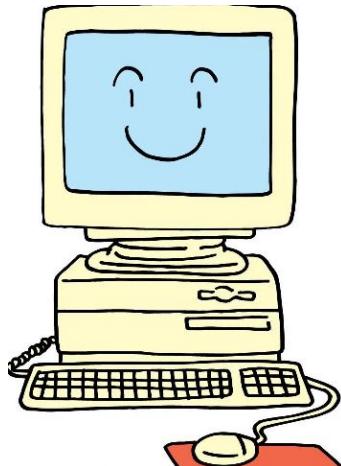
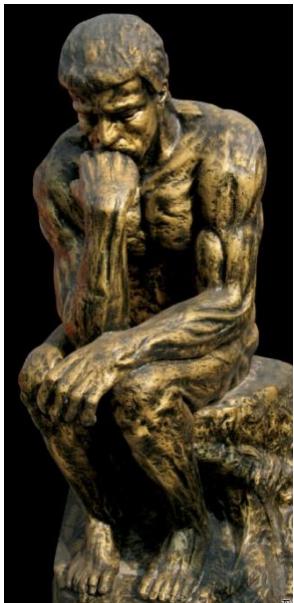


计算物理 第一部分

第1讲



实践是检验真理
的唯一标准
邓小平

李强 北京大学物理学院中楼411
qliphy0@pku.edu.cn, 15210033542

周二 3-4(10:10-12:00) 周五 5-6(13:00-14:50) 理教211

	周一习题	周二	周五	备注	5次作业 70%
1		9.14 (引言)	9.17(计算)		
2	9.20	9.21	9.24(计算、线代)		
3		9.28(线代、线方)HW1	10.1		
4	10.4	10.5	10.8 (线方)		
5		10.12(线方、内插)	10.15 (内插)	10.16周六下午6点前交HW1	
6	10.18	10.19(内插、积分)	10.22(积分)HW2		
7		10.26(积分、求根)	10.29(求根、极值)		
8	11.01	11.02(求根、极值)	11.05(本征值)	11.6周六下午6点前交HW2	
9		11.09(本征值)	11.12(本征值)HW3		
10	11.15	11.16 (ODE)	11.19 (ODE)		
11		11.23 (ODE)	11.26 (PDE)	11.27周六下午6点前交HW2	
12	11.29	11.30 (PDE)	12.03 (PDE、FFT)		
13		12.07 (FFT)	12.10 (随机数)		
14	12.13	12.14 (随机数)	12.17 (随机数)		
15		12.21(蒙特卡洛)	12.24(蒙特卡洛)		
16	12.27	12.28(蒙特卡洛)	12.31 (有限元、其它)		

5次作业

70%

期末笔试
30%

习题课 (具体时间待通知): 双周周一 10-11(18:40-20:30) 三教105

助教信息1

姓名	学号	手机	邮箱
高昂	2001110231	13716400125	2001110231@stu.pku.edu.cn

助教信息2

姓名	学号	手机	邮箱
张明	2001110128	15901020238	2001110128@stu.pku.edu.cn

答疑时间：周二下午3点-4点 物理中楼411

期末 2022.01.04 上午

课件网盘

教学微信群



计算物理2021秋



该二维码7天内(9月19日前)有效，重新进入将更新



[https://disk.pku.edu.cn:443/link/40F3FCFA2EFDEA
814D52A354CC9D40D2](https://disk.pku.edu.cn:443/link/40F3FCFA2EFDEA814D52A354CC9D40D2)

五次大作业，最终成绩是5次作业成绩的平均，占总成绩的70%。**期末闭卷考试，占总成绩的30%。**

这些作业中，有相当的内容是同学们利用所学的算法自己编写程序解决一些具体的数学或物理问题，你们**需要附上源代码以及数值结果，加上对结果的分析和讨论**。严禁从任何渠道抄袭（包括网上、书籍或(往届)同学等），**一旦发现一次，该次作业成绩为0；发现二次及以上，平时作业为0。**

本课程不讲授具体高级程序涉及语言，默认选课同学已掌握或者将自学至少下面一种高级预言：Fortran, C, C++或者Python等。**我们不接受利用Mathematica或者Matlab等集成软件，或者调用Python库编写的程序完成作业！**

每次作业会安排抽查，由助教约谈几位同学，要求同学当场演示讲解。

- **计算物理**, 是利用计算机来求解物理问题或者分析物理实验结果的一个重要的物理学分支, 有其相对的独立性; 同时, 它是**物理学、数学、计算机科学三者的交叉学科**, 与理论物理和实验物理有着密切的联系, 为物理学的发展起着极大的推动作用。
现代科学分析三大手段之一：理论分析、科学实验、数值计算。
- 计算物理的**研究目标是物理, 但其核心是数值分析, 也称为数值计算(或科学计算)**。在数学科学里, 数值分析也叫计算数学, 是数学科学的一个重要分支, 其研究对象是利用计算机求解各种数学问题的数值方法及相关理论。
- **数值计算和分析**在众多的学科和应用中占有举足轻重的地位, 几乎涉及到自然科学、工程应用和社会科学的各方面。

大型科学计算的发展水平也是一个国家综合国力的重要标志之一(硬件+软件): <http://www.top500.org> (全球超算500)

无论是国防、材料科学、天文学，还是距离我们生活更近的天气预报、灾害监测、交通管理，超算都大有用武之地。

中国「天河二号」超算协助搭建起「15秒断诊」的新冠CT影像智能诊断平台，并助力筛选能抑制病毒的小分子药物。美国Summit超算同样参与到新药研制中，模拟新冠病毒与不同化合物的反应。2020/2021夺冠的日本Fugaku超算也为探索新药启动了试验。

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	Sierra - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

数值计算方法的历史：

- 有不少计算方法是非常传统的，它们在18-19世纪或者更早些时候就已经建立。**20世纪40年代电子计算机的问世及之后的飞速发展**，为大规模的数值计算创造了条件，也使得集中而系统的研究适用于数字计算机的数值算法变得十分迫切和必要。
- 数值分析作为一门学科分支，不仅仅是一些数值方法的简单积累，而**需要揭示包含在多种数值方法之间的相同结构和统一的原理**。它在大量数值计算实践和理论分析工作的相互促进中迅速发展着。有些原有方法沿用至今，有一些被淘汰，而新的方法和理论不断诞生。

- 数值计算的早期历史：

H. H. Goldstine. *A history of Numerical Analysis from the 16th through the 19th Century*. Springer-Verlag, New York, 1977.

- 科学计算在计算机时代的历史：

S. G. Nash, editor. *A History of Scientific Computing*. ACM Press, New York, 1990.

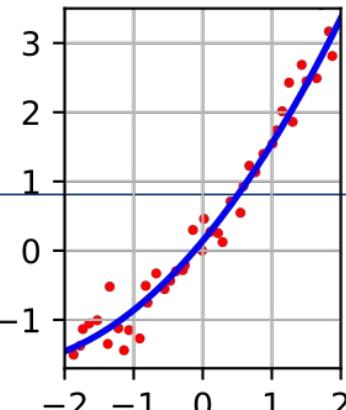
Table of Contents

Table of Contents

Introduction	xii
1. The Sixteenth and Early Seventeenth Centuries	1
1.1. Introduction	1
1.2. Napier and Logarithms	2
1.3. Briggs and His Logarithms	13
1.4. Bürgi and His Antilogarithms	20
1.5. Interpolation	23
1.6. Vieta and Briggs	33
1.7. Kepler	41
2. The Age of Newton	51
2.1. Introduction	51
2.2. Logarithms and Finite Differences	54
2.3. Trigonometric Tables	62
2.4. The Newton–Raphson and Other Iterative Methods	64
2.5. Finite Differences and Interpolation	68
2.6. Maclaurin on the Euler–Maclaurin Formula	84
2.7. Stirling	97
2.8. Leibniz	117
3. Euler and Lagrange	119
3.1. Introduction	119
3.2. Summation of Series	124
3.3. Euler on the Euler–Maclaurin Formula	126
3.4. Applications of the Summation Formula	129
3.5. Euler on Interpolation	137
3.6. Lunar Theory	142
3.7. Lagrange on Difference Equations	145
3.8. Lagrange on Functional Equations	149
3.9. Lagrange on Fourier Series	154
3.10. Lagrange on Partial Difference Equations	158
3.11. Lagrange on Finite Differences and Interpolation	161
3.12. Lagrange on Hidden Periodicities	171
3.13. Lagrange on Trigonometric Interpolation	182

x

4. Laplace, Legendre, and Gauss	185
4.1. Introduction	185
4.2. Laplace on Interpolation	185
4.3. Laplace on Finite Differences	191
4.4. Laplace Summation Formula	192
4.5. Laplace on Functional Equations	195
4.6. Laplace on Finite Sums and Integrals	198
4.7. Laplace on Difference Equations	203
4.8. Laplace Transforms	206
4.9. Method of Least Squares	209
4.10. Gauss on Least Squares	212
4.11. Gauss on Numerical Integration	224
4.12. Gauss on Interpolation	233
4.13. Gauss on Rounding Errors	258
5. Other Nineteenth Century Figures	261
5.1. Introduction	261
5.2. Jacobi on Numerical Integration	261
5.3. Jacobi on the Euler–Maclaurin Formula	266
5.4. Jacobi on Linear Equations	270
5.5. Cauchy on Interpolation	276
5.6. Cauchy on the Newton–Raphson Method	278
5.7. Cauchy on Operational Methods	280
5.8. Other Nineteenth Century Results	284
5.9. Integration of Differential Equations	285
5.10. Successive Approximation Methods	294
5.11. Hermite	298
5.12. Sums	309
Bibliography	316
Index	337



A scatter plot showing red data points forming a curve, with a solid blue line representing a polynomial fit passing through the origin.

Colin Maclaurin



Drawing by [David Steuart Erskine](#) c. 1795, from a portrait by [James Ferguson](#)

Born	February 1698 Kilmidan, Cowal, Argyll, Scotland
Died	14 June 1746 (aged 48) Edinburgh, Scotland

Adrien-Marie Legendre



1820 watercolor [caricature](#) of Adrien-Marie Legendre by French artist [Julien-Léopold Boilly](#) (see [mistaken portrait](#)), the only existing portrait known^[1]

Born	18 September 1752 Paris, France
Died	9 January 1833 (aged 80) Paris, France

Leonhard Euler



Portrait by [Jakob Emanuel Handmann](#) (1753)

Born	15 April 1707 Basel, Switzerland
Died	18 September 1783 (aged 76)

Carl Friedrich Gauss



Portrait of Gauss by [Christian Albrecht Jensen](#) (1840)

Born	Johann Carl Friedrich Gauss 30 April 1777 Brunswick, Principality of Brunswick-Wolfenbüttel
Died	23 February 1855 (aged 77)

Joseph-Louis Lagrange



Joseph-Louis (Giuseppe Luigi), comte de Lagrange

Born	Giuseppe Lodovico Lagrangia 25 January 1736 Turin, Kingdom of Sardinia
Died	10 April 1813 (aged 77) Paris, First French Empire

Augustin-Louis Cauchy



Cauchy around 1840. Lithography by Zéphirin Belliard after a painting by Jean Roller.

Born	21 August 1789 Paris, France
Died	23 May 1857 (aged 67) Sceaux, France

Pierre-Simon Laplace



Pierre-Simon Laplace as Chancellor of the Senate under the [First French Empire](#)

Born	23 March 1749 Beaumont-en-Auge, Normandy, Kingdom of France
Died	5 March 1827 (aged 77) Paris, Kingdom of France

Carl Gustav Jacob Jacobi



Born	10 December 1804 Potsdam, Kingdom of Prussia
Died	18 February 1851 (aged 46) Berlin, Kingdom of Prussia

朗道《力学》：“理论物理的目标是建立物理定律，即建立物理量之间的关系。确定物理量的具体数值一般不是理论物理的任务，实验在处理这些问题方面相对比较容易，因为在绝大多数情况下，实验不必进行花费大量时间和人力的计算。当然，用理论可以直接算出数值的简单的情况除外”。在朗道看来，从理论出发算出数值的结果是很困难，朗道之所以有这样的论断是因为他写《力学》这本书的时候是在上世纪中叶，那时的计算手段很落后，数值计算比实验还要困难。但我们今天借助庞大的计算机，可以做很多前人想都无法想象的事情。

这门课如果是数学系开可能会被称为Numerical analysis，如果是计算机系开，可能会被称为Numerical computing，我们物理学院开，称之为**计算物理**。事实上，**随着计算机技术的发展，计算物理是作为实验物理、理论物理之外的第三大分支存在的**。它的作用是让我们通过数值的方法去解决我们在科研、工程中所遇到的各种物理问题。**这与我们学院开的另外一门课，数学物理方法正好相辅相成**。数理方法是要通过解析的手段去解决物理问题。

ТЕОРЕТИЧЕСКАЯ ФИЗИКА ТОМ I

Л. Д. ЛАНДАУ
Е. М. ЛИФШИЦ

МЕХАНИКА

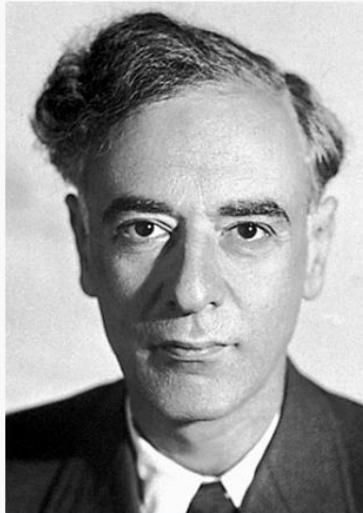
理论物理学教程 第一卷

力学 (第五版)

Л. Д. 朗道 Е. М. 栗弗席兹 著 李俊峰 鞠国兴 译校

高等教育出版社

Lev Landau



Born

Lev Davidovich Landau
22 January 1908

Baku, Baku Governorate,
Russian Empire

Died

1 April 1968 (aged 60)
Moscow, Soviet Union

理论物理学教程·力学

众所周知,物理学是由实验物理和理论物理两个学科组成的.我们已知的大量物理定律可以由为数不多的最一般规律推演出来.这样的推演和最一般规律的建立需要独特的方法,这些构成了特别的学科——理论物理学的任务.

理论物理利用数学手段和方法得出自己的结果和结论,但理论物理与数学的截然不同在于前者与实验结果的直接联系,且不说最一般规律的建立只能以实验数据为基础,甚至从一般规律中得到推论也需要对现象做预先的实验研究,没有这样的实验研究,就无法从大量参与因素中判断哪些因素是重要的,哪些因素是可以忽略的.在得到只考虑重要因素的方程之后,从本质上说,理论物理的任务就基本完成了.进一步应用这些方程于复杂程度不同的具体情况很快成为数学的研究对象,由称为数学物理的一个数学分支来研究.

理论物理的目标是建立物理定律,即建立物理量之间的关系.确定物理量的具体数值一般不是理论物理的任务,实验在处理这些问题方面相对比较容易,因为在绝大多数情况下,实验不必进行花费大量时间和人力的类似的计算.当然,用理论可以直接算出数值的简单情况除外^①.

