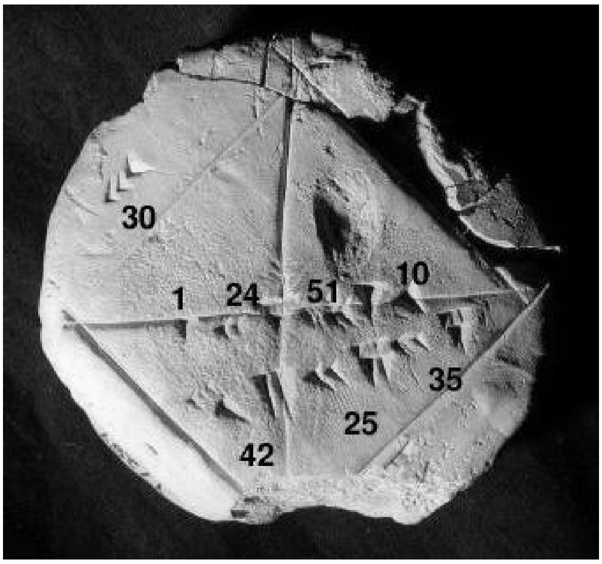
Numerical analysis

Numerical analysis is the study of algorithms that use numerical approximation (as opposed to general symbolic manipulations) for the problems of continuous mathematics (as distinguished from discrete mathematics).



Babylonian clay tablet YBC 7289 (c. 1800–1600 BCE) with annotations. The approximation of the square root of 2 is four sexagesimal figures, which is about six decimal figures. $1 + 24/60 + 51/60^2 + 10/60^3 = 1.41421296...$[1] Image by Bill Casselman.[2]

One of the earliest mathematical writings is the Babylonian tablet YBC 7289, which gives a sexagesimal numerical approximation of $\sqrt{2}$, the length of the diagonal in a unit square. Being able to compute the sides of a triangle (and hence, being able to compute square roots) is extremely important, for instance, in carpentry and construction.[3]

Numerical analysis continues this long tradition of practical mathematical calculations. Much like the Babylonian approximation of $\sqrt{2}$, modern numerical analysis does not seek exact answers, because exact answers are often impossible to obtain in practice. Instead, much of numerical analysis is concerned with obtaining approximate solutions while maintaining reasonable bounds on errors.

Numerical analysis naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century, the life sciences and even the arts have adopted elements of scientific computations. Ordinary differential equations appear in the movement of heavenly bodies (planets, stars and galaxies); optimization occurs in portfolio management; numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology.

Before the advent of modern computers numerical methods often depended on hand interpolation in large printed tables. Since the mid 20th century, computers calculate the required functions instead. The interpolation algorithms nevertheless may be used as part of the software for solving differential equations.

General introduction

The overall goal of the field of numerical analysis is the design and analysis of techniques to give approximate but accurate solutions to hard problems, the variety of which is suggested by the following.

- Advanced numerical methods are essential in making numerical weather prediction feasible.
- Computing the trajectory of a spacecraft requires the accurate numerical solution of a system of ordinary differential equations.
- Car companies can improve the crash safety of their vehicles by using computer simulations of car crashes. Such simulations essentially consist of solving partial differential equations numerically.
- Hedge funds (private investment funds) use tools from all fields of numerical analysis to calculate the value of stocks and derivatives more precisely than other market participants.
- Airlines use sophisticated optimization algorithms to decide ticket prices, airplane and crew assignments and fuel needs. This field is also called operations research.
- Insurance companies use numerical programs for actuarial analysis.

The rest of this section outlines several important themes of numerical analysis.

History

The field of numerical analysis predates the invention of modern computers by many centuries. Linear interpolation was already in use more than 2000 years ago. Many great mathematicians of the past were preoccupied by numerical analysis, as is obvious from the names of important algorithms like Newton's method, Lagrange interpolation polynomial, Gaussian elimination, or Euler's method.

To facilitate computations by hand, large books were produced with formulas and tables of data such as interpolation points and function coefficients. Using these tables, often calculated out to 16 decimal places or more for some functions, one could look up values to plug into the formulas given and achieve very good numerical estimates of some functions. The canonical work in the field is the NIST publication edited by Abramowitz and Stegun, a 1000-plus page book of a very large number of commonly used formulas and functions and their values at many points. The function values are no longer very useful when a computer is available, but the large listing of formulas can still be very handy.

The mechanical calculator was also developed as a tool for hand computation. These calculators evolved into electronic computers in the 1940s, and it was then found that these computers were also useful for administrative purposes. But the invention of the computer also influenced the field of numerical analysis, since now longer and more complicated calculations could be done.

Direct and iterative methods

Direct vs iterative methods

Consider the problem of solving

$$3x^3+4=28$$

for the unknown quantity x.

	$3x^3 + 4 = 28.$
Subtract 4	$3x^3 = 24.$
Divide by 3	$x^3 = 8.$
Take cube roots	$x = 2.$

For the iterative method, apply the bisection method to $f(x) = 3x^3 - 24$. The initial values are $a = 0$, $b = 3$, $f(a) = -24$, $f(b) = 57$.

a	*b*	mid	*f*(mid)
0	3	1.5	-13.875
1.5	3	2.25	10.17...
1.5	2.25	1.875	-4.22...
1.875	2.25	2.0625	2.32...

We conclude from this table that the solution is between 1.875 and 2.0625. The algorithm might return any number in that range with an error less than 0.2.

Discretization and numerical integration

In a two hour race, we have measured the speed of the car at three instants and recorded them in the following table.

Time	0:20	1:00	1:40
km/h	140	150	180

A **discretization** would be to say that the speed of the car was constant from 0:00 to 0:40, then from 0:40 to 1:20 and finally from 1:20 to 2:00. For instance, the total distance traveled in the first 40 minutes is approximately (2/3h x 140 km/h)=93.3 km. This would allow us to estimate the total distance traveled as 93.3 km + 100 km + 120 km = 313.3 km, which is an example of **numerical integration** (see below) using a Riemann sum, because displacement is the integral of velocity.

Ill posed problem: Take the function $f(x) = 1/(x - 1)$. Note that $f(1.1) = 10$ and $f(1.001) = 1000$: a change in x of less than 0.1 turns into a change in $f(x)$ of nearly 1000. Evaluating $f(x)$ near $x = 1$ is an ill-conditioned problem.

Well-posed problem: By contrast, the function $f(x)=\sqrt{x}$ is continuous and so evaluating it is well-posed, at least for x being not close to zero.

Direct methods compute the solution to a problem in a finite number of steps. These methods would give the precise answer if they were performed in infinite precision arithmetic. Examples include Gaussian elimination, the QR factorization method for solving systems of linear equations, and the simplex method of linear programming. In practice, finite precision is used and the result is an approximation of the true solution (assuming stability).

In contrast to direct methods, iterative methods are not expected to terminate in a number of steps. Starting from an initial guess, iterative methods form successive approximations that converge to the exact solution only in the limit. A convergence test is specified in order to decide when a sufficiently accurate solution has (hopefully) been found.

Even using infinite precision arithmetic these methods would not reach the solution within a finite number of steps (in general). Examples include Newton's method, the bisection method, and Jacobi iteration. In computational matrix algebra, iterative methods are generally needed for large problems.

Iterative methods are more common than direct methods in numerical analysis. Some methods are direct in principle but are usually used as though they were not, e.g. GMRES and the conjugate gradient method. For these methods the number of steps needed to obtain the exact solution is so large that an approximation is accepted in the same manner as for an iterative method.

Discretization

Furthermore, continuous problems must sometimes be replaced by a discrete problem whose solution is known to approximate that of the continuous problem; this process is called *discretization*. For example, the solution of a differential equation is a function. This function must be represented by a finite amount of data, for instance by its value at a finite number of points at its domain, even though this domain is a continuum.

The generation and propagation of errors

The study of errors forms an important part of numerical analysis. There are several ways in which error can be introduced in the solution of the problem.

Round-off

Round-off errors arise because it is impossible to represent all real numbers exactly on a machine with finite memory (which is what all practical digital computers are).

Truncation and discretization error

Truncation errors are committed when an iterative method is terminated or a mathematical procedure is approximated, and the approximate solution differs from the exact solution. Similarly, discretization induces a discretization error because the solution of the discrete problem does not coincide with the solution of the continuous problem. For instance, in the iteration in the sidebar to compute the solution of $3x^3 + 4 = 28$, after 10 or so iterations, we conclude that the root is roughly 1.99 (for example). We therefore have a truncation error of 0.01.

Once an error is generated, it will generally propagate through the calculation. For instance, we have already noted that the operation + on a calculator (or a computer) is inexact. It follows that a calculation of the type a+b+c+d+e is even more inexact.

What does it mean when we say that the truncation error is created when we approximate a mathematical procedure. We know that to integrate a function exactly requires one to find the sum of infinite trapezoids. But numerically one can find the sum of only finite trapezoids, and hence the approximation of the mathematical procedure. Similarly, to differentiate a function, the differential element approaches to zero but numerically we can only choose a finite value of the differential element.

Numerical stability and well-posed problems

Numerical stability is an important notion in numerical analysis. An algorithm is called *numerically stable* if an error, whatever its cause, does not grow to be much larger during the calculation. This happens if the problem is *well-conditioned*, meaning that the solution changes by only a small amount if the problem data are changed by a small amount. To the contrary, if a problem is *ill-conditioned*, then any small error in the data will grow to be a large error.

Both the original problem and the algorithm used to solve that problem can be *well-conditioned* and/or *ill-conditioned*, and any combination is possible.

So an algorithm that solves a well-conditioned problem may be either numerically stable or numerically unstable. An art of numerical analysis is to find a stable algorithm for solving a well-posed mathematical problem. For instance, computing the square root of 2 (which is roughly 1.41421) is a well-posed problem. Many algorithms solve this problem by starting with an initial approximation x_1 to $\sqrt{2}$, for instance x_1=1.4, and then computing improved guesses x_2, x_3, etc... One such method is the famous Babylonian method, which is given by $x_{k+1} = x_k/2 + 1/x_k$. Another iteration, which we will call Method X, is given by $x_{k+1} = (x_k^2-2)^2 + x_k$.[4] We have calculated a few iterations of each scheme in table form below, with initial guesses $x_1 = 1.4$ and $x_1 = 1.42$.

Babylonian	Babylonian	Method X	Method X
$x_1 = 1.4$	$x_1 = 1.42$	$x_1 = 1.4$	$x_1 = 1.42$
$x_2 = 1.4142857...$	$x_2 = 1.41422535...$	$x_2 = 1.4016$	$x_2 = 1.42026896$
$x_3 = 1.414213564...$	$x_3 = 1.41421356242...$	$x_3 = 1.4028614...$	$x_3 = 1.42056...$
		...	...
		$x_{1000000} = 1.41421...$	$x_{28} = 7280.2284...$

Observe that the Babylonian method converges fast regardless of the initial guess, whereas Method X converges extremely slowly with initial guess 1.4 and diverges for initial guess 1.42. Hence, the Babylonian method is numerically stable, while Method X is numerically unstable.

Areas of study

The field of numerical analysis is divided into different disciplines according to the problem that is to be solved.

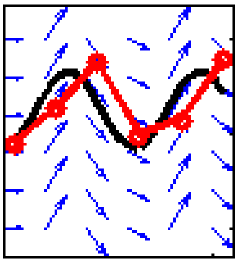
Computing values of functions

Interpolation: We have observed the temperature to vary from 20 degrees Celsius at 1:00 to 14 degrees at 3:00. A linear interpolation of this data would conclude that it was 17 degrees at 2:00 and 18.5 degrees at 1:30pm.

Extrapolation: If the gross domestic product of a country has been growing an average of 5% per year and was 100 billion dollars last year, we might extrapolate that it will be 105 billion dollars this year.

Regression: In linear regression, given *n* points, we compute a line that passes as close as possible to those *n* points.

Optimization: Say you sell lemonade at a lemonade stand, and notice that at $1, you can sell 197 glasses of lemonade per day, and that for each increase of $0.01, you will sell one less lemonade per day. If you could charge $1.485, you would maximize your profit, but due to the constraint of having to charge a whole cent amount, charging $1.49 per glass will yield the maximum income of $220.52 per day.



Differential equation: If you set up 100 fans to blow air from one end of the room to the other and then you drop a feather into the wind, what happens? The feather will follow the air currents, which may be very complex. One approximation is to measure the speed at which the air is blowing near the feather every second, and advance the simulated feather as if it were moving in a straight line at that same speed for one second, before measuring the wind speed again. This is called the Euler method for solving an ordinary differential equation.

One of the simplest problems is the evaluation of a function at a given point. The most straightforward approach, of just plugging in the number in the formula is sometimes not very efficient. For polynomials, a better approach is using the Horner scheme, since it reduces the necessary number of multiplications and additions. Generally, it is important to estimate and control round-off errors arising from the use of floating point arithmetic.

Interpolation, extrapolation, and regression

Interpolation solves the following problem: given the value of some unknown function at a number of points, what value does that function have at some other point between the given points?

Extrapolation is very similar to interpolation, except that now we want to find the value of the unknown function at a point which is outside the given points.

Regression is also similar, but it takes into account that the data is imprecise. Given some points, and a measurement of the value of some function at these points (with an error), we want to determine the unknown function. The least squares-method is one popular way to achieve this.

Solving equations and systems of equations

Another fundamental problem is computing the solution of some given equation. Two cases are commonly distinguished, depending on whether the equation is linear or not. For instance, the equation $2x+5=3$ is linear while $2x^2+5=3$ is not.

Much effort has been put in the development of methods for solving systems of linear equations. Standard direct methods, i.e., methods that use some matrix decomposition are Gaussian elimination, LU decomposition, Cholesky decomposition for symmetric (or hermitian) and positive-definite matrix, and QR decomposition for non-square matrices. Iterative methods such as the Jacobi method, Gauss–Seidel method, successive over-relaxation and conjugate gradient method are usually preferred for large systems.

Root-finding algorithms are used to solve nonlinear equations (they are so named since a root of a function is an argument for which the function yields zero). If the function is differentiable and the derivative is known, then Newton's method is a popular choice. Linearization is another technique for solving nonlinear equations.

Solving eigenvalue or singular value problems

Several important problems can be phrased in terms of eigenvalue decompositions or singular value decompositions. For instance, the spectral image compression algorithm [5] is based on the singular value decomposition. The corresponding tool in statistics is called principal component analysis.

Optimization

Optimization problems ask for the point at which a given function is maximized (or minimized). Often, the point also has to satisfy some constraints.

The field of optimization is further split in several subfields, depending on the form of the objective function and the constraint. For instance, linear programming deals with the case that both the objective function and the constraints are linear. A famous method in linear programming is the simplex method.

The method of Lagrange multipliers can be used to reduce optimization problems with constraints to unconstrained optimization problems.

Evaluating integrals

Numerical integration, in some instances also known as numerical quadrature, asks for the value of a definite integral. Popular methods use one of the Newton–Cotes formulas (like the midpoint rule or Simpson's rule) or Gaussian quadrature. These methods rely on a "divide and conquer" strategy, whereby an integral on a relatively large set is broken down into integrals on smaller sets. In higher dimensions, where these methods become prohibitively expensive in terms of computational effort, one may use Monte Carlo or quasi-Monte Carlo methods (see Monte Carlo integration), or, in modestly large dimensions, the method of sparse grids.

Differential equations

Numerical analysis is also concerned with computing (in an approximate way) the solution of differential equations, both ordinary differential equations and partial differential equations.

Partial differential equations are solved by first discretizing the equation, bringing it into a finite-dimensional subspace. This can be done by a finite element method, a finite difference method, or (particularly in engineering) a finite volume method. The theoretical justification of these methods often involves theorems from functional analysis. This reduces the problem to the solution of an algebraic equation.

Software

Since the late twentieth century, most algorithms are implemented in a variety of programming languages. The Netlib repository contains various collections of software routines for numerical problems, mostly in Fortran and C. Commercial products implementing many different numerical algorithms include the IMSL and NAG libraries; a free alternative is the GNU Scientific Library.

There are several popular numerical computing applications such as MATLAB, S-PLUS, LabVIEW, and IDL as well as free and open source alternatives such as FreeMat, Scilab, GNU Octave (similar to Matlab), IT++ (a C++ library), R (similar to S-PLUS) and certain variants of Python. Performance varies widely: while vector and matrix operations are usually fast, scalar loops may vary in speed by more than an order of magnitude.[6] [7]

Many computer algebra systems such as Mathematica also benefit from the availability of arbitrary precision arithmetic which can provide more accurate results.

Also, any spreadsheet software can be used to solve simple problems relating to numerical analysis.

See also

- Scientific computing
- List of numerical analysis software
- List of numerical analysis topics
- Gram-Schmidt process
- Numerical differentiation
- Symbolic-numeric computation
- Analysis of algorithms
- Numerical Recipes

References

- Gilat, Amos (2004). *MATLAB: An Introduction with Applications* (2nd edition ed.). John Wiley & Sons. ISBN 0-471-69420-7.
- Hildebrand, F. B. (1974). *Introduction to Numerical Analysis* (2nd edition ed.). McGraw-Hill. ISBN 0-070-28761-9.
- Leader, Jeffery J. (2004). *Numerical Analysis and Scientific Computation*. Addison Wesley. ISBN 0-201-73499-0.
- Trefethen, Lloyd N. (2006). "Numerical analysis" [8], 20 pages. In: Timothy Gowers and June Barrow-Green (editors), *Princeton Companion of Mathematics*, Princeton University Press.

External links

Journals

- Numerische Mathematik [9], volumes 1-66, Springer, 1959-1994 (searchable; pages are images). **(English) (German)**
- Numerische Mathematik at SpringerLink [10], volumes 1-112, Springer, 1959–2009
- SIAM Journal on Numerical Analysis [11], volumes 1-47, SIAM, 1964–2009

Software and Code

- Lists of free software for scientific computing and numerical analysis [12] **(English) (French)**
- Numerical methods for Fortran programmers [13]
- Java Number Cruncher [14] features free, downloadable code samples that graphically illustrate common numerical algorithms
- Excel Implementations [15]
- Several Numerical Mathematical Utilities (in Javascript) [16]

Online Texts

- *Numerical Recipes* [17], William H. Press (free, downloadable previous editions)
- *First Steps in Numerical Analysis* [18], R.J.Hosking, S.Joe, D.C.Joyce, and J.C.Turner
- *Numerical Analysis for Engineering* [19], D. W. Harder
- *CSEP* (Computational Science Education Project) [20], U.S. Department of Energy

Online Course Material

- Numerical Methods [21], Stuart Dalziel University of Cambridge
- Lectures on Numerical Analysis [22], Dennis Deturck and Herbert S. Wilf University of Pennsylvania
- Numerical methods [23], John D. Fenton University of Karlsruhe

- Numerical Methods for Science, Technology, Engineering and Mathematics [24], Autar Kaw University of South Florida
- Numerical Analysis Project [25], John H. Mathews California State University, Fullerton
- Numerical Methods - Online Course [26], Aaron Naiman Jerusalem College of Technology
- Numerical Methods for Physicists [27], Anthony O'Hare Oxford University
- Lectures in Numerical Analysis [18], R. Radok Mahidol University
- Introduction to Numerical Analysis for Engineering [28], Henrik Schmidt Massachusetts Institute of Technology

References

[1] Photograph, illustration, and description of the *root(2)* tablet from the Yale Babylonian Collection (http://it.stlawu.edu/~dmelvill/mesomath/tablets/YBC7289.html)
[2] YBC 7289, Bill Casselman (http://www.math.ubc.ca/~cass/Euclid/ybc/ybc.html)
[3] The New Zealand Qualification authority specifically mentions this skill in document 13004 version 2, dated 17 October 2003 titled CARPENTRY THEORY: Demonstrate knowledge of setting out a building (http://www.nzqa.govt.nz/nqfdocs/units/pdf/13004.pdf)
[4] This is a fixed point iteration for the equation $x=(x^2-2)^2+x=f(x)$, whose solutions include $\sqrt{2}$. The iterates always move to the right since $f(x)\geq x$. Hence $x_1=1.4<\sqrt{2}$ converges and $x_1=1.42>\sqrt{2}$ diverges.
[5] The Singular Value Decomposition and Its Applications in Image Compression (http://online.redwoods.cc.ca.us/instruct/darnold/maw/single.htm)
[6] Speed comparison of various number crunching packages (http://www.sciviews.org/benchmark/)
[7] Comparison of mathematical programs for data analysis (http://www.scientificweb.com/ncrunch/ncrunch5.pdf) Stefan Steinhaus, ScientificWeb.com
[8] http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/NAessay.pdf
[9] http://www-gdz.sub.uni-goettingen.de/cgi-bin/digbib.cgi?PPN362160546
[10] http://www.springerlink.com/content/0029-599X
[11] http://siamdl.aip.org/dbt/dbt.jsp?KEY=SJNAAM
[12] http://norma.mas.ecp.fr/wikimas/ScientificComputingSoftware
[13] http://people.sc.fsu.edu/~tomek/Fortran/num_meth.html
[14] http://www.apropos-logic.com/nc/
[15] http://www.ifh.uni-karlsruhe.de/people/fenton/Lectures.html
[16] http://www.akiti.ca/Mathfxns.html
[17] http://www.nr.com/oldverswitcher.html
[18] http://kr.cs.ait.ac.th/~radok/math/mat7/stepsa.htm#Numerical%20Analysis
[19] http://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/
[20] http://www.phy.ornl.gov/csep/CSEP/TEXTOC.html
[21] http://www.damtp.cam.ac.uk/user/fdl/people/sd103/lectures/nummeth98/index.htm#L_1_Title_Page
[22] http://www.math.upenn.edu/~wilf/DeturckWilf.pdf
[23] http://www.ifh.uni-karlsruhe.de/people/fenton/LectureNotes/Numerical-Methods.pdf
[24] http://numericalmethods.eng.usf.edu/
[25] http://math.fullerton.edu/mathews/numerical.html
[26] http://www.math.jct.ac.il/~naiman/nm/
[27] http://www-teaching.physics.ox.ac.uk/computing/NumericalMethods/NMfP.pdf
[28] http://ocw.mit.edu/OcwWeb/Mechanical-Engineering/2-993JSpring-2005/CourseHome/

Article Sources and Contributors

Numerical analysis *Source*: http://en.wikipedia.org/w/index.php?oldid=384345232 *Contributors*: A bit iffy, APH, Aither, Aitter, AmadeusKlocker, Arnero, Arthena, Asyndeton, Audiosmurf, Beliavsky, BenFrantzDale, Berland, Bethnim, BigJohnHenry, Blainster, Bocianski, Bookandcoffee, Brianboonstra, Bubba73, Bugg, CBM, CES1596, CMaes, CRGreathouse, Carrionluggage, Cat2020, Chaos, Charles Matthews, Charvest, Chrismacgr, Christina Silverman, Closedmouth, Conversion script, Crust, DSP-user, Darklilac, Darthhappyface, David Binner, David Haslam, Decrease789, Decrypt3, Didimos, Dominus, Dontaskme, Drunken Pirate, Dungodung, Dysprosia, EXTER7, EagleFan, Eijkhout, Ekojekoj, Elwikipedista, ExamplePuzzle, FatBastardInk, Fintor, Fleminra, Gadfium, Gene Nygaard, Gesslein, Giftlite, Ginkgo100, Gombang, Google Child, Graham87, Greenleaf, Gtpjg, Guanaco, Hede2000, Hlangeveld, Hongooi, HussainAbbas, IronGargoyle, JJL, JLaTondre, Jagged 85, Jaredwf, JesseW, Jitse Niesen, Jmath666, JonMcLoone, Jtir, Jurgen, Justin W Smith, KSmrq, Kate, Kawautar, Kingpin13, Koavf, KymFarnik, Lambiam, Ldo, Lethe, Lightmouse, Linas, Loisel, Loupeter, Lunch, MER-C, Matt Crypto, Mattisse, Meaghan, Michael Hardy, MisterSheik, Mlpkr, Msh210, MystRivenExile, NevilleDNZ, Nikolas Karalis, Ninly, Nixdorf, Nk, Nonphixion, Nostraticispeak, Ntsimp, Oleg Alexandrov, Oliver202, Oyz, Paul Matthews, PedroPVZ, Peter L, PhotoBox, Pred, Quantufinity, Redgecko, Rege, Requestion, RexNL, Robertgreer, Robinh, Rogper, Saforrest, Salih, SamShearman, Sina2, SlackerMom, Sligocki, Stephen B Streater, Stephenkirkup, Stevan White, Svick, Systemlover, Tarquin, Taw, Taxman, TeemuN, The Anome, The undertow, Timothy Clemans, Tobias Hoevekamp, Tom harrison, Tomchen, Tomtzigt, Topbanana, Troworld, Turidoth, Unmerklich, Urdutext, VictorAnyakin, WikipedianMarlith, Windharp, Wolfrock, Wordsoup, Wygk, Xask Linus, Xiaojeng, Yuval madar, Yvwv, Zeno Gantner, Zowie, Zzuuzz, 石庭豐, 183 anonymous edits

Image Sources, Licenses and Contributors

Image:Ybc7289-bw.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ybc7289-bw.jpg *License*: Creative Commons Attribution 2.5 *Contributors*: Billcasselman, Jtir, 1 anonymous edits

Image:Schumacher (Ferrari) in practice at USGP 2005.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Schumacher_(Ferrari)_in_practice_at_USGP_2005.jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: User:Rdsmith4

Image:LemonadeJuly2006.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:LemonadeJuly2006.JPG *License*: GNU Free Documentation License *Contributors*: Evrik, Gveret Tered, Opponent, Polylerus

Image:Wind-particle.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Wind-particle.png *License*: Public Domain *Contributors*: BenFrantzDale, Loisel, 2 anonymous edits

License