# Lecture 1

## Introduction to Numerical Methods

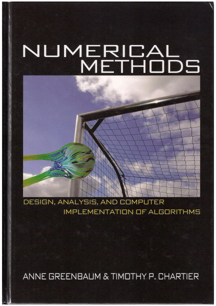
T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

June 16, 2014

# Course Info

http://courses.engr.illinois.edu/cs357/su2014/




- Book: Numerical Methods Design, Analysis, and Computer Implementation of Algorithms, by Anne Greenbaum & Timothy P. Chartier

Contact Information:

- Office: 2209 SC
- Office Hour: 1:00-2:00 PM Monday or by appointment
- email: gambill AT illinois DOT edu

- TA:
  Sweta Yamini Seethamaraju
  email: ta1cs357 AT cs DOT illinois DOT edu
- Office Hour: tbd
- Office Hour Location: 209 Siebel Center

# Assessment

`http://courses.engr.illinois.edu/cs357/su2013/`

Grades:

| | |
|---|---|
| Compass quizzes | 150 points maximum |
| MPs | 150 points maximum |
| Midterm Exam | 150 points |
| Final Exam | 250 points |

Homework:

- no dropped scores
- May discuss MPs with TAs or other students, but **do not** copy!
  - copied partial solutions is still copying
  - see departmental policy re: cheating ( `https://agora.cs.illinois.edu/display/undergradProg/Honor+Code`)

# Finally. . .

`http://courses.engr.illinois.edu/cs357/su2013/`

Schedule and Notes:

- Final exam is on Friday August $8^{th}$ time and location: TBA

Questions?

## Topics

- Section 1: What is Numerical Analysis / Numerical Methods?
- Section 2: Numbers
- Section 3: IEEE754
- Section 4: Errors-Data, Roundoff,Truncation,machine epsilon($\epsilon_m$),unit roundoff($\mu$)
- Section 5: Operations on Numbers
- Section 6: Functions

# Section 1: What is Numerical Analysis / Numerical Methods?

## Definition

Numerical Analysis - The study of algorithms (methods) for problems involving quantities that take on continuous (as opposed to discrete) values.

# Numerical Calculation vs. Symbolic Calculation

- Numerical Calculation: involve numbers directly
  - manipulate numbers to produce a **numerical** result
- Symbolic Calculation: symbols represent numbers
  - manipulate symbols according to mathematical rules to produce a symbolic result

**Example (numerical)**

$$\frac{(17.36)^2 - 1}{17.36 + 1} = 16.36$$

**Example (symbolic)**

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

# Analytic Solution vs. Numerical Solution

- Analytic Solution (a.k.a. symbolic): The exact numerical or symbolic representation of the solution
  - may use special characters such as $\pi$, $e$, or $\tan(83)$
- Numerical Solution: The computational representation of the solution
  - entirely numerical

**Example (analytic)**

$$\frac{1}{4}$$

$$\frac{1}{3}$$

$$\pi$$

$$\tan(83)$$

**Example (numerical)**

$$0.25$$

$$0.33333\ldots(?)$$

$$3.14159\ldots(?)$$

$$0.88472\ldots(?)$$

# Numerical Computation and Approximation

- Numerical Approximation is needed to carry out the steps in the numerical calculation. The overall process is a numerical computation.

**Example (symbolic computation, numerical solution)**

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} - 1 = \frac{1}{12} = 0.083333333\ldots$$

**Example (numerical computation, numerical approximation)**

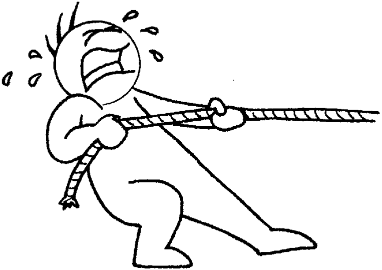
$$0.500 + 0.333 + 0.250 - 1.000 = 0.083$$

