PHSA-CC-1-1-P : Mathematical Physics I

Instructor : Amit Kumar Bhattacharjee (AKB)

Course Webpage : https://amitbny.github.io/akb.github.io/scomphsa.html

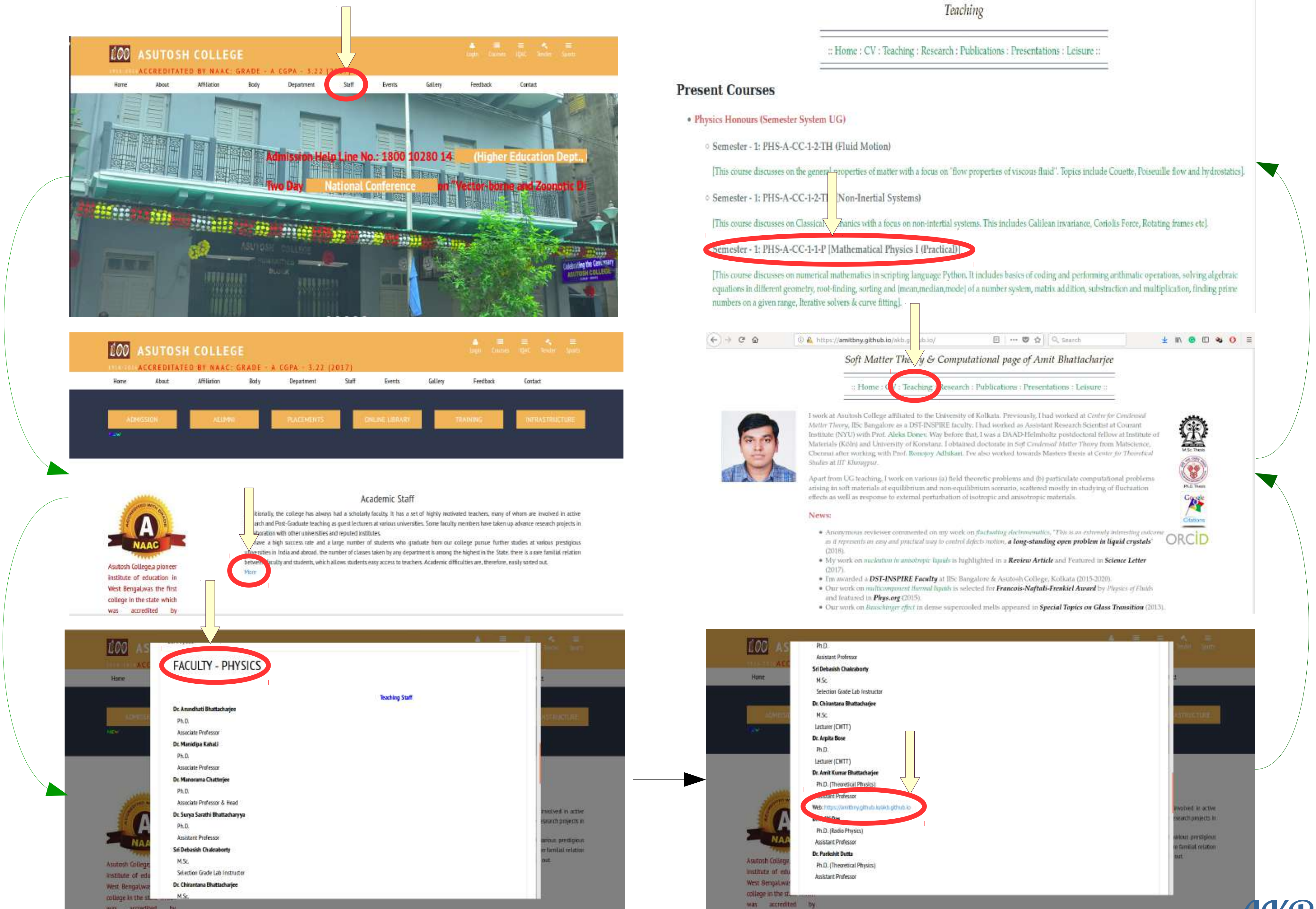
Course timeline : Jul–Nov, 2018

Evaluation : Assignments/Classtest followed by Semester examination

Ebook resources : National digital library: https://ndl.iitkgp.ac.in
http://nlist.inflibnet.ac.in

