 0;
>>
>> for k = 1:n

       sum = sum + x(k) * x(k);

   end
>>
>> x_norm = sqrt(sum);
>>
>> disp([norm(x),abs(x_norm-norm(x))])
   1.9621e+201           Inf
>>
```

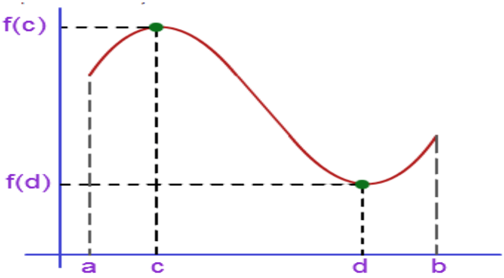
Class algorithms vs. Commercial software (III)

```
>>
>> xmax = max(abs(x));
>>
>> if (xmax == 0)
        x_norm = 0;
   else
        y = x/xmax;
        sum = 0;
        for k = 1:n
            sum = sum + y(k) * y(k);
        end
        x_norm = xmax * sqrt(sum);
   end
>>
>> disp([norm(x),abs(x_norm-norm(x))])
   1.9621e+201    0.0000e+00
```

Material to skip in Chapter 2

- False position in Section 2.3

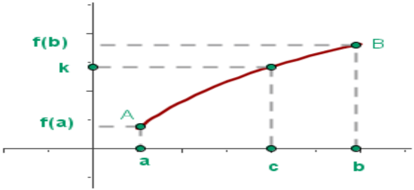
Extreme Value Theorem



- Maximum $f(c)$ and minimum $f(d)$ attainable in $[a, b]$ if $f(x)$ continuous.
- Basis of much of data analysis, artificial intelligence.
- IF $c \in (a, b)$ AND $f(x)$ differentiable, then

$$f'(c) = 0.$$

Intermediate Value Theorem



- If $f(x)$ continuous, then c exists in $[a, b]$ so $f(c) = k$ for any k between $f(a)$ and $f(b)$.
- Basis of methods for solving $f(x) = 0$.

We will actually find c in equation $f(c) = 0$ to some TOLERANCE.

§2.1 Bisection Method

theorem: Given continuous function $f(x)$ on an interval $[a, b]$ with $f(a) \cdot f(b) < 0$, there must be a root p in (a, b) so that $\boxed{f(p) = 0.}$

§2.1 Bisection Method

theorem: Given continuous function $f(x)$ on an interval $[a, b]$ with $f(a) \cdot f(b) < 0$, there must be a root p in (a, b) so that $\boxed{f(p) = 0.}$

PROOF: By Intermediate Value Thm, 0 is between $f(a), f(b)$. $\square$

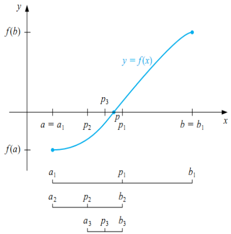
§2.1 Bisection Method

theorem: Given continuous function $f(x)$ on an interval $[a, b]$ with $f(a) \cdot f(b) < 0$, there must be a root p in (a, b) so that $\boxed{f(p) = 0.}$

PROOF: By Intermediate Value Thm, 0 is between $f(a), f(b)$. □

- To find a root p: set $[a_1, b_1] = [a, b]$.
- set $p_1 = \frac{a_1+b_1}{2}$ and compute $f(p_1)$.
  - if $f(p_1) = 0$, then quit with root p_1 (NEED BE VERY LUCKY, BUT COULD HAPPEN.)
  - if $f(a_1) \cdot f(p_1) < 0$, then set $[a_2, b_2] = [a_1, p_1]$,
  - otherwise $(f(p_1) \cdot f(b_1) < 0)$ set $[a_2, b_2] = [p_1, b_1]$,

  In both cases, new interval $\boxed{\text{half}}$ as long as old one.
- repeat with $p_2 = \frac{a_2+b_2}{2}$.



Bisection Method in Cartoon

Naive Bisection Method

```
% Bisection Method

%Input: f(x) continuous on [a, b]
%       f(a) * f(b) < 0

%Output: p in (a, b) so f(p) = 0.