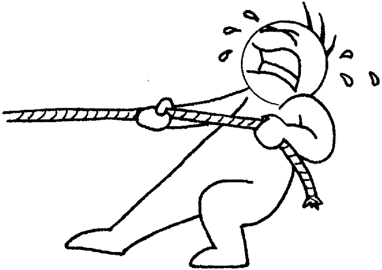
# Method vs. Algorithm vs. Implementation

- Method: a general (mathematical) framework describing the solution process
- Algorithm: a detailed description of executing the method
- Implementation: a particular instantiation of the algorithm

- Is it a “good” method?
- Is it a robust (stable) algorithm?
- Is it a fast implementation?

# The Big Theme





Accuracy

Cost

# Numerical Analysis

**Definition (Trefethen)**

Study of algorithms for the problems of continuous mathematics

We've been doing this since Calculus (and before!)

- Riemann sum for calculating a definite integral
- Newton's Method
- Taylor's Series expansion + truncation

# History: Numerical Algorithms

- date to 1650 BCE: The Rhind Papyrus of ancient Egypt contains 85 problems; many use numerical algorithms (T. Chartier, Davidson)

- Approximates $\pi$ with $(8/9)^2 * 4 \approx 3.1605$

# History: Archimedes

- 287-212BC developed the ”Method of Exhaustion”
- Method for determining $\pi$
  - find the length of the permieter of a polygon inscribed inside a circle of radius $1/2$
  - find the permiter of a polygon circumscribed outside a circle of radius $1/2$
  - the value of $\pi$ is between these two lengths

# History: Method of Exhaustion

- A circle is not a polygon
- A circle **is** a polygon with an infinite number of sides
- $C_n$ = circumference of an n-sided polygon inscribed in a circle of radius $1/2$
- $\lim_{n\to\infty} = \pi$
- Archimedes deterimined

$$\frac{223}{71} < \pi < \frac{22}{7}$$
$$3.1408 < \pi < 3.1429$$

- two places of accuracy....
- `http://www.pbs.org/wgbh/nova/archimedes/pi.html`

# History: Method of Machin

- Around 1700, John Machin discovered the trig identity

$$\pi = 16 \arctan\left(\frac{1}{5}\right) - 4 \arctan\left(\frac{1}{239}\right)$$

- Led to calculation of the first 100 digits of $\pi$
- Uses the Taylor series of $\arctan$ in the algorithm

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \ldots$$

- Used until 1973 to find the first Million digits

# Big Questions

- How algorithms work and how they fail
- Why algorithms work and why they fail

- Connects mathematics and computer science
- Need mathematical theory, computer programming, and scientific inquiry

# Numerical Analysis

A Numerical Analyst needs

- computational knowledge (e.g. programming skills)
- understanding of the application (physical intuition for validation)
- mathematical ability to construct and meaningful algorithm

# Numerical Analysis

Numerical focus:

- **Approximation** An approximate solution is sought. How close is this to the desired solution?
- **Efficiency** How fast and cheap (memory) can we compute a solution?
- **Stability** Is the solution sensitive to small variations in the problem setup?
- **Error** What is the role of finite precision of our computers?

Why?

- Numerical methods improve scientific simulation
- Some disasters attributable to bad numerical computing (Douglas Arnold)
    - The Patriot Missile failure, in Dharan, Saudi Arabia, on February 25, 1991 which resulted in 28 deaths, is ultimately attributable to poor handling of rounding errors.
    - The explosion of the Ariane 5 rocket just after lift-off on its maiden voyage off French Guiana, on June 4, 1996, was ultimately the consequence of a simple overflow.
    - The sinking of the Sleipner A offshore platform in Gandsfjorden near Stavanger, Norway, on August 23, 1991, resulted in a loss of nearly one billion dollars. It was found to be the result of inaccurate finite element analysis.