AKB

Way to access the webpage



AKB

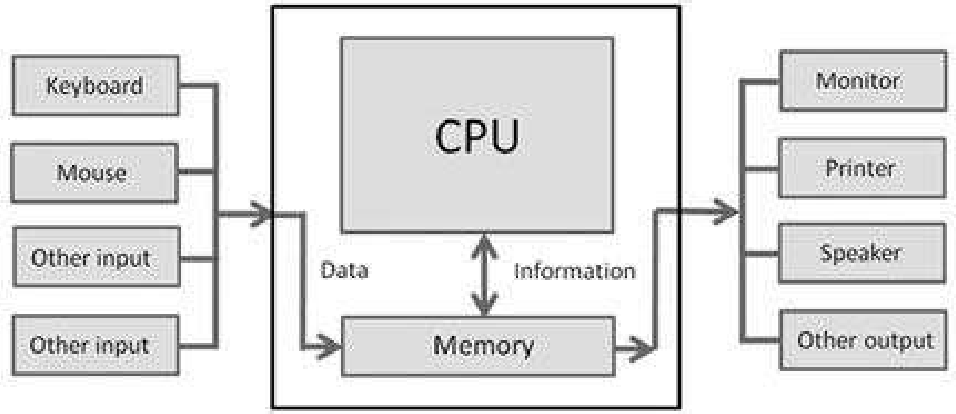
Course Marks : TBD; Credits - 2

- Introduction and overview ➡ Computer architecture and organization, memory and Input/output devices.
- Basics of scientific computing ➡ Binary and decimal arithmetic, Floating point numbers, algorithms, Sequence, Selection and Repetition, single and double precision arithmetic, underflow & overflow - importance of making equations in terms of dimensionless variables, Iterative methods.

AKB

What is a Computer?

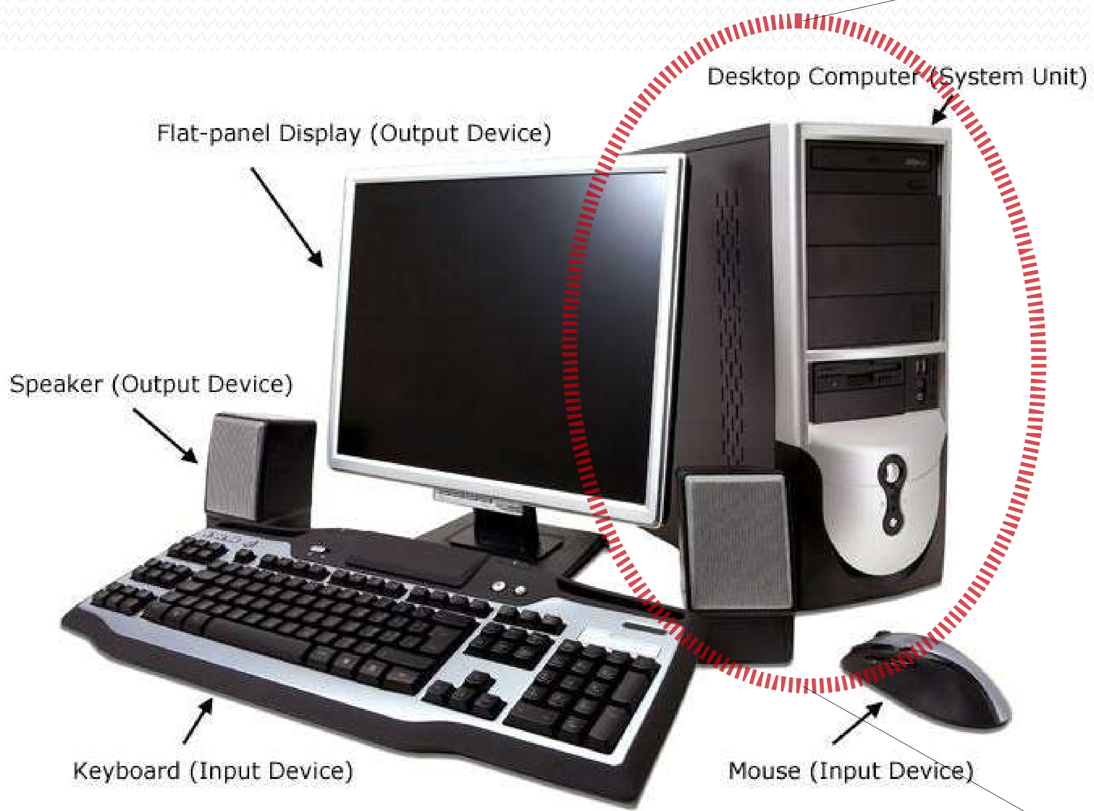
A computer is a device that can be instructed to carry out sequences of arithmetic or logical set of operations automatically via computer programming.



Hardware

What is a Computer?

A computer is a device that can be instructed to carry out sequences of arithmetic or logical set of operations automatically via computer programming.



AKB

What is a Computer?

CPU consists of Arithmetic Logic Unit (ALU), Memory, Input/Output (I/O)

HDD

RAM

Graphics Card

& other components, like motherboard, heat-sink, power supply, Fan etc.

HDD (data-storage) is the secondary memory while RAM (volatile memory) is the primary memory. Graphics card is necessary for data-heavy applications.

ALU, Memory, I/O

- ALU performs 2 class of operations (arithmetic & logic), e.g. +, -, *, /, sin(), sqrt() etc. Some machines can operate only on whole numbers (integers), while others use floating point (real) numbers, but with limited precision. Also Boolean logic operations (AND, OR, NOT, XOR) are performed in ALU.