fa = f(a);
fb = f(b);

repeat
   c = (a+b)/2;
   fc = f(c);
   if (fc ==0)
      p = c;
      return;
   end
   if (fc * fa < 0)
      b = c;
   else
      a = c;
   end
end
```

```
function [x, out] = bisect(Fcn, Intv, params)
%
%   To find a solution to f(x) = 0 given the continuous function
%   f on the interval [a,b], where f(a) and f(b) have
%   opposite signs:
%
%   INPUT:   function f(x) defined by function handle Fcn,
%            interval [a,b]= [Intv.a, Intv.b]
%            tolerence params.tol, max # of iterations = params.MaxIt
%   OUTPUT:  root x, and data structure out.
%            The success flag out.flg, is 0 for successful
%            execution and non-zero otherwise. out.it is the number
%            of iterations to reach within tolerance.
%
% Written by Ming Gu for Math128A, Spring 2021

TOL   = params.tol;
NO    = params.MaxIt;
a     = Intv.a;
b     = Intv.b;
if (a > b)
    a = Intv.b;
    b = Intv.a;
end
fa    = sign(Fcn(a));
fb    = sign(Fcn(b));
if (fa*fb >0)
     error('Initial Interval may not contain root',msg);
end
if a==b
     error('Initial values for a and b must not equal',msg);
end

