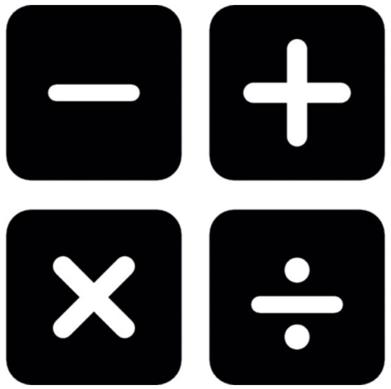
CPE 310: Numerical Analysis for Engineers

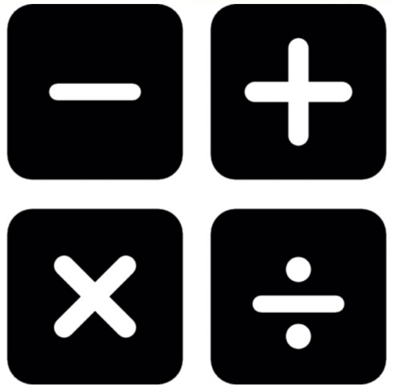
Chapter 0: Accuracy, Arithmetic, and Error

Ahmed Tamrawi

This is a course to teach you **computer algorithms** for analyzing and solving science and engineering problems in numerical ways

We will learn how a computer can be used to solve problems that **may not be solvable** by the techniques that are taught in most calculus courses



1 2 3 4 5
6 7 8 9 10


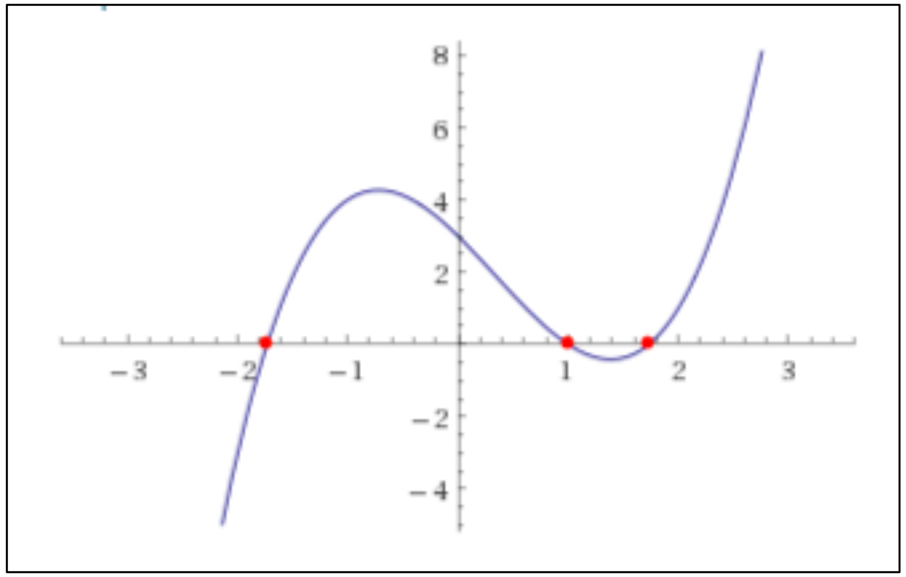
Analysis in Mathematics

Solve a problem through equations. The equations must then be reduced to an answer through the procedures of algebra, calculus, differential equations, partial differential equations, or the like.

Analysis in Mathematics

Solve a problem through equations. The equations must then be reduced to an answer through the procedures of algebra, calculus, differential equations, partial differential equations, or the like.

$$f(x) = x^3 - x^2 - 3x + 3$$



$$x^3 - x^2 - 3x + 3 = 0$$

$$(x^3 - x^2) + (-3x + 3) = 0$$

$$x^2(x-1) - 3(x-1) = 0$$

$$(x-1)(x^2-3) = 0$$

$$x = 1, \pm\sqrt{3}$$

Analytical Answer

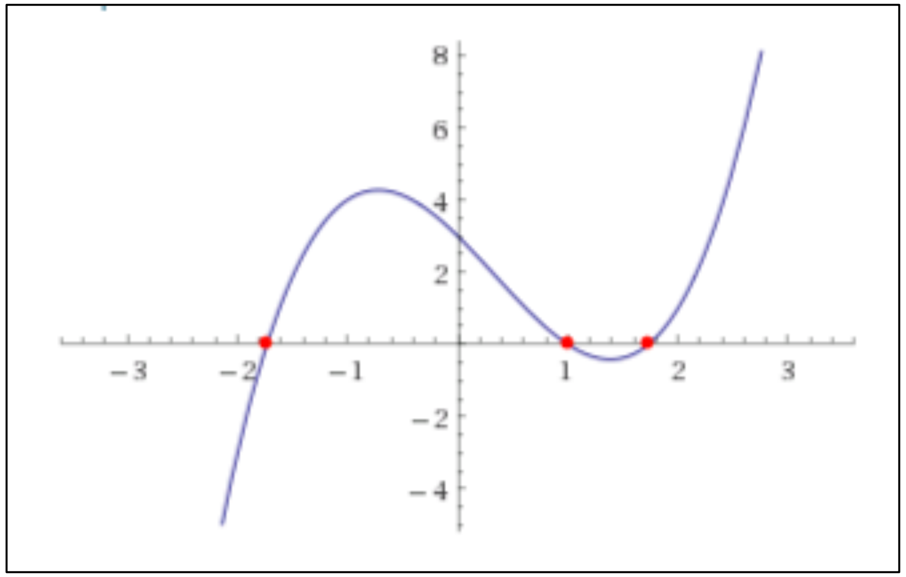
Numerical Analysis

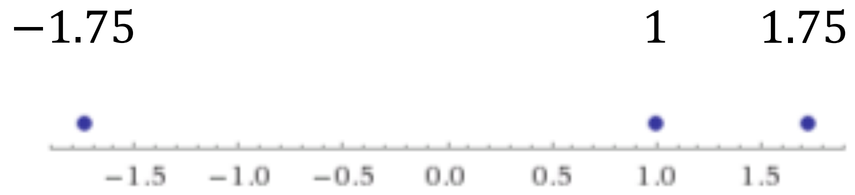
The only procedures to solve the problem are arithmetic: add, subtract, multiply, divide, and compare. Since these operations are exactly those that computers can do

Numerical Analysis

The only procedures to solve the problem are arithmetic: add, subtract, multiply, divide, and compare. Since these operations are exactly those that computers can do

$$f(x) = x^3 - x^2 - 3x + 3$$





$$\sqrt{3} = 1.73205080757$$

Analytical Answer

Numerical Answer

Analytical Answer

The equations reduced to an answer through the procedures of algebra, calculus, differential equations, partial differential equations, or the like.

Numerical Answer

The only procedures that are used are arithmetic: add, subtract, multiply, divide, and compare

"Always an Approximation"

While the numerical result is an approximation, this can usually ***be as accurate as needed****. The necessary accuracy is, of course,* ***determined by the application****.*

$$\boxed{\sqrt{2}}$$

$$\approx$$

$$\boxed{1.414}$$

$$\boxed{1.41421}$$

$$\boxed{1.41421356237}$$

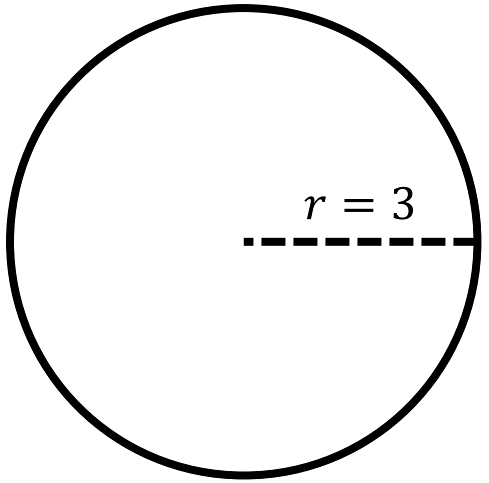
Analytical Answer

The equations reduced to an answer through the procedures of algebra, calculus, differential equations, partial differential equations, or the like.

