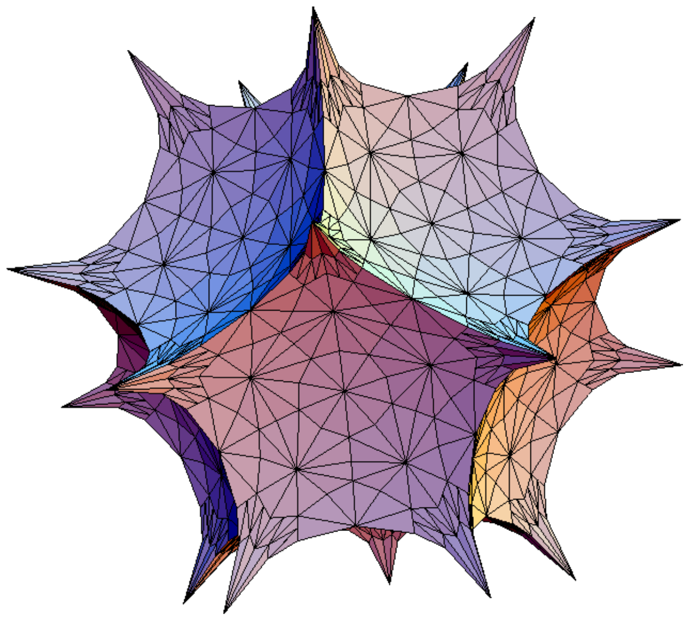
# Introduction to Computer Algebra and Mathematica

Edoardo Milotti

Dipartimento di Fisica, Università di Trieste

## List of topics

- Numerical methods: how to compute square roots
- Symbolic computation
- Basics of Mathematica
- The Euclidean algorithm
- Euclidean division



## How to compute square roots: the Newton-Raphson algorithm

Problem: find a numerical solution of the (nonlinear) equation

$$f(x) = 0$$

starting from the initial guess $x_0$. The true solution is close to the starting value, i.e.,

$$x_s = x_0 + \delta x$$

Then

$$0 = f(x_0 + \delta x) \approx f(x_0) + f'(x_0)\delta x$$

$$0 = f(x_0 + \delta x) \approx f(x_0) + f'(x_0)\delta x$$

$$\Downarrow$$

$$\delta x \approx -\frac{f(x_0)}{f'(x_0)}$$

Successessive iterations $\Rightarrow$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$\ldots$$

Each Newton-Raphson step is equivalent to solving the linear equation

$$y = f(x_0) + f'(x_0)x = 0$$

where the function represents the tangent to the function $f(x)$ in $x_0$.

$$\arctan x = 0$$

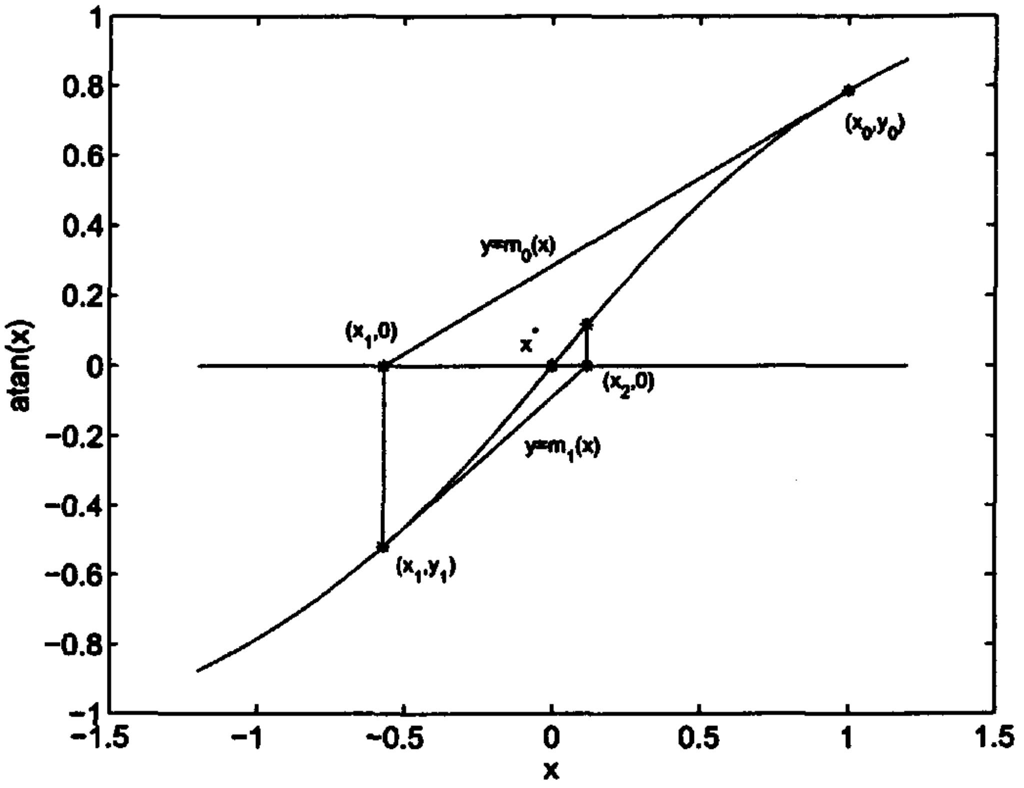


**Figure 1.1.** *Newton iteration for the arctan function.*

## Computing the square root: an application of the Newton-Raphson method

$$x = \sqrt{S} \quad \Longleftrightarrow \quad f(x) = x^2 - S = 0$$

$$f'(x) = 2x; \qquad -\frac{f(x)}{f'(x)} = -\frac{x^2 - S}{2x} = -\frac{x}{2} + \frac{S}{2x}$$

Successive iterates:

$$x_{n+1} = \frac{x_n}{2} + \frac{S}{2x_n}$$

This is the Babylonian method for computing the square root

# Example with the Babylonian method: square root of 2

Successive iterates
with 15 figures precision

| | |
|---|---|
| 1 | 1.5 |
| 2 | 1.41666666666667 |
| 3 | 1.41421568627451 |
| 4 | 1.41421356237469 |
| 5 | 1.414213562373095 |
| 6 | 1.414213562373095 |
| 7 | 1.414213562373095 |
| 8 | 1.414213562373095 |
| 9 | 1.414213562373095 |
| 10 | 1.414213562373095 |

No change from
this point on

## Convergence of the Babylonian method is very fast

Consider the fractional error

$$\epsilon_n = \frac{x_n - \sqrt{S}}{\sqrt{S}}$$

then we find

$$\begin{aligned}
\epsilon_{n+1} &= \frac{|x_{n+1} - \sqrt{S}|}{\sqrt{S}} = \frac{|(x_n + S/x_n)/2 - \sqrt{S}|}{\sqrt{S}} \\
&= \frac{|x_n^2 - 2x_n\sqrt{S} + S|}{2x_n\sqrt{S}} \\
&= \frac{(x_n - \sqrt{S})^2}{2x_n\sqrt{S}} \\
&\approx \frac{(x_n - \sqrt{S})^2}{2S} \\
&= \frac{\epsilon_n^2}{2}
\end{aligned}$$

$$\Rightarrow \qquad \epsilon_{n+1} \approx \frac{\epsilon_n^2}{2}$$

**The fractional error decreases quadratically !**

Moreover, the method is stable.

## The quadratic convergence is a general property of the NR method

The generic iteration step is

$$x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)}$$

therefore, when we introduce the errors $\epsilon_n$ such that

$$x_n = x + \epsilon_n$$

we find

$$\boxed{\epsilon_{n+1} = \epsilon_n - \frac{f'(x_n)}{f(x_n)}}$$

Next, notice that the derivatives can be rewritten in the following way

$$f(x_n) = f(x+\epsilon_n) \approx f(x) + f'(x)\epsilon_n + \frac{1}{2}f''(x_n)\epsilon_n^2$$

$$= f'(x)\epsilon_n + \frac{1}{2}f''(x_n)\epsilon_n^2$$

$$f'(x_n) = f'(x+\epsilon_n) \approx f'(x) + f''(x)\epsilon_n$$

$$\epsilon_{n+1} = \epsilon_n - \frac{f(x_n)}{f'(x_n)} \approx \frac{f'(x_n)\epsilon_n - f(x_n)}{f'(x_n)}$$

$$\approx \frac{f'(x)\epsilon_n + f''(x)\epsilon_n^2 - f'(x)\epsilon_n - \frac{1}{2}f''(x)\epsilon_n^2}{f'(x)}$$

$$= \frac{f''(x)}{2f'(x)}\epsilon_n^2$$

a quadratic formula again !
Fast convergence is guaranteed for the NR method.