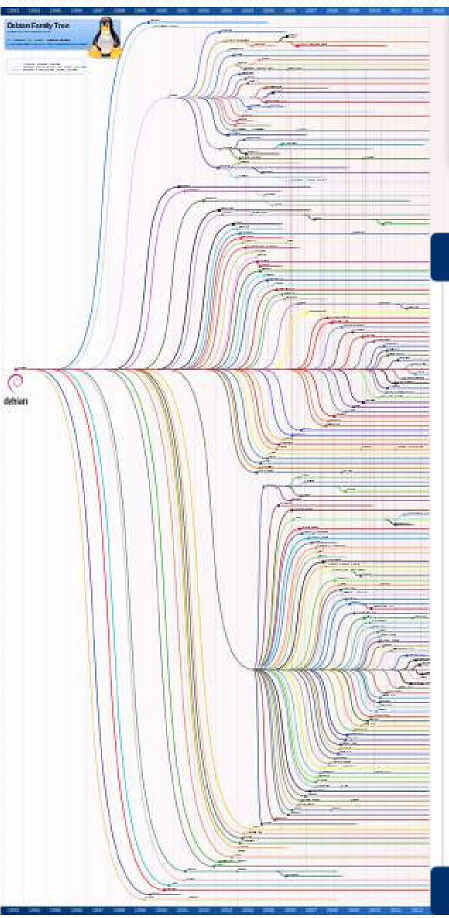
- Memory cell can store binary numbers in groups of 8 bits (or 1 byte). Each byte represents 256 different numbers ($2^8 = 256$): $\mathbb{R} \in [0,255], [-128,127]$. CPU contains memory cells (Registers) which are read/written more rapidly than RAM.

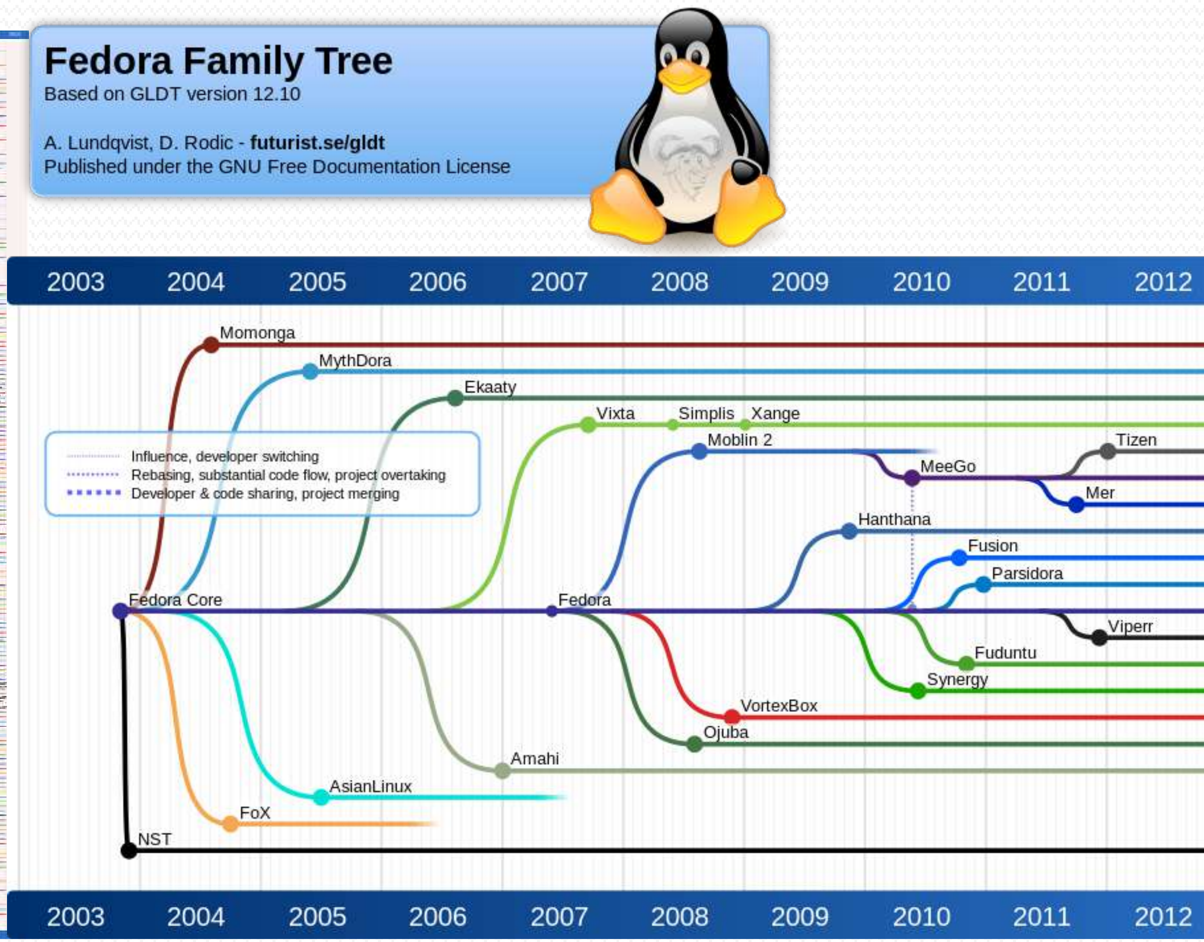
- I/O is the way a CPU exchanges information with the outside world, through Peripherals e.g. keyboard, mouse etc (input devices) & display, printer etc (output devices). HDD, optical disk drives, computer networking serve as both input and output devices.