## Sets of Numbers

- Natural Numbers = $\mathbb{N} = \{1, 2, 3, ...\}$
- Integers = $\mathbb{Z} = \{0, \pm 1, \pm 2, \pm 3, ...\}$
- Rationals = $\mathbb{Q} = \{a/b \mid a \in \mathbb{Z},\ b \in \mathbb{Z},\ b \neq 0\}$
- Reals = $\mathbb{R} = \{\pm d_n d_{n-1} ... d_2 d_1 . d_{-1} d_{-2} ... \mid n \in \mathbb{N}$ and $d_j \in \{0, 1, 2 ..., 9\},\ j = n, n-1, n-2, ..., 1, -1, -2, ...\}$
- n-tuples of Reals = $\mathbb{R}^n = \{(r_1, r_2, ..., r_n) \mid r_i \in \mathbb{R}$ and $n \in \mathbb{N}\}$
- Complex Numbers = $\mathbb{C} = \{(a, b) = a + bi \mid a \in \mathbb{R},\ b \in \mathbb{R},\ \ i^2 = -1\}$
- Extended Reals = $\overline{\mathbb{R}} = \mathbb{R} \cup \pm\infty$.
- Interval Numbers = $\mathbb{IR} = \{[a, b] \mid a \leqslant b,\ a \in \mathbb{R}\ ,\ b \in \mathbb{R}\}$

## Computer Representation of Real Numbers

- Normalized Floating Point Numbers
  $F(\beta, t, L, U) =$
  $\{\pm\beta^e(d_0.d_1d_2...d_{t-1}) \mid 0 \leqslant d_i \leqslant \beta - 1, i = 0, ..., t-1, \;\; L \leqslant e \leqslant U\}$
- $\beta$ is called the ”base”
- t is called the ”precision”
- L,U represent the range of the exponent
- $d_0.d_1d_2...d_{t-1}$ is called the mantissa or significand
- if $d_0 = 0$ then the number is zero (Normalization)
- e is called the exponent

## Computer Representation of Real Numbers

- (Toy)Normalized Binary Floating Point Numbers
  $F(10, 2, -1, 1)$
  $= \{\pm 10^e (d_0.d_1) \mid 0 \leqslant d_i \leqslant 9, i = 0, 1, \quad -1 \leqslant e \leqslant 1\}$
  $= \{\pm 10^e (d_0 + d_1/10) \mid 0 \leqslant d_i \leqslant 9, i = 0, 1, \quad -1 \leqslant e \leqslant 1, \ d_0 =$
  $0$ implies number is zero$\}$

## Problem: Write out all numbers on the number line

Hint:

$$\begin{array}{rc} 0.d_0 d_1 & e = -1 \\ d_0.d_1 & e = 0 \\ d_0 d_1.0 & e = 1 \end{array}$$

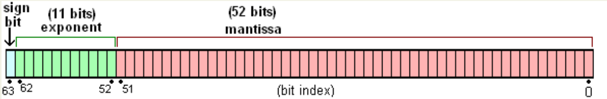
# Python double precision real numbers

## Computer Representation of Real Numbers

For Python floating point numbers:
$\beta = 2, t = 53, L = -1022, U = 1023$ so an individual floating point number has the bit form,

$$s_1 e_1 e_2 \ldots e_{11} d_1 d_2 \ldots d_{52}$$

as seen from the picture in memory



# Python representation of real numbers

## Computer Representation of Real Numbers

To convert the bit string $s_1e_1e_2...e_{11}d_1d_2...d_{52}$ to a decimal (base ten) real number,

1. Convert $e_1e_2...e_{11}$ to a decimal number E.
2. Multiply $2^{E-1023}$ with $1.d_1d_2...d_{52}$ (just shift the decimal point $E-1023$ bits) and convert this to a decimal real number R.
3. Compute $(-1)^{s_1} * R$

The above sequence can be compactly expressed in the formula,

$$number = (-1)^{s_1} * (1.d_1d_2...d_{52}) * 2^{E-1023} \text{ where } 0 < E < 2047$$