必须指出,由于理论物理的任务是建立刻画给定现象的物理量之间的联系,因此只有在自然界确实存在这种联系时,才能建立理论.但是,经常是我们感兴趣的物理量之间毫无关系,亦即在不同的自然现象中可以在极为不同的组合中遇到这些物理量.因此,缺乏某个现象的理论并不意味着它无法解释.如同在其它情况下规律性可以由最一般的规律得出一样,在这种情况下不存在规律性的

① 朗道写这个序言是在 1940 年,那时的计算手段很落后,所以数值计算比实验还困难.——译者注

数值计算于国防: 氢弹之父于敏

【校友撷英】于敏：民族的脊梁北大人的榜样：“而国内当时(1965年左右)仅有一台每秒万次的电子管计算机，并且95%的时间分配给有关原子弹的研究，只剩下5%的时间留给于敏负责的氢弹设计。”



研制原子弹时也使用更酷的手摇计算机

手摇计算机是1878年由一位在俄国工作的瑞典发明家奥涅尔制造的。

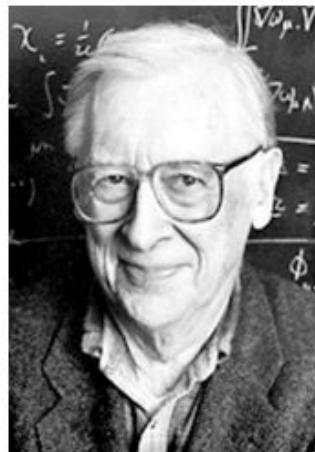
手摇计算机一般只能做四则运算、平方数、立方数、开平方、开立方。如果需要输入三角函数和对数，都需要查表。如果计算中有括号，麻烦极了。使用中必须正摇几圈，反摇几圈，还要用纸笔不断记录。



数值计算于科研：1998和2013年诺贝尔化学奖



Walter Kohn



John A. Pople



Martin Karplus



Michael Levitt

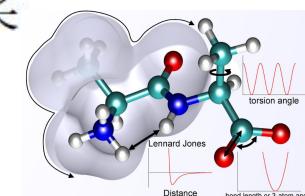


Arieh Warshel

Walter Kohn “for his development of the density-functional theory” and John A. Pople “for his development of computational methods in quantum chemistry”.

“for the development of multiscale models for complex chemical systems”.

这三位都是分子模拟的开山鼻祖，他们开创的领域广泛应用于化学、材料、生物，终于在30多年后获得了诺贝尔奖



The Nobel Prize in Physics 1999

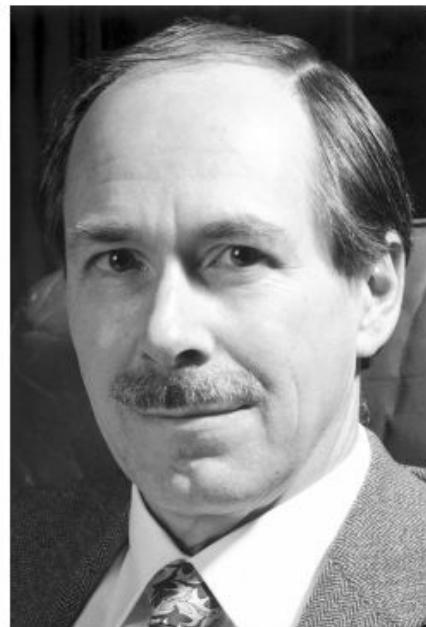


Photo from the Nobel Foundation archive.

Gerardus 't Hooft

Prize share: 1/2

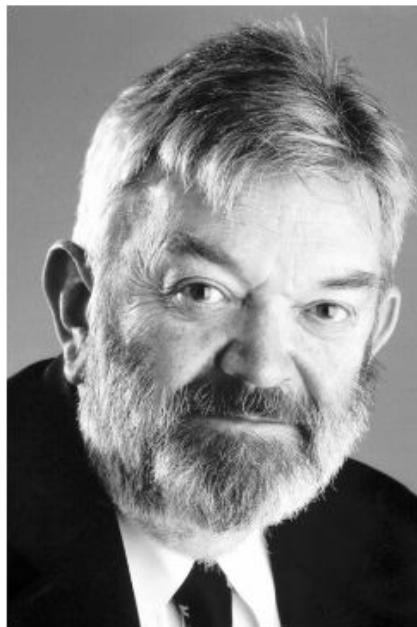


Photo from the Nobel Foundation archive.

Martinus J.G. Veltman

Prize share: 1/2

Tini Veltman (1931-2021): From Assembly Language to a Nobel Prize 汇编语言 符号运算

January 21, 2021

[link](#)

CERN COMPUTER CENTRE
PROGRAM LIBRARY
In Addition to LONG WRITE-UP



SCHOONSCHIP, A PROGRAM FOR ALGEBRAIC MANIPULATIONS

Written in 1967 by M. Veltman at CERN. Version of January 1, 1975.

SCHOONSCHIP is a computer program, mainly written in COMPASS, the CDC 6000 and 7000 assembler language. It is able to perform algebraic manipulations, which means that it deals with formulae rather than with numbers.

SCHOONSCHIP is likely to be of help when the problem to be solved meets the following specifications:

- * It must be a long, but in principle straightforward, algebraic calculation. A computation can easily involve 100,000 terms. The program performs a purely mechanical task of substituting expressions, comparing and selecting terms, re-arranging function arguments, etc.
- * The problem must be formulated by specifying an expression followed by a set of substitutions or commands to be applied on it.

The following types of operations are built in:

The Nobel Prize in Physics 1999 was awarded jointly to Gerardus 't Hooft and Martinus J.G. Veltman "for elucidating the quantum structure of electroweak interactions in physics."

Stephen Wolfram:

Within days after finishing my PhD in theoretical physics at Caltech in November 1979, I flew to Geneva, Switzerland, to visit CERN for a couple of weeks. **And it was during that visit that I started designing SMP (“Symbolic Manipulation Program”)—the system that would be the forerunner of Mathematica and the Wolfram Language.** And when I mentioned what I was doing to people at CERN they said “You should talk to Tini Veltman”.



Stephen Wolfram



Wolfram in 2008

Born	29 August 1959 (age 62) London, England
Nationality	British, American
Education	Dragon School ^[1] Eton College
Alma mater	St. John's College, Oxford (no degree) California Institute of Technology (PhD, 1980)
Known for	Mathematica

IEEE, 也就是电器与电子工程师学会主办的《Computing in science & engineering》杂志曾经选出过 20世纪的十大算法, 也就是对科学和工程发展影响最大的十个算法:

1. 1946 年, 美国 Los Alamos 国家实验室的 John Von Neumann(冯诺伊曼)、S. Ulam、N. Metropolis 开创的 **Metropolis 算法, 也就是蒙特卡罗方法**。
2. 1947 年, 美国 RAND 公司的 G. Dantzig 开创的 Simplex 算法 (解线性规划问题)
3. 1950 年, 美国国家标准局数值分析研究所的 M. Hestenes、E. Stiefel 和 C. Lanczos 开创的 **Krylov 子空间迭代法 (例如: 解稀疏矩阵本征值和本征矢量)**
4. 1951 年, 美国 Oak Ridge 国家实验室的 A. Householder(豪斯霍尔德) 形式化的 **矩阵计算的分解方法, 也称 Householder 约化**
5. 1957 年, 美国 IBM 公司的 J. Backus 领导开发的 **FORTRAN 最优编译器算法**
6. 1959 - 1961 年, 英国伦敦 Ferranti 公司的 J. Francis 开创的 **QR 算法, 是一种计算矩阵特征值的稳定算法** (如果待求的矩阵不是特别巨大 ($n \geq 3000$), 那么 QR 算法实际上是最为合适的算法。)
7. 1962 年, 英国伦敦 Elloit Brothers 公司的 T. Hoare 提出 Quicksort 算法, 也即快速排序算法 (这个算法可能大家比较熟悉, 从数列中挑出一个元素, 比这个基准元素小的数放在基准前面, 大的数放在基准后面, 然后进行递归操作就行了。)
8. 1965 年, 美国 IBM 公司的 J. Cooley 和普林斯顿大学及美国贝尔实验室的 J. Tukey 共同提出的 **快速傅立叶变换算法: FFT 算法**
9. 1977 年, 美国 Brigham Young 大学的 H. Ferguson 和 R. Forcade 提出的整数检测算法 (integer relation detection)
10. 1987 年, 美国耶鲁大学的 L. Greengard 和 V. Rokhin 发明的快速多极算法 (fast multipole)

1947: George Dantzig(丹齐克), at the RAND Corporation(兰德公司), creates the simplex method for linear programming.

In terms of widespread application, Dantzig's algorithm is one of the most successful of all time: Linear programming dominates the world of industry, where economic survival depends on the ability to optimize within budgetary and other constraints. (Of course, the “real” problems of industry are often nonlinear; the use of linear programming is sometimes dictated by the computational budget.) **The simplex method is an elegant way of arriving at optimal answers.** Although theoretically susceptible to exponential delays, the algorithm in practice is highly efficient—which in itself says something interesting about the nature of computation.

George Dantzig



Dantzig with President Gerald Ford in 1976

Born	George Bernard Dantzig November 8, 1914 Portland, Oregon, US
Died	May 13, 2005 (aged 90) Stanford, California, US
Citizenship	American
Alma mater	University of Maryland (BS) University of Michigan (MS) University of California, Berkeley (PhD)

假设有n个变量和m个约束。线性规划的标准形式如下：

$$\begin{aligned} & \max \sum_{1 \leq k \leq n} c_k x_k \\ \text{s.t. } & \sum_{1 \leq k \leq n} A_{1,k} x_k \leq b_1, \\ & \sum_{1 \leq k \leq n} A_{2,k} x_k \leq b_2, \\ & \dots \\ & \sum_{1 \leq k \leq n} A_{m,k} x_k \leq b_m \\ & x_1, x_2, \dots, x_n \geq 0 \end{aligned}$$

1951: Alston Householder of Oak Ridge National Laboratory formalizes the decompositional approach to matrix computations.

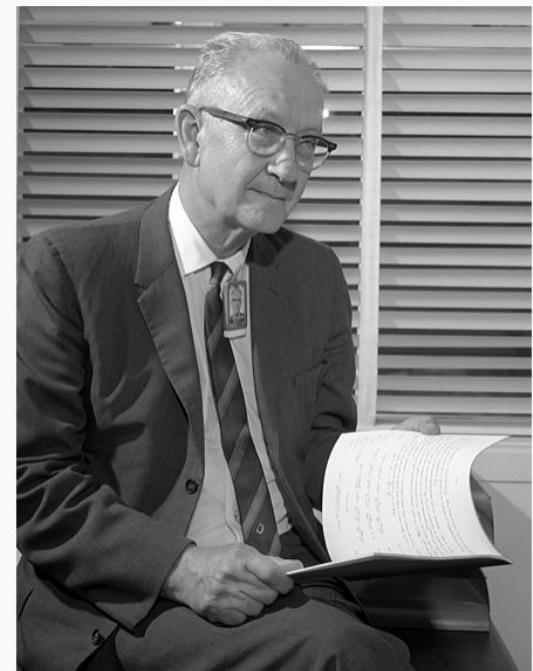
The ability to factor matrices into triangular, diagonal, orthogonal, and other special forms has turned out to be extremely useful.

The decompositional approach has enabled software developers to produce flexible and efficient matrix packages.

It also facilitates the analysis of rounding errors, one of the big bugbears of numerical linear algebra.

(In 1961, James Wilkinson of the National Physical Laboratory in London published a seminal paper in the Journal of the ACM, titled “Error Analysis of Direct Methods of Matrix Inversion,” based on the LU decomposition of a matrix as a product of lower and upper triangular factors.)

Alston Scott Householder



Born	5 May 1904
Died	4 July 1993 (aged 89)
Alma mater	University of Chicago
Known for	Householder transformation

1950: Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of Krylov subspace iteration methods.

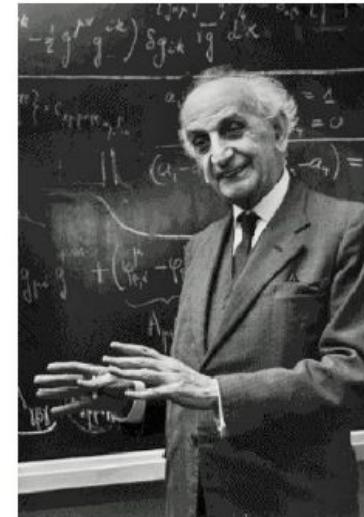
These algorithms address the seemingly simple task of solving equations of the form $\mathbf{Ax} = \mathbf{b}$. The catch, of course, is that \mathbf{A} is a huge $n \times n$ matrix, so that the algebraic answer $\mathbf{x} = \mathbf{b}/\mathbf{A}$ is not so easy to compute. (Indeed, matrix “division” is not a particularly useful concept.)

Iterative methods—such as solving equations of the form $\mathbf{Kx}_{i+1} = \mathbf{Kx}_i + \mathbf{b} - \mathbf{Ax}_i$ with a simpler matrix \mathbf{K} that’s ideally “close” to \mathbf{A} —lead to the study of Krylov subspaces. Named for the Russian mathematician Nikolai Krylov, Krylov subspaces are spanned by powers of a matrix applied to an initial “remainder” vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$.

Lanczos found a nifty way to generate an orthogonal basis for such a subspace when the matrix is symmetric. Hestenes and Stiefel proposed an even niftier method, known as the conjugate gradient method, for systems that are both symmetric and positive definite.



Magnus Hestenes



Cornelius
Lanczos
(1893-1974)

Eduard Stiefel (1909-1978)

1977: Helaman Ferguson and Rodney Forcade of Brigham Young University advance an integer relation detection algorithm.

The problem is an old one: **Given a bunch of real numbers, say x_1, x_2, \dots, x_n , are there integers a_1, a_2, \dots, a_n (not all 0) for which**

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = 0?$$

For $n=2$, the venerable [Euclidean algorithm](#) (计算最大公约数) does the job, computing terms in the continued-fraction expansion of x_1/x_2 . If x_1/x_2 is rational, the expansion terminates and, with proper unraveling, gives the “smallest” integers a_1 and a_2 . If the Euclidean algorithm doesn’t terminate—or if you simply get tired of computing it—then the unraveling procedure at least provides lower bounds on the size of the smallest integer relation.

Ferguson and Forcade’s generalization, although much more difficult to implement (and to understand), is also more powerful. Their detection algorithm, for example, has been used to find the precise coefficients of the polynomials satisfied by the third and fourth bifurcation points, $B_3 = 3.544090$ and $B_4 = 3.564407$, of the logistic map. ([The latter polynomial is of degree 120; its largest coefficient is \$257^{30}\$.](#)) **It has also proved useful in simplifying calculations with Feynman diagrams in quantum field theory.**

GENERALIZATION OF THE EUCLIDEAN ALGORITHM FOR
REAL NUMBERS TO ALL DIMENSIONS HIGHER THAN TWO

BY H. R. P. FERGUSON AND R. W. FORCADE

[Bull. Amer. Math. Soc. \(N.S.\) 1\(6\):
912-914 \(November 1979\).](#)

Empirical determinations of Feynman integrals using integer relation algorithms

Kevin Acres, David Broadhurst

Integer relation algorithms can convert numerical results for Feynman integrals to exact evaluations, when one has reason to suspect the existence of reductions to linear combinations of a basis, with rational or algebraic coefficients. Once a tentative reduction is obtained, confidence in its validity is greatly increased by computing more decimal digits of the terms and verifying the stability of the result. Here we give examples of how the PSLQ and LLL algorithms have yielded remarkable reductions of Feynman integrals to multiple polylogarithms and to the periods and quasi-periods of modular forms. Moreover, these algorithms have revealed quadratic relations between Feynman integrals. A recent application concerning black holes involves quadratic relations between combinations of Feynman integrals with algebraic coefficients.

1987: Leslie Greengard and Vladimir Rokhlin of Yale University invent the fast multipole algorithm.

This algorithm overcomes one of the biggest headaches of **N-body simulations**: the fact that accurate calculations of the motions of N particles interacting via gravitational or electrostatic forces (think stars in a galaxy, or atoms in a protein) would seem to require $O(N^2)$ computations—one for each pair of particles.