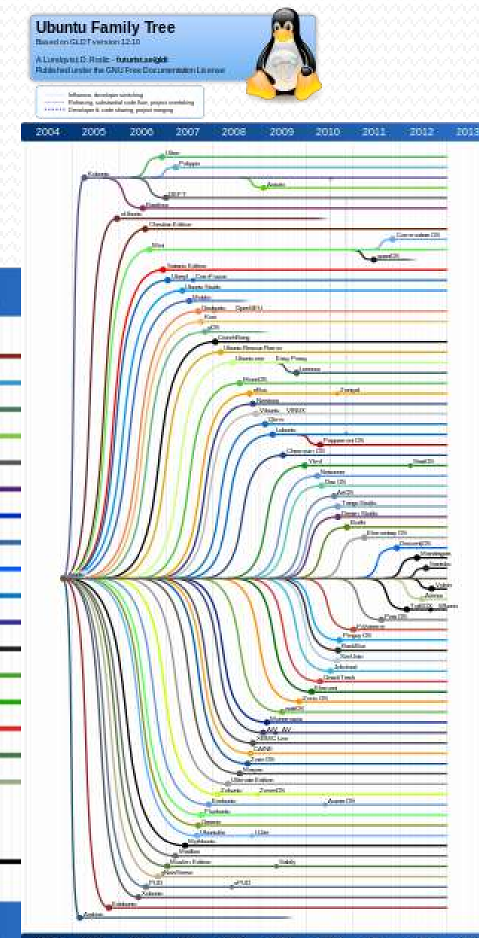
What is a Computer?

Computer Programs, libraries, Operating Systems (OS) etc. OS has many Variant : (i) Unix distro (Solaris Sun OS), IRIX etc,

(ii) GNU/Linux [CentOS, Fedora (Redhat), SUSE, Ubuntu/Mint]







What is a Computer?

Computer Programs, libraries, Operating Systems (OS) etc. OS has many Variant : (i) Unix distro (Solaris Sun OS), IRIX etc,
(ii) GNU/Linux [CentOS, Fedora (Redhat), SUSE, Ubuntu/Mint]

Library : (i) Multimedia [DirectX, OpenGL, OpenAL, Vulkan (API)]
(ii) Programming Library (GSL, NRCP etc)

Data : (i) Protocol (TCP/IP, FTP, HTTP, SMTP etc),
(ii) File format (HTML, XML, JPEG, MPEG, PNG etc)

User Interface : GUI

Application Software : Office-suite, Graphics, Audio, Games, Software Engineering (Compiler, Assembler, Interpreter, Debugger, Text editor etc).

What is a Computer?

Programming Languages : (i) Low-level (e.g. Assembly language),
(ii) High-level (e.g. Basic, C/C++, Fortran 90/95 Java, Pascal),
(iii) Scripting (Python, Ruby, Perl).

Mathematical Softwares :
(i) Coding : LAPACK, LINPACK.

(ii) Coding/Visualization/Post-processing : Mathematica, Matlab/Octave, Maple.

(iii) Visualization : OpendX, Ovito, Paraview, VisIt, PyMol.

(iv) Supercomputing : LAMMPS, BoxLib, PETSc, Sundials.