If we write $1.d_1d_2...d_{52}$ as $1.f$ then the above expression can be written in the compact form,

$$number = (-1)^{s_1} * 1.f * 2^{E-1023} \text{ where } 0 < E < 2047$$

# What happened to zero?

## "Special" numbers

Special numbers use values of $E = 0$ or $f = 0$.
For example, zero is represented by $E = 0, f = 0$
$(-1)^s 2^{-1022}(0.0)$

| $s$ | $0...0$ | $0...0$ |
|---|---|---|
| $bit : 63$ | $bits : 62 \leftarrow 52$ | $bits : 51 \leftarrow 0$ |

subnormal(denormalized) numbers by $E = 0, f \neq 0$
$(-1)^s 2^{-1022}(0.f)$

| $s$ | $0...0$ | $f$ |
|---|---|---|
| $bit : 63$ | $bits : 62 \leftarrow 52$ | $bits : 51 \leftarrow 0$ |

Infinity is represented by $E = 2047, f = 0$
$(-1)^s 2^{1024}(1.0)$
NaN is represented by $E = 2047, f \neq 0$
$(-1)^s 2^{1024}(1.f)$

# Test your understanding

## Which values in $\mathbb{Q}$ are exactly representable with a finite binary decimal expansion?

- Hint: Use the Fundamental Theorem of Arithmetic: Any integer greater than 1 can be written as a unique product (up to ordering of the factors) of prime factors.
- Is 0.1 exactly representable in Python?
- How do we display the value Python assigns to 0.1?

## Are floating point values equally spaced?

- For a toy Normalized Binary Floating point system $F(2,3,-1,1) = \{(-1)^s 2^e (1.d_1 d_2) \mid s = 0 \; or \; s = 1 \; , \; d_i = 0$ or $1, \; L \leqslant e \leqslant U\}$ are the values equally spaced? (We are ignoring ”special” numbers.)

# Test your understanding

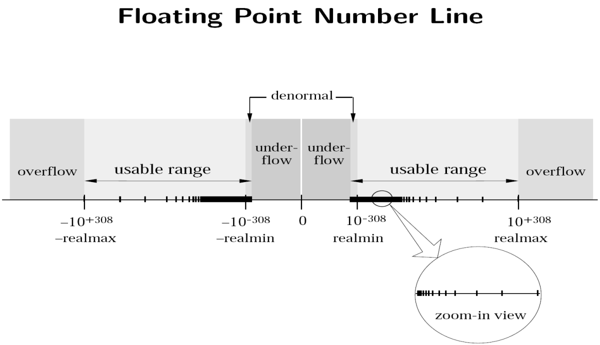
**What are the largest and smallest(positive both subnormal and not subnormal) IEEE numbers?**

- Check your answer by typing:

```
import numpy as np
np.finfo(np.float).tiny
np.finfo(np.float).max
```

  at the Python prompt.

# IEEE representable numbers




## Overflow/Underflow

- computations too close to zero may result in *underflow*
- computations too large may result in *overflow*
- overflow error is considered more severe
- underflow can just fall back to 0

# Test your understanding

## Example

Effects of spacing of floating point values

```python
first = []
a=1.0
while a > 0.0:
    a = a/2.0
    first.append(a)
second = []
a=1.0
while a+1.0 > 1.0:
    a = a/2.0
    second.append(a)
print('len(first) =', len(first))
print('first[-1] = ', first[-1])
print('len(second) = ',len(second))
print('second[-1] = ', second[-1])
```

# Fraction Algorithm

An algorithm to compute the binary representation of a fraction $x$:

$$\begin{aligned} x &= 0.b_1 b_2 b_3 b_4 \ldots \\ &= b_1 \cdot 2^{-1} + \ldots \end{aligned}$$