Numerical Answer

The only procedures that are used are arithmetic: add, subtract, multiply, divide, and compare

"Always an Approximation"



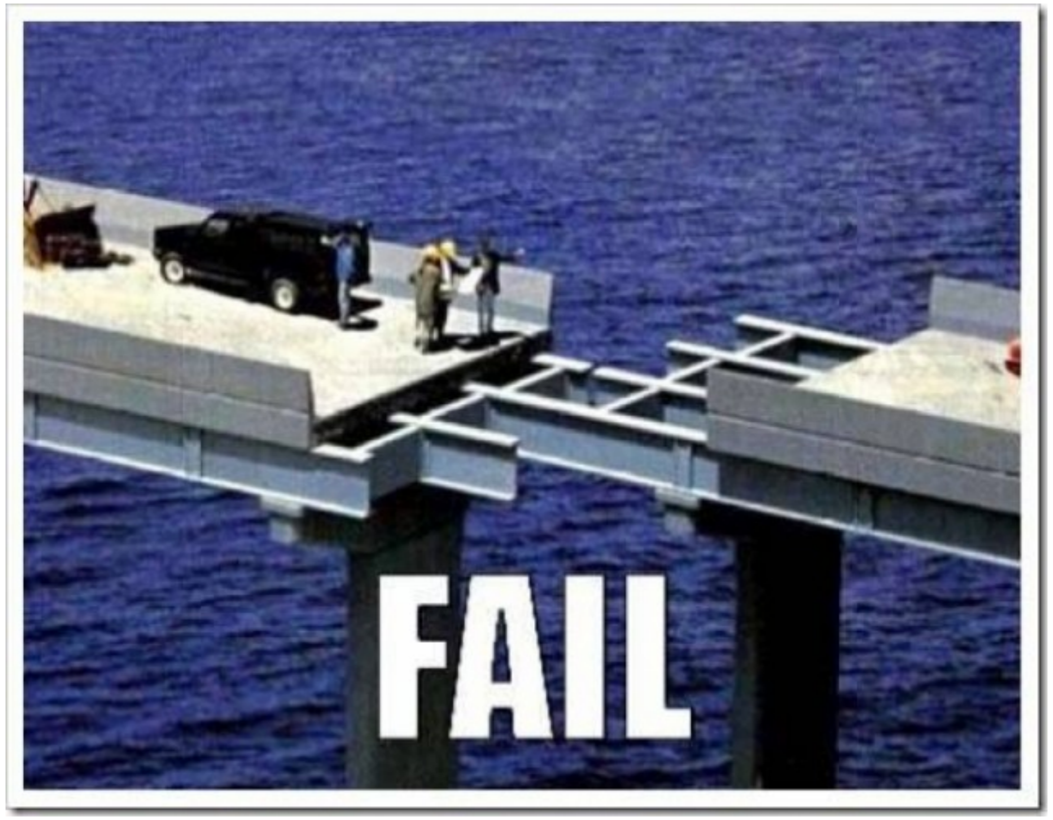
Area = 9π

$$(3.14)(3)^2 \approx \boxed{28.26}$$

$$(3.141592)(3)^2 \approx \boxed{28.274328}$$

$$(3.14159265359)(3)^2 \approx \boxed{28.2743338823}$$

Numerical methods require such **tedious and repetitive arithmetic operations** that only when we have a computer to carry out these many separate operations it is practical to solve problems in this way

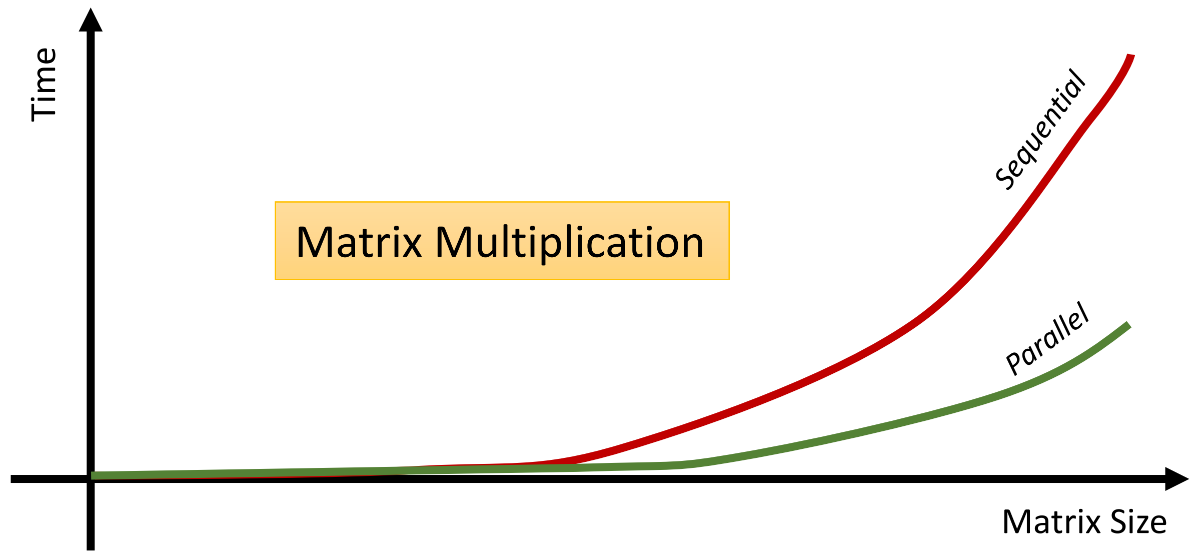


Why not Ask Human to do it?

A human would make so many mistakes that there would be little confidence in the result

The manpower cost would be more than could normally be afforded

One important trend in computer operations is the use of several processors working in parallel to carry out procedures with greater speed than can be obtained with a single processor