## Variants?

One of many variants uses the extraction of the inverse square root

$$f(x) = \frac{1}{x^2} - S = 0 \quad \Rightarrow \quad f'(x) = -2/x^3; \quad \delta x = -\frac{f(x)}{f'(x)} = \frac{1/x^2 - S}{2/x^3}$$

then  we find

$$x_{n+1} = x_n + \frac{x_n}{2}(1 - Sx_n^2) = \frac{x_n}{2}(3 - Sx_n^2)$$

This method is interesting because each iterations involves one subtraction and four multiplications instead of one addition, two multiplications and one division. Since divisions are considerably slower than the other elementary operations, this variant is faster. One finds the square root with one final multiplication:

$$\sqrt{S} = S \times \frac{1}{\sqrt{S}}$$

**Note, however, that the variant is not stable !!!**

$$x_{n+1} = \frac{x_n}{2}(3 - Sx_n^2)$$

The parenthesis may be negative. This happens for

$$x_0 > \sqrt{3/S}$$

and in this case the solution picks up the wrong sign.

Moreover, if

$$x_0 > \sqrt{5/S}$$

then successive iterates are larger and larger (in absolute value) and the solution diverges (**the solution is unstable**).

## The NR method may have stability problems when the trial solution is far from the actual solution and there are derivative flips

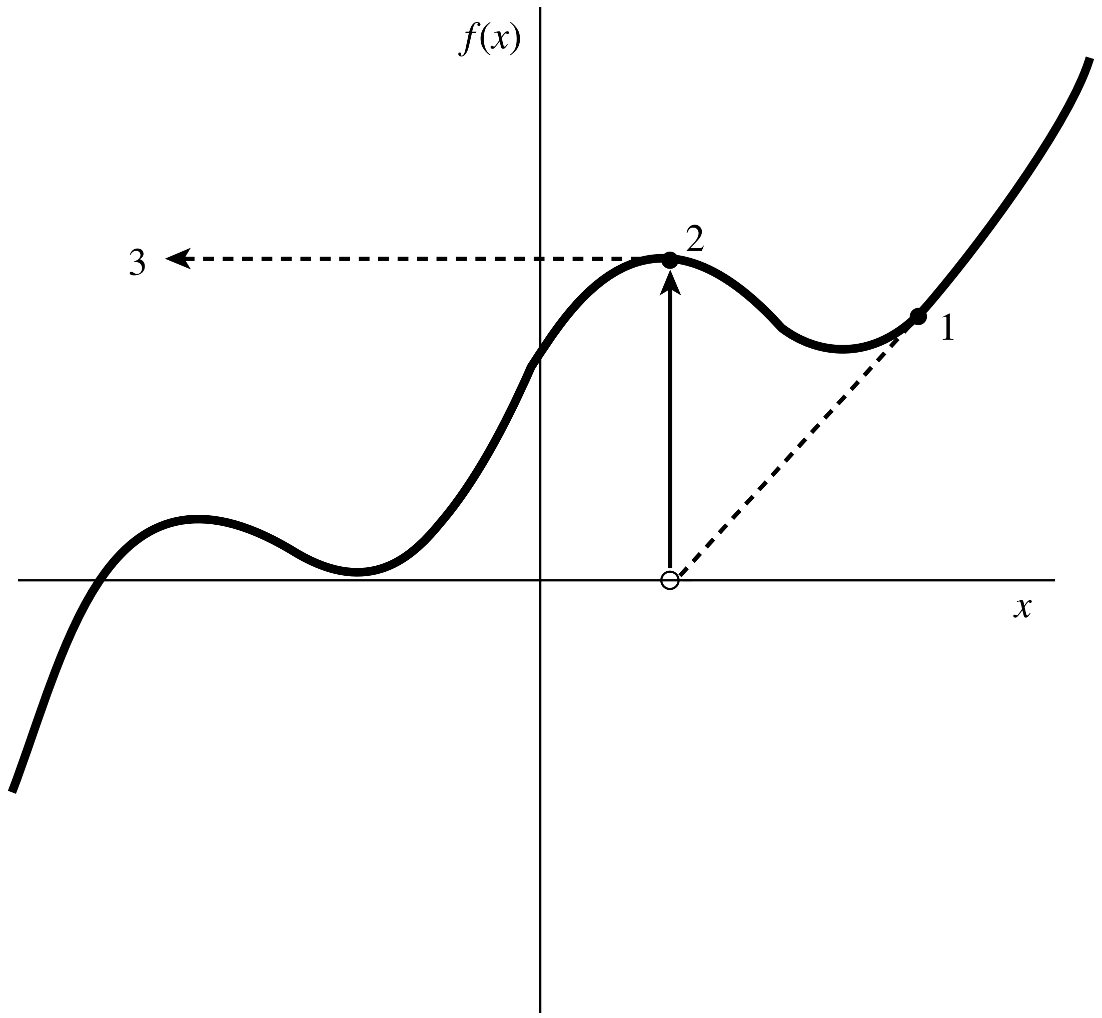


Figure 9.4.2. Unfortunate case where Newton's method encounters a local extremum and shoots off to outer space. Here bracketing bounds, as in `rtsafe`, would save the day.

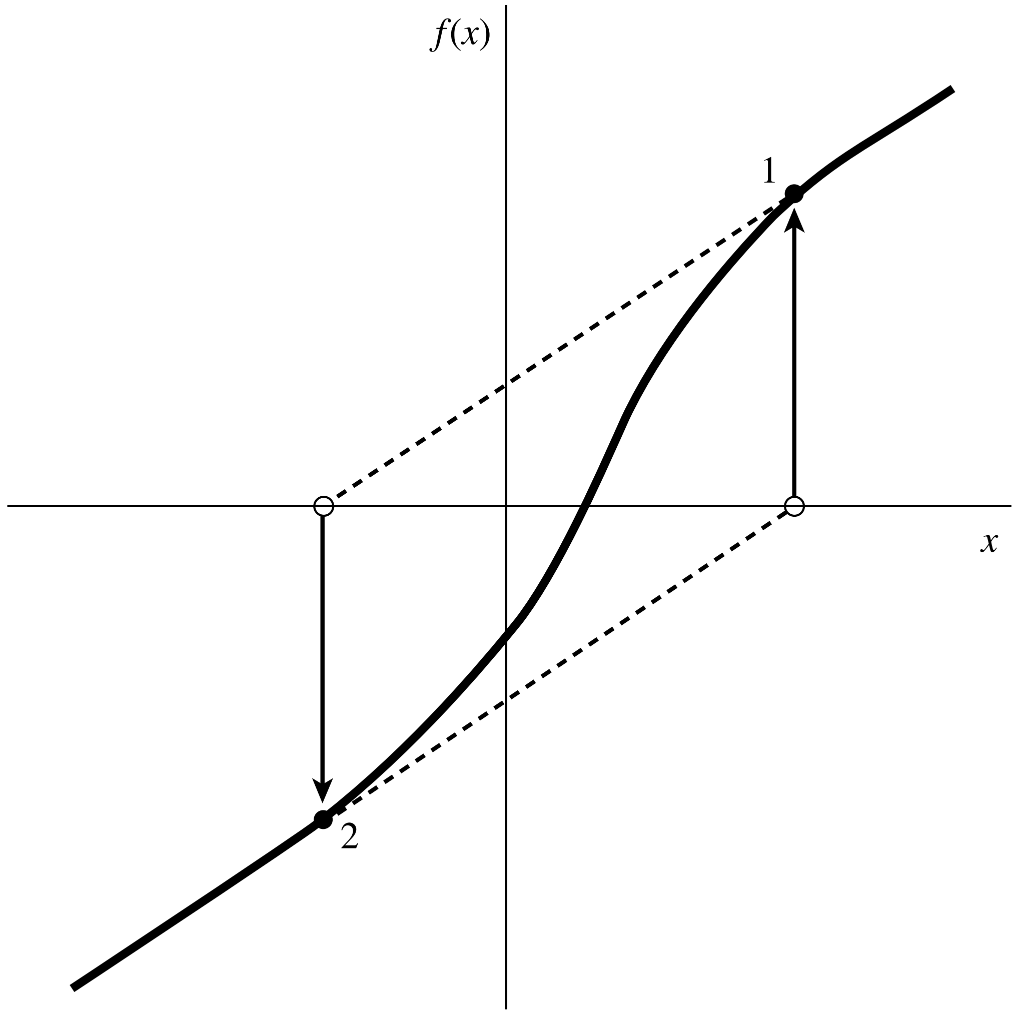
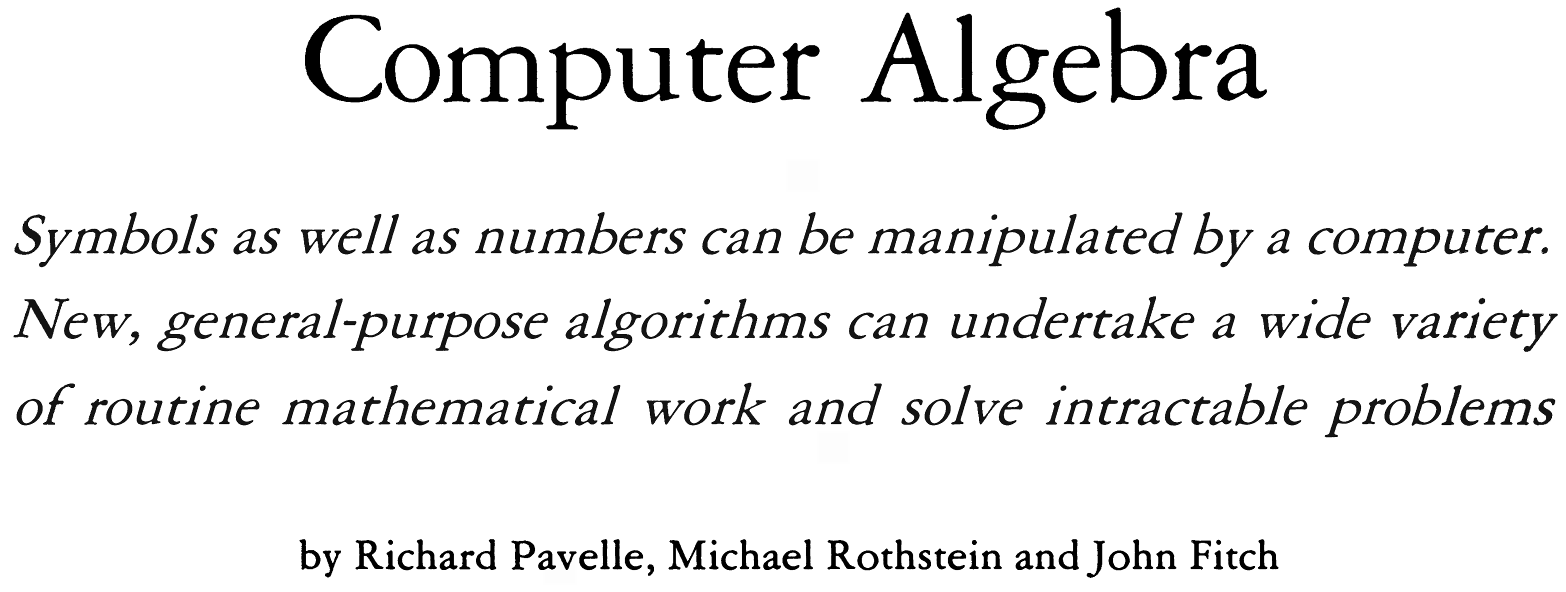