The fast multipole algorithm gets by with $O(N)$ computations. It does so by using multipole expansions (net charge or mass, dipole moment, quadrupole, and so forth) to approximate the effects of a distant group of particles on a local group. A hierarchical decomposition of space is used to define ever-larger groups as distances increase. **One of the distinct advantages of the fast multipole algorithm is that it comes equipped with rigorous error estimates, a feature that many methods lack.**

Fast Multipole Methods for N -body Simulations of Collisional Star Systems

Diptajyoti Mukherjee, Qirong Zhu, Hy Trac, Carl L. Rodriguez

Direct N -body simulations of star clusters are accurate but expensive, largely due to the numerous $\mathcal{O}(N^2)$ pairwise force calculations. To solve the post-million-body problem, it will be necessary to use approximate force solvers, such as tree codes. In this work, we adapt a tree-based, optimized Fast Multipole Method (FMM) to the collisional N -body problem. The use of a rotation-accelerated translation operator and an error-controlled cell opening criterion leads to a code that can be tuned to arbitrary accuracy. We demonstrate that our code, Taichi, can be as accurate as direct summation when $N > 10^4$. This opens up the possibility of performing large- N , star-by-star simulations of massive stellar clusters, and would permit large parameter space studies that would require years with the current generation of direct summation codes. Using a series of tests and idealized models, we show that Taichi can accurately model collisional effects, such as dynamical friction and the core-collapse time of idealized clusters, producing results in strong agreement with benchmarks from other collisional codes such as NBODY6++GPU or PeTar. Parallelized using OpenMP and AVX, Taichi is demonstrated to be more efficient than other CPU-based direct N -body codes for simulating large systems. With future improvements to the handling of close encounters and binary evolution, we clearly demonstrate the potential of an optimized FMM for the modeling of collisional stellar systems, opening the door to accurate simulations of massive globular clusters, super star clusters, and even galactic nuclei.

QR算法：计算矩阵特征值

John G.F. Francis (born 1934) is an English computer scientist, who in 1961 published the QR algorithm for computing the eigenvalues and eigenvectors of matrices,^[1] which has been named as one of the ten most important algorithms of the twentieth century.^{[2][3]} The algorithm was also proposed independently by Vera N. Kublanovskaya of the Soviet Union in the same year.^[4]

Francis was born in London in 1934. In 1954 he worked for the National Research Development Corporation (NRDC). In 1955–1956 he attended Cambridge University, but did not complete a degree. He then returned to the NRDC, where he served as assistant to Christopher Strachey. At this time he devised the QR transformation. In 1961 he left the NRDC to work at Ferranti Corporation, Ltd. and then at the University of Sussex. Subsequently, he had positions with various industrial organizations and consultancies. His interests encompassed artificial intelligence, computer languages, and systems engineering, although he never returned to the field of numerical computation.^[5]

By 1962, Francis had left the field of numerical analysis, and subsequently had no idea of the impact his work on the QR algorithm had had, until re-contacted by Gene Golub and Frank Uhlig in 2007, by which time he was retired and living in Hove, England (near Brighton).^[5] Still in good health, he was the opening speaker at a mini-symposium that marked 50 years of the QR algorithm, held at the 23rd Biennial Conference on Numerical Analysis in Glasgow in June 2009.^[6] Francis was awarded a University of Sussex honorary doctorate in July 2015.^[7]

John G.F. Francis	
Born	1934 London
Known for	QR algorithm



John Francis in July 2008

好的数值算法的特点

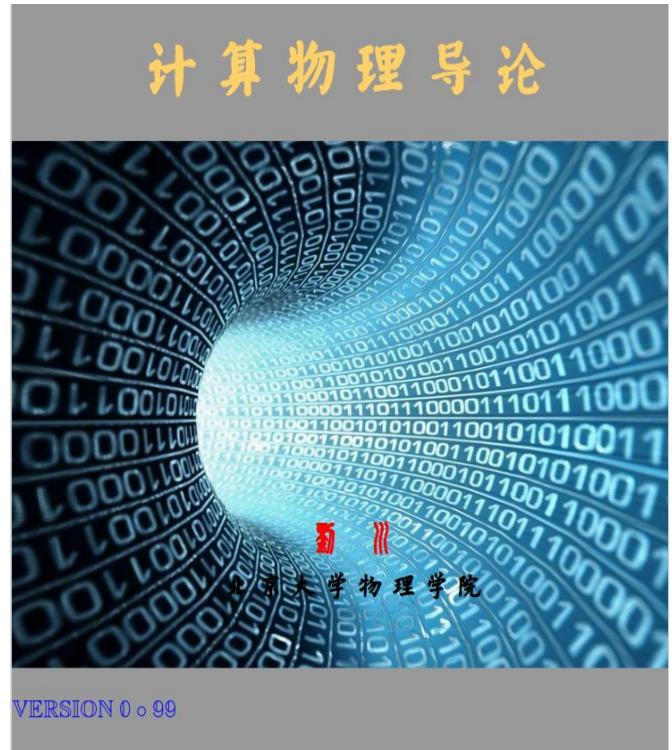
- 面向计算机，提供切实可行的算法。只能包括计算机能够直接处理的加减乘除运算、逻辑运算、以及内置的简单函数计算等。
- 能任意逼近，并有可靠的理论分析(收敛性、稳定性、误差分析)，能达到精度要求，对近似算法要保证收敛性和稳定性。
- 省时间、省资源，并有良好的计算复杂性。计算复杂性是指算法所包含的运算次数和所需的储存量。好的算法计算效率高、计算复杂度低(耗时少，存储小)。
- 多次广泛的上机试验证明算法稳定、高效。

简单来说，好的数值算法本质上有三个要求：好，快，省。好，指算的准确，误差小，计算效果好；快，指计算步骤少，效率高；省，指计算的时间和存储代价小。【参考书】N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 1996.

- 国内外各个大学的计算物理(或者以其他名称出现但实际上与计算物理等价)课程是所有课程中**差异性最大的一门课程**。我们这个课程经过讨论后形成的共识是, 计算物理课程**主要侧重于物理学各个领域都会遇到的、具有普遍性的数值问题**。
- 而一些更加具体的、仅仅涉及到某个物理学领域的问题则可以放在更加专门的计算物理+ 的课程中。
- 计算物理当然是需要利用计算机进行求解的问题。这就涉及到人与机器的结合等多个问题:用什么机器, 用什么语言或者用什么软件等等。**在计算物理中, 我们更侧重于问题的基本解决的思路**(也就是算法方面更侧重一些), 至于说该问题的求解用什么语言则交给同学们自己选择。
- 我们课程的主要目并不是教会同学们如何使用某个具体软件包, 而是希望让大家了解数值计算的一些基本规则和方法。诚然, 多数目前的商业软件包都提供了不错的用户界面, 但是本课程的目的是让大家了解一下这些界面后面的一些东西。**在你以后面对完全不同的应用时, 可以知道如何去求解一个全新的数值问题。**

1. “计算”介绍
2. 数值计算基础
3. 线性方程组求解
4. 内插与函数的计算
5. 数值积分
6. 方程求根与函数极值
7. 本征值与本征矢量

上半部分



参考材料：

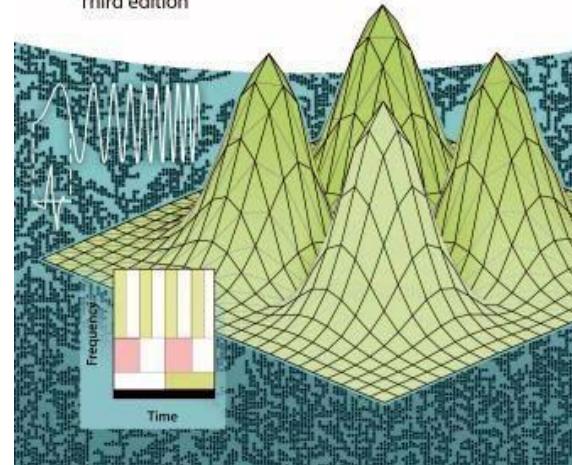
- W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery: **Numerical Recipes: The art of scientific computing**, 3rd Ed., Cambridge University Press, 2007.
- R. L. Burden and J. D. Faires, **Numerical Analysis**, 9th ed., Brooks/Cole, Cengage Learning, 2011.
- L. R. Scott, **Numerical Analysis**, Princeton University Press, 2011.

Rubin H. Landau, Manuel J. Páez
and Cristian C. Bordeianu

Computational Physics

Problem Solving with Python

Third edition



Landau Book

1. 常微分方程
2. 偏微分方程
3. 快速傅里叶变换
4. 随机数
5. 蒙特卡洛方法
6. 有限元方法
7. 机器学习初步

下半部分

参考材料：

科学出版社, 计算物理学, 刘金远等

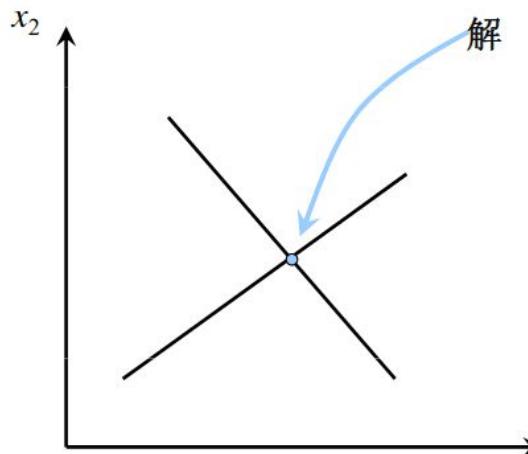
世界图书出版公司, An Introduction to Computational Physics, Tao Pang

科学出版社, 计算物理学, 马文淦

世界图书出版公司, A First Course in Computational Physics and Object-Oriented Programming with C++, David Yevick

线性代数方程组 (System of Linear Algebraic Equations)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= c_1 \\ a_{21}x_1 + a_{22}x_2 &= c_2 \end{aligned}$$



A.X=b
A可以是
巨大矩阵

问：今有

上禾三秉，中禾二秉，下禾一秉，实三十九斗；

上禾二秉，中禾三秉，下禾一秉，实三十四斗；

上禾一秉，中禾二秉，下禾三秉，实二十六斗。

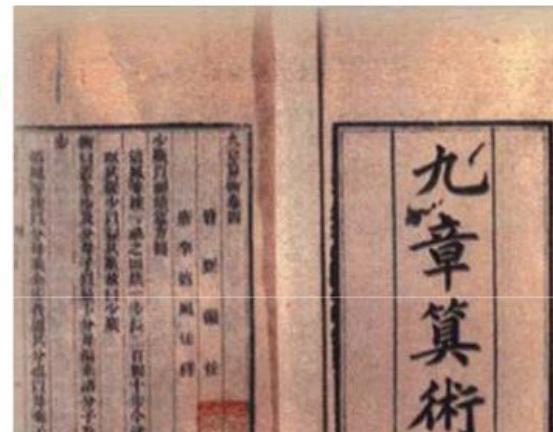
问上、中、下禾实一秉各几何？

——《九章算术》

$$3x + 2y + z = 39$$

$$2x + 3y + z = 34$$

$$x + 2y + 3z = 26$$

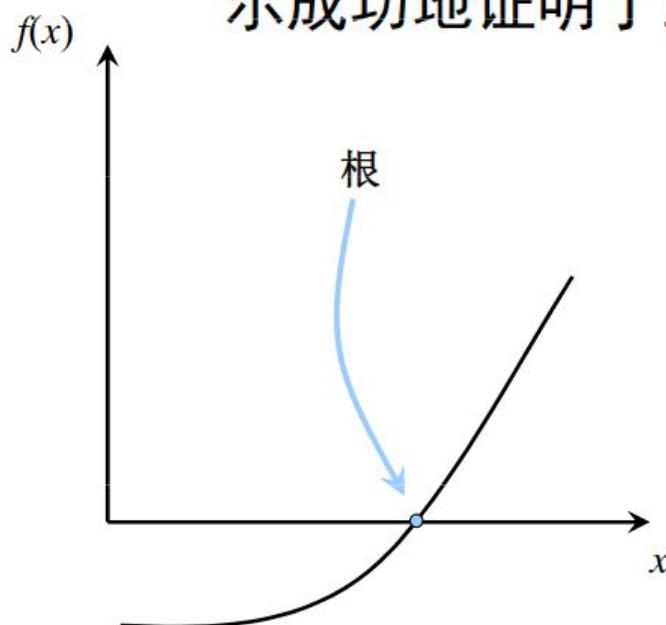


例：代数求根问题

$$f(x) = ax^2 + bx + c = 0$$

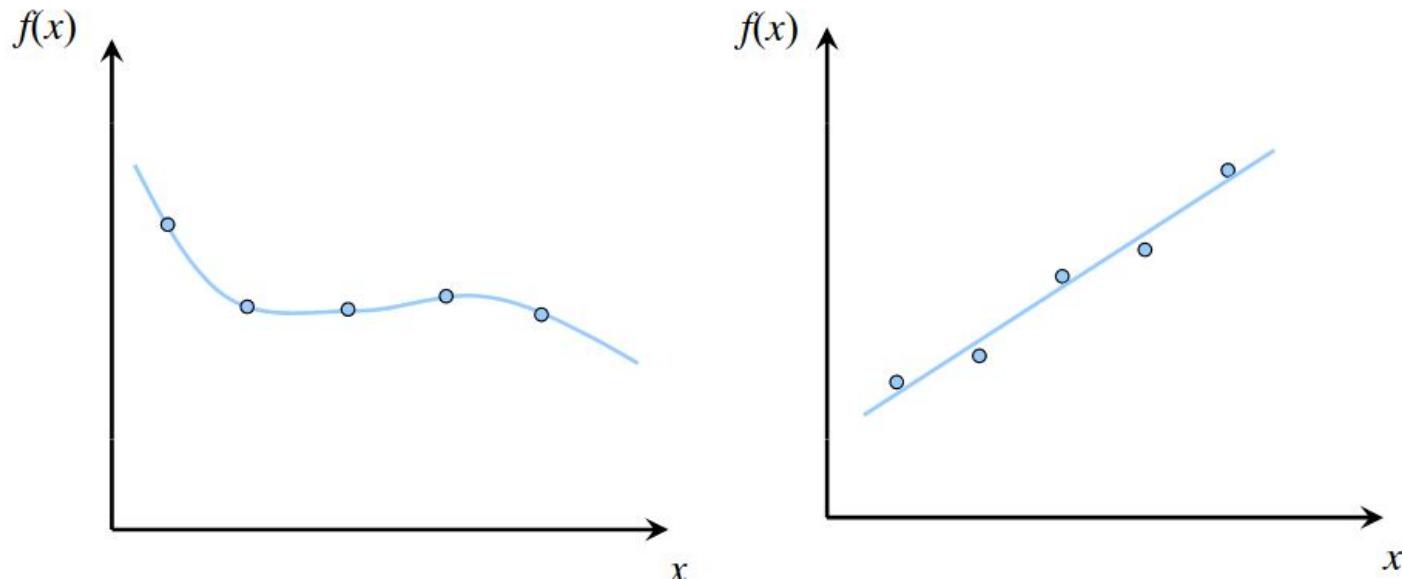
$$\Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- 公元1541年意大利数学家塔尔塔利亚给出了三次方程的一般解法；公元1545年意大利数学家卡尔达诺的名著《大术》记载了费拉里的四次方程的一般解法。
- 公元1778年，法国数学大师拉格朗日提出了五次方程解不存在的猜想。公元1824年，挪威年轻数学家阿贝尔成功地证明了五次以上一般方程没有根式解。



$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \Rightarrow x = ?$$
$$\sin x + x = 0 \Rightarrow x = ?$$

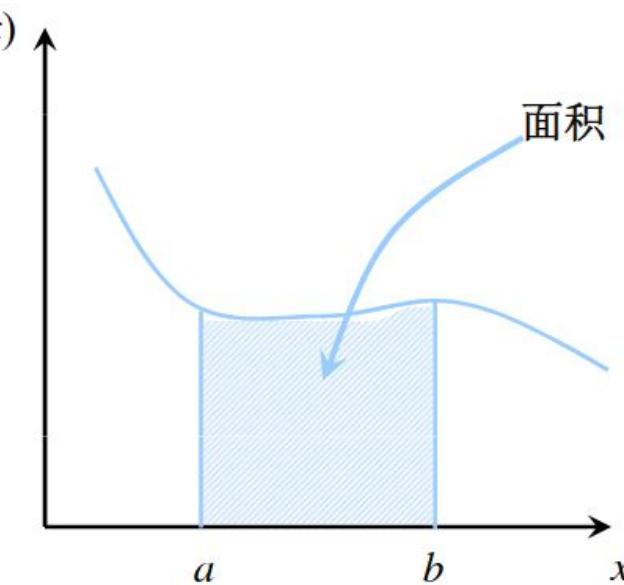
插值(Interpolation)和曲线拟合(Curve Fitting)