Multiply $x$ by 2. The integer part of $2x$ is $b_1$

$$2x = b_1 \cdot 2^0 + b_2 \cdot 2^{-1} + b_3 \cdot 2^{-2} + \ldots$$

## Example

Example:Compute the binary representation of 0.625

$$\begin{aligned} 2 \cdot 0.625 = 1.25 &\quad \Rightarrow \quad b_1 = 1 \\ 2 \cdot 0.25 = 0.5 &\quad \Rightarrow \quad b_2 = 0 \\ 2 \cdot 0.5 = 1.0 &\quad \Rightarrow \quad b_3 = 1 \end{aligned}$$

So $(0.625)_{10} = (0.101)_2$

## Data Error

*Data Error* can occur when making a measurement of a physical quantity. But *Data Error* can also occur by means of a data entry error.

## Numerical Error: Roundoff

*Roundoff* occurs when digits in a decimal point (0.3333...) are lost (0.3333) due to a limit on the memory available for storing one numerical value.

## Numerical Error: Truncation

*Truncation* error occurs when approximations such as truncating an infinite series, replacing a derivative by a finite difference quotient, replacing an arbitrary function by a polynomial, or terminating an iterative sequence before it converges.

# Floating Point Errors

- Not all reals can be exactly represented as a machine floating point number. Then what?
- Roundoff error
- IEEE options:
  - Round to next nearest FP (preferred), Round to 0, Round up, and Round down
  - We will use the notation $fl(x)$ to denote the floating point value representing a real number x.

Let $x_+$ and $x_-$ be the two floating point machine numbers closest to $x$

- round to nearest: $round(x) = x_-$ or $x_+$, whichever is closest
- round toward 0: $round(x) = x_-$ or $x_+$, whichever is between 0 and $x$
- round toward $-\infty$ (down): $round(x) = x_-$ (used in representing interval numbers)
- round toward $+\infty$ (up): $round(x) = x_+$(used in representing interval numbers)

## Machine Epsilon

The machine epsilon $\epsilon_m$ is the smallest positive machine number such that

$$1 + \epsilon_m \neq 1$$

- The double precision machine epsilon is $2^{-52}$.

```
>> eps
ans = 2.2204e-16
```

# Measuring Error - Accuracy

- Here we define the absolute error:

$$\text{Absolute Error}(x_a) = x_a - x_t$$
$$= \text{approximate value} - \text{true value}$$

- This doesn't tell the whole story. For example, if the values are large, like billions, then an Error of 100 is small. If the values are smaller, say around 10, then an Error of 100 is large. We need the relative error:

$$\text{Relative Error}(x_a) = \frac{x_a - x_t}{x_t}$$
$$= \frac{\text{approximate value} - \text{true value}}{\text{true value}}$$

when true value $\neq 0$

- After some simple algebra on the previous equation above we can write,

$$\text{approximate value} = \text{true value} * (1 + \text{Relative Error})$$

# Floating Point Errors

## How big is roundoff error?

Suppose $x$ is a real number and $fl(x) = x_-$ where $x_-$ is not a subnormal floating point number or zero. We further assume that rounding is performed by "rounding to nearest".

$$
\begin{aligned}
x &= (1.d_1 d_2 d_3 \ldots d_{52} \ldots)_2 \times 2^e \\
x_- &= (1.d_1 d_2 \ldots d_{52})_2 \times 2^e \\
x_+ &= \left((1.d_1 d_2 \ldots d_{52})_2 + 2^{-52}\right) \times 2^e \\
|x_- - x| &\leqslant \frac{|x_+ - x_-|}{2} = 2^{e-53} \\
\left|\frac{x_- - x}{x}\right| &\leqslant \frac{2^{e-53}}{2^e} = 2^{-53} = \epsilon_m/2
\end{aligned}
$$