It     = 0;
out.x = [a;b];
out.f = [Fcn(a);Fcn(b)];
while (It <= NO)
    c      = (a+b)/2;
    out.x = [out.x;c];
    out.f =[out.f;Fcn(c)];
    fc     = sign(Fcn(c));
    if (fc ==0)
        x         = c;
        out.flg = 0;
        out.it   = It;
        return;
    end
    if (fc * fa < 0)
        b = c;
    else
        a = c;
    end
    if (abs(b-a)<=TOL)
        x         = (a+b)/2;
        out.flg = 0;
        out.it   = It;
        return;
    end
    It = It + 1;
end
out.flg =1;
out.it   = NO;
x         = (a+b)/2;
```

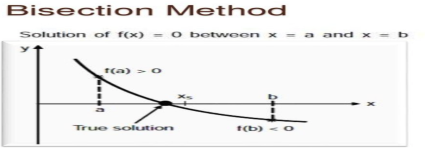
Theorem 2.1 Suppose that $f \in C[a, b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \leq \frac{b-a}{2^n}, \quad \text{when} \quad n \geq 1. \qquad \blacksquare$$

Theorem 2.1 Suppose that $f \in C[a,b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \le \frac{b-a}{2^n}, \quad \text{when} \quad n \ge 1.$$

$\boxed{\text{Most versatile root-finder}}$



- $\boxed{\text{Always}}$ works as long as $f(a)\,f(b) > 0$.

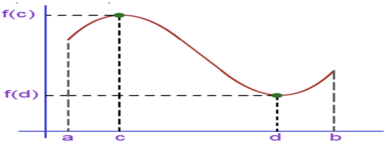
Theorem 2.1 Suppose that $f \in C[a,b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \leq \frac{b-a}{2^n}, \quad \text{when} \quad n \geq 1.$$

Theorem 2.1 Suppose that $f \in C[a,b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \le \frac{b-a}{2^n}, \quad \text{when} \quad n \ge 1.$$

$\boxed{\text{Potential problems with Thm. 2.1 in optimization applications}}$



- Both maximum $f'(c) = 0$ and minimum $f'(d) = 0$. Thm. 2.1 can't tell which one.
- Thm. 2.1 condition does not work: $f'(a)\,f'(b) > 0$.

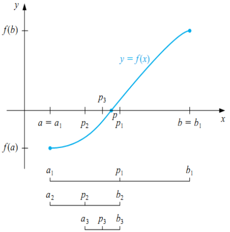
Proof of Thm 2.1, Assume that $f(p_n) \neq 0$ for all n

- By construction

$$a_1 \leq a_2 \leq \cdots \leq a_n \leq \cdots \leq \cdots \leq b_n \leq \cdots \leq b_2 \leq b_1.$$

Thus sequences $\{a_n\}$ and $\{b_n\}$ monotonically converge to limits $a_\infty \leq b_\infty$, respectively.

- Since $f(a_n) \cdot f(b_n) < 0$ for all n, it follows that $f(a_\infty) \cdot f(b_\infty) \leq 0$, thus a root $p \in [a_\infty, b_\infty] \subset [a_n, b_n]$ exists.
- Since $p_n = \frac{a_n + b_n}{2}$, it follows that $|p_n - p| \leq \frac{b_n - a_n}{2}$.



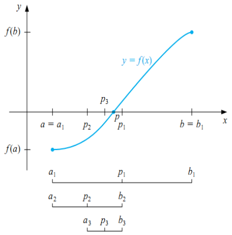
Proof of Thm 2.1, Assume that $f(p_n) \neq 0$ for all n

- By construction

$$a_1 \leq a_2 \leq \cdots \leq a_n \leq \cdots \leq \cdots \leq b_n \leq \cdots \leq b_2 \leq b_1.$$

Thus sequences $\{a_n\}$ and $\{b_n\}$ monotonically converge to limits $a_\infty \leq b_\infty$, respectively.