数值积分(numerical integration)和数值微分(numerical differentiation)

$$I = \int_a^b f(x)dx = F(x)\Big|_a^b$$

$$\frac{dy}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$

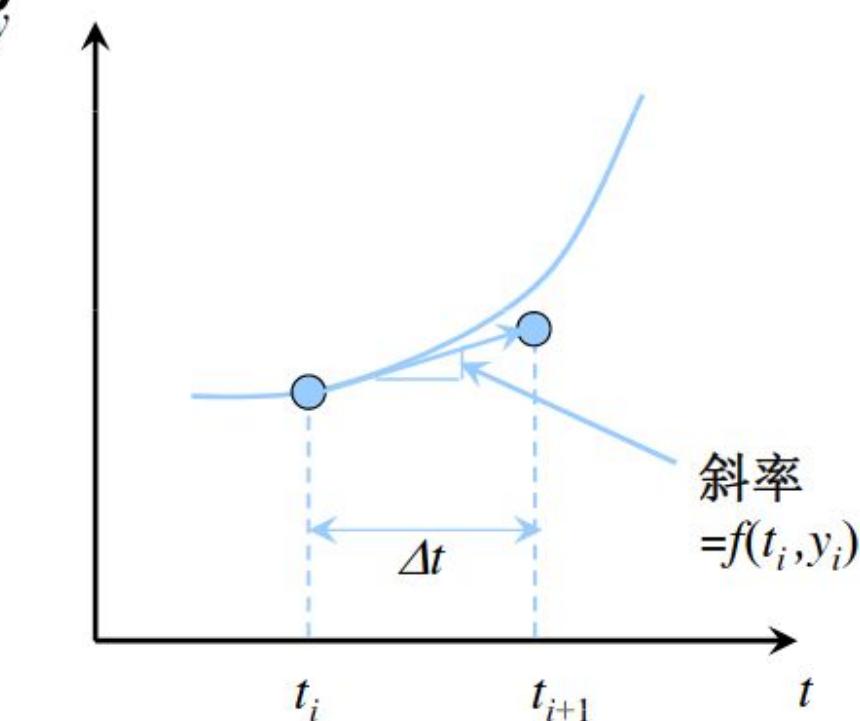


- 常微分方程 (Ordinary Differential Equation)

$$\frac{dy}{dt} = f(t, y)$$

求关于自变量 t 的函数 y

$$y_{i+1} = y_i + f(t_i, y_i) \Delta t$$



初值问题(给出积分曲线在初始点的状态)
边值问题(给出积分曲线首尾两端的状态)

1. “计算”介绍, 常微分方程

Linux操作系统

Ubuntu

Fortran、C++简介

Euler法示例

Python、Numpy简介

CERN ROOT数据处理软件

Histogram, bin, error-bar

Pi计算示例

作图: gnuplot、ROOT、Python

动态图示例

并行计算、版本维护

1983 图灵奖：

Ken Thompson, Dennis M. Ritchie
for their development of generic operating
systems theory and specifically for the
implementation of the UNIX operating
system.



- In 80's, Microsoft's DOS (disk operating system) was the dominated OS for PC
- Apple MAC was better, but expensive
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- **People was looking for a UNIX based system, which is cheaper and can run on PC**
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees



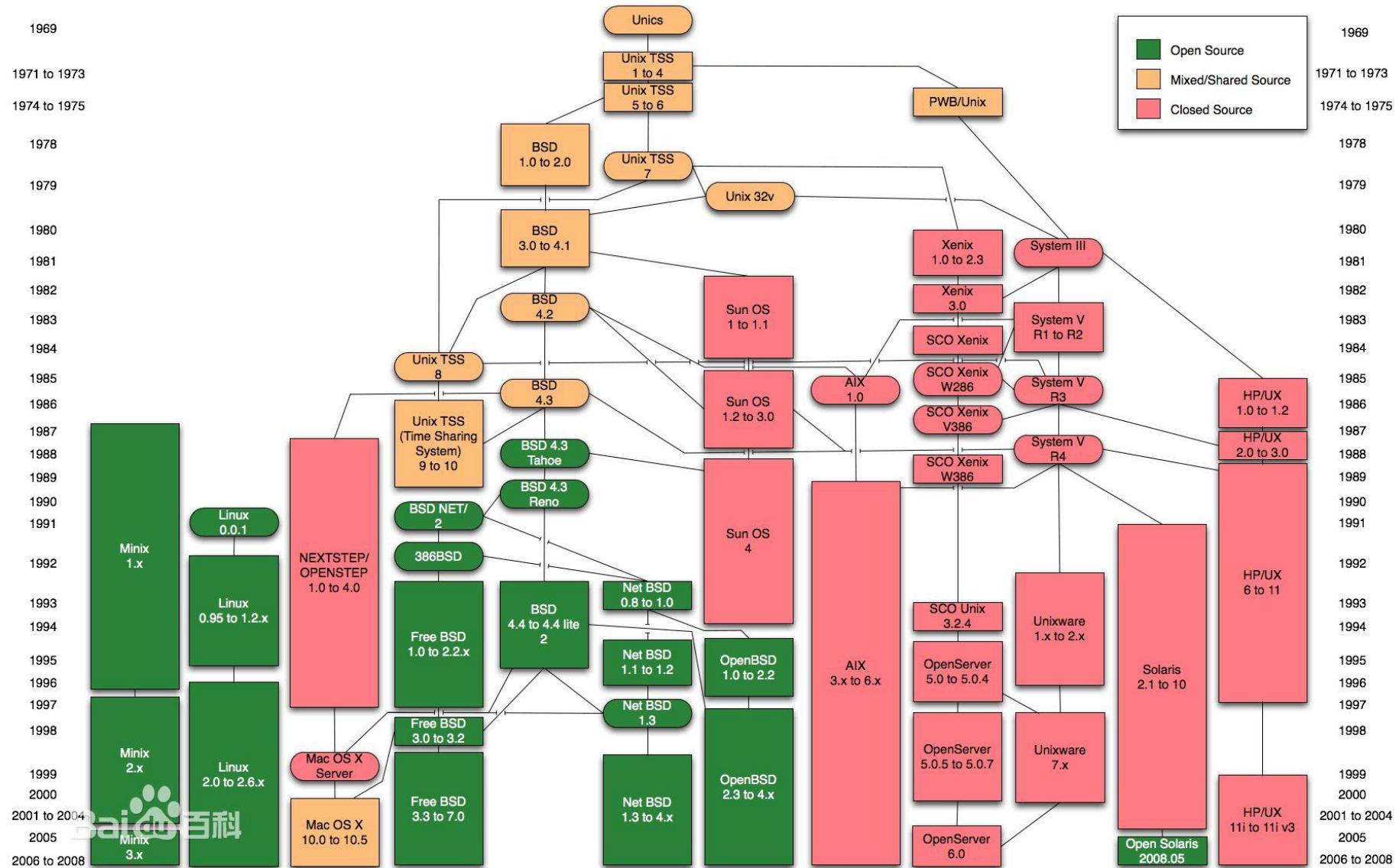
- Established in 1984 by **Richard Stallman**, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
- GNU通用公共许可证(GNU General Public License, GPL)。即“反版权”(或称Copyleft)概念。

1991年Linus Torvalds编写出了与UNIX兼容的Linux操作系统内核并在GPL条款下发布。Linux之后在网上广泛流传，许多程序员参与了开发与修改。1992年Linux与其他GNU软件结合，完全自由的操作系统正式诞生。该操作系统往往被称为“GNU/Linux”或简称Linux。



GNU/Linux
The Soft Revolution





Linux系统

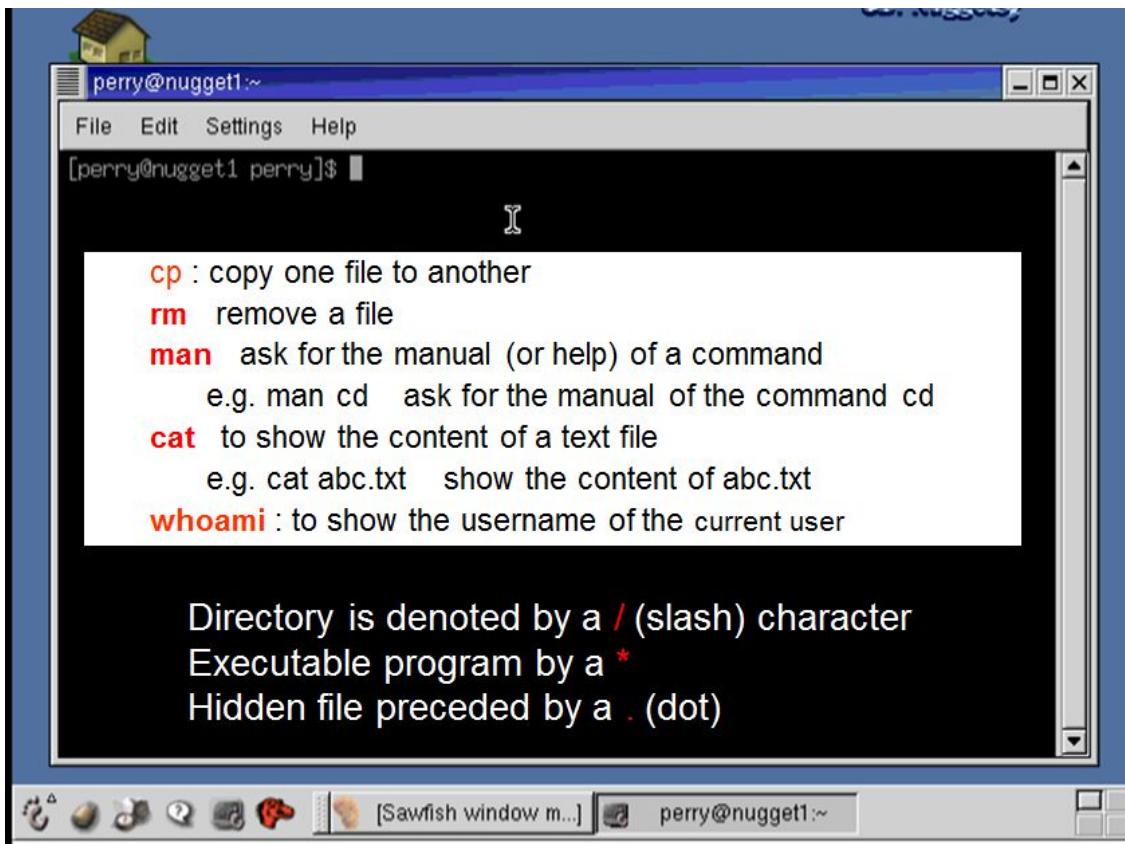
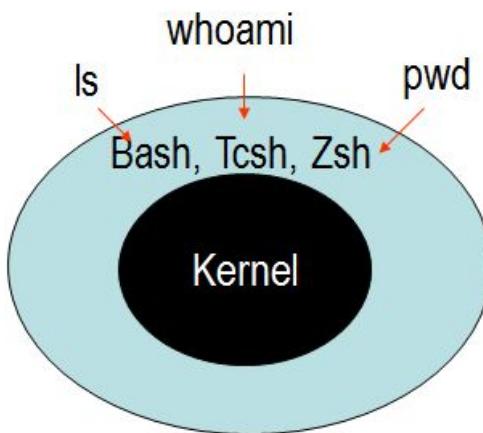
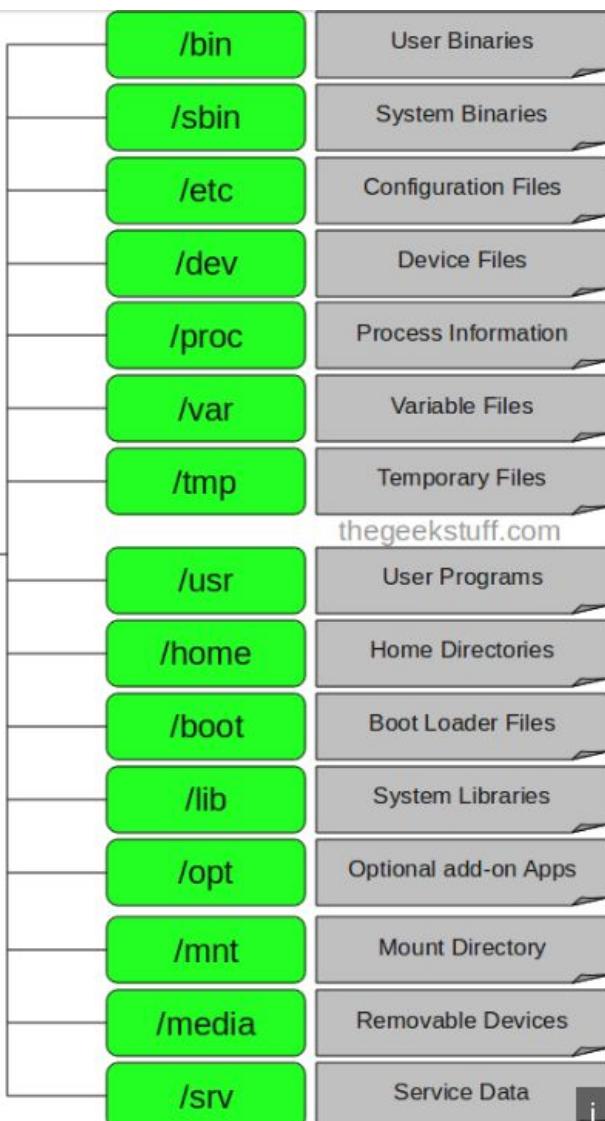
Arch Linux, CentOS, Debian, Fedora, Gentoo Linux, Linux Mint, Mageia, openSUSE and Ubuntu, together with commercial distributions such as Red Hat Enterprise Linux and SUSE Linux Enterprise Server.



Statistics about the Linux distributions

Name	Machine
Ubuntu	39,276
Debian GNU/Linux	26,936
Fedora	10,079
Slackware Linux	9,764
SuSE Linux	9,393
Gentoo Linux	7,413
Arch Linux	4,744
CentOS	4,740
Red Hat Linux	4,672
Kubuntu	2,713
Mandrake	2,645
Mandriva	2,385
Linux Mint	2,248
unknown	2,175
openSUSE	1,750

Linux Shell以及文件系统



Ubuntu是一个以桌面应用为主的开源GNU/Linux操作系统, Ubuntu 是基于Debian GNU/Linux, 支持x86、amd64(即x64)和ppc架构, 由全球化的专业开发团队(Canonical Ltd)打造的



Ubuntu历史版本一览表

版本号	代号	发布时间
17.04	Zesty Zapus	2017/04(即将发布)
16.10	Yakkety Yak	2016/10/20
16.04LTS	Xenial Xerus	2016/04/21
15.10	Wily Werewolf	2015/10/23
15.04	Vivid Vervet	2015/04/22
14.10	Utopic Unicorn	2014/10/23
14.04 LTS	Trusty Tahr	2014/04/18
13.10	Saucy Salamander	2013/10/17
13.04	Raring Ringtail	2013/04/25
12.10	Quantal Quetzal	2012/10/18
12.04 LTS	Precise Pangolin	2012/04/26
11.10	Oneiric Ocelot	2011/10/13
11.04 (Unity成为默认桌面环境)	Natty Narwhal	2011/04/28
10.10	Maverick Meerkat	2010/10/10
10.04 LTS	Lucid Lynx	2010/04/29
9.10	Karmic Koala	2009/10/29
9.04	Jaunty Jackalope	2009/04/23
8.10	Intrepid Ibex	2008/10/30
8.04 LTS	Hardy Heron	2008/04/24
7.10	Gutsy Gibbon	2007/10/18
7.04	Feisty Fawn	2007/04/19
6.10	Edgy Eft	2006/10/26
6.06 LTS	Dapper Drake	2006/06/01
5.10	Breezy Badger	2005/10/13
5.04	Hoary Hedgehog	2005/04/08
4.10 (初始发布版本)	Warty Warthog	2004/10/20

Arguably *the most user-friendly version of Linux.*

Huge repository of (free) software available - b

Ubuntu 16.04 LTS is here

Discover Ubuntu's sixth long-term support release.