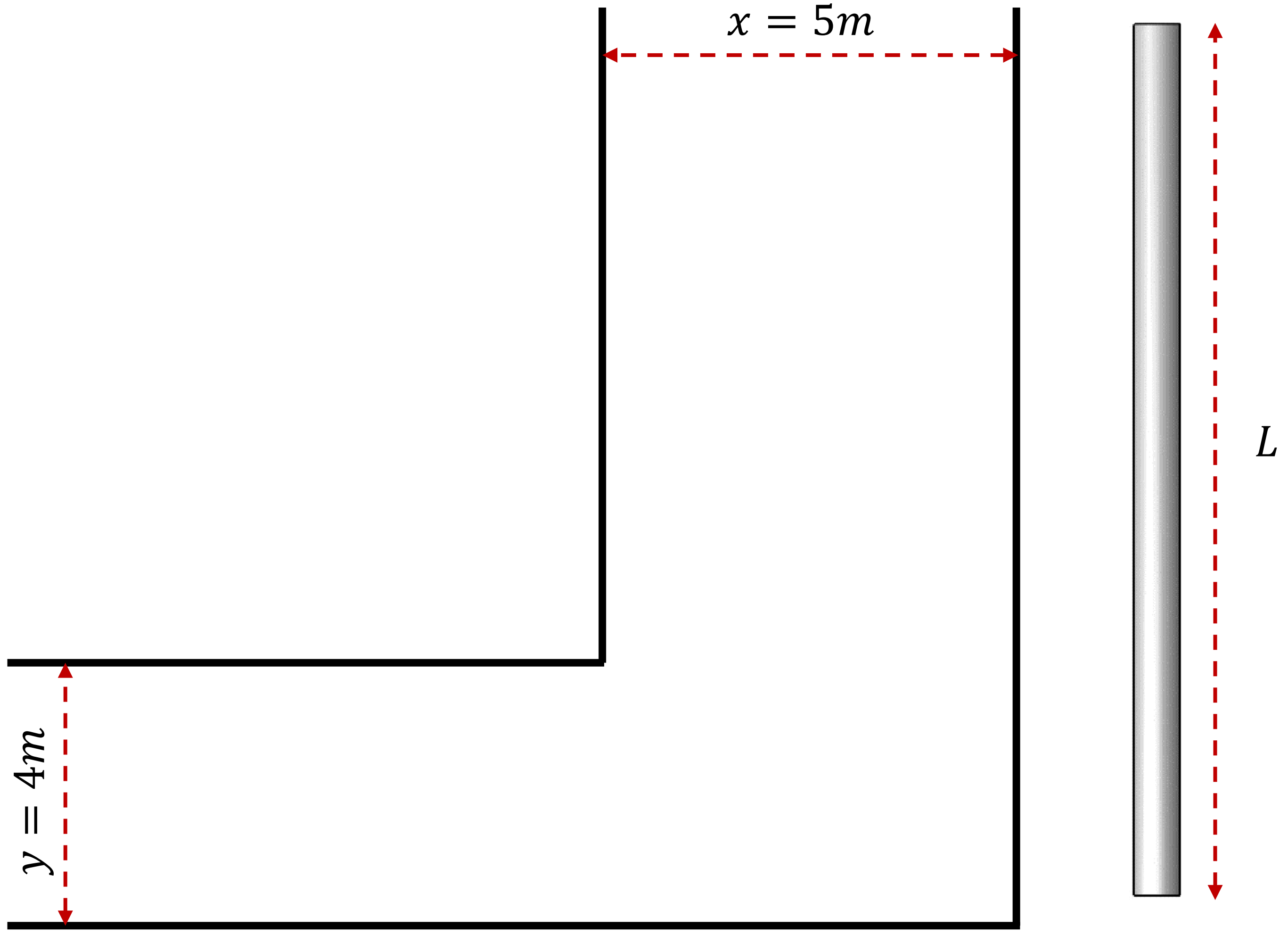


MATLAB

The Language of Technical Computing

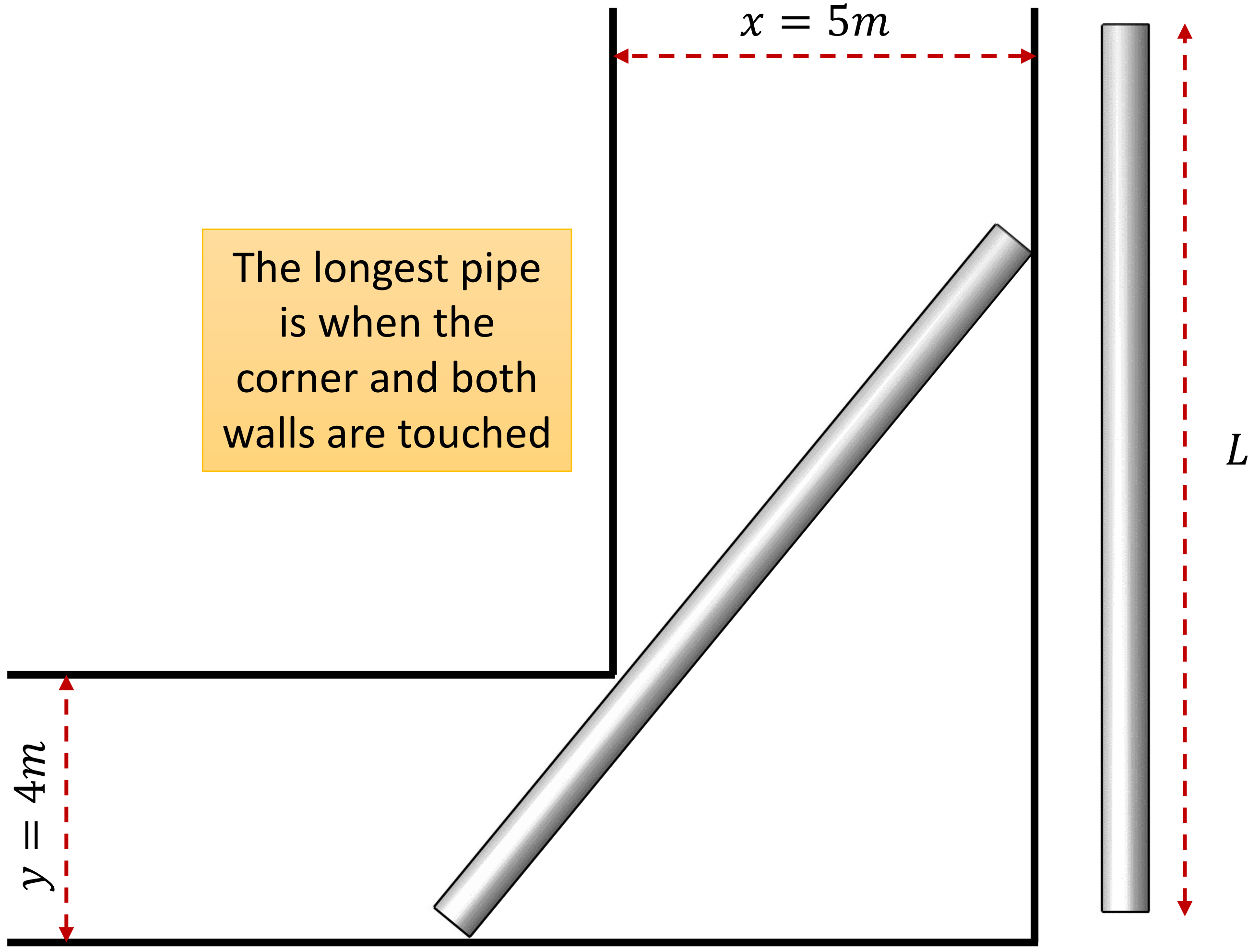
We will learn Matlab to code the algorithms we learn in this course

A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.



What is the longest pipe that can be pushed into the 4-m corridor?

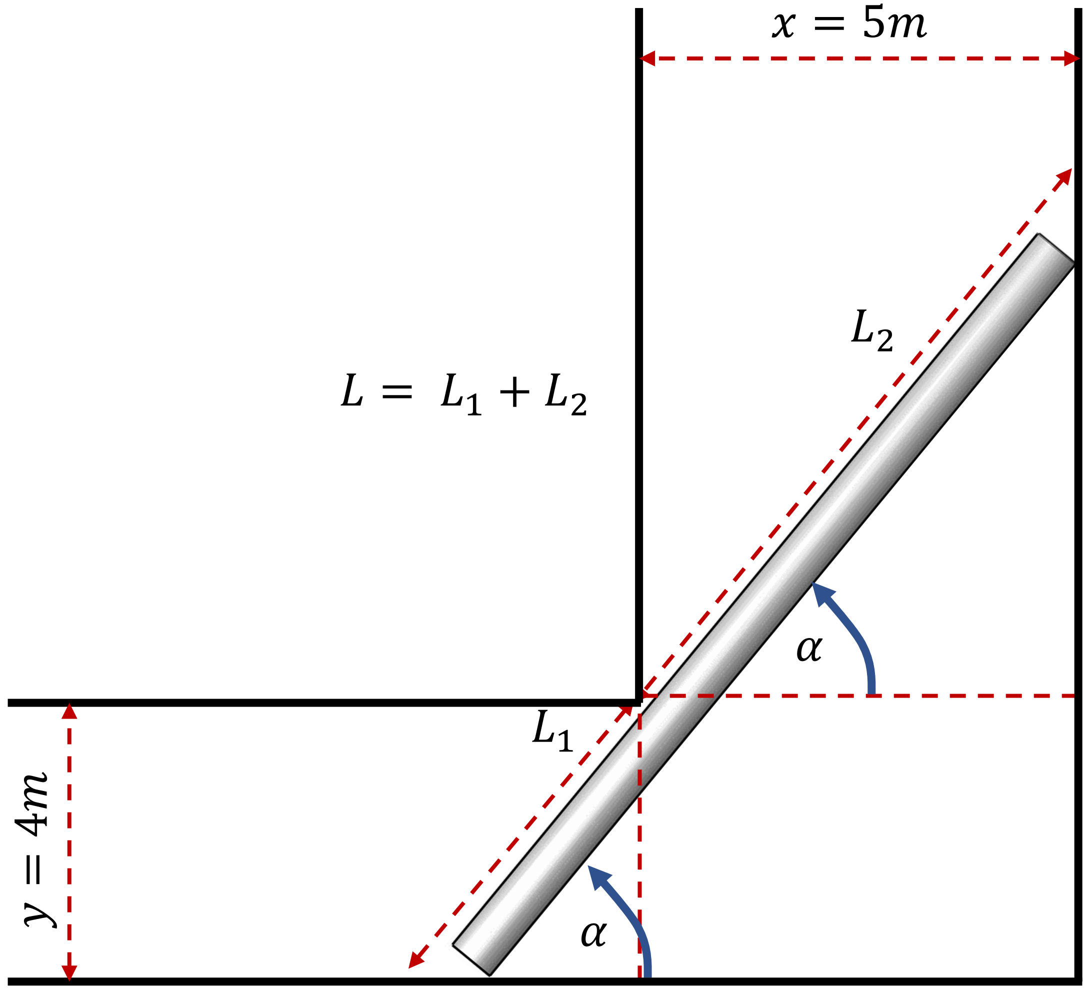
A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.