**Figure 9.4.3.** Unfortunate case where Newton's method enters a nonconvergent cycle. This behavior is often encountered when the function $f$ is obtained, in whole or in part, by table interpolation. With a better initial guess, the method would have succeeded.

## Symbolic computation



# Computer Algebra

*Symbols as well as numbers can be manipulated by a computer. New, general-purpose algorithms can undertake a wide variety of routine mathematical work and solve intractable problems*

by Richard Pavelle, Michael Rothstein and John Fitch

Charles-Eugène Delaunay (9 April 1816 – 5 August 1872) was a French astronomer and mathematician.

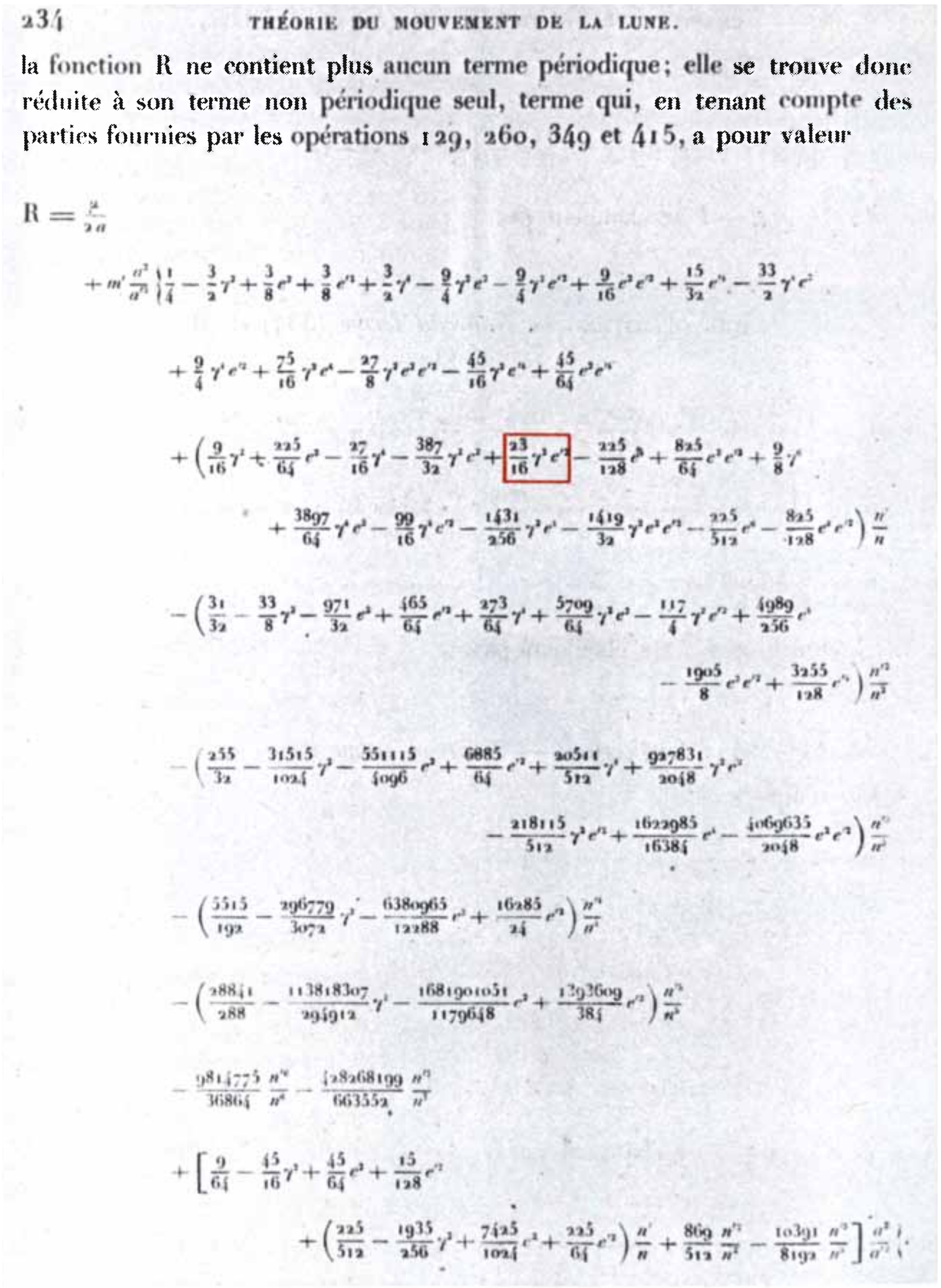
His lunar motion studies were important in advancing both the theory of planetary motion and mathematics.

From the year 1846, M. Delaunay devoted all his available time to the carrying out of this important work. After fourteen years of assiduous labour, he published in 1860, the first volume, entitled *La Théorie de la Lune*, forming one of the volumes of the Memoirs of the French Institute. During this interval, he, in separate memoirs, published papers on several questions relating to the theory, notably that on the secular acceleration of the mean motion of the Moon, which at this time was a subject exciting much controversy. The magnitude even of this supplementary work may be conceived from an extract from an interesting review of the state of this controversy in 1859, by the Rev. R. Main, then President of the Society. "The reader, who desires to see the exact expression for the acceleration arrived at by M. Delaunay, must consult the work referred to, as it is too long for insertion here, consisting, as it does, of *forty-two* terms. Thus much may be said of it, that it takes in every possible term as far as the eighth order, including those terms depending on the inclination and excentricity of the lunar orbit; and that all the terms calculated by Mr. Adams are in exact accordance with it." In 1867, the second volume appeared in continuation of the first, completing by far the most difficult portion of the work. The third volume, of which the contents have, for the greater part, been prepared for some time, will bring this great undertaking to a conclusion so far as concerns the theory.

“This enormous labour, which has occupied M. Delaunay for nearly twenty years, has been performed by him without assistance from anyone. Indeed, from the nature of the calculations which are required, it would not have been easy to obtain any effective assistance. In order to ensure accuracy, M. Delaunay has omitted no means of verification, and he has performed all the calculations, without exception, at two separate times, with a sufficient interval between them to prevent any special risk of committing the same error twice in succession.

“The volumes before us are perfect models of orderly arrangement. Notwithstanding the great length and complication of the calculations, the whole work is so disposed that any part of it may be speedily examined with the utmost readiness by anyone who may wish to test its accuracy.