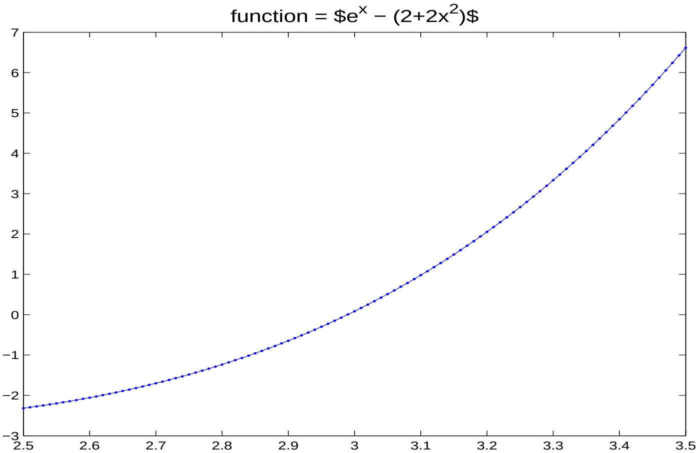
- Since $f(a_n) \cdot f(b_n) < 0$ for all n, it follows that $f(a_\infty) \cdot f(b_\infty) \leq 0$, thus a root $p \in [a_\infty, b_\infty] \subset [a_n, b_n]$ exists.
- Since $p_n = \frac{a_n + b_n}{2}$, it follows that $|p_n - p| \leq \frac{b_n - a_n}{2}$.

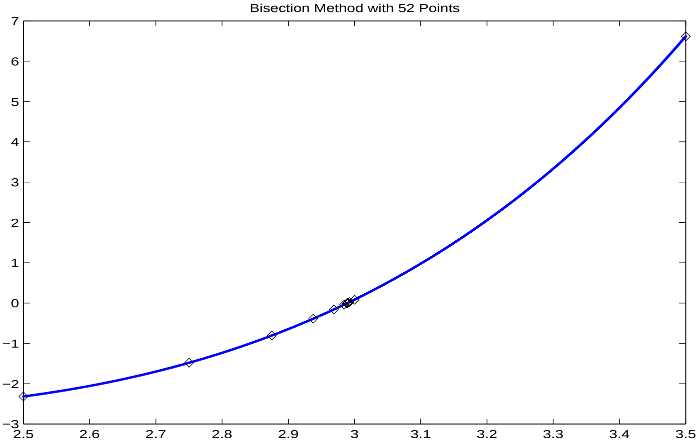


By construction $\displaystyle b_n - a_n = \frac{b_{n-1} - a_{n-1}}{2} = \frac{b_{n-2} - a_{n-2}}{2^2} = \cdots = \frac{b_1 - a_1}{2^{n-1}} = \frac{b - a}{2^{n-1}}.$

Put together, $\displaystyle |p_n - p| \leq \frac{b-a}{2^n}.$ In fact, $a_\infty = b_\infty = p$.

Example Function with Root





Bisection Method with 52 Points
7
6
5
4
3
2
1
0
−1
−2
−3
2.5
2.6
2.7
2.8
2.9
3
3.1
3.2
3.3
3.4
3.5

§2.2 Fixed Point Iteration

The number p is a **fixed point** for a given function g if $g(p) = p$.

- $\boxed{\text{Given}}$ a root-finding problem $f(p) = 0$, we can define functions $g(x)$ with a fixed point at p in multiple ways:

$$g(x) = x - f(x), \quad g(x) = x + 3\,f(x), \quad \text{etc.}$$

- Conversely, $\boxed{\text{given}}$ function g with fixed point at p, then the function

$$f(x) = x - g(x)$$

has a root at p.

§2.2 Fixed Point Iteration

The number p is a **fixed point** for a given function g if $g(p) = p$.

- $\boxed{\text{Given}}$ a root-finding problem $f(p) = 0$, we can define functions $g(x)$ with a fixed point at p in multiple ways:

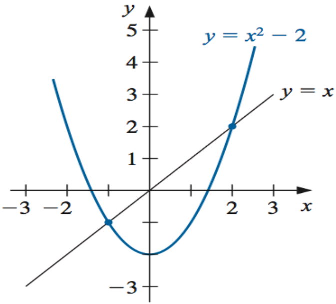
$$g(x) = x - f(x), \quad g(x) = x + 3\,f(x), \quad \text{etc.}$$

- Conversely, $\boxed{\text{given}}$ function g with fixed point at p, then the function

$$f(x) = x - g(x)$$

has a root at p.

Fixed Point Example



Fixed Point Iteration

Given initial approximation p_0, define *Fixed Point Iteration*

$$p_n = g(p_{n-1}), \quad n = 1, 2, \cdots,$$

If iteration converges to p, then

$$p = \lim_{n \to \infty} p_n = \lim_{n \to \infty} g(p_{n-1}) = g(p).$$

Fixed Point Iteration

Given initial approximation p_0, define *Fixed Point Iteration*

$$p_n = g(p_{n-1}), \quad n = 1, 2, \cdots,$$

If iteration converges to p, then

$$p = \lim_{n \to \infty} p_n = \lim_{n \to \infty} g(p_{n-1}) = g(p).$$

$$\boxed{\text{Easy to define. How does it work?}}$$

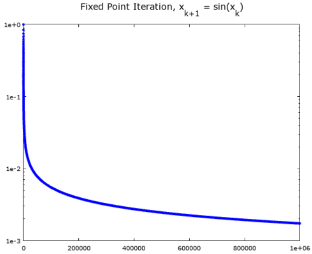
Fixed Point Example $x - \sin(x) = 0$: slow convergence

$$
\begin{aligned}
g(x) &= \sin(x) \in [-1, 1] \quad \text{for} \quad x \in [-1, 1], \\
|g'(x)| &\leq 1 \in [0, 1].
\end{aligned}
$$