[Get Ubuntu 16.04 LTS](#)

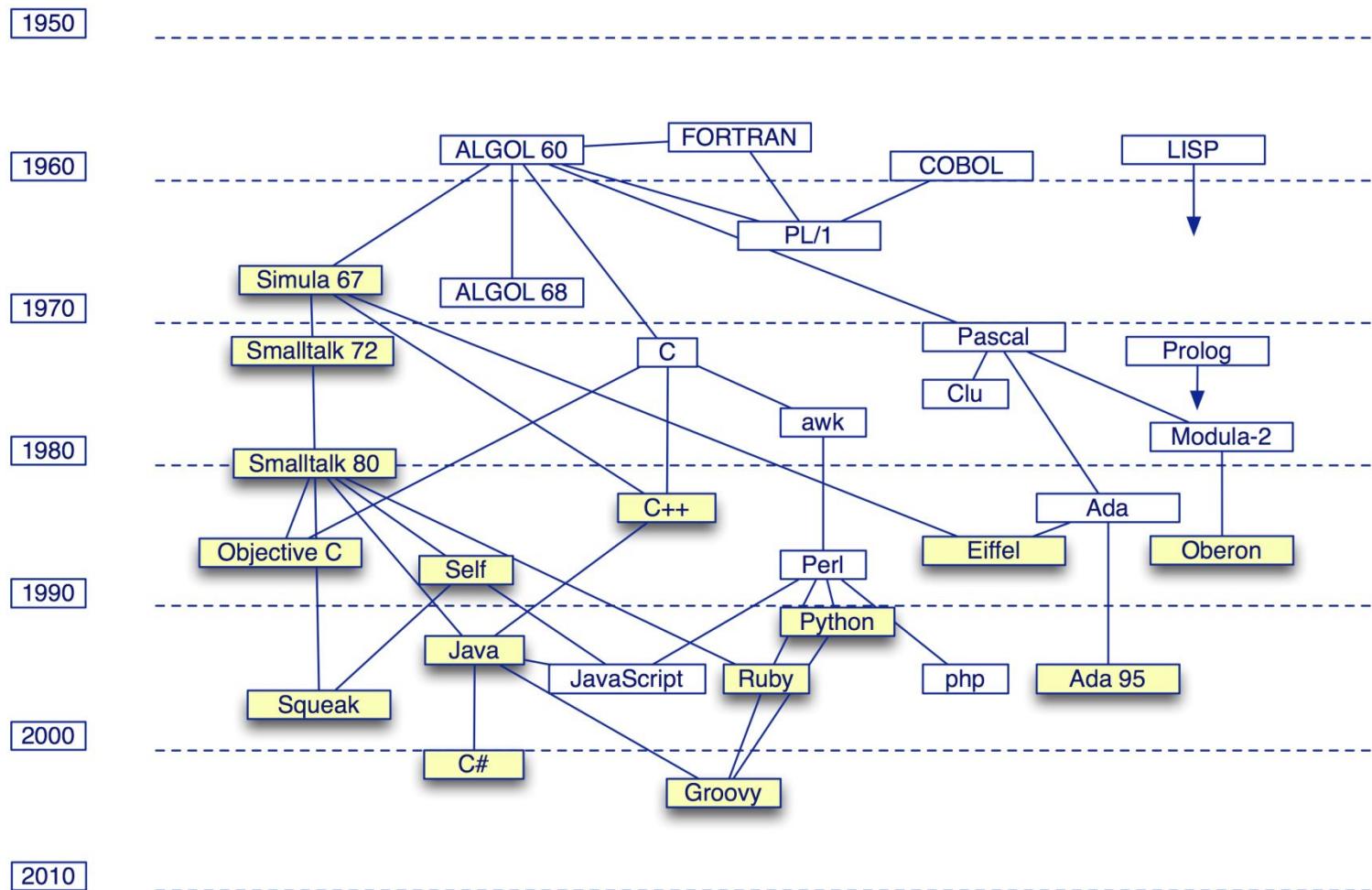
[Learn more about Ubuntu >](#)



可与Windows双系统安装;可硬盘, U盘,光盘安装; 安装过程很自动
安装编译环境

```
sudo apt-get install build-essential  
      gcc, g++,.....  
sudo apt-get install gfortran
```

Object-oriented language genealogy



1979年，Bjarne Stroustrup到了Bell实验室，开始从事将C改良为带类的C(C with classes)的工作。1983年该语言被正式命名为C++。

Simula is considered the first object-oriented programming language. As its name suggests, Simula was designed for doing simulations, and the needs of that domain provided the framework for many of the features of object-oriented languages today.

Simula

Paradigm Object-oriented

Designed by Ole-Johan Dahl, Kristen Nygaard

First appeared 1965

Influenced by

ALGOL 60

Influenced

Object-oriented programming languages



ALAN KAY



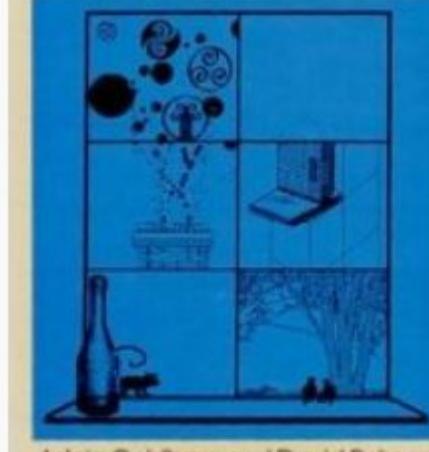
United States – 2003

CITATION

For pioneering many of the ideas at the root of contemporary object-oriented programming languages, leading the team that developed Smalltalk, and for fundamental contributions to personal computing.

SMALLTALK-80

THE LANGUAGE AND ITS IMPLEMENTATION



Paradigm Object-oriented

Designed by Alan Kay, Dan Ingalls, Adele Goldberg

Developer Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Diana Merry, Scott Wallace, Peter Deutsch and Xerox PARC

First appeared 1972; 46 years ago
(development began in 1969)

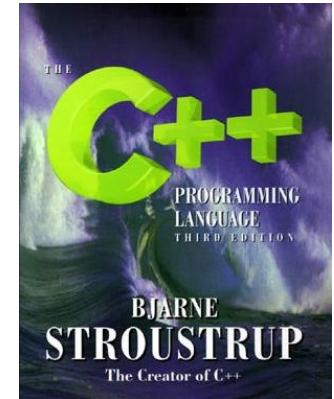
Stable release Smalltalk-80 version 2 / 1980; 38 years ago

C++简介

A “*better C*” that supports:
Systems programming
Object-oriented programming (*classes & inheritance*)
Programming-in-the-large (*namespaces, exceptions*)



Generic programming (*templates*)
Reuse (large class & template libraries)



Most C programs are also C++ programs.

`gcc hello.c -o hello`

“Hello World” in C++

A preprocessor directive

```
#include <stdio.h>
```

Include standard io declarations

```
int main(void)
```

Write to standard output

```
{
```

char array

```
printf("hello, world\n");
```

Indicate correct termination

```
return 0;
```

Use the standard namespace

```
using namespace std;
```

Include standard iostream classes

```
#include <iostream>
```

A C++ comment

```
// My first C++ program!
```

cout is an instance of ostream

```
int main(void)
```

operator overloading (two different argument types!)

```
{
```

cout << "hello world!" << endl;

```
return 0;
```

编译:Use gcc, Visual Studio, etc.

**File types:.cc, .cp, .cpp, .CPP, .cxx, .c++, .C , .h,
.H**

```
1 // Fig. 18.3: fig18_03.cpp
2 // Using an inline function to calculate the volume of a cube.
3 #include <iostream>
4 using std::cout;           ← using avoids repeating std::
5 using std::cin;
6 using std::endl;
7
8 // Definition of inline function cube. Definition of function appears
9 // before function is called, so a function prototype is not required.
10 // First line of function definition acts as the prototype.
11 inline double cube( const double side ) ← inline qualifier
12 {
13     return side * side * side; // calculate the cube of side
14 } // end function cube
15
16 int main()
17 {
18     double sideValue; // stores value entered by user
19
20     for ( int i = 1; i <= 3; i++ )
21     {
22         cout << "\nEnter the side length of your cube: ";
23         cin >> sideValue; // read value from user
24
25         // calculate cube of sideValue and display result
26         cout << "Volume of cube with side "
27             << sideValue << " is " << cube( sideValue ) << endl;
28     }
29
30     return 0; // indicates successful termination
31 } // end main
```

Complete function definition so the compiler knows how to expand a cube function call into its inlined code.

cube function call that could be inlined

Enter the side length of your cube: 1.0
Volume of cube with side 1 is 1

Enter the side length of your cube: 2.3
Volume of cube with side 2.3 is 12.167



JOHN BACKUS

United States – 1977

CITATION

For profound, influential, and lasting contributions to the design of practical high-level programming systems, notably through his work on FORTRAN, and for seminal publication of formal procedures for the specification of programming languages.

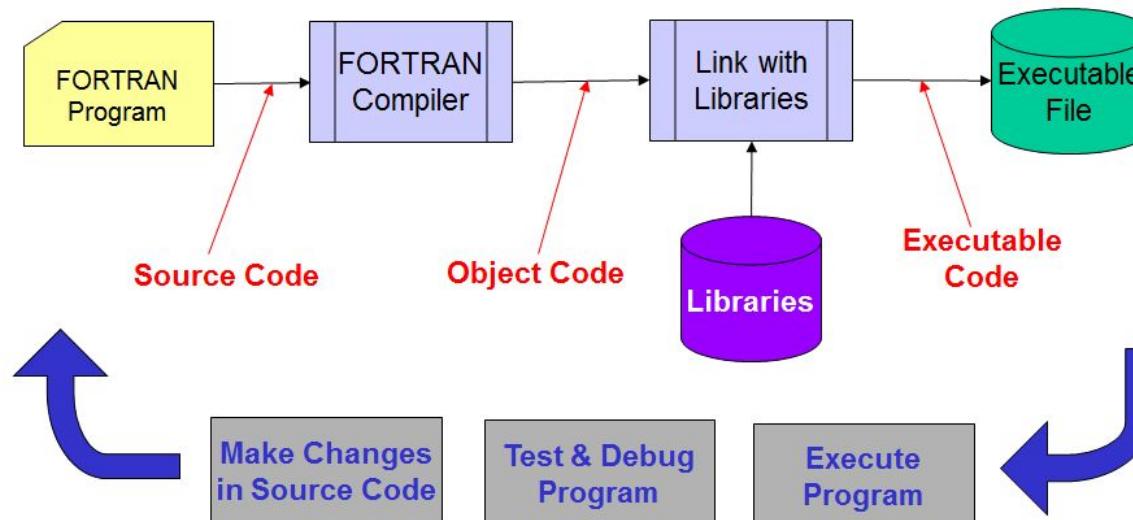
1977图灵奖

- One of the oldest computer languages
 - created by John Backus and released in 1957
 - designed for scientific and engineering computations
- Version history
 - FORTRAN 1957
 - FORTRAN II
 - FORTRAN IV
 - FORTRAN 66 (released as ANSI standard in 1966)
 - FORTRAN 77 (ANSI standard in 1977)
 - FORTRAN 90 (ANSI standard in 1990)
 - FORTRAN 95 (ANSI standard version)
 - FORTRAN 2003 (ANSI standard version)

Fortran

- **FORTRAN was created to solve scientific and engineering problems**
- **Introduced integer and floating point variables**
- Introduced array data types for math computations
- Introduced subroutines and subfunctions
- There is a free compiler in Unix-Linux systems
 - f77, g77 – **g95, gfortran**

- FORTRAN is a compiled language (like C) so the source code (what you write) must be converted into machine code before it can be executed (e.g. Make command)



Fortran 结构

- Skeleton of a program...

```
PROGRAM MAIN
REAL KGLO,FORC,KEL
COMMON /STIF/KGLO(100,100)/LOAD/FORC(100)/DEF/D(100)
C   ...read in data and initialize problem...
DO 100 IELEM=1,NELEMS
C     ...assemble global stiffness matrix...
    CALL KELEM(IELEM,KEL)          Calculate stiffness matrix,
                                   KEL, for a single element
    CALL ASMBK(IELEM,KEL)
100 CONTINUE
DO 200 ILOAD=1,NLOADS
C     ...assemble load vector...
    CALL LODVEC(ILOAD,LOAD)
200 CONTINUE
CALL CONSTR(KDOFS)
CALL SOLVE(NDOFS)
C   ...print out results, etc. ...
END
```

partial list of declarations

Calculate stiffness matrix,
KEL, for a single element

Add KEL to global stiffness
matrix, KGLO

Construct FORC from
individual loads defined in
LOAD array

Must constrain problem at specified
DOF's (or no solution possible)

Compute solution for displacements



- Open source general-purpose language.
- Object Oriented, Modular
- Easy to interface with C++/C/Java/Fortran
- Interactive environment
- Interpreted and therefore slower than compiled languages

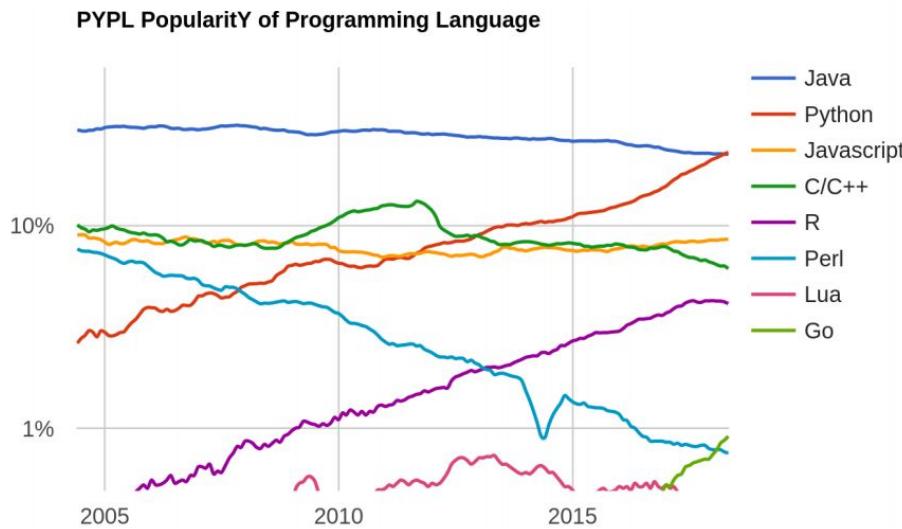
斐波那契数列

```
def fib(n):  
    a, b = 0, 1  
    while a < n:  
        print(a, end=' ')  
        a, b = b, a+b  
    print()  
fib(1000)
```



```
$ python3.4 1.py  
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610  
0.87' 不换行是python3版本的用法, python2版本无法编译
```

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales



<http://pypl.github.io/PYPL.html>

```
>>> import antigravity  
>>> import this
```

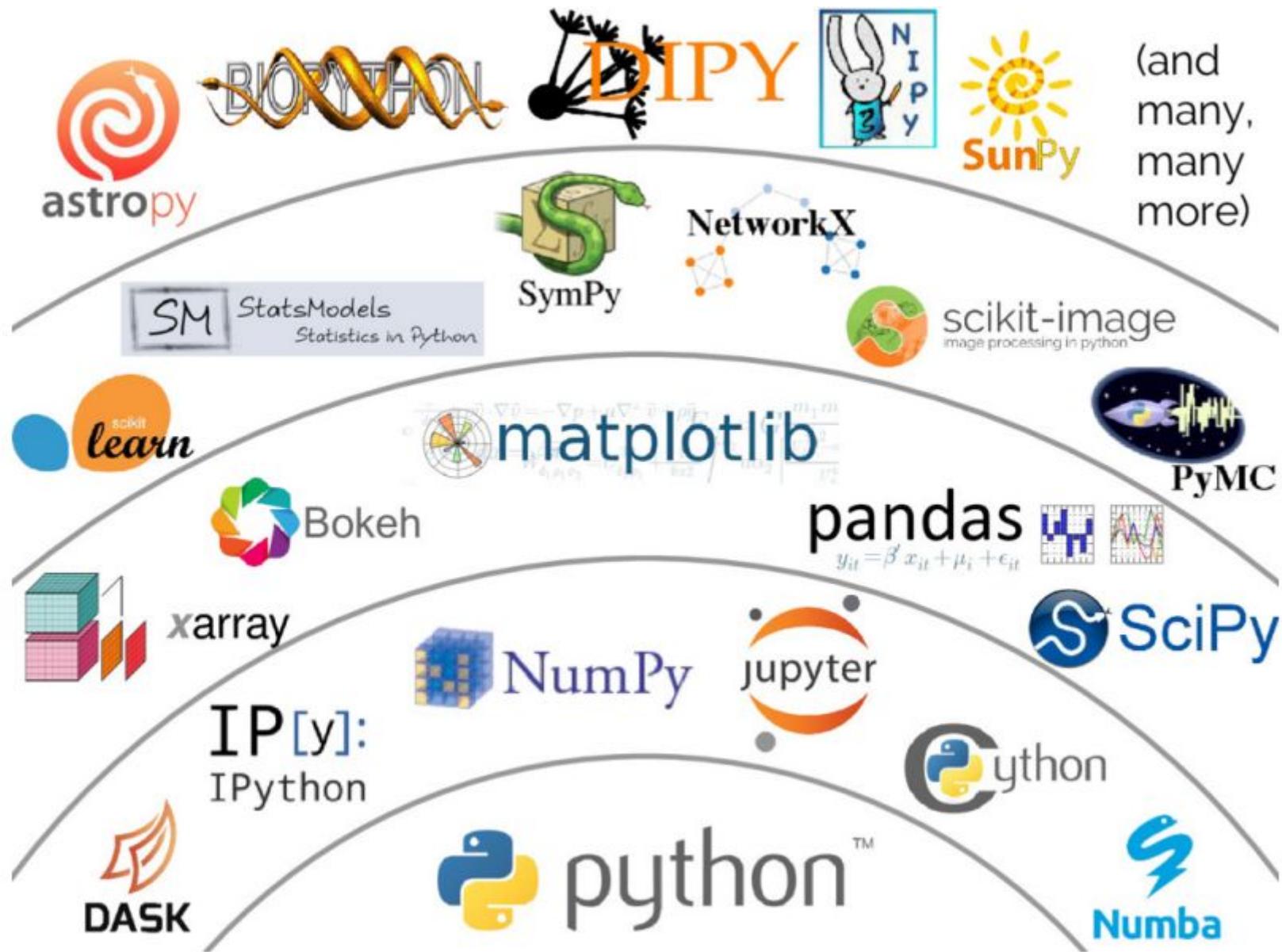
```
sudo apt install python3-pip  
(python2: sudo apt install  
python-pip)
```

- C++:
 - Fast
 - Compiled
 - Statically typed
 - int i = 0;
 - Access to pointers
 - Whitespace irrelevant
- python:
 - Slower
 - Interpreted
 - Dynamically typed:
 - i = 0
 - No pointers
 - Whitespace matters

```
pip3 install matplotlib  
pip3 install numpy  
pip3 install pandas  
pip3 install scipy  
pip3 install vpython  
  
pip3 install --upgrade pip  
pip3 install jupyter  
jupyter notebook
```

Best case is to have your “human” handling with python and your hardcore computer code in C++ .
Then call the C++ code from python.
This is what scipy, numpy do, etc.

Python-ecosystem



nature > review articles > article

Review Article | Open Access | Published: 16 September 2020

Array programming with NumPy

Charles R. Harris, K. Jarrod Millman  [...] Travis E. Oliphant

Nature 585, 357–362(2020) | Cite this article

192k Accesses | 3 Citations | 2075 Altmetric | Metrics

NumPy

in 2005 NumPy emerged as a ‘best of both worlds’ unification⁷—combining the features of Numarray with the small-array performance of Numeric and its rich C API.

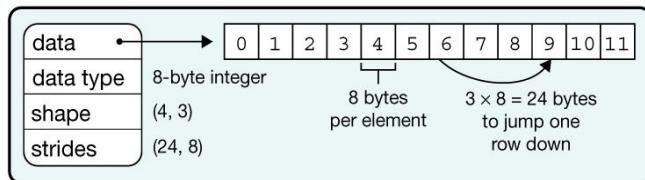
Array programming provides a powerful, compact and expressive syntax for accessing, manipulating and operating on data in vectors, matrices and higher-dimensional arrays.

NumPy is the primary array programming library for the Python language. It has an essential role in research analysis pipelines in fields as diverse as physics, chemistry, astronomy, geoscience, biology, psychology, materials science, engineering, finance and economics. For example, in astronomy, NumPy was an important part of the software stack used in the discovery of gravitational waves¹ and in the first imaging of a black hole². Here we review how a few fundamental array concepts lead to a simple and powerful programming paradigm for organizing, exploring and analysing scientific data. NumPy is the foundation upon which the scientific Python ecosystem is constructed. It is so pervasive that several projects, targeting audiences with specialized needs, have developed their own NumPy-like interfaces and array objects. Owing to its central position in the ecosystem, NumPy increasingly acts as an interoperability layer between such array computation libraries and, together with its application programming interface (API), provides a flexible framework to support the analysis of scientific and industrial applications.

Numpy

a Data structure

$x = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$



b Indexing (view)

$x[:, 1:] \rightarrow \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$ with slices $x[:, :, ::2] \rightarrow \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$ with slices with steps

Slices are **start:end:step**, any of which can be left blank

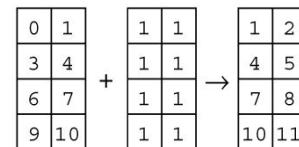
c Indexing (copy)

$x[1, 2] \rightarrow 5$ with scalars $x[x > 9] \rightarrow \boxed{10 \ 11}$ with masks

$x[\boxed{0 \ 1}, \boxed{1 \ 2}] \rightarrow [\boxed{x[0, 1]}, \boxed{x[1, 2]}] \rightarrow \boxed{1 \ 5}$ with arrays

$x[\boxed{1 \ 2}, \boxed{1 \ 0}] \rightarrow x[\boxed{\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}}, \boxed{\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}}] \rightarrow \boxed{4 \ 3}$ with arrays with broadcasting

d Vectorization



g Example

```
In [1]: import numpy as np
```

```
In [2]: x = np.arange(12)
```

```
In [3]: x = x.reshape(4, 3)
```

```
In [4]: x
```

```
Out[4]:
```

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

```
In [5]: np.mean(x, axis=0)
```

```
Out[5]: array([4.5, 5.5, 6.5])
```

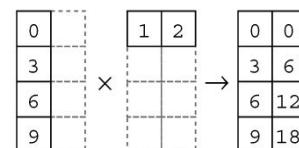
```
In [6]: x = x - np.mean(x, axis=0)
```

```
In [7]: x
```

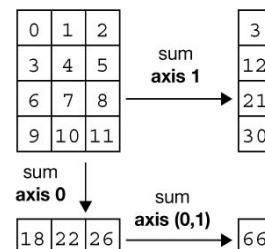
```
Out[7]:
```

```
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

e Broadcasting



f Reduction



Numpy

```
import time
import numpy as np

size_of_vec = 1000
def pure_python_version():
    t1 = time.time()
    X = range(size_of_vec)
    Y = range(size_of_vec)
    Z = [X[i] + Y[i] for i in range(len(X)) ]
    return time.time() - t1
def numpy_version():
    t1 = time.time()
    X = np.arange(size_of_vec)
    Y = np.arange(size_of_vec)
    Z = X + Y
    return time.time() - t1
t1 = pure_python_version()
t2 = numpy_version()
print(t1, t2)
print("Numpy is in this example " + str(t1/t2) + " faster!")
```

0.0004336833953857422 2.2649765014648438e-05

Numpy is in this example 19.147368421052633 faster!

基于Cint(C/C++ interpreter, C-int)是一个
C++解释器, 和GCC、VC等编译器不同,它是
解释执行C++代码的

<https://root.cern.ch/>

The screenshot shows the official ROOT website at <https://root.cern.ch/>. The page features a blue header with the ROOT logo and "Data Analysis Framework". Below the header is a navigation bar with links: Download, Documentation, News, Support, About, Development, and Contribute. The main content area includes sections for "Getting Started", "Reference Guide", "Forum", and "Gallery". A large red arrow points from the bottom left towards the "Getting Started" button. To the right, there is a code editor window displaying C++ code related to the ROOT interpreter.

```
import ROOT
cppFunctionCode = """
void f() {
    std::cout << "Hi jitted C++ world!" << std::endl
}
"""
ROOT.gInterpreter.Declare(cppFunctionCode)
ROOT.f() # Hello!
```

Under the Spotlight

16-12-2015 [Try the new ROOTbooks on Binder \(beta\)](#)

Try the new [ROOTbooks on Binder \(Beta\)](#)! Use ROOT Interactively in notebooks and explore the examples.

Other News

16-04-2016 [The status of reflection in C++](#)

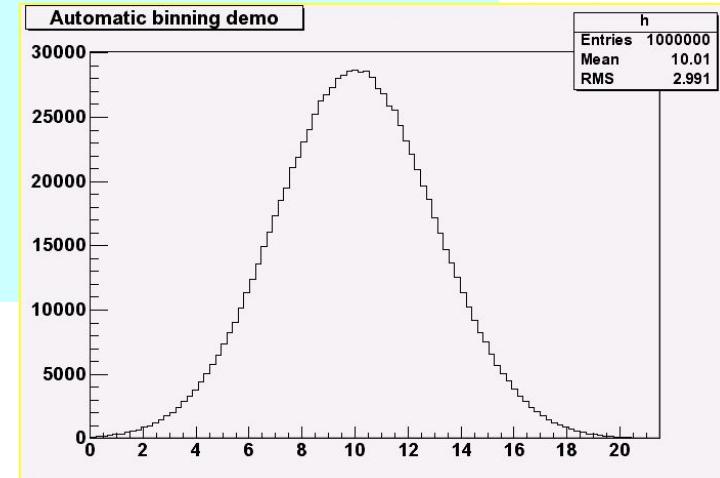
05-01-2016 [Wanted: A tool to 'warn' user of inefficient \(construct in data model](#)

CERN Root: Histogram

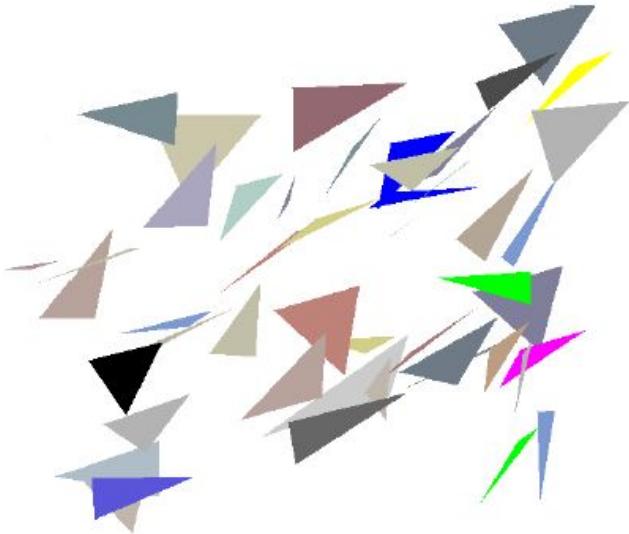
Bin, Events, Mean, RMS

```
#include "TH1.h"
#include "TF1.h"

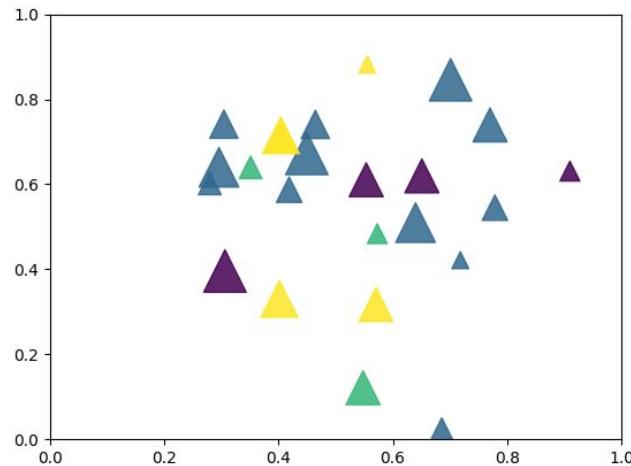
void demoauto() {
    TF1 *f1 = new TF1("f1","gaus",0,30);
    f1->SetParameters(1,10,3);
    TH1F *h = new TH1F("h","Automatic binning
demo",100,0,20);
    for (Int_t i=0;i<1000000;i++) {
        h->Fill(f1->GetRandom());
    }
    h->Draw();
}
```



CERN Root 例1：三角图形



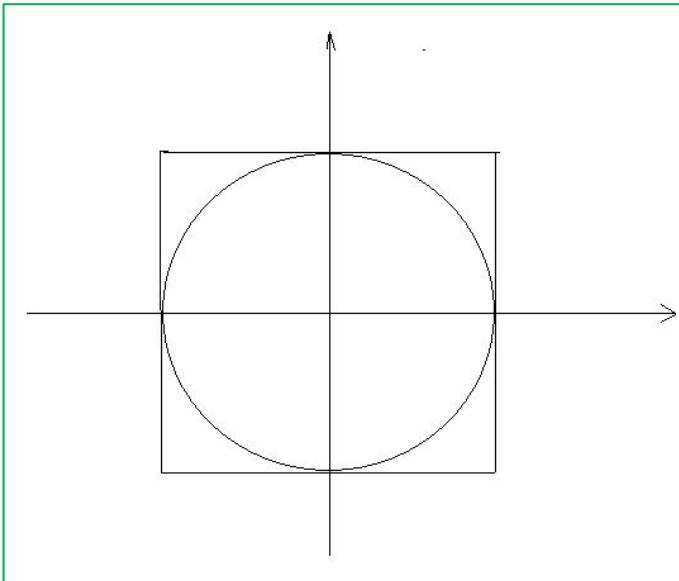
```
| TCanvas *c1 = new TCanvas("c1","triangles",10,10,700,700);  
TRandom r;  
Double_t dx = 0.2; Double_t dy = 0.2;  
Int_t ncolors = gStyle->GetNumberOfColors();  
Double_t x[4],y[4];  
TColor *c;  
Int_t ci;  
for (Int_t i=0;i<ntriangles;i++) {  
    x[0] = r.Uniform(.05,.95); y[0] = r.Uniform(.05,.95);  
    x[1] = x[0] + dx*r.Rndm(); y[1] = y[0] + dy*r.Rndm();  
    x[2] = x[1] - dx*r.Rndm(); y[2] = y[1] - dy*r.Rndm();  
    x[3] = x[0]; y[3] = y[0];  
    TPolyLine *pl = new TPolyLine(4,x,y);  
    pl->SetUniqueID(i);  
    ci = ncolors*r.Rndm();  
    c = gROOT->GetColor(ci);  
    c->SetAlpha(r.Rndm());  
    pl->SetFillColor(ci);  
    pl->Draw("f");  
}  
c1->AddExec("ex","TriangleClicked()");
```



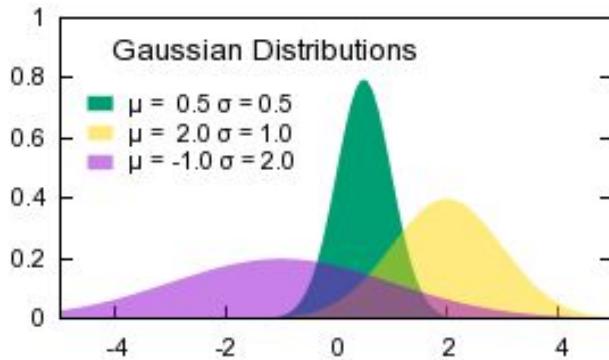
```
import math
import random
import matplotlib.pyplot as plt
# create random data
no_of_balls = 25
x = [random.triangular() for i in
range(no_of_balls)]
y = [random.gauss(0.5, 0.25) for i in
range(no_of_balls)]
colors = [random.randint(1, 4) for i in
range(no_of_balls)]
areas = [math.pi * random.randint(5, 15)**2
for i in range(no_of_balls)]
# draw the plot
plt.figure()
plt.scatter(x, y, s=areas, marker='^',
c=colors, alpha=0.85)
#plt.scatter(x, y, s=areas, marker='o',
c=colors, alpha=0.85)
plt.axis([0.0, 1.0, 0.0, 1.0])
plt.savefig('3.png')
plt.show()
```

CERN Root例2：随机数计算Pi

```
void random()
{
//TRandom3, is based on the "Mersenne
Twister generator", and is the
recommended one, since it has good
random proprieties (period of about
10**6000 ) and it is fast
    // create random number generator
    gRandom = new TRandom3(0);
    gRandom2 = new TRandom3(1);
    int n=20000;
    int ftot=0;
    for (int i = 0; i < n; ++i) {
        double x=gRandom->Uniform(-1,1);
        double y=gRandom2->Uniform(-1,1);
        double r=sqrt(x*x+y*y);
        if(r<1.0) {ftot++;}
    }
    cout<<4*float(ftot)/float(n)<<endl;
}
```



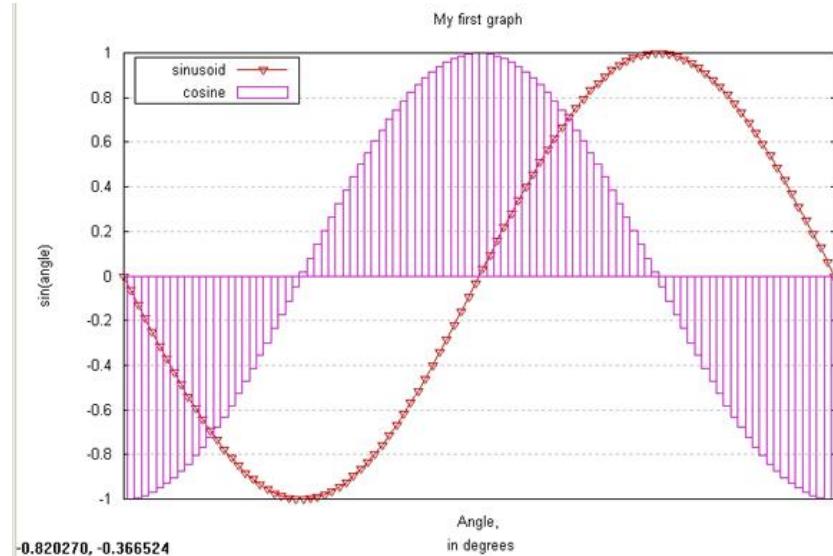
gnuplot homepage



[FAQ](#)
[Documentation](#)
[Demos](#)
[Download](#)

[Contributed scripts](#)
[External Links](#)
[Tutorials, learning, and help](#)
[Books](#)

Gnuplot is a portable **command-line driven graphing utility** for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms. The source code is copyrighted but freely distributed (i.e., you don't have to pay for it). It was **originally created to allow scientists and students to visualize** mathematical functions and data **interactively**, but has grown to support many non-interactive uses such as web scripting. It is also used as a plotting engine by third-party applications like Octave. Gnuplot has been supported and under active development since 1986.



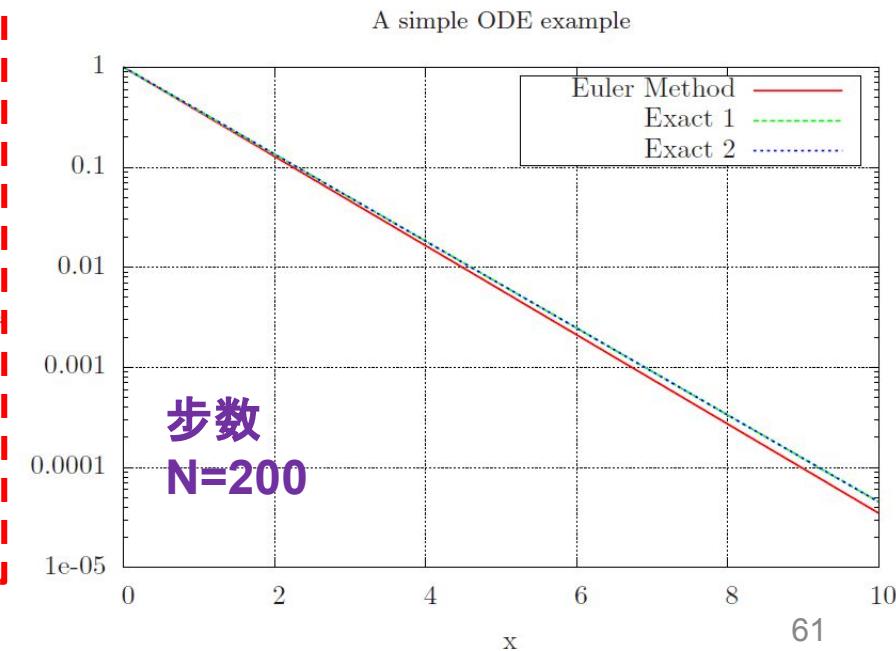
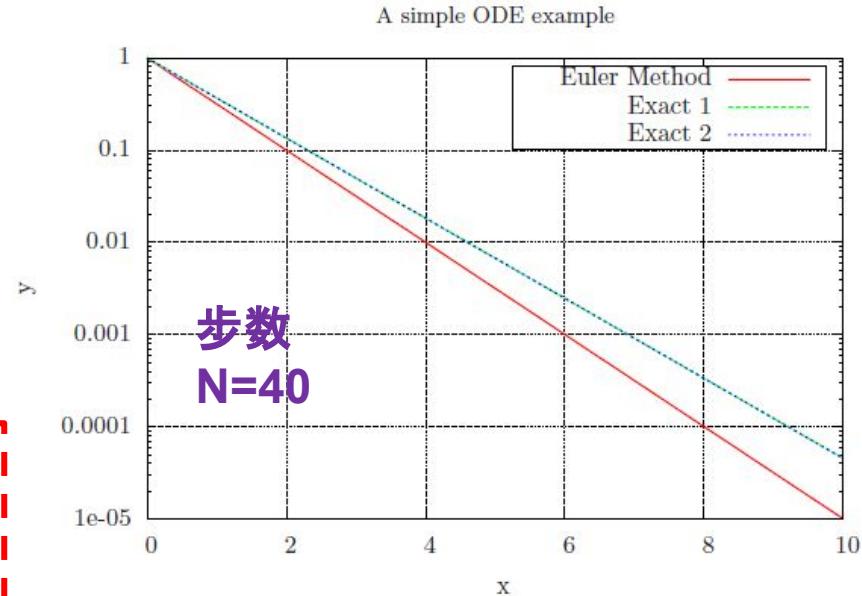
例：常微分方程求解

An ODE Example: $y' = -y$,

with $y(0)=1$. $x:[0-10]$

Explicit Method

t	output	exact
0.000000E+00	0.1000000E+01	0.1000000E+01
0.200000E+00	0.8000000E+00	0.8187308E+00
0.400000E+00	0.6400000E+00	0.6703200E+00
0.600000E+00	0.5120000E+00	0.5488116E+00
0.800000E+00	0.4096000E+00	0.4493290E+00
0.100000E+01	0.3276800E+00	0.3678794E+00
.....		
0.200000E+01	0.1073742E+00	0.1353353E+00
.....		
0.500000E+01	0.3777893E-02	0.6737947E-02
0.800000E+01	0.1329228E-03	0.3354626E-03
.....		
0.940000E+01	0.2787593E-04	0.8272407E-04
0.960000E+01	0.2230075E-04	0.6772874E-04
0.980000E+01	0.1784060E-04	0.5545160E-04
0.100000E+02	0.1427248E-04	0.4539993E-04

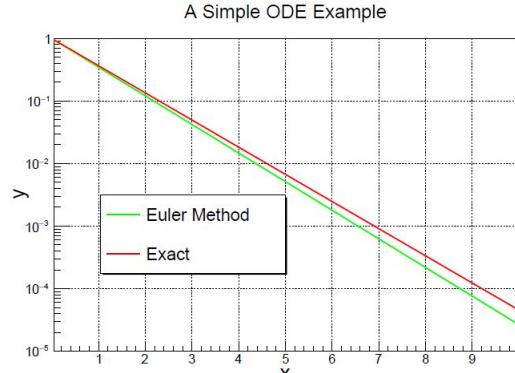


例2：CERN ROOT作图

```
void rootplot()
{
    TCanvas *c1 = new
    TCanvas("c1","",200,10,700,500);
    c1->SetGrid();

    Int_t n = 101;
    float Rbin[101],R2U[101],R2C[101];

    float xbin,xp,yp;
    int flag=0;
    int k0=0;
    int ka=0;
    int kb=0;
    FILE *fp = fopen("plot2.gnu","r");
    while (1) {
        flag=fscanf(fp,"%f %f %f",&xbin,&xp,&yp);
        if(flag!=3)break;
        Rbin[k0]=xbin;
        R2U[k0]=xp;
        R2C[k0]=yp;
        k0++;
    }
}
```



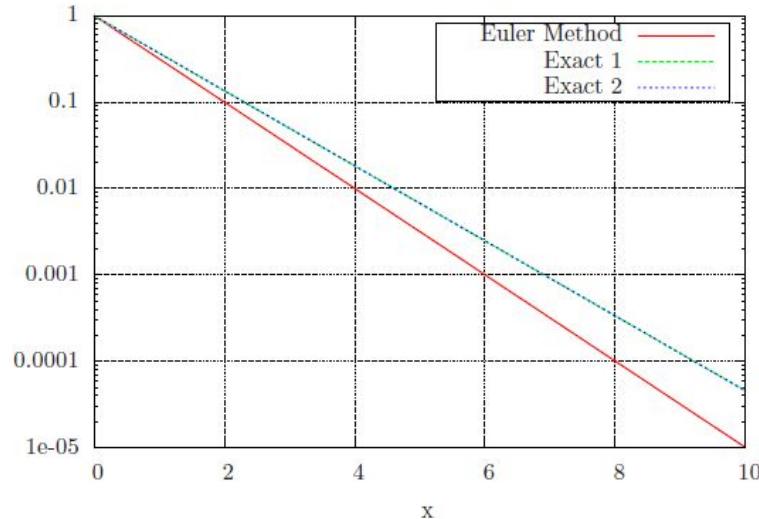
```
TMultiGraph *mg = new TMultiGraph();
mg->SetTitle("A Simple ODE Example; x; y");
gr = new TGraph(n,Rbin,R2U);
gr->SetLineWidth(2);
gr->SetLineColor(3);
gr2 = new TGraph(n,Rbin,R2C);
gr2->SetLineWidth(2);
gr2->SetLineColor(2);
mg->Add(gr);
mg->Add(gr2);
mg->Draw("AC");
mg->GetXaxis()->SetRangeUser(0.1,9.9);
mg->GetYaxis()->SetRangeUser(0.00001,1.);
mg->GetXaxis()->SetTitle("x");
mg->GetXaxis()->SetTitleSize(.06);
mg->GetYaxis()->SetTitle("y");
mg->GetYaxis()->SetTitleSize(.06);
mg->GetXaxis()->CenterTitle();
mg->GetYaxis()->CenterTitle();
mg->GetXaxis()->SetTitleOffset(0.6);
mg->GetYaxis()->SetTitleOffset(0.6);
c1->SetLogy();
gPad->Modified();
TLegend *l1 = new TLegend(0.18,0.3,0.5,0.5);
l1->SetBorderSize(2);
l1->SetFillColor(0);
l1->AddEntry(gr,"Euler Method","");
l1->AddEntry(gr2,"Exact","");
l1->Draw();
c1->SaveAs("example-N100-2.pdf");
}
```

例3：GNUPLOT作图

tot.tex

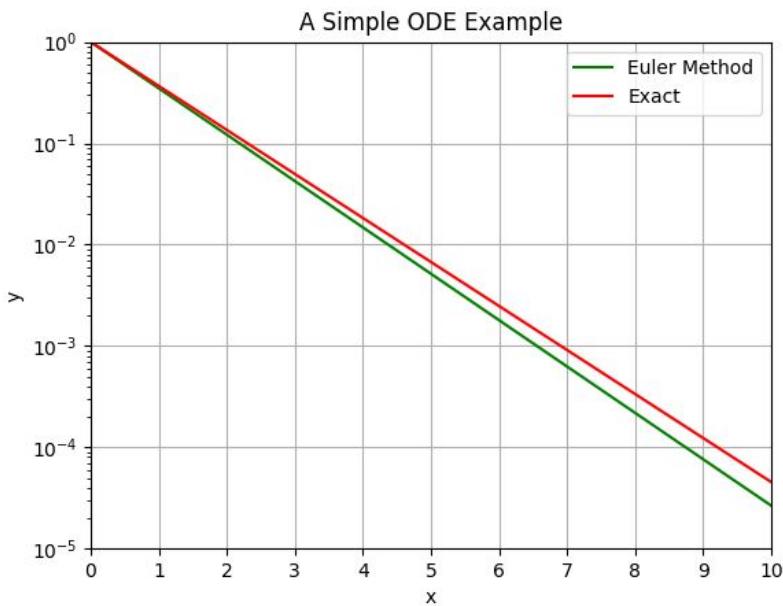
```
\documentclass[17pt]{article}
\usepackage{graphicx,color}
\begin{document}
\thispagestyle{empty}
\input{data.tex}
\end{document}
```

A simple ODE example



```
set pointsize 2.0
set tics in
set log y
set mytics 10
set grid
set xrange[0:10]
set terminal epslatex color
set output "data.tex"
set title 'A simple ODE example'
set ylabel 'y'
set xlabel 'x'
set xlabel offset 0.5
set ylabel offset 2.5
set key box
plot "plot.gnu" index 0 u ($1):($2) title
'Euler Method' w l lt 1 lw 3 ,
"plot.gnu" index 0 u ($1):($3) title
'Exact 1' w l lt 2 lw 3 ,
exp(-x) title'Exact 2' w l lt 3 lw 3
set output
!l $\atop{a}tex tot.tex
!dvips -E tot.dvi -o example1.eps
!epstopdf example1.eps$ 
```

例3：Python作图



```
import matplotlib.pyplot as plt
import csv
import numpy as np

x = []
y = []
z = []

with open('plot.gnu','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=' ',
skipinitialspace=True)
    for row in plots:
        print(row)
        x.append(float(row[0]))
        y.append(float(row[1]))
        z.append(float(row[2]))

plt.semilogy()
plt.xticks(np.arange(0, 11, 1))
plt.xlim(0, 10)
plt.ylim(0.00001, 1)
plt.plot(x,y, color='green', label='Euler Method')
plt.plot(x,z, color='red', label='Exact')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.title('A Simple ODE Example')
plt.legend()
plt.savefig('4.png')
plt.show()
```

Modern parallel programming: **open-MP** and **MPI**.

–Lots of processors connected and coordinating to solve large problems

M P I = Message Passing Interface.

<https://www mpi-forum.org/docs/>

A standardized collection of routines (functions) which is implemented for each programming language (Fortran, C, C++, Python).

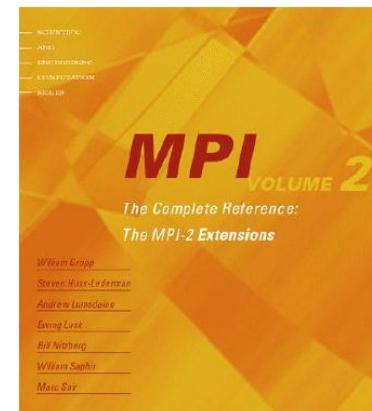
First standardized in 1994 (MPI-1.0) and second in 1997 (MPI-2.0) and (MPI-3.0) after 2012. Currently **MPI-2.0** is most widely used.

The two most widely used implementations are

- **MPICH** <http://www.mpich.org>
- **open-MPI** <http://www.open-mpi.org>

Python implementation mpi4py

(Note that open-MPI has nothing to do with openMP)



并行计算

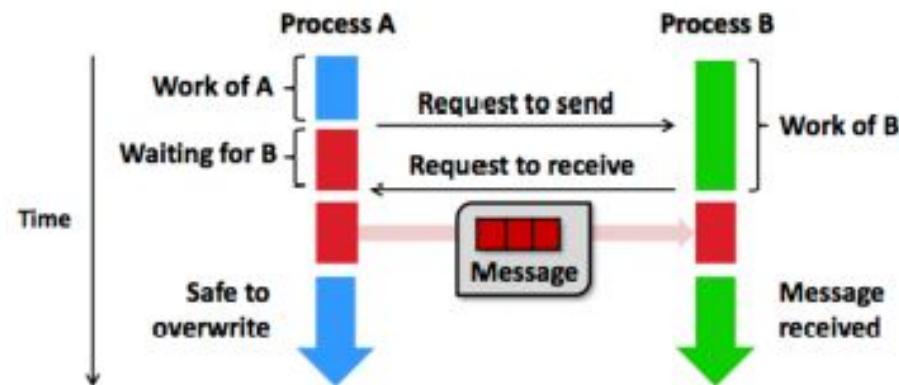
```
sudo apt-get install libcr-dev mpich2 mpich2-doc
```

```
mpic++ -o example example.c  
mpirun -n 2 ./example
```

MPI include file

Initialize MPI environment

Do work and make message passing calls



Terminate MPI Environment

并行计算

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char ** argv)
{
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("I am %d of %d\n", rank, size);

    MPI_Finalize();
    return 0;
}
```

Basic requirements for an MPI program

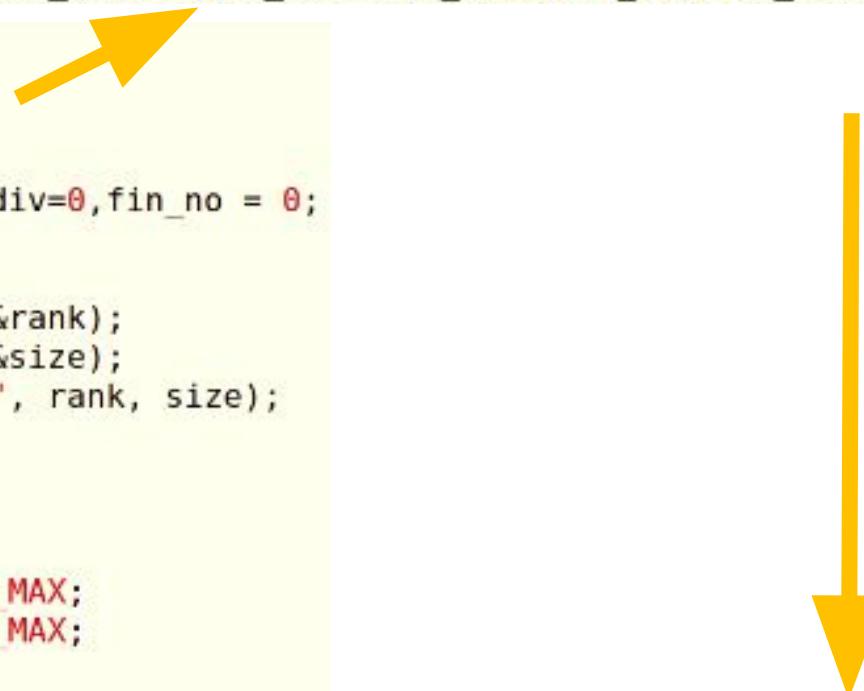
Data communication in MPI is like email exchange
–One process sends a copy of the data to another process (or a group of processes), and the other process receives it

并行计算

```
printf("For rank %d count = %d itrr = %d\n",rank,count,no_div);
MPI_Reduce(&count,&total_count,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
MPI_Reduce(&no_div,&fin_no,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);

double t1, t2;
int rank,size,i;
double x=0,y=0,pi,z;
int no = atoi(argv[1]);
int count=0,total_count=0,no_div=0,fin_no = 0;
MPI_Init(&argc,&argv);
t1 = MPI_Wtime();
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);
printf("rank= %i ; size= %i\n", rank, size);
no_div = no/size;
srand ( time(NULL) );
for(i=0;i<no_div;i++)
{
    x=rand()/(double)RAND_MAX;
    y=rand()/(double)RAND_MAX;
    z=x*x+y*y;
    if(z<=1)
        count++;
}
t2 = MPI_Wtime();
printf("total time for rank %i: = %lf\n", rank, t2-t1);

if(rank ==0){
    printf("\n");
    printf("*** Average over all processors ***\n");
    printf("Total count = %d, total itrr = %d\n",total_count,fin_no);
    pi = ((double)total_count)/fin_no*4.0000;
    printf("Pi value = %lf\n",pi);
}
MPI_Finalize();
```



并行计算

```
qliphy@qliphy-IdeaCentre-B520:~/Desktop/CP2019-new/MPI$ mpirun -n 1 ./pi 80000000
rank= 0 ; size= 1
For rank 0 count = 62831445 itrr = 80000000
total time for rank 0: = 1.796201

*** Average over all processors ***
Total count = 62831445, total itrr = 80000000
Pi value = 3.141572
```

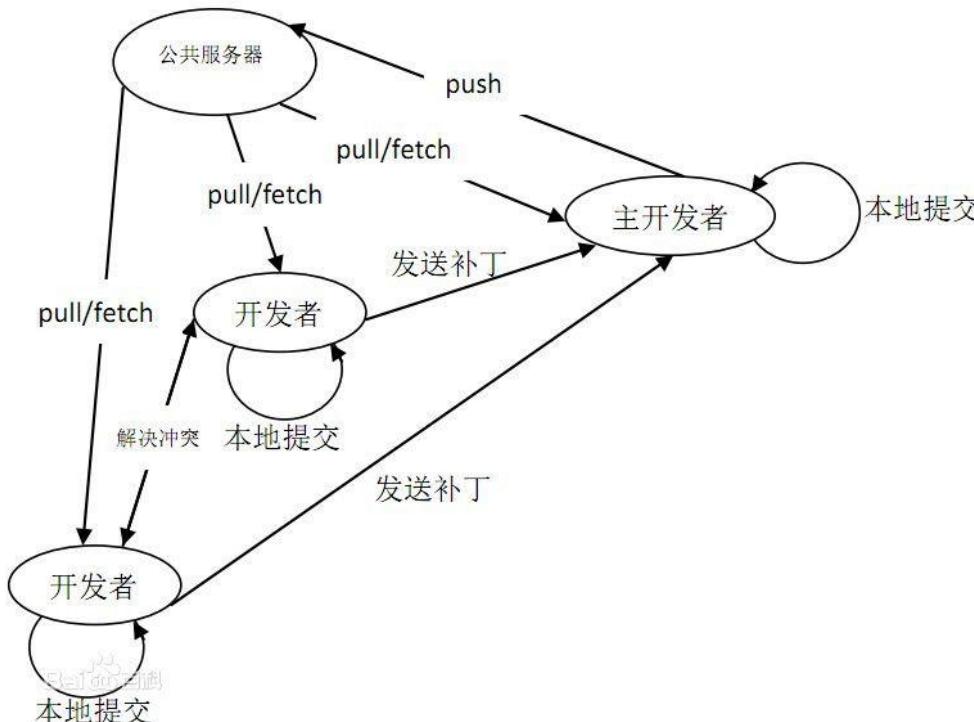
```
qliphy@qliphy-IdeaCentre-B520:~/Desktop/CP2019-new/MPI$ mpirun -n 4 ./pi 80000000
rank= 0 ; size= 4
rank= 1 ; size= 4
rank= 2 ; size= 4
rank= 3 ; size= 4
For rank 2 count = 15710006 itrr = 20000000
For rank 3 count = 15710006 itrr = 20000000
total time for rank 3: = 0.739523
total time for rank 2: = 0.739545
For rank 1 count = 15710006 itrr = 20000000
total time for rank 1: = 0.792446
For rank 0 count = 15710006 itrr = 20000000
total time for rank 0: = 0.816587

*** Average over all processors ***
Total count = 62840024, total itrr = 80000000
Pi value = 3.142001
```

版本控制

- CVS (concurrent versions system)
- SVN (subversion)
- git

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere



Sign Up a Github Account

The screenshot shows the GitHub sign-up page. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for Explore, Features, Enterprise, Pricing, Sign up (in a green button), and Sign in. The main heading "Where software is built" is displayed prominently. Below it, a sub-headline states "Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free." A note indicates "Private plans start at \$7/mo." To the right, there is a form for entering account information: "Pick a username", "Your email", and "Create a password". A note below the password field says "Use at least one lowercase letter, one numeral, and seven characters." A large green "Sign up for GitHub" button is located at the bottom right of the form area. A small disclaimer at the bottom left of the form area states: "By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally."

You will receive an activation email to confirm your email address

Set Up Git

<https://help.github.com/articles/set-up-git/>

1. Download and install the latest version of Git.

In terminal:

1. `$ git config --global user.name "YOUR NAME"`
2. `$ git config --global user.email "YOUR EMAIL ADDRESS"`

Authenticating with GitHub from Git

- Connecting over HTTPS (recommended)

If you [clone with HTTPS](#), you can

[cache your GitHub password in Git using a credential helper](#).

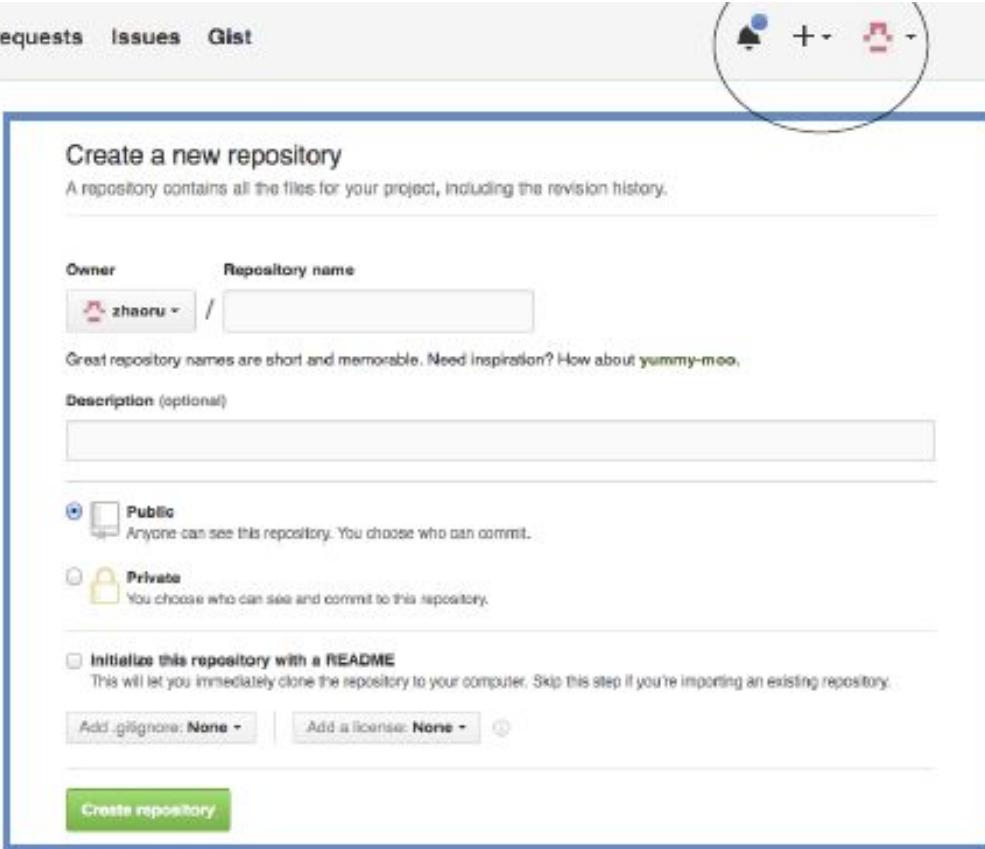
<https://help.github.com/articles/caching-your-github-password-in-git/>

- Connecting over SSH

If you [clone with SSH](#), you must [generate SSH keys](#) on each computer you use to push or pull from GitHub.

Create a Repo

<https://help.github.com/articles/create-a-repo/>



The screenshot shows the GitHub interface for creating a new repository. At the top, there's a navigation bar with a search bar, pull requests, issues, and a gist button. On the right, there's a user profile icon and a plus sign for creating new repositories. The main area is titled "Create a new repository" with a sub-instruction: "A repository contains all the files for your project, including the revision history." It asks for the "Owner" (set to "zhaoru") and "Repository name". Below that, it says "Great repository names are short and memorable. Need inspiration? How about yummy-moo." There's a "Description (optional)" field with a placeholder "yummy-moo". Under "Visibility", the "Public" option is selected, with a note: "Anyone can see this repository. You choose who can commit." The "Private" option is also available. A checkbox for "Initialize this repository with a README" is checked, with a note: "This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository." At the bottom, there are buttons for "Add .gitignore: None" and "Add a license: None", followed by a large green "Create repository" button.

- Set up “Repo Name”
- Public
- **Don’t** select “Initialize this repository with a README”
- Create Repository

Upload your file to the Repo

In terminal:

```
$ cd "target folder"  
$ echo "# TEST4" >> README.md  
$ git init  
$ git add README.md  
$ git add *  
$ git commit -m "first commit"  
$ git remote add origin  
git@github.com:XXX/TEST4.git (or https://)  
$ git push -u origin master
```

Then refresh your website. Upload successfully!

版本控制

The screenshot shows a GitHub repository page for 'qliphy / ComputationalPhysics2019'. The top navigation bar includes 'Watch 0', 'Star 0', 'Fork 0', and tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area has three sections: 'Quick setup — if you've done this kind of thing before', '...or create a new repository on the command line', and '...or push an existing repository from the command line'. Each section contains a code snippet and a copy icon.

Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** git@github.com:qliphy/ComputationalPhysics2019.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# ComputationalPhysics2019" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin git@github.com:qliphy/ComputationalPhysics2019.git  
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:qliphy/ComputationalPhysics2019.git  
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

💡 **ProTip!** Use the URL for this page when adding GitHub as a remote.

版本控制

qliphy / ComputationalPhysics2021

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

qliphy Create README.md a2d7603 now 1 commit

README.md Create README.md now

README.md

ComputationalPhysics2021

About No description, website, or topics provided.

Readme

Releases No releases published Create a new release

Packages No packages published Publish your first package

2021年的小作业：每位同学push一个以学号命名的文件夹到 XXX/ComputationalPhysics2021 @github