Scientific Computing

17 Equations That Changed the World
by Ian Stewart

1.	**Pythagoras's Theorem**	$a^2 + b^2 = c^2$	Pythagoras,530 BC
2.	**Logarithms**	$\log xy = \log x + \log y$	John Napier, 1610
3.	**Calculus**	$\frac{\mathrm{d}f}{\mathrm{d}t} = \lim_{h\to 0} = \frac{f(t+h) - f(t)}{h}$	Newton, 1668
4.	**Law of Gravity**	$F = G\frac{m_1 m_2}{r^2}$	Newton, 1687
5.	**The Square Root of Minus One**	$i^2 = -1$	Euler, 1750
6.	**Euler's Formula for Polyhedra**	$V - E + F = 2$	Euler, 1751
7.	**Normal Distribution**	$\Phi(x) = \frac{1}{\sqrt{2\pi\rho}} e^{\frac{(x-\mu)^2}{2\rho^2}}$	C.F. Gauss, 1810
8.	**Wave Equation**	$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$	J. d'Almbert, 1746
9.	**Fourier Transform**	$f(\omega) = \int_{\infty}^{\infty} f(x) e^{-2\pi i x \omega}\,\mathrm{d}x$	J. Fourier, 1822
10.	**Navier-Stokes Equation**	$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -\nabla p + \nabla\cdot\mathbf{T} + \mathbf{f}$	C. Navier, G. Stokes, 1845
11.	**Maxwell's Equations**	$\nabla\cdot\mathbf{E} = 0$, $\nabla\cdot\mathbf{H} = 0$, $\nabla\times\mathbf{E} = -\frac{1}{c}\frac{\partial\mathbf{H}}{\partial t}$, $\nabla\times\mathbf{H} = \frac{1}{c}\frac{\partial E}{\partial t}$	J.C. Maxwell, 1865
12.	**Second Law of Thermodynamics**	$\mathrm{d}S \geq 0$	L. Boltzmann, 1874
13.	**Relativity**	$E = mc^2$	Einstein, 1905
14.	**Schrodinger's Equation**	$i\hbar\frac{\partial}{\partial t}\Psi = H\Psi$	E. Schrodinger, 1927
15.	**Information Theory**	$H = -\sum p(x)\log p(x)$	C. Shannon, 1949
16.	**Chaos Theory**	$x_{t+1} = kx_t(1 - x_t)$	Robert May, 1975
17.	**Black-Scholes Equation**	$\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} + \frac{\partial V}{\partial t} - rV = 0$	F. Black, M. Scholes, 1990

= ?

Scientific Computing

- Domain beyond analytical approach (due to non-linearity, integrability, non-inversion and many other reasons): Numerical Mathematics/Applied Mathematics/Computational Science. Applications include, Computational Finance, Computational Biology, Computational Engineering, Computational Physics, Computational Chemistry, Computational Materials Science & so on.

- A well-executed computation can reproduce lab-based experiments quantitatively, and therefore can predict new phenomena by "numerical experiments" often hard to realize on a lab due to financial / timeframe / workforce restrictions.

- There goes the catch! Given a computer, is every computation is a well-executed computation? Answer is NO.

AKB

Binary and Decimal

- Decimal (or denary) numeral system represents integer and non-integer numbers in base-10 positional number system. Decimal refers to digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in the decimal system containing a “decimal separator”, e.g. 3.14. In general,

$$a_m a_{m-1} \dots a_0 . b_1 b_2 \dots b_n = a_m 10^m + a_{m-1} 10^{m-1} + \dots + a_0 10^0 + \frac{b_1}{10^1} + \frac{b_2}{10^2} + \dots + \frac{b_n}{10^n}.$$

- Binary numeral system represents only two numbers 0 and 1 in base-2 number system. A human-understood decimal is converted to computer-understood binary to perform computation and back converted to decimal to decipher.
- Conversion binary → decimal: $a_m a_{m-1} \dots a_1 = a_m 2^{m-1} + a_{m-1} 2^{m-2} + \dots + a_1 2^0.$

Most Significant Bit (MSB)

Least Significant Bit (LSB)

$$101100101_2 = 1*2^8 + 0*2^7 + 1*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 357_{10}.$$

- Conversion decimal ➔ binary: Repeated division-by-2 method:

294_{10} : $\text{divide by } 2 \rightarrow 147 \{\text{remainder } 0 (LSB)\}$, $\text{divide by } 2 \rightarrow 73 \ (\text{remainder } 1)$,
$\text{divide by } 2 \rightarrow 36 \ (\text{remainder } 1)$, $\text{divide by } 2 \rightarrow 18 \ (\text{remainder } 0)$,
$\text{divide by } 2 \rightarrow 9 \ (\text{remainder } 0)$, $\text{divide by } 2 \rightarrow 4 \ (\text{remainder } 1)$,
$\text{divide by } 2 \rightarrow 2 \ (\text{remainder } 0)$, $\text{divide by } 2 \rightarrow 1 \ (\text{remainder } 0)$,
$\text{divide by } 2 \rightarrow 0 \{\text{remainder } 1 (MSB)\}$ ➔ 100100110_2.

- Fractions in binary terminate, if the denominator has 2 as the only prime factor. 1/10 doesn't have a finite binary representation which causes 10×0.1 not to be precisely equal to 1 in floating point arithmetic. To interpret the binary expression for $\frac{1}{3} = .010101\ldots$ means $= 0*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4} + \ldots = 0.3125 + \ldots$. So 1 and 0's alternate forever, if we want to reach the exact expression as a sum of inverse powers of 2 ➔ *source of Error* !!

Scientific Computing

The Patriot Missile Failure → On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Saudi Arabia failed to track & intercept an incoming Iraqi Scud missile. It killed 28 soldiers & injured 100s of people. Cause of the failure turned out to be inaccurate calculation of the time due to arithmetic errors!! How????