```
>> n = 1000000;
>> x = zeros(n,1);
>> x(1) = 1;
>> for k=2:n
x(k) = sin(x(k-1));
end
>> semilogy(abs(x),'b.-')
>> title('Fixed Point Iteration, x_{k+1} = sin(x_k)', 'FontSize', 14)
```

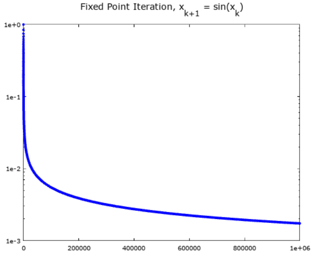
Fixed Point Example $x - \sin(x) = 0$: VERY slow convergence

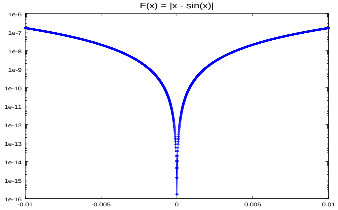
$$g(x) \quad = \quad \sin(x) \in [-1, 1] \quad \text{for} \quad x \in [-1, 1].$$



Fixed Point Example $x - \sin(x) = 0$: VERY slow convergence

$$g(x) \quad = \quad \sin(x) \in [-1, 1] \quad \text{for} \quad x \in [-1, 1].$$





Fixed Point: $x - (1 - \cos(x)) = 0$: VERY fast convergence

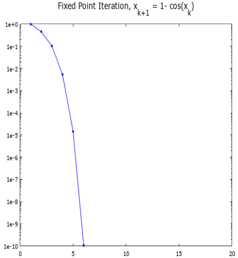
$$
\begin{aligned}
g(x) &= 1 - \cos(x) \in [-1, 1] \quad \text{for} \quad x \in [-1, 1], \\
|g'(x)| &= |\sin x| \leq \sin 1.
\end{aligned}
$$