## Unit Roundoff Error

unit roundoff error = $\mu = \epsilon_m/2$

# Floating Point Errors - Accuracy

## From the previous slides

$$fl(x) = x * (1 + \text{Relative Error})$$

$$\left|\frac{fl(x) - x}{x}\right| \leqslant \mu$$

## We have established that for $x \in \mathbb{R}$ and $fl(x)$ not subnormal,

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

# Relationship between error and Significant Digits

## Example

Given $x = 123.01234$ and $y = 123.0123$ is an approximation of $x$ then

$$|y - x| = 0.00004 = 0.4 * 10^{-4}$$

$$\frac{|y - x|}{|x|} \leqslant 3.26 * 10^{-7}$$

From the above example we would like to make the following assertions:

- If absolute error is less than $0.5 * 10^{-t}$ then there are t equal digits to the right of the decimal point between $y$ and $x$, when both numbers are in the non-scientific notation form,

$$y = d_1 d_2 ... d_n . d_{n+1} d_{n+2} ... d_{n+t} ...$$

- If the relative error is less than $5.0 * 10^{-t}$ then there are t equal digits digits total between $y$ and $x$ when both numbers are in the scientific notation form, .e.g. no leading zeros)

$$x = 0.d_1 d_2 ... d_t ... * 10^e$$

# Relationship between error and Significant Digits

However, the following example shows that the above assertions are not strictly correct.

## Example

Given $x = 0.00351$ and an approximation $y = .00346$ then

$$|y - x| = 0.00005 = 0.5 * 10^{-4}$$

$$\frac{|y - x|}{|x|} \leqslant 1.43 * 10^{-2}$$

Now $x$ and $y$ agree in three digits to the right of the decimal but the first assertion says it should be four. However if we rounded (to nearest) the fifth decimal digits of $x$ and $y$ then the assertion would be true.
If we re-write $x = 0.351 * 10^{-2}$ and $y = 0.346 * 10^{-2}$ these numbers agree only with one digit but the assertion says it should be two. Again, however, if we round (to nearest) the third decimal digits of $x$ and $y$ then the assertion would be true. How can we overcome this discrepancy in our assertions?

# Significant Digits - Accuracy

We overcome the previous problem be re-defining "equal digits" as follows:

## Significant Decimal Digits

If $y = d_1 d_2 ... d_n . d_{n+1} d_{n+2} ... d_{n+t} ...$ is an approximation of $x$ and we have $|y - x| \leqslant 0.5 * 10^{-t}$ then the $t$ digits starting in the position $\geqslant 10^{-t}$ of $y$ are called "significant decimal digits".

## Significant Digits

If $y$ is an approximation of $x = 0.d_1 d_2 ... d_t ... * 10^e$ and we have $\frac{|y-x|}{|x|} \leqslant 5.0 * 10^{-t}$ then the $t$ digits starting in the position $\geqslant 10^{-t}$ of $y$ are called "significant digits".

# Our Job

Given a specific problem. A numerical analyst will do the following:

- Determine the condition of the problem.
  - A problem is ill-conditioned if small changes to input values create large errors in the solution, assuming that our implementation is mathematically perfect, i.e. does not introduce round-off or truncation errors.
  - A problem is well-conditioned if it is not ill-conditioned.
- Choose a method and specific algorithm that is stable.
  - An algorithm is stable if it achieves the level of accuracy defined by the condition of the problem.
- Implement the algorithm so that the calculation is not susceptible to large roundoff error and the approximation has a *tolerable* truncation error.

How?

- Compute the condition "number" for the problem.
- incorporate roundoff-truncation knowledge into
  - the method
  - the algorithm
  - the software design
- awareness $\rightarrow$ correct interpretation of results

## For $\mathbb{C}$ - Complex Numbers

- $(a+bi)+(c+di)=(a+c)+(b+d)i$.
- $(a+bi)-(c+di)=(a-c)+(b-d)i$.
- $(a+bi)*(c+di)=(ac-bd)+(bc+ad)i$.
- $(a+bi)/(c+di)=\frac{a+bi}{c+di}\frac{c-di}{c-di}=\frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}i$ where $c^2+d^2\neq 0$.

# $\mathbb{C}$ - Complex Numbers

## An alternative view

- $a + bi \leftrightarrow \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$.