“Finally, the analytical expressions which have been obtained for the Moon's co-ordinates are converted into numbers, by substituting for the elements the most accurate numerical values which the comparison of theory with observation has made known. * * * The work is complete in itself; in it the very difficult and complicated problem of determining the Moon's motion is attacked by a perfectly original method, and that one as powerful and beautiful as it is new. The work has been planned with admirable skill, and has been carried out with matchless perseverance. The result is an enduring scientific monument of which our age may well be proud.” *



234 THÉORIE DU MOUVEMENT DE LA LUNE.

la fonction R ne contient plus aucun terme périodique; elle se trouve donc réduite à son terme non périodique seul, terme qui, en tenant compte des parties fournies par les opérations 129, 260, 349 et 415, a pour valeur

$$
\begin{aligned}
R = \frac{\mu}{2a}\\
+ m'\frac{a^2}{a'^3}\Big\{ & \frac{1}{4} - \frac{3}{2}\gamma^2 + \frac{3}{8}e^2 + \frac{3}{8}e'^2 + \frac{3}{2}\gamma^4 - \frac{9}{4}\gamma^2e^2 - \frac{9}{4}\gamma^2e'^2 + \frac{9}{16}e^2e'^2 + \frac{15}{32}e'^4 - \frac{33}{2}\gamma^4e^2\\
& + \frac{9}{4}\gamma^4e'^2 + \frac{75}{16}\gamma^2e^4 - \frac{27}{8}\gamma^2e^2e'^2 - \frac{45}{16}\gamma^2e'^4 + \frac{45}{64}e^2e'^4\\
& + \Big(\frac{9}{16}\gamma^2 + \frac{225}{64}e^2 - \frac{27}{16}\gamma^4 - \frac{387}{32}\gamma^2e^2 + \boxed{\frac{23}{16}\gamma^2e'^2} - \frac{225}{128}e^4 + \frac{825}{64}e^2e'^2 + \frac{9}{8}\gamma^6\\
& \qquad + \frac{3897}{64}\gamma^4e^2 - \frac{99}{16}\gamma^4e'^2 - \frac{1431}{256}\gamma^2e^4 - \frac{1419}{32}\gamma^2e^2e'^2 - \frac{225}{512}e^6 - \frac{825}{128}e^4e'^2\Big)\frac{n'}{n}\\
& - \Big(\frac{31}{32} - \frac{33}{8}\gamma^2 - \frac{971}{32}e^2 + \frac{465}{64}e'^2 + \frac{273}{64}\gamma^4 + \frac{5709}{64}\gamma^2e^2 - \frac{117}{4}\gamma^2e'^2 + \frac{4989}{256}e^4\\
& \qquad - \frac{1905}{8}e^2e'^2 + \frac{3255}{128}e'^4\Big)\frac{n'^2}{n^2}\\
& - \Big(\frac{255}{32} - \frac{31515}{1024}\gamma^2 - \frac{551115}{4096}e^2 + \frac{6885}{64}e'^2 + \frac{20511}{512}\gamma^4 + \frac{927831}{2048}\gamma^2e^2\\
& \qquad - \frac{218115}{512}\gamma^2e'^2 + \frac{1622985}{16384}e^4 - \frac{4069635}{2048}e^2e'^2\Big)\frac{n'^3}{n^3}\\
& - \Big(\frac{5515}{192} - \frac{296779}{3072}\gamma^2 - \frac{6380965}{12288}e^2 + \frac{16285}{24}e'^2\Big)\frac{n'^4}{n^4}\\
& - \Big(\frac{28841}{288} - \frac{113818307}{294912}\gamma^2 - \frac{1681901051}{1179648}e^2 + \frac{1293609}{384}e'^2\Big)\frac{n'^5}{n^5}\\
& - \frac{9814775}{36864}\frac{n'^6}{n^6} - \frac{428268199}{663552}\frac{n'^7}{n^7}\\
& + \Big[\frac{9}{64} - \frac{45}{16}\gamma^2 + \frac{45}{64}e^2 + \frac{15}{128}e'^2\\
& \qquad + \Big(\frac{225}{512} - \frac{1935}{256}\gamma^2 + \frac{7425}{1024}e^2 + \frac{225}{64}e'^2\Big)\frac{n'}{n} + \frac{869}{512}\frac{n'^2}{n^2} - \frac{10391}{8192}\frac{n'^3}{n^3}\Big]\frac{a^2}{a'^2}\Big\}.
\end{aligned}
$$

MASSIVE ALGEBRAIC CALCULATION completed by hand in 1867 was recomputed for the first time in 1970 with a computer-algebra system.

The original calculation was undertaken by the French astronomer Charles Delaunay and was published in two volumes; a page of the second volume is reproduced here. Delaunay's calculation, which took 10 years to finish and another 10 years to check, gave the position of the moon as a function of time to a precision never before attained. Three errors were detected when the calculation was checked by Andre Deprit, Jacques Henrard and Arnold Rom of the Boeing Scientific Research Laboratories in Seattle. The major error, which gave rise to the other two errors, is outlined in color. The checking required about 20 hours of running time on a computer. Although computer algebra was responsible for exposing Delaunay's errors, the tables have since been turned: the remarkable precision of the calculation has been employed to test a few new computer-algebra systems for accuracy before they were put into service.

(from Pavelle, Rothstein & Fitch, Sci. Am. Issue 6, 1981)

THE ASTRONOMICAL JOURNAL VOLUME 76, NUMBER 3 APRIL 1971

# Analytical Lunar Ephemeris: Delaunay's Theory

André Deprit, Jacques Henrard, and Arnold Rom
*Boeing Scientific Research Laboratories, Seattle, Washington*

(Received 1 December 1970)

Delaunay's constants have been substituted into our analytical solution of the main problem of lunar theory. The results are compared with Delaunay's reduced formulas. Corrections are proposed to four terms in the mean motion of the perigee, three in the mean motion of the node, 45 in the reduced expression for the latitude, and 49 in that for the longitude.

## Computational and algorithmic tools

- **LISP**

- **Gosper’s algorithm** (a procedure for finding sums of hypergeometric terms that are themselves hypergeometric terms. That is: suppose we have a(1) + ... + a(n) = S(n) − S(0), where S(n) is a hypergeometric term (i.e., S(n + 1)/S(n) is a rational function of n); then necessarily a(n) is itself a hypergeometric term, and given the formula for a(n) Gosper's algorithm finds that for S(n))

- **Risch’s algorithm** (a semi-algorithm for indefinite integration. It is used in some computer algebra systems to find antiderivatives. It is named after the American mathematician Robert Henry Risch, a specialist in computer algebra who developed it in 1968)

- ...

*Proc. Natl. Acad. Sci. USA*
Vol. 75, No. 1, pp. 40–42, January 1978
Mathematics

# Decision procedure for indefinite hypergeometric summation

**(algorithm/binomial coefficient identities/closed form/symbolic computation/linear recurrences)**

R. WILLIAM GOSPER, JR.

Xerox Palo Alto Research Center, Palo Alto, California 94304

*Communicated by Donald E. Knuth, September 26, 1977*

**ABSTRACT** Given a summand $a_n$, we seek the "indefinite sum" $S(n)$ determined (within an additive constant) by

$$\sum_{n=1}^{m} a_n = S(m) - S(0) \qquad [0]$$

or, equivalently, by

$$a_n = S(n) - S(n-1). \qquad [1]$$

An algorithm is exhibited which, given $a_n$, finds those $S(n)$ with the property

$$\frac{S(n)}{S(n-1)} = \text{a rational function of } n. \qquad [2]$$

With this algorithm, we can determine, for example, the three identities

$$\sum_{n=1}^{m} \frac{\prod_{j=1}^{n-1} bj^2 + cj + d}{\prod_{j=1}^{n} bj^2 + cj + e} = \frac{1 - \prod_{j=1}^{m} \frac{bj^2 + cj + d}{bj^2 + cj + e}}{e - d}, \qquad [3a]$$

$$\sum_{n=1}^{m} \frac{\prod_{j=1}^{n-1} aj^3 + bj^2 + cj + d}{\prod_{j=1}^{n} aj^3 + bj^2 + cj + e} = \frac{1 - \prod_{j=1}^{m} \frac{aj^3 + bj^2 + cj + d}{aj^3 + bj^2 + cj + e}}{e - d}, \qquad [3b]$$

and
$$\sum_{n=1}^{m} \frac{\prod_{j=1}^{n-1} bj^2 + cj + d}{\prod_{j=1}^{n+1} bj^2 + cj + e} = \frac{\frac{2b}{e-d} - \frac{3b + c + d - e}{b + c + e} - \left(\frac{2b}{e-d} - \frac{b(2m+3) + c + d - e}{b(m+1)^2 + c(m+1) + e}\right) \prod_{j=1}^{m} \frac{bj^2 + cj + d}{bj^2 + cj + e}}{b^2 - c^2 + d^2 + e^2 + 2bd - 2de + 2eb} \qquad [3c]$$

and we can also conclude that

$$\sum_{n=1}^{m} \frac{\prod_{j=1}^{n-1} j^3}{\prod_{j=1}^{n+1} j^3 + 1} \qquad [3d]$$

is inexpressible as $S(m) - S(0)$, for any $S(n)$ satisfying Eq. 2.

Bull. Amer. Math. Soc.
Volume 76, Number 3 (1970), 605-608.