- Time in tenths of second (measured by system's internal clock) was multiplied by 0.1 to produce the time in seconds, using a 24-bit Register. Specifically, value of 1/10 (having non-terminating binary expansion) was truncated at 24-bits. This small chopping error when multiplied by large number led to significant error. $\frac{1}{10} = \frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^8}+\frac{1}{2^9}+\frac{1}{2^{12}}+\frac{1}{2^{13}} \rightarrow \text{binary expansion} \rightarrow$ $0.0001100110011001100110011001100$

What patriot saved in the system → $0.00011001100110011001100$

error in binary 0.0000000000000000000000011001100 lead to 0.000000095 in decimal. Multiplying by the number of tenths of a second in 100 hours gives 0.000000095×100×60×60×10=0.34 seconds.

Floating Point Number System

Significant Digits ➡ These are the first nonzero digit & all succeeding digits, *e.g.* 1.7320 has 5 significant digits, while 0.0491 has only 3.

A floating point (real) number system have elements of the form $y = \pm m \times \beta^{e-t}$, is characterized with 4 integer parameters:

- Base (or radix) β, precision t, exponent e & significand (or mantissa) m. Here $e_{min} \le e \le e_{max}$ & $0 \le m \le \beta^t - 1$. This gives the range of nonzero floating point numbers $\beta^{e_{min}-1} \le y \le \beta^{e_{max}}(1-\beta^{-t})$.
- Floating point numbers aren't equally spaced !! If $\beta = 2, t = 3, e_{min} = -1, e_{max} = 3$ then non-negative numbers are $0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875, 1.0, 1.25, 1.50, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0$.
- Spacing of the floating point numbers jumps by a factor 2 at each power of 2. Spacing can be characterized in terms of machine epsilon, which is the distance from 1.0 to the next larger floating point number. In Python, this can be seen as :

"import numpy as np", and then, "np.spacing(1)" yields 2.2204460492503131e-16.

Floating Point Number System

In MATLAB/Octave, “eps” gives the same value. “realmax” & “realmin” represent the largest positive & smallest positive normalized floating point number. In Python, “import numpy as np”, and then, “np.finfo(np.double).max” yields 1.7976931348623157e+308, while “np.finfo(np.double).tiny” yields 2.2250738585072014e-308.

IEEE Arithmetic IEEE standard defines a binary floating point system. The standard specifies floating point number formats, results of the basic floating point operations & comparisons, rounding modes, floating point exceptions & handling, conversion between different arithmetic formats.

Two main floating point formats are defined:

	Type	Size	Significand	Exponent	Unit roundoff	Range
(precision)	Single	32 bits	23+1 bits	8 bits	$2^{-24} \approx 5.96\times10^{-8}$	$10^{\pm 38}$
	Double	64 bits	52+1 bits	11 bits	$2^{-53} \approx 1.11\times10^{-16}$	$10^{\pm 308}$

Floating Point Number System

In both formats one bit is reserved as a sign bit. The most significant bit is always 1 & not stored. This hidden bit accounts for the "+1" in the table.

- NaN (Not a number) is a special bit pattern with arbitrary significand. It's generated by operations such as $0/0, 0\times\infty, \infty/\infty, (+\infty)+(-\infty)$. Infinity symbol is represented by zero significand & same exponent field as NaN, sign bit distinguishes between $\pm\infty$ with property, $\infty+\infty=\infty, (-1)\times\infty=-\infty, \text{finite}/\infty=0$. Zero is represented by a zero exponent field & zero significand, with $+0=-0$.
- In MATLAB/Fortran 90/95, A(p:q, r:s) denotes submatrix of A formed of rows p to q & columns r to s. A(:, j) is the jth column of A, and A(i, :) the ith row of A.
- Evaluation of an expression in floating point arithmetic denoted by $fl(.)$ is

$$fl(x\, op\, y)=(x\, op\, y)(1+\delta), \quad |\delta|\le u$$

  u is called the ***unit roundoff*** (machine precision) $\approx 10^{-8}(\text{single}), 10^{-16}(\text{double}),$ $10^{-10}-10^{-12}(\text{pocket calculators})$.
- Computed quantities are denoted with ***hat***. So, $\hat{x}$ is the computed approximation of x.

AKB

Floating Point Number System

- $\lfloor x \rfloor$ (floor x) is the largest integer $\leq x$ & $\lceil x \rceil$ (ceil x) is the smallest integer $\geq x$. Check with Python: "import math; math.floor(1.9) = 1.0", "math.ceil(1.9)=2.0".
- Remember, we compute single precision arithmetic ($u \approx 6 \times 10^{-8}$) by rounding, say, a double precision result with *unit roundoff* ($u \approx 1.1 \times 10^{-16}$) to single precision as well rounding result of every elementary operation to single precision.