## Example

$$(2+3i)*(-1-4i) = 10 - 11i$$

since

$$\begin{bmatrix} 2 & 3 \\ -3 & 2 \end{bmatrix} * \begin{bmatrix} -1 & -4 \\ 4 & -1 \end{bmatrix} = \begin{bmatrix} 10 & -11 \\ 11 & 10 \end{bmatrix}$$

# Operators

## For $\mathbb{IR}$ - Interval Numbers

Where $\circ$ denotes the operators $+, -, *, /$.

- $[a, b] \circ [c, d] = \{x \circ y \mid x \in [a, b], y \in [c, d]\}$.

### Example

$[-1, 1] + [-0.1, 0.1] = [-1.1, 1.1]$

### Example

$[2, 3] - [2, 3] = [-1, 1]$ thus $x - x \neq 0$ except for intervals of zero width, e.g. $[3, 3]$.

### Example

$[-1, 2] * ([-2, 3] + [3, 4]) = [-1, 2] * [1, 7] = [-7, 14]$ but $[-1, 2] * [-2, 3] + [-1, 2] * [3, 4] = [-4, 6] + [-4, 8] = [-8, 14]$ so the distributive law doesn't hold.

# Operators

## For Floating Point Numbers

Denote the double precision floating point numbers $F(2, 53, -1022, 1023)$ as $DPFP$. We will assume that the following holds where $\circ$ denotes the operators $+, -, *, /$.

- $fl(x \circ y) = (x \circ y)(1 + \delta)$ where $x, y \in DPFP$ and $|\delta| \leqslant \mu$.

### Example

$x \circ y$ need not be in $DPFP$ even if both $x, y \in DPFP$. Consider $x = 1, y = 3$ and $x/y$

### Example

Note that $fl(fl(x + y) + z) = fl(x + fl(y + z))$ fails for some choice of $x, y, z \in DPFP$. For example, when $x = 1.111...100 * 2^0$ 1.—fifty bits of 1's followed by two bits of 0's and $y = z = 1.0...0 * 2^-53$.

# Fields

The Rationals, Reals and Complex Numbers form a field based on the operations $+$ , $*$.

## Properties of a Field F

- closure: If $a \in F$ and $b \in F$ then $a + b \in F$ and $a * b \in F$.
- associativity: $a + (b + c) = (a + b) + c$ and $a * (b * c) = (a * b) * c$ for all $a \in F,\ b \in F,\ c \in F$.
- commutativity: $a + b = b + a \textit{ and } a * b = b * a$ for all $a \in F,\ b \in F$.
- additive and multiplicative identity: $a + 0 = a$ and $a * 1 = a$ for all $a \in F$.
- additive and multiplicative inverses: $a + (-a) = 0$ and $b * (1/b) = 1$ for all $a \in F$ and $b \in F,\ b \neq 0$.
- distributivity: $a * (b + c) = a * b + a * c$ for all $a \in F,\ b \in F,\ c \in F$.

# Completeness of the Real Numbers

## Upper Bound

Given any non-empty set $S \subset \mathbb{R}$ then we say that $S$ has an *upper bound* if there exists a real number $r \in \mathbb{R}$ with the property that for any $s \in S$ then $s \leqslant r$.

## Least Upper Bound or Supremum

Given any non-empty set $S \subset \mathbb{R}$ that has an upper bound then we say that $S$ has a *least upper bound* if there exists an upper bound $r_0 \in \mathbb{R}$ for $S$ with the property that for any upper bound $r$ for $S$ then $r_0 \leqslant r$.

The Real Numbers satisfy the Dedekind Completeness Property.

## Dedekind Completeness

For any non-empty set $S \subset \mathbb{R}$ that has an upper bound, then $S$ has a *least upper bound*.