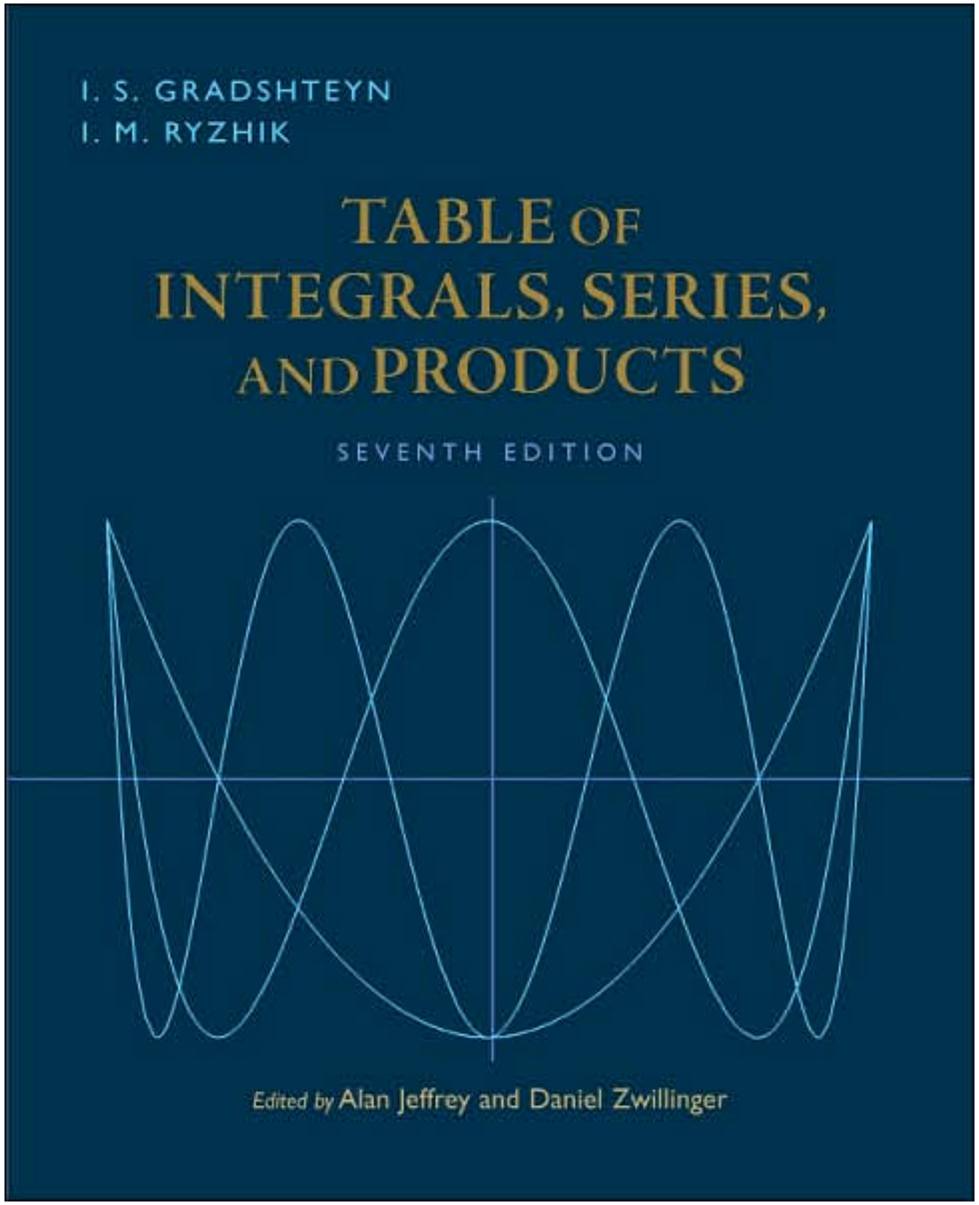
# THE SOLUTION OF THE PROBLEM OF INTEGRATION IN FINITE TERMS

BY ROBERT H. RISCH

Communicated by M. H. Protter, October 22, 1969

**Introduction.** The problem of integration in finite terms asks for an algorithm for deciding whether an elementary function has an elementary indefinite integral and for finding the integral if it does. "Elementary" is used here to denote those functions built up from the rational functions using only exponentiation, logarithms, trigonometric, inverse trigonometric and algebraic operations.

This vaguely worded question has several precise, but inequivalent formulations. The writer has devised an algorithm which solves the classical problem of Liouville. A complete account is planned for a future publication. The present note is intended to indicate some of the ideas and techniques involved.


I. S. GRADSHTEYN
I. M. RYZHIK
TABLE OF INTEGRALS, SERIES, AND PRODUCTS
SEVENTH EDITION
Edited by Alan Jeffrey and Daniel Zwillinger

(first published in 1943)

## Computer Algebra Systems

For a current list of CAS go to

https://en.wikipedia.org/wiki/List_of_computer_algebra_systems

UM–TH–91–18
June 9, 1993

# Schoonschip '91

Martinus J. G. Veltman and David N. Williams

*Randall Laboratory of Physics*
*The University of Michigan*
*Ann Arbor, MI 48109-1120*

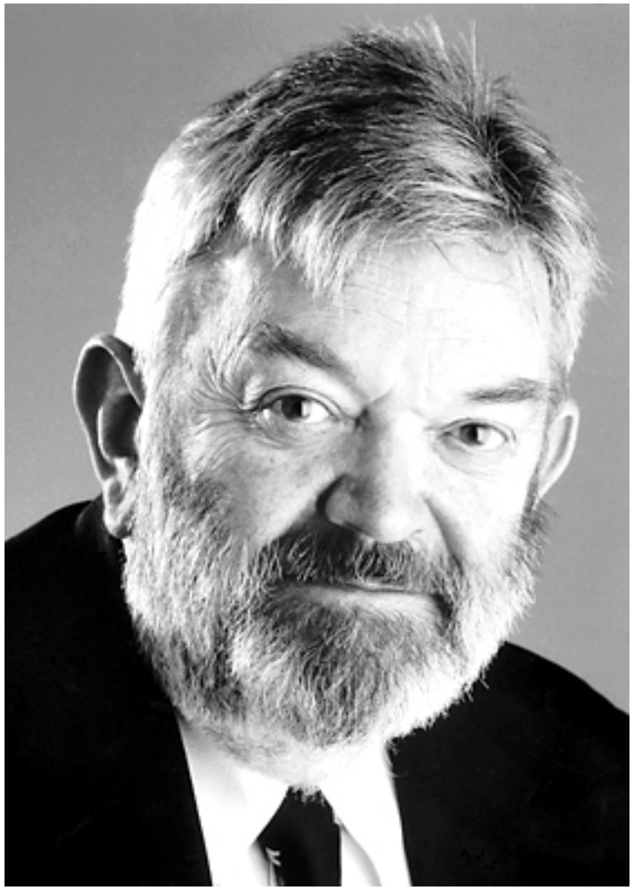
## ABSTRACT

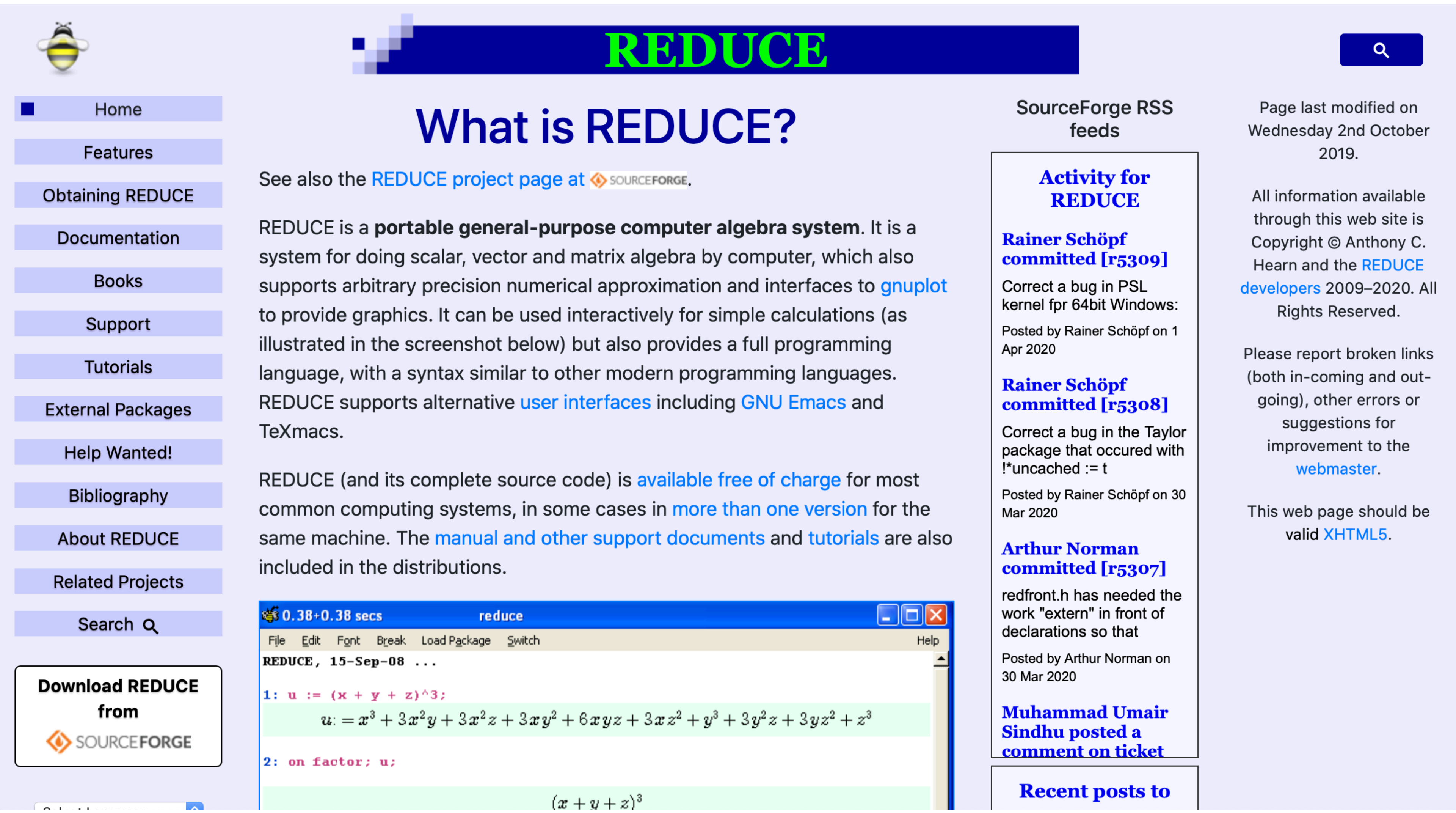
The symbolic manipulation program Schoonschip is being made freely available for a number of computers with Motorola 680x0 cpu's. It can run on machines with relatively modest memory and disk resources, and is designed to run as fast as possible given the host constraints. Memory and disk utilization can be adapted to tune performance. Recently added capabilities include a system for efficient generation of diagrams, gamma algebra for continuous dimensions, algorithmic improvements for handling large problems, and an increase in the allowed number of `X` expressions.