```
>>
>> n=20;
>> x = zeros(n,1);
>> x(1) = 1;
>> for k=2:n
x(k) = 1- cos(x(k-1));
end
>> semilogy(abs(x),'b.-')
warning: axis: omitting non-positive data in log plot
```

Fixed Point: $x - (1 - \cos(x)) = 0$: $\text{V}\textsc{ery}$ fast convergence

$$
\begin{aligned}
g(x) &= 1 - \cos(x) \in [-1, 1] \quad \text{for} \quad x \in [-1, 1], \\
|g'(x)| &= |\sin x| \leq \sin 1.
\end{aligned}
$$

```
>>
>> n=20;
>> x = zeros(n,1);
>> x(1) = 1;
>> for k=2:n
x(k) = 1- cos(x(k-1));
end
>> semilogy(abs(x),'b.-')
warning: axis: omitting non-positive data in log plot
```



Fixed Point Theorem (I)

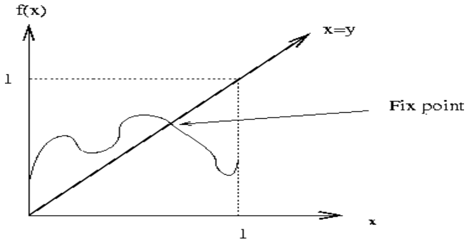
Theorem 2.3

(i) If $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$, then g has at least one fixed point in $[a,b]$.

(ii) If, in addition, $g'(x)$ exists on (a,b) and a positive constant $k < 1$ exists with

$$|g'(x)| \le k, \quad \text{for all } x \in (a,b),$$

then there is exactly one fixed point in $[a,b]$. (See Figure 2.4.) ■



Proof of Thm 2.3

- If $g(a) = a$ or $g(b) = b$,then g has a fixed point at an endpoint.
- Otherwise, $g(a) > a$ and $g(b) < b$. The function $h(x) = g(x) - x$ is continuous on $[a, b]$, with

$$h(a) = g(a) - a > 0 \quad \text{and} \quad h(b) = g(b) - b < 0.$$

- This implies that there exists $p \in (a, b)$, $h(p) = 0$.
- $g(p) - p = 0$, or $p = g(p)$.

If $|g'(x)| \leq k < 1$ for all x in (a, b), and p and q are two $\boxed{\text{distinct}}$ fixed points in $[a, b]$. Then a number ξ exists (Mean Value Theorem)

$$\frac{g(p) - g(q)}{p - q} = g'(\xi) < 1.$$

So

$$1 = \frac{p - q}{p - q} = \frac{g(p) - g(q)}{p - q} = g'(\xi) < 1.$$

Proof of Thm 2.3

- If $g(a) = a$ or $g(b) = b$,then g has a fixed point at an endpoint.
- Otherwise, $g(a) > a$ and $g(b) < b$. The function $h(x) = g(x) - x$ is continuous on $[a, b]$, with

$$h(a) = g(a) - a > 0 \quad \text{and} \quad h(b) = g(b) - b < 0.$$

- This implies that there exists $p \in (a, b)$, $h(p) = 0$.
- $g(p) - p = 0$, or $p = g(p)$.

If $|g'(x)| \leq k < 1$ for all x in (a, b), and p and q are two $\boxed{\text{distinct}}$ fixed points in $[a, b]$. Then a number ξ exists (Mean Value Theorem)

$$\frac{g(p) - g(q)}{p - q} = g'(\xi) < 1.$$

So

$$1 = \frac{p - q}{p - q} = \frac{g(p) - g(q)}{p - q} = g'(\xi) < 1. \boxed{\Longrightarrow \boxed{\text{distinct}} \Longleftarrow}$$

This contradiction implies uniqueness of fixed point.

Fixed Point Iteration

Given initial approximation p_0, define *Fixed Point Iteration*

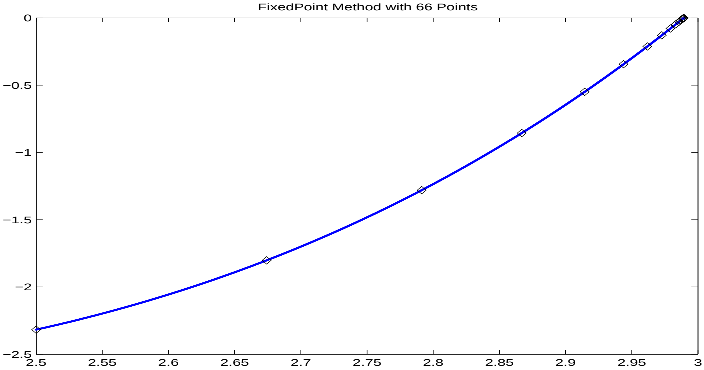
$$p_n = g(p_{n-1}), \quad n = 1, 2, \cdots,$$

If iteration converges to p, then

$$p = \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g(p).$$

Fixed Point Example $x - \log(2 + 2x^2) = 0$: normal convergence

$$
\begin{aligned}
g(x) &= \log(2 + 2x^2) \in [2, 3] \quad \text{for} \quad x \in [2, 3], \\
|g'(x)| &\leq \frac{4}{5} < 1.
\end{aligned}
$$



Fixed Point Theorem (II)

Theorem 2.4 **(Fixed-Point Theorem)**

Let $g \in C[a,b]$ be such that $g(x) \in [a,b]$, for all x in $[a,b]$. Suppose, in addition, that g' exists on (a,b) and that a constant $0 < k < 1$ exists with

$$|g'(x)| \leq k, \quad \text{for all } x \in (a,b).$$

Then for any number p_0 in $[a,b]$, the sequence defined by

$$p_n = g(p_{n-1}), \quad n \geq 1,$$

converges to the unique fixed point p in $[a,b]$. ∎

Fixed Point Theorem (II)

Theorem 2.4 (Fixed-Point Theorem)

Let $g \in C[a,b]$ be such that $g(x) \in [a,b]$, for all x in $[a,b]$. Suppose, in addition, that g' exists on (a,b) and that a constant $0 < k < 1$ exists with

$$|g'(x)| \leq k, \quad \text{for all } x \in (a,b).$$

Then for any number p_0 in $[a,b]$, the sequence defined by

$$p_n = g(p_{n-1}), \quad n \geq 1,$$

converges to the unique fixed point p in $[a,b]$. ∎

PRO: simple iteration

CON: conditions hard to verify

No algorithm for finding $[a, b]$

Proof of Thm 2.4

- A unique fixed point $p \in [a, b]$ exists.
- $$|p_n - p| = |g(p_{n-1}) - g(p)| = |g'(\xi_n)(p_{n-1} - p)| \le k|p_{n-1} - p|$$
- $$|p_n - p| \le k|p_{n-1} - p| \le k^2|p_{n-2} - p| \le \cdots \le k^n|p_0 - p|.$$
- Since
  $$\lim_{n\to\infty} k^n = 0,$$
  $\{p_n\}_{n=0}^{\infty}$ converges to p.

$\boxed{\text{No Harm Principle}}$ in numerical algorithm design

What we do not know never harms us

$\boxed{\text{No Harm Principle}}$ in numerical algorithm design

What we do not know never harms us

(NOT REALLY!!!)

$\boxed{\text{No Harm Principle}}$ in numerical algorithm design

What we do not know never harms us

(NOT REALLY!!!)

Trust but Verify