Absolute & Relative Error ➔ If $\hat{x}$ is an approximation to real number x, then

$$E_{\text{abs}}(\hat{x}) = |x - \hat{x}|, \quad E_{\text{rel}}(\hat{x}) = \frac{|x - \hat{x}|}{|x|}$$

Note that relative error is scale independent: $x \to \alpha x, \hat{x} \to \alpha \hat{x}$, doesn't change $E_{\text{rel}}(\hat{x})$.

- Relative error is connected with the notion of *Correct significant digits*, however relative error is a more precise, base independent measure.
- Sources of Error ➤ (i) rounding, (ii) data uncertainty & (iii) truncation. Uncertainty in data can arise in several ways ➔ from errors of measurement, storing data on

AKB

computer. Data errors can be analysed using perturbation theory, while intermediate rounding errors require an analysis specific to the given method & thus harder to understand.

- Truncation/discretization errors is when in Taylor's series to derive numerical methods, such as Trapezium rule for Quadrature, Euler's method for differential equations etc, finite terms are kept and later are omitted. This depends on choice of "h": $f(x+h)=f(x)+hf'(x)+\frac{h^2}{2!}f''(x)+\frac{h^3}{3!}f'''(x)+O(h^4)$
- "Rounding errors and instability are important & numerical analysts will always be the experts in these subjects & at pains to ensure that the unwary are not tripped up by them. But our central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed".

– Nick Trefethen, FRS, Univ. of Oxford.

Precision vs Accuracy ➡ Accuracy refers to the absolute/relative error of an approximate quantity. Precision is the accuracy with which the basic arithmetic operations (+, -, *, /) are performed & for floating point arithmetic is measured by the *unit roundoff* u. Accuracy & precision are the same for the scalar computation c = a x b, but accuracy can be much worse than precision in the solution of a linear system of equations, e.g. stiff equations.

Forward and Backward Errors ➡ Suppose that an approximation $\hat{y}$ of $y=f(x)$ is computed in an arithmetic of precision u with $E_{rel}(\hat{y})\approx u$. This doesn't mean we know for any $\triangle x$, $\hat{y}=f(x+\triangle x)$. The absolute and relative errors of $\hat{y}$ are called forward errors and $min|\triangle x|$ and $\frac{min|\triangle x|}{|x|}$ is called the backward error. Stability of numerical recipe lies on backward stable algorithm where rounding errors are most significant. Recipe for cosine functions do not satisfy $\hat{y}=f(x+\triangle x)$ but $\hat{y}+\triangle y=f(x+\triangle x)$, $\triangle y\leq\epsilon|y|$, $\triangle x\leq\eta|x|$, are called mixed forward-backward error result.

AKB

Conditioning ➔ Forward & Backward error is governed by sensitivity of solution to perturbations in the data or "conditioning" of the problem. Then,

$$\hat{y}-y=f(x+\triangle x)-f(x)=f'(x)\triangle x+\frac{(\triangle x)^2}{2}f''(x+\triangle x)+O((\triangle x)^3),$$

or $\dfrac{\hat{y}-y}{y}=\left(\dfrac{xf'(x)}{f(x)}\right)\dfrac{\triangle x}{x}+O((\triangle x)^2)$. Here $c(x)=\left|\dfrac{xf'(x)}{f(x)}\right|$ measures the relative

change in output for relative change in input or condition number of f. For example, consider $f(x)=\log x$, $c(x)=|1/\log x|$ → ∞ for $x\approx 1$. So a small relative change in x can produce large relative change in log x for x~1.

Rule of thumb: $\text{forward error} \le \text{condition number} \times \text{backward error}$. So, computed solution to an ill-conditioned problem can have a large forward error, even if computed solution has *small* backward error. Backward stability implies forward stability. Cramer's rule for solving 2x2 linear system is forward stable but not backward stable.

AKB

Cancellation ➔ Consider the function $f(x)=\frac{(1-\cos x)}{x^2}$ which for all $x\neq 0$ is $0\leq f(x)<1/2$. However, say, for $x=1.2\times10^{-5}$, $\cos x=0.9999999999$ rounded to 10 significant digits, so as $1-\cos x=0.0000\,0000\,01$ and then, $\frac{(1-\cos x)}{x^2}=\frac{10^{-10}}{1.44\times10^{-10}}$ = 0.6944…, which is wrong !!

The problem lies in the fact that, even though 1-c is exact, it has only 1 significant figure, so subtraction produces a result of the same size as the error in c. However, if the subtraction is avoided by rewriting $\cos x=1-2\sin^2(x/2)$, $f(x)=\frac{1}{2}(\frac{\sin(x/2)}{x/2})^2$. The same procedure now yields f(x) = 0.5 correct to 10 significant digits.

- Error in cancellation can be avoided by *estimating the damage*, it can't be unavoidable. Or computing ratio of differences of the same order of error so that numerator & denominator cancels out. Or, for example computing x + (y – z) for $x\gg y\approx z>0$.

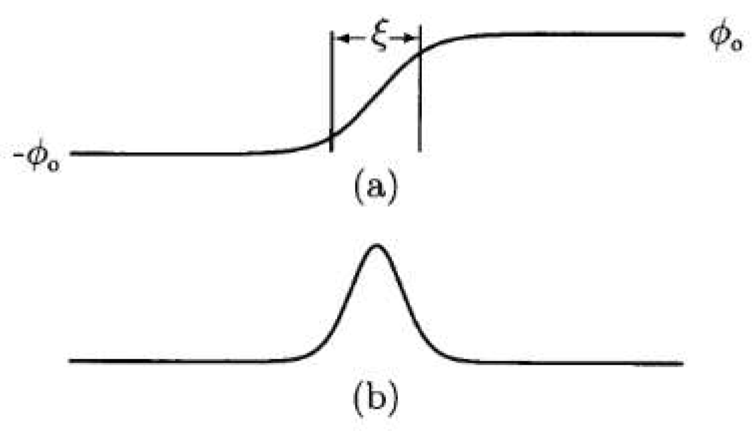
Roots of a Quadratic Equation ➲ Depending on the sign of the remainder b^2-4ac, for $a\neq 0, ax^2+bx+c=0$ have two roots (real-unequal, real-equal, imaginary) $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$. If $b^2\gg 4ac$, then $x=\frac{-b\pm b}{2a}$ and for "+" sign it suffers massive cancellation that brings prominence of earlier rounding errors. To avoid, the largest (in absolute value) root is chosen $x_1=\frac{-b-\sqrt{b^2-4ac}}{2a}$ and the other from $x_1x_2=\frac{c}{a}$. But when $b^2\approx 4ac$, accuracy is lost & only way to guarantee accuracy is to use extended precision.

Overflow & Underflow ➲ If we apply $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ in single precision arithmetic to equation $10^{20}x^2-3\times 10^{20}x+2\times 10^{20}=0$, even when the roots are not harmful (x=1 & x=2), overflow occurs since the maximum floating point number is $\approx 10^{38}$. Analytically/numerically dividing by maximum (|a|, |b|, |c|) = 3×10^{20} is OK, but same strategy doesn't work for, say, $10^{-20}x^2-3x+2\times 10^{20}=0$ whose roots are 10^{20} & 2×10^{20}. Scaling the variable $x=10^{20}y$ yields, $10^{20}y^2-3\times 10^{20}y+2\times 10^{20}=0$ which is the initial equation we started from.

AKB

Need for non-dimensionalization:

A well-known example in Condensed Matter Physics is the ϕ^4 kink, that gives a tanh solution of the domain wall formed between liquid-gas interface/magnetic domain walls having diverse consequences in many branches of physics.



$$\phi = \pm\phi_0 \tanh\left[\frac{x - x_0}{\sqrt{2c/r}}\right] \qquad \phi = \text{density (say)}$$

$$\mathscr{F} = -\frac{1}{2} r \phi^2 + u \phi^4 + \frac{1}{2} c (\nabla \phi)^2$$

- To have a control over the dynamics of density variation, clearly one needs to control these parameters, r, u, c or a multi-dimensional diagram which is nearly impossible to control often because of many parameters with less-known activity.

AKB

Need for non-dimensionalization:

- Notice that $\mathscr{F}=-\frac{1}{2}r\phi^2+u\phi^4$ have two minima, that we see from $\frac{\partial\mathscr{F}}{\partial\phi}=0=-r\phi+4u\phi^3$ or $\phi=\pm\phi_0=\sqrt{\frac{r}{4u}}$. Then $\mathscr{F}_0=-\frac{1}{2}r\phi^2+u\phi^4=-\frac{r^2}{16u}=-\frac{r\phi_0^2}{4}$
- But now notice that, $\mathscr{F}=-\frac{1}{2}r\phi^2+u\phi^4=-\frac{1}{2}r\frac{\phi^2}{\phi_0^2}\phi_0^2+u\frac{\phi^4}{\phi_0^4}\phi_0^4=-\frac{r^2}{8u}\hat{\phi}^2+\frac{r^2}{16u}\hat{\phi}^4$

$$=-\frac{r^2}{8u}\hat{\phi}^2+\frac{r^2}{16u}\hat{\phi}^4=-\frac{r^2}{16u}(2\hat{\phi}^2-\hat{\phi}^4)=\mathscr{F}_0(2\hat{\phi}^2-\hat{\phi}^4)$$

- Therefore, $\frac{\mathscr{F}}{\mathscr{F}_0}=\hat{\mathscr{F}}=2\hat{\phi}^2-\hat{\phi}^4$ & so, $\partial_t\hat{\phi}=4(\hat{\phi}-\hat{\phi}^3)$. The dynamics is completely free of parameter. According to the scale, we can choose what to compute !!

Beautiful Example : Kibble mechanism in Cosmology → LCD screen defect applications. Both are governed by same equation !!

AKB