In 1963/64, during an extended stay at SLAC, Martinus J. G. Veltman designed the computer program Schoonschip for symbolic manipulation of mathematical equations, which is now considered the very first computer algebra system.

Veltman initially developed the program to compute the quadrupole moment of the W boson, the computation of which led to "a monstrous expression involving in the order of 50,000 terms in intermediate stages".

Martinus Justinus Godefriedus "Tini" Veltman (born 27 June 1931) is a Dutch theoretical physicist. He shared the 1999 Nobel Prize in physics with his former student Gerardus 't Hooft for their work on particle theory.





## What is REDUCE?

See also the REDUCE project page at SOURCEFORGE.

REDUCE is a **portable general-purpose computer algebra system**. It is a system for doing scalar, vector and matrix algebra by computer, which also supports arbitrary precision numerical approximation and interfaces to gnuplot to provide graphics. It can be used interactively for simple calculations (as illustrated in the screenshot below) but also provides a full programming language, with a syntax similar to other modern programming languages. REDUCE supports alternative user interfaces including GNU Emacs and TeXmacs.

REDUCE (and its complete source code) is available free of charge for most common computing systems, in some cases in more than one version for the same machine. The manual and other support documents and tutorials are also included in the distributions.

```
REDUCE, 15-Sep-08 ...

1: u := (x + y + z)^3;
```

$$u := x^3 + 3x^2y + 3x^2z + 3xy^2 + 6xyz + 3xz^2 + y^3 + 3y^2z + 3yz^2 + z^3$$

```
2: on factor; u;
```

$$(x + y + z)^3$$

SourceForge RSS feeds

**Activity for REDUCE**

**Rainer Schöpf committed [r5309]**

Correct a bug in PSL kernel fpr 64bit Windows:

Posted by Rainer Schöpf on 1 Apr 2020

**Rainer Schöpf committed [r5308]**

Correct a bug in the Taylor package that occured with !*uncached := t

Posted by Rainer Schöpf on 30 Mar 2020

**Arthur Norman committed [r5307]**

redfront.h has needed the work "extern" in front of declarations so that

Posted by Arthur Norman on 30 Mar 2020

**Muhammad Umair Sindhu posted a comment on ticket**

**Recent posts to**

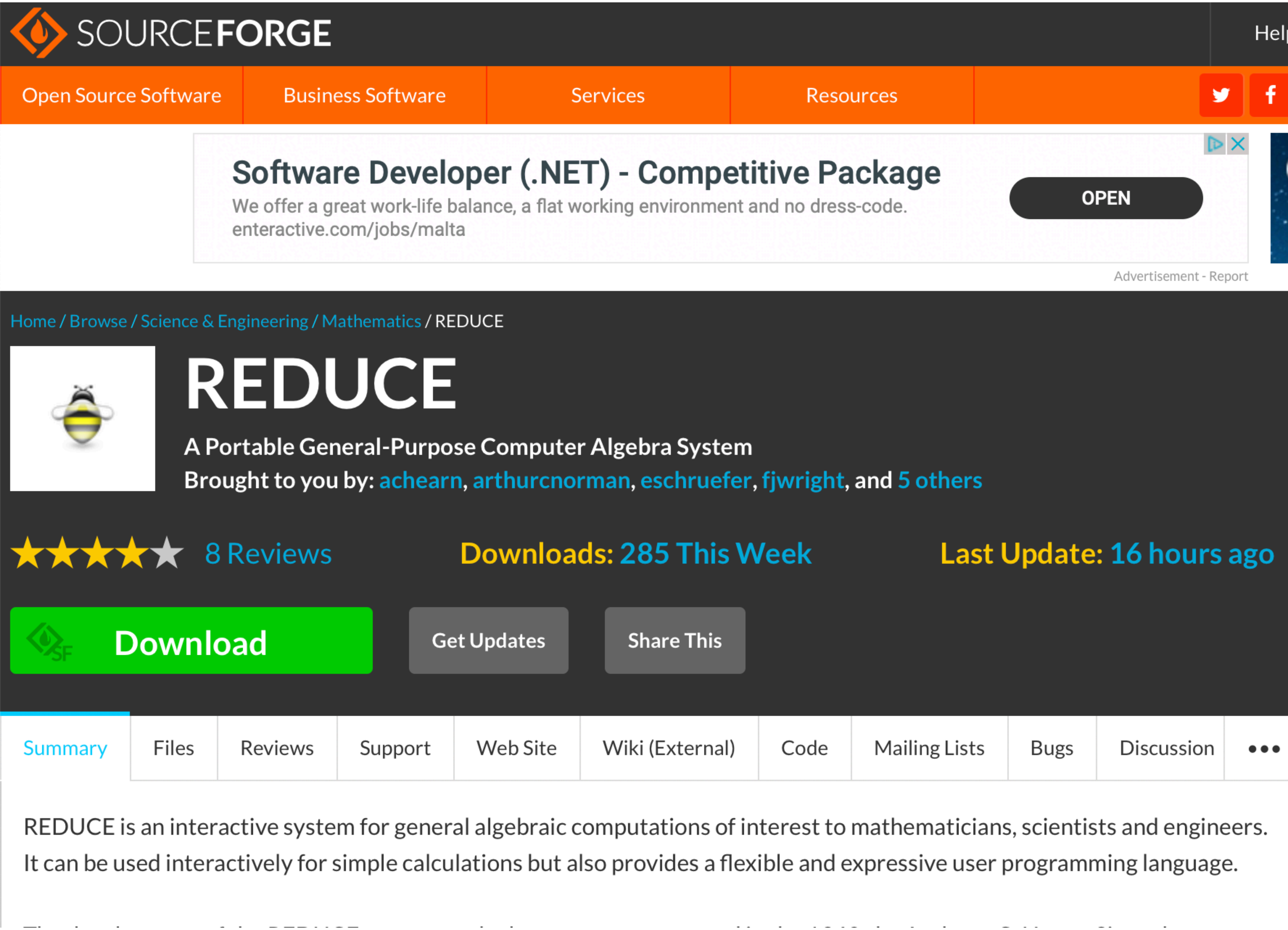
Page last modified on Wednesday 2nd October 2019.

All information available through this web site is Copyright © Anthony C. Hearn and the REDUCE developers 2009–2020. All Rights Reserved.

Please report broken links (both in-coming and out-going), other errors or suggestions for improvement to the webmaster.

This web page should be valid XHTML5.

https://reduce-algebra.sourceforge.io/index.php



Home / Browse / Science & Engineering / Mathematics / REDUCE

REDUCE

A Portable General-Purpose Computer Algebra System

Brought to you by: achearn, arthurcnorman, eschruefer, fjwright, and 5 others

★★★★☆ 8 Reviews

Downloads: 285 This Week

Last Update: 16 hours ago

REDUCE is an interactive system for general algebraic computations of interest to mathematicians, scientists and engineers. It can be used interactively for simple calculations but also provides a flexible and expressive user programming language.