**Definition**

A function is an ordered triple of sets $f = (Domain, CoDomain, Graph)$ where $Graph = \{(x,y) \mid x \in Domain, y \in CoDomain\}$ and $Graph$ has the property that if $(x,y) \in Graph$ and $(x,z) \in Graph$ then $y = z$.

The set $\{y \mid (x,y) \in Graph\}$ is called the *Range* of the function.
If $Domain = A$ and $CoDomain = B$ we say that $f$ maps $A$ into $B$ and write,

$$f : A \to B$$

If $(x,y) \in Graph$ we can write $y = f(x)$.

**Example**

The triple $\left([-1,1],[-1,1],\{(x,y) \mid x^2+y^2 = 1, x,y \in [-1,1]\}\right)$ is NOT a function, however $\left([-1,1],[0,1],\{(x,y) \mid x^2+y^2 = 1, x \in [-1,1], y \in [0,1]\}\right)$ is a function.

# Classification of Functions

## Definition

Given a function,

$$f = (Domain, CoDomain, Graph)$$

- $f$ is called *onto* or *surjective* if $CoDomain = Range$.
- $f$ is called *1-1* or *injective* if $(x_1, y), (x_2, y) \in Graph$ implies $x_1 = x_2$.
- $f$ is called *bijective* if it is both *surjective* and *injective*.

## Example

The function $\left(\mathbb{R}, \mathbb{R}, \{(x, y) \mid y = x^3, x, y \in \mathbb{R}\}\right)$ is a bijection.

# Inverse of a Function

## Definition

Given a bijective function,

$$f = (Domain, CoDomain, Graph)$$

then we can define an *inverse* function $f^{-1}$ as follows:

$$f^{-1} = (CoDomain, Domain, GraphInv) \; where$$
$$GraphInv = \{(y, x) \mid x \in Domain, y \in CoDomain\}$$

## Example

The function below is a bijection.

$$\big([0, +\infty), [0, +\infty), \{(x, y) \mid y = x^2, x, y \in [0, +\infty)\}\big)$$

To determine it's inverse swap $x$ with $y$ in the equation $y = x^2$ to obtain, $x = y^2$.

$$\big([0, +\infty), [0, +\infty), \{(x, y) \mid x = y^2 (y = +\sqrt{x}), x \in [0, +\infty)\}\big)$$

# Inverse of a Function

## Example

The function below is a bijection.

$$\left(\mathbb{R}^2, \mathbb{R}^2, \left\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} = M * \mathbf{x}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^2, M = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}\right\}\right)$$

To determine it's inverse swap $\mathbf{x}$ with $\mathbf{y}$ in the equation $\mathbf{y} = M * \mathbf{x}$ to obtain, $\mathbf{x} = M * \mathbf{y}$ (or $M^{-1}\mathbf{x} = \mathbf{y}$).

$$\left(\mathbb{R}^2, \mathbb{R}^2, \left\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} = M^{-1} * \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^2, M^{-1} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}\right\}\right)$$

# Norm Function

## Example - Measuring the "size" of Numbers

The function below is called an $\ell^2$ "$\ell$ two-norm".

$$\left(\mathbb{R}^n, \mathbb{R}, \left\{(\mathbf{x}, y) \mid y = |\mathbf{x}|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}\right\}\right)$$

The $\ell^2$ function has the following properties:

- $|\mathbf{u}|_2 \geqslant 0$ and $|\mathbf{u}|_2 = 0$ only if $\mathbf{u} = (0, ..., 0) \in \mathbb{R}^n$
- $|r * \mathbf{u}|_2 = |r| * |\mathbf{u}|_2$ for all $r \in \mathbb{R}$ and $\mathbf{u} \in \mathbb{R}^n$ ($|r|$ is just the absolute value of $r$)
- $|\mathbf{u} + \mathbf{v}|_2 \leqslant |\mathbf{u}|_2 + |\mathbf{v}|_2$ for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ (triangle inequality)

# Outlook

Next time:

- Taylor Series
- Order of Convergence
- Condition Number
- Stability