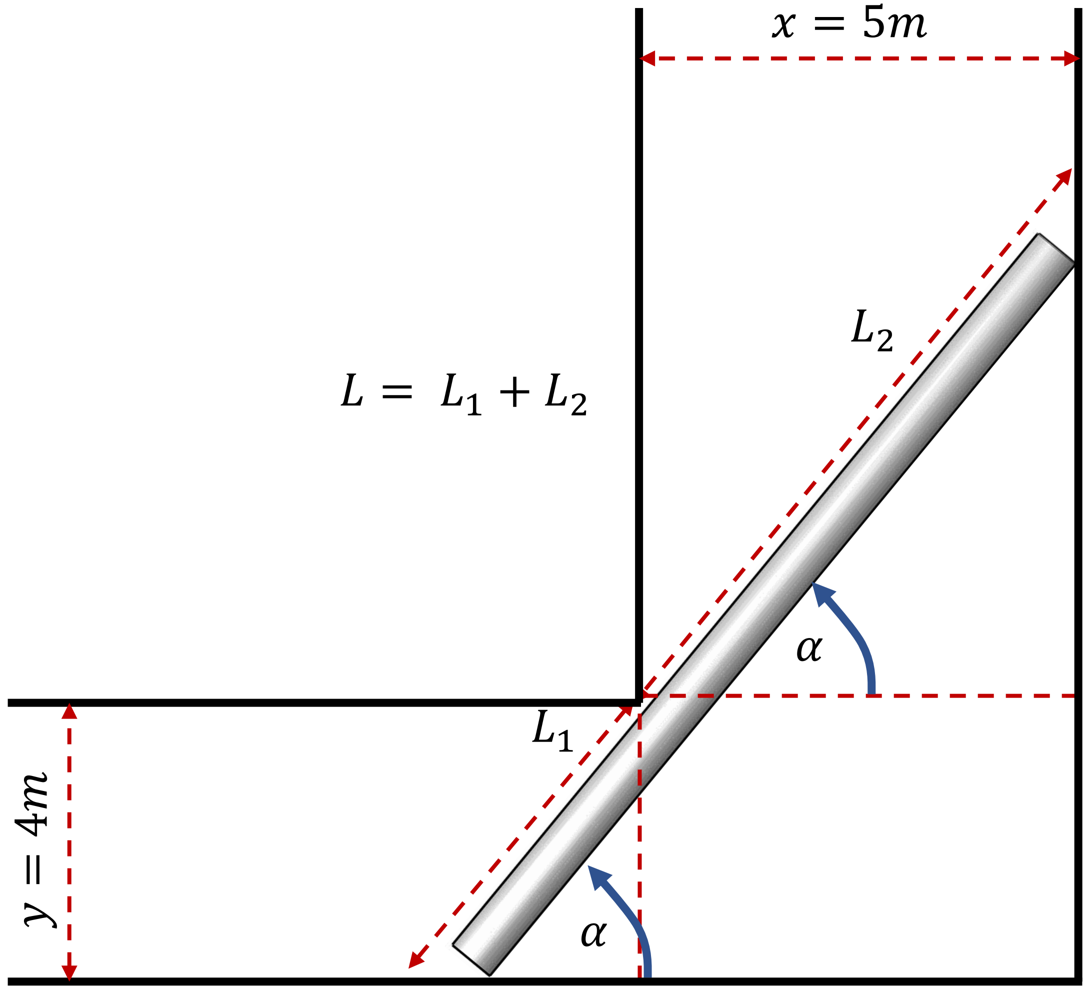
The longest pipe is when the corner and both walls are touched

What is the longest pipe that can be pushed into the 4-m corridor?

A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.



A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.

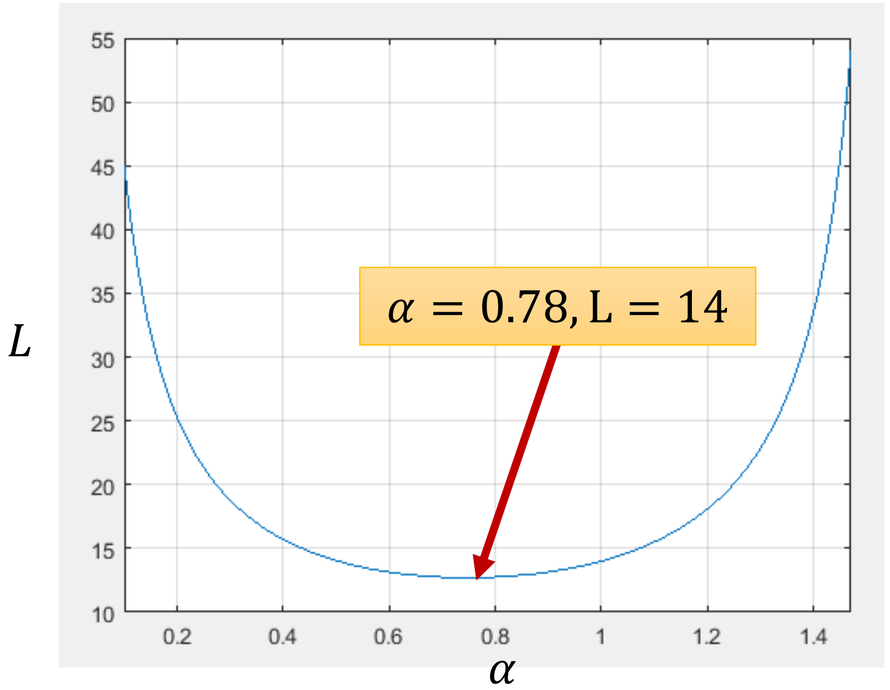


$$L_1 = \frac{y}{\sin\alpha} \qquad L_2 = \frac{x}{\cos\alpha}$$

$$L = L_1 + L_2$$

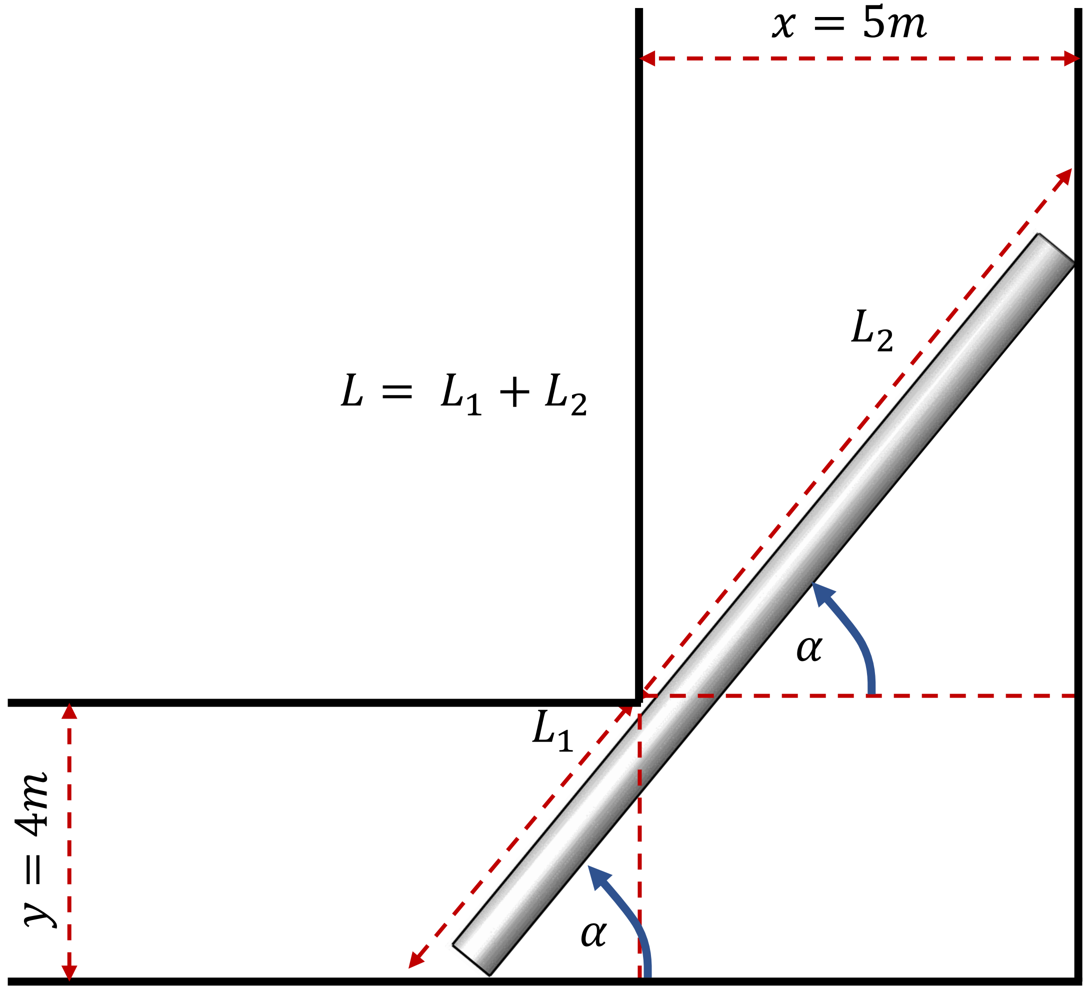
$$L = \frac{y}{\sin\alpha} + \frac{x}{\cos\alpha}$$

$$L = \frac{4}{\sin\alpha} + \frac{5}{\cos\alpha}$$



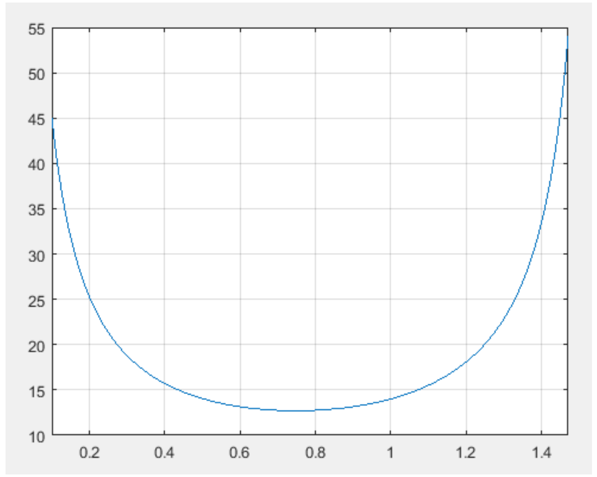
A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.

$$L = \frac{4}{\sin\alpha} + \frac{5}{\cos\alpha}$$

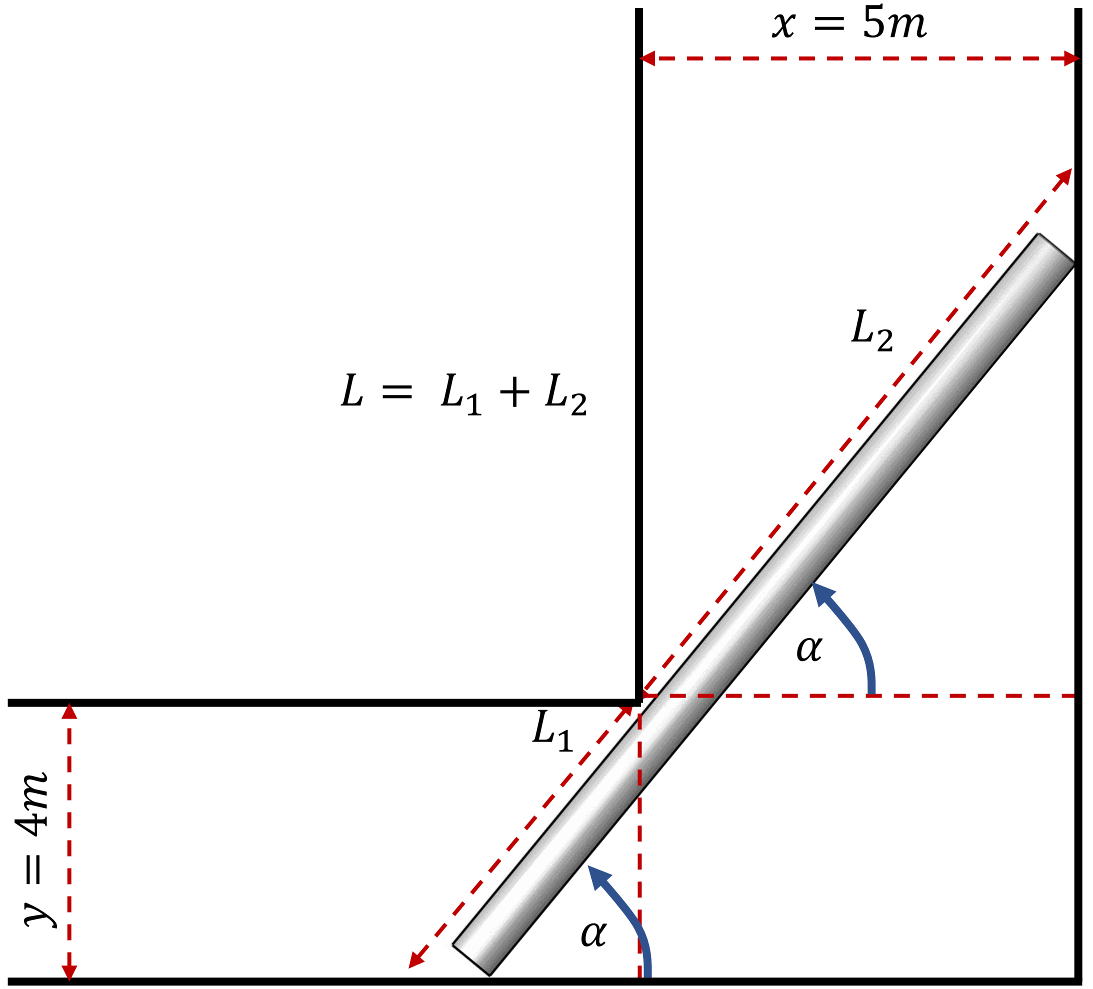


MATLAB SOLUTION:

```
>> L = inline('4/sin(a) + 5/cos(a)')

L =

      Inline Function:
      L(a) = 4/sin(a) + 5/cos(a)

>> fplot(L, [0.1, pi/2 – 0.1]); grid on
```



A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.

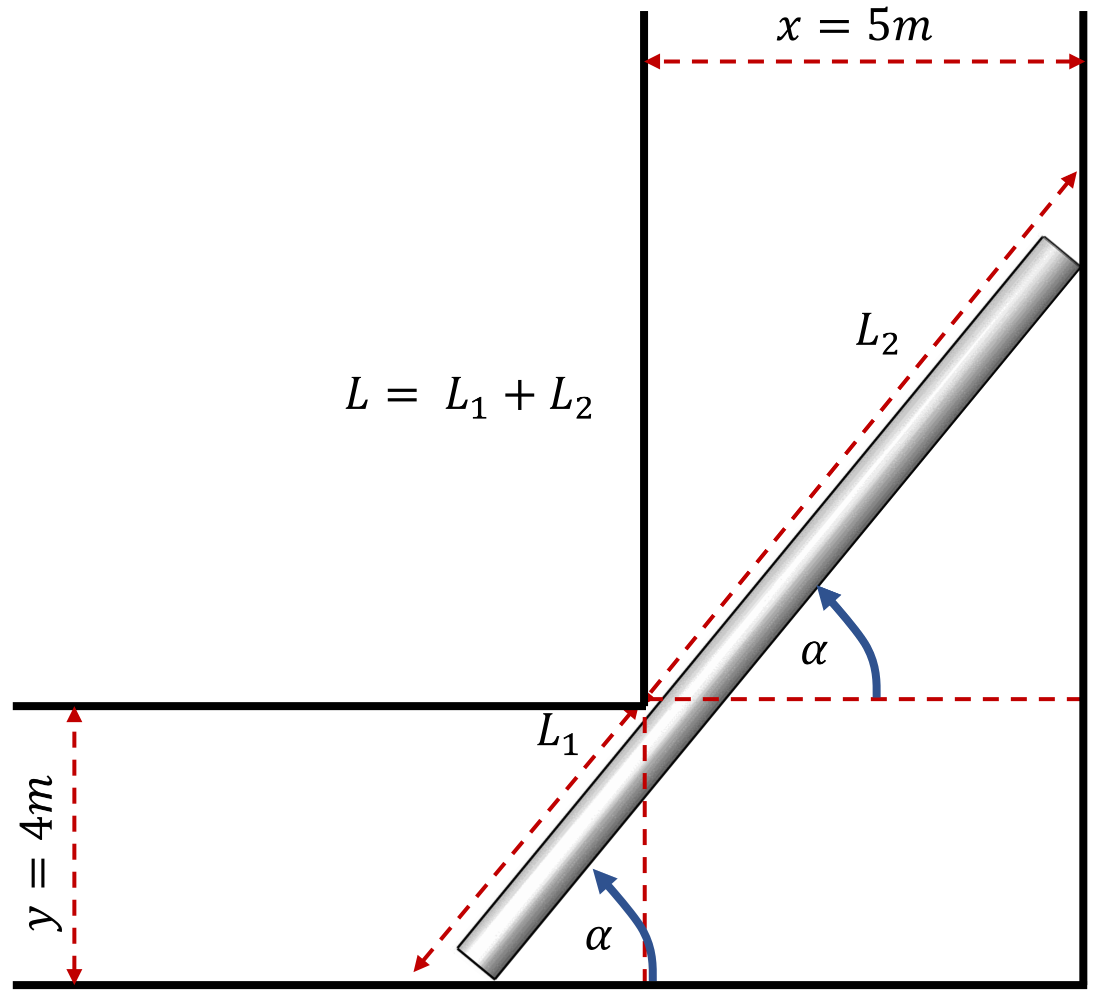


$$L = \frac{4}{\sin\alpha} + \frac{5}{\cos\alpha}$$

MATLAB SOLUTION:

```
>> L = inline('4/sin(a) + 5/cos(a)')

L =
       Inline Function:
       L(a) = 4/sin(a) + 5/cos(a)

>> fplot(L, [0.1, pi/2 - 0.1]); grid on
>> angle = fminbond(L, [0.1, pi/2 - 0.1])

angle =
       0.7482

>> L(angle)

ans =
       12.7017
```

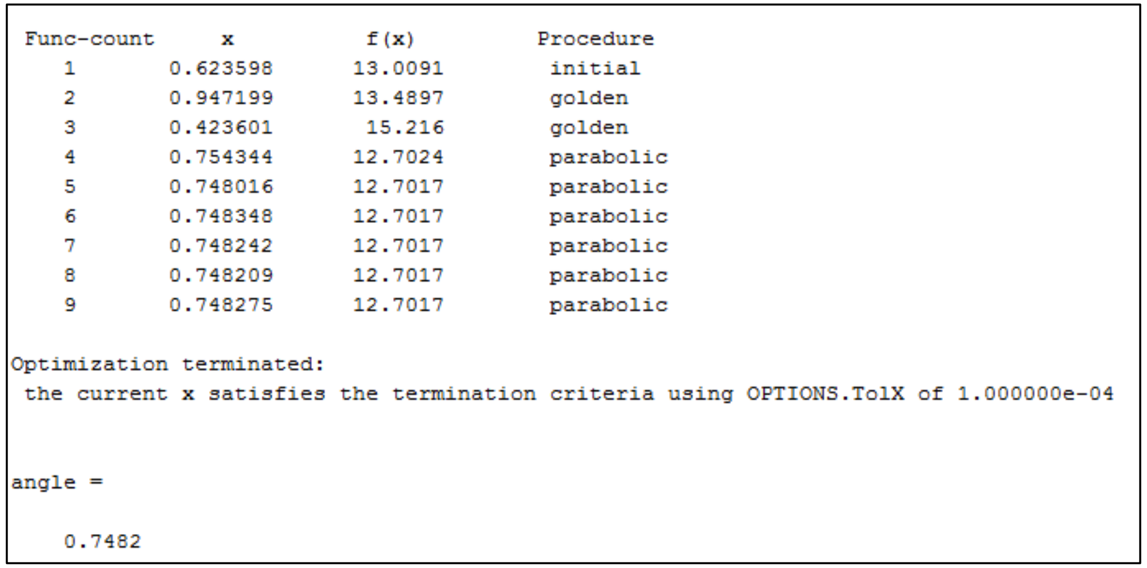
A metal pipe L-m long is pushed on the floor from a corridor 5-m wide into another corridor 4-m wide and at a right angle.



$$L = \frac{4}{\sin\alpha} + \frac{5}{\cos\alpha}$$

MATLAB SOLUTION:

```
>> L = inline('4/sin(a) + 5/cos(a)')
L =
         Inline Function:
         L(a) = 4/sin(a) + 5/cos(a)

>> fplot(L, [0.1, pi/2 - 0.1]); grid on
>> angle = fminbond(L, [0.1, pi/2 - 0.1],
optimset('Display', 'Iter'))
```



```
 Func-count     x          f(x)         Procedure
    1        0.623598    13.0091         initial
    2        0.947199    13.4897         golden
    3        0.423601     15.216         golden
    4        0.754344    12.7024         parabolic
    5        0.748016    12.7017         parabolic
    6        0.748348    12.7017         parabolic
    7        0.748242    12.7017         parabolic
    8        0.748209    12.7017         parabolic
    9        0.748275    12.7017         parabolic

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04


angle =

    0.7482
```

There is more than one way to attack the problem

There are prewritten programs that can help

The accuracy that is needed in the answer dictates how you should get the solution

It is critically important to realize that **errors** can occur in doing *numerical procedures*

Errors Original Data

Blunders

Truncation Error

Propagated Error

Round-Off Error

Error in Original Data

Real-world problems (e.g., Physical System) is modeled by a mathematical equation, will nearly always **have coefficients that are imperfectly known**.

The reason is that the problems often depend on measurements of doubtful accuracy OR the model itself may not reflect the behavior of the situation perfectly

Blunders *a stupid or careless mistake.*

We always use a computer or programmable calculators in numerical analysis. Such machines make mistakes **very infrequently**, but because computers and calculators are programmed by humans blunders or gross errors do occur more frequently than we like to admit.

Running test cases with known results, but it is no guarantee of freedom from foolish error

Truncation Errors

The error made by truncating an infinite sum and approximating it by a finite sum

$$\boxed{e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}} \xrightarrow{\hspace{3em}} \boxed{e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}}$$

Infinite Sum Series *Finite Sum Series*

Life is finite and computer time is costly, we must be satisfied with an approximation to the exact analytical answer.

Propagated Error

The error in the succeeding steps of a process due to an occurrence of an earlier error

Step 1

Step 2

Step 3

Step 4

Step *n*

Unstable Propagated Error
Error magnified continuously as the method continues, eventually overshadowing/destroying the true value

Stable Propagated Error
Error made at early points die out as the method continues

Round-Off Error

Difference between result produced by given algorithm using exact arithmetic and result produced by same algorithm using limited precision arithmetic

Due to inexact representation of real numbers and arithmetic operations upon them

$$\frac{1}{6} \approx 1.67$$

$$\frac{1}{6} \approx 1.666667$$

$$\frac{1}{6} \approx 1.666666666666666666666666666666667$$

Computing Error

The accuracy of any computation is always of great importance. There are two common ways to express the size of the error in a computed result

$$\text{Absolute Error} = |\text{True Value} - \text{Approximate Value}|$$

Absolute error is usually more serious when the magnitude of the true value is small as its dependent on the scale of the value

$$\text{Relative Error} = \frac{\text{Absolute Error}}{|\text{True Value}|}$$

The relative error is more independent of the scale of the value

$1{,}036.52 \pm 0.010$

0.005 ± 0.010

Computing Error

The accuracy of any computation is always of great importance. There are two common ways to express the size of the error in a computed result

Absolute Error = |True Value – Approximate Value|

Absolute error is usually more serious when the magnitude of the true value is small as its dependent on the scale of the value

$$\text{Relative Error} = \frac{\text{Absolute Error}}{|\text{True Value}|}$$

The relative error is more independent of the scale of the value

Example

True Value = $\frac{10}{3}$ **Approximate Value** = 3.333

Absolute Error = $\left|\frac{10}{3} - 3.333\right| = 0.00033333 = \frac{1}{3000}$

Relative Error = $\frac{1}{3000}\Big/\frac{10}{3} = \frac{1}{10000}$

Floating-Point Arithmetic

Even though a computer follows exactly the instructions that it is given, when it performs an arithmetic operation it does not get exact answers unless only integers or exact powers of 2 are involved.

A computer stores numbers as floating-point quantities that resemble scientific notation.

13.524 is stored as a floating-point number $.13524 \times 10^2$

.13524E2

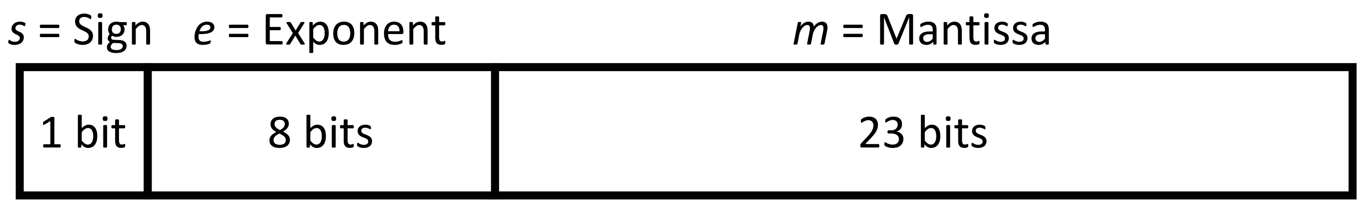
-0.0442 is stored as a floating-point number $.442 \times 10^{-1}$

.442E-1

IEEE 754 Floating Point standard is by far the most common

http://www.c-jump.com/bcc/common/Talk2/Cxx/IEEE_754_fp_standard/IEEE_754_fp_standard.html

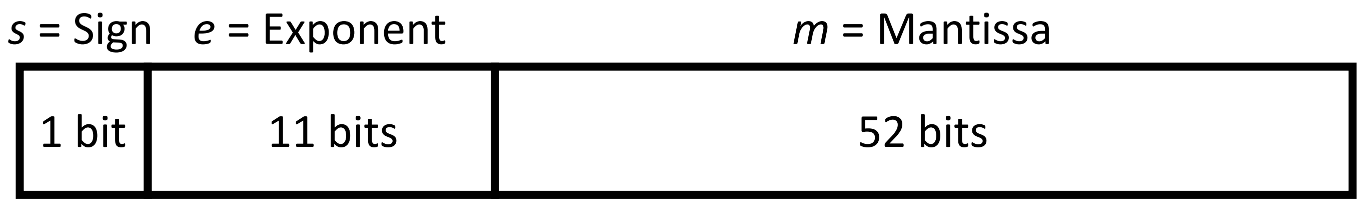
Single Precision



Number = $(-1)^s \times (1.m) \times 2^{(e-127)}$

Range = $10^{\pm 38}$

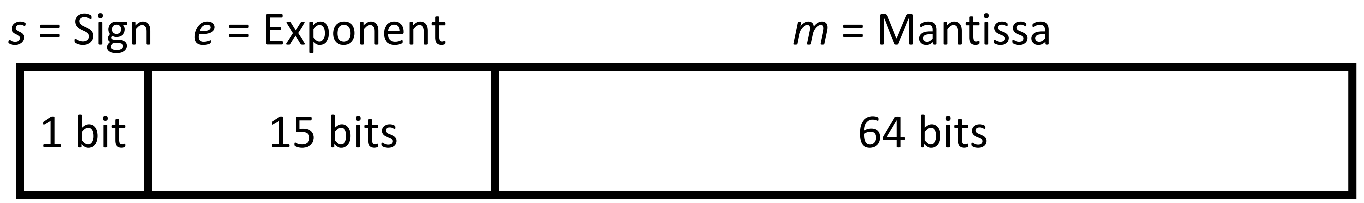
Double Precision



Number = $(-1)^s \times (1.m) \times 2^{(e-1023)}$

Range = $10^{\pm 308}$

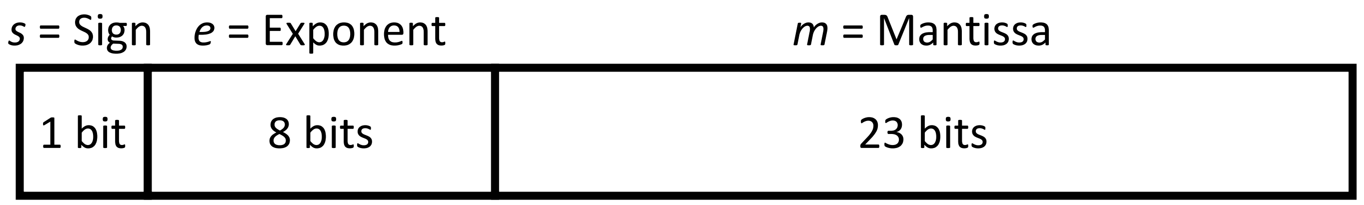
Extended Precision



Number = $(-1)^s \times (1.m) \times 2^{(e-16383)}$

Range = $10^{\pm 4931}$

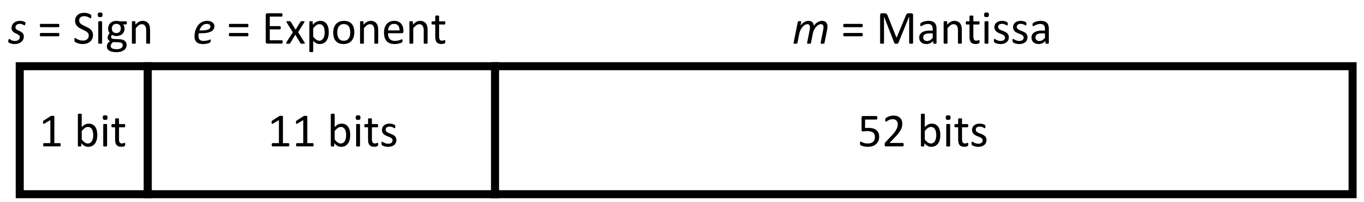
Single Precision



Number = $(-1)^s \times (1.m) \times 2^{(e-127)}$

Range = $10^{\pm 38}$

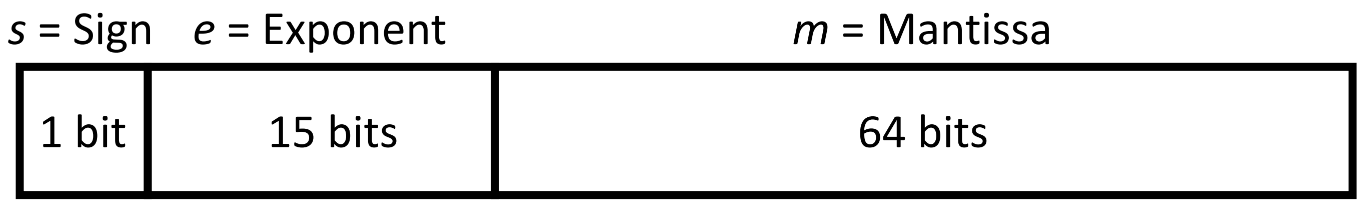
Double Precision



Number = $(-1)^s \times (1.m) \times 2^{(e-1023)}$

Range = $10^{\pm 308}$

Extended Precision

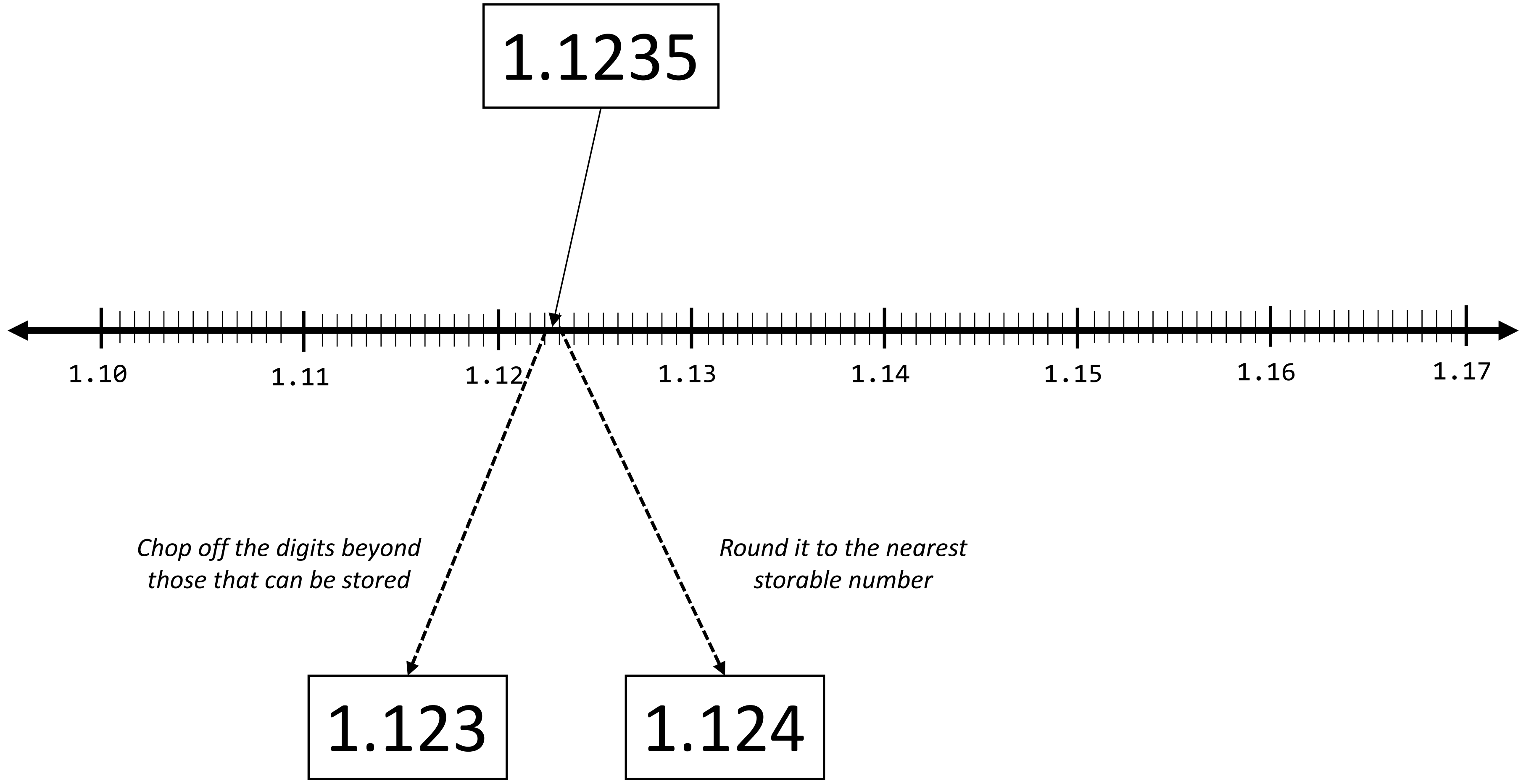


Number = $(-1)^s \times (1.m) \times 2^{(e-16383)}$

Range = $10^{\pm 4931}$

Since there is an **infinite number of real numbers**, it is clear that there **must be gaps between the stored values**.

This is the source of round-off error


1.1235
1.10
1.11
1.12
1.13
1.14
1.15
1.16
1.17
Chop off the digits beyond those that can be stored
Round it to the nearest storable number
1.123
1.124

Floating-Point Numbers Comparison for Equality

Floating point math is not exact

Slight changes in the order of operations or the precision of intermediates can change the floating-point arithmetic results.

Comparing two floats to see if they are equal is **usually not what you want**. GCC has a warning: “`warning: comparing floating point with == or != is unsafe`”.

C Example:

```
float f = 0.1f;
float sum;
sum = 0;
for (int i = 0; i < 10; ++i){
    sum += f;
}
float product = f * 10;
printf("sum = %1.15f\n", sum);
printf("mul = %1.15f\n", product);
printf("mul2 = %1.15f\n", f * 10);
```

```
sum= 1.000000119209290
mul= 1.000000000000000
mul2= 1.000000014901161
```

https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/

Floating-Point Numbers Comparison for Equality

Floating point math is not exact

✗

```
if(A == B){
        // some code
}else{
        // some code
}
```

```
if(|A – B| <= tolerance){
        // some code
}else{
        // some code
}
```

✓

Integer data type is 32 bit in Java

`Integer.MAX_VALUE = 2147483647`

`Integer.MIN_VALUE = -2147483648`

`2147483650 > Integer.MAX_VALUE`

*Causes **overflow***

`-2147483650 < Integer.MIN_VALUE`

*Causes **underflow***

In Java, any value that surpasses `Integer.MAX_VALUE` gets rolled over

`Integer.MAX_VALUE + 1 = Integer.MIN_VALUE`

In Java, any value that surpasses `Integer.MIN_VALUE` gets rolled over.

`Integer.MIN_VALUE – 1 = Integer.MAX_VALUE`

https://dzone.com/articles/overflow-and-underflow-data

Epsilon (ε)

It represents the **smallest machine value** that can be added to 1.0 that gives a result distinguishable from 1.0

MATLAB:

```
>> eps

ans =
       2.2204e-016
```

$$\boxed{\alpha < \varepsilon}$$

$$(1+\alpha)+\alpha = 1$$

$$1+(\alpha+\alpha) > 1$$

Round-off Error Versus Truncation Error

Multiple errors can affect the accuracy and precision

Round-off error occurs, even when the procedure is exact, due to the imperfect precision of the computer

Truncation error is caused by using a procedure that does not give precise results even though the arithmetic is precise.

$$\boxed{f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}}$$

Depending on the precision of the computer, round-off errors will dominate and the results become less exact.

Finding an approximate value for $f'(x)$ by using a small value for *h*. The smaller *h* is, the result is closer to the true value for the derivative (*the truncation error is reduced*)

Computational Error = Round-Off Error + Truncation Error

The accuracy of a numerical solution depends not only on the computer's accuracy; **the problem itself is a factor**

Computing Surface Area of Earth using $V = 4\pi r^2$

- Earth is modeled as **sphere**, idealizing its true shape
- Value for radius is based on **empirical measurements and previous computations**
- Value for π requires **truncating infinite process**
- Values for input data and **results of arithmetic operations are rounded** in computer