# PHOTON-PHOTON SCATTERING CONTRIBUTION TO THE SIXTH-ORDER MAGNETIC MOMENT OF THE MUON*

Janis Aldins† and Toichiro Kinoshita
Laboratory of Nuclear Studies, Cornell University, Ithaca, New York 14850

and

Stanley J. Brodsky and Andrew J. Dufner
Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305

(Received 25 July 1969)

We report a calculation of the three-photon—exchange (electron-loop) contribution to the sixth-order anomalous magnetic moment of the muon. Our result, which contains a logarithmic dependence on the muon-to-electron mass ratio, brings the theoretical prediction into agreement with the CERN measurements, within the 1—standard-deviation experimental accuracy.

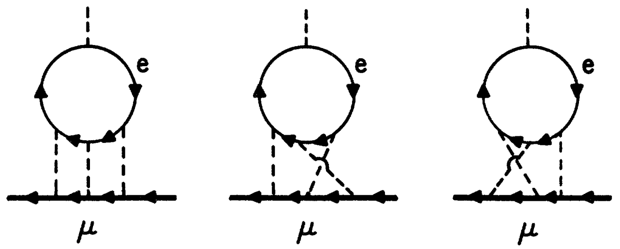


FIG. 1. Feynman diagrams containing subdiagrams of photon-photon scattering type. The heavy, thin, and dotted lines represent the muon, electron, and photon, respectively. There are three more diagrams obtained by reversing the direction of the electron loop.

We are greatly in debt to Dr. A. C. Hearn for providing his algebraic computation program REDUCE and his assistance in its use.

PHYSICAL REVIEW A VOLUME 28, NUMBER 2 AUGUST 1983

# Rate for positronium decay to five photons

Gregory S. Adkins and Frank R. Brown
*Joseph Henry Laboratories, Princeton University, Princeton, New Jersey 08544*
(Received 1 March 1983)

In this report we calculate $\Gamma_5$, the rate for orthopositronium to decay to five photons. We find that $\Gamma_5 = [0.0189(11)]\alpha^2\Gamma_{\rm LO}$, where $\Gamma_{\rm LO}$ is the lowest-order orthopositronium decay rate. The relevance of our result to the experimental situation is discussed.

The positronium atom is involved in a number of tests of QED. Although they are generally not as precise as the beautiful studies of the electron and muon magnetic moments, the positronium tests complement that work because they involve bound-state physics and annihilation processes. In this Brief Report we present a calculation of the five-photon decay rate of orthopositronium (the spin-triplet state) and discuss its relevance to recent measurements of the total orthopositronium decay rate.

Orthopositronium decays to an odd number of photons. Current experiments do not determine the total number of final-state photons, so the measurable decay rate can be written as a sum of partial rates,

$$\Gamma = \Gamma_3 + \Gamma_5 + \Gamma_7 + \cdots \quad . \tag{1}$$

The five-photon parital decay rate is (to lowest order)

$$\Gamma_5 = \int d\tilde{k}_1 \cdots d\tilde{k}_5 \frac{1}{2P^0}(2\pi)^4\delta(P-k_1-\cdots-k_5)\frac{1}{5!}\overline{|M|^2} \quad , \tag{5}$$

The invariant matrix element is

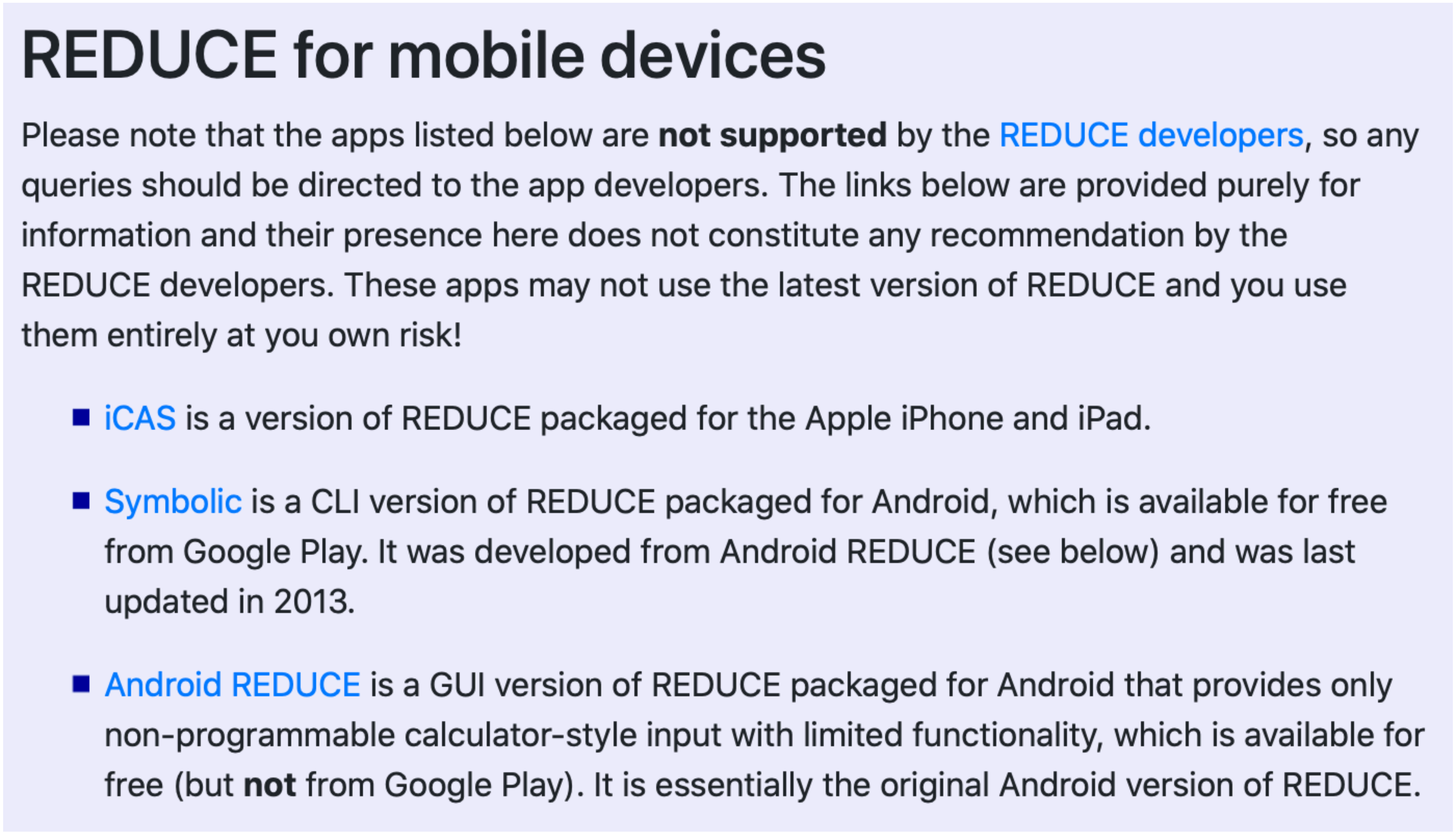
$$M = -16\pi^2 i\left(\frac{\alpha^4}{m^2}\right)\sum_{\sigma\in S_5}\operatorname{tr}\Big[\gamma\epsilon_{\sigma(5)}(-\gamma R_{\sigma(5)}-1)^{-1}\gamma\epsilon_{\sigma(4)}(-\gamma Q_{\sigma(5),\sigma(4)}-1)^{-1} \times\gamma\epsilon_{\sigma(3)}(\gamma Q_{\sigma(2),\sigma(1)}-1)^{-1}\gamma\epsilon_{\sigma(2)}(\gamma R_{\sigma(1)}-1)^{-1}\gamma\epsilon_{\sigma(1)}\begin{pmatrix}0 & \vec{\sigma}\cdot\hat{\epsilon}_m\\ 0 & 0\end{pmatrix}\Big] \quad .$$

As a check of our methods the four-photon partial decay rate of parapositronium (the spin-singlet state) was also obtained. Our result

$$\Gamma_4 = [0.01389(6)]m\alpha^7 \tag{16}$$

agrees well with previous evaluations of this quantity.[6, 7]

The evaluation of $\Gamma_4$ was first attempted by McCoyd.[8] McCoyd's result was about a factor of 4 too low because of an incorrect assumption about the phase-space population. However, McCoyd carried out a monumental amount of algebra by hand, and we have verified that his result for the "nonrelativistic limit of reduced cross section: $e^+ + e^- \rightarrow 4\gamma$" is correct. This result, consisting of 149 simple terms, is much shorter than the equivalent, but not fully simplified, expression that we obtained using REDUCE.

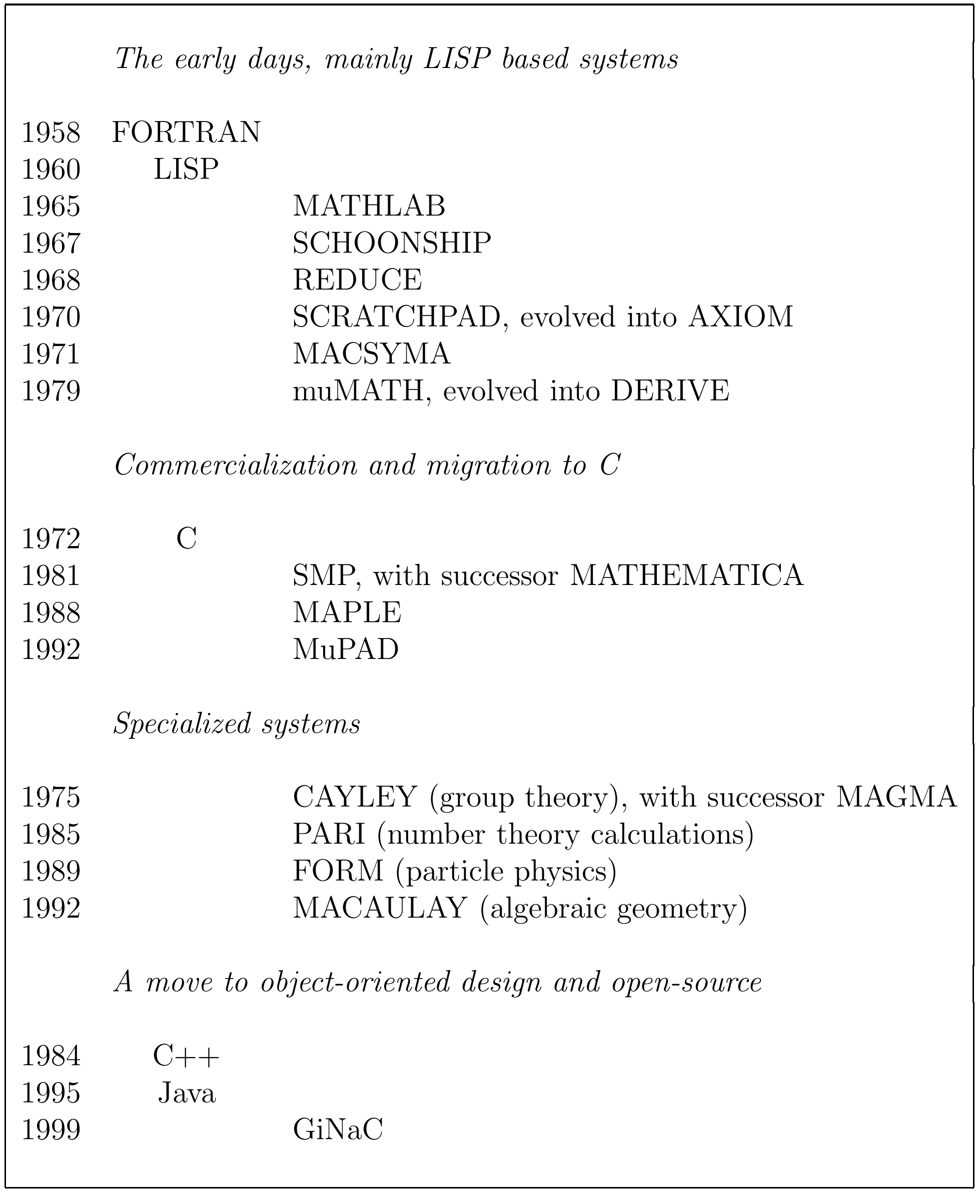


# REDUCE for mobile devices

Please note that the apps listed below are **not supported** by the REDUCE developers, so any queries should be directed to the app developers. The links below are provided purely for information and their presence here does not constitute any recommendation by the REDUCE developers. These apps may not use the latest version of REDUCE and you use them entirely at you own risk!

- iCAS is a version of REDUCE packaged for the Apple iPhone and iPad.
- Symbolic is a CLI version of REDUCE packaged for Android, which is available for free from Google Play. It was developed from Android REDUCE (see below) and was last updated in 2013.
- Android REDUCE is a GUI version of REDUCE packaged for Android that provides only non-programmable calculator-style input with limited functionality, which is available for free (but **not** from Google Play). It is essentially the original Android version of REDUCE.

## Computer algebra systems in particle physics
## (from Weinzierl, ''Computer Algebra in Particle Physics'' https://arxiv.org/abs/hep-ph/0209234v1)



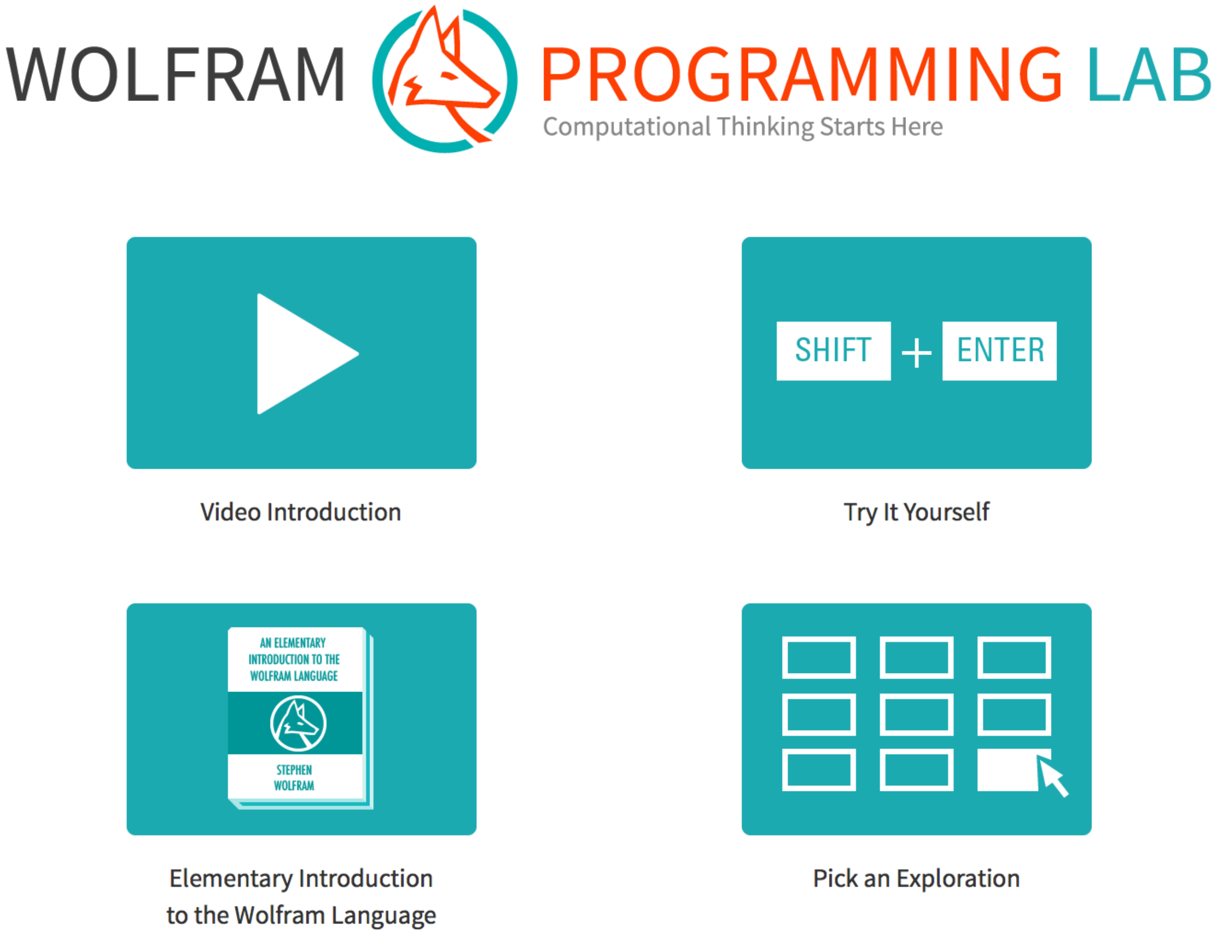
| | | |
|---|---|---|
| | *The early days, mainly LISP based systems* | |
| 1958 | FORTRAN | |
| 1960 | LISP | |
| 1965 | | MATHLAB |
| 1967 | | SCHOONSHIP |
| 1968 | | REDUCE |
| 1970 | | SCRATCHPAD, evolved into AXIOM |
| 1971 | | MACSYMA |
| 1979 | | muMATH, evolved into DERIVE |
| | *Commercialization and migration to C* | |
| 1972 | C | |
| 1981 | | SMP, with successor MATHEMATICA |
| 1988 | | MAPLE |
| 1992 | | MuPAD |
| | *Specialized systems* | |
| 1975 | | CAYLEY (group theory), with successor MAGMA |
| 1985 | | PARI (number theory calculations) |
| 1989 | | FORM (particle physics) |
| 1992 | | MACAULAY (algebraic geometry) |
| | *A move to object-oriented design and open-source* | |
| 1984 | C++ | |
| 1995 | Java | |
| 1999 | | GiNaC |

For an updated list see also

https://en.wikipedia.org/wiki/List_of_computer_algebra_systems

## Setup your environment

Go to [http://lab.wolframcloud.com](http://lab.wolframcloud.com) and register



WOLFRAM PROGRAMMING LAB

Computational Thinking Starts Here

Video Introduction

Try It Yourself

Elementary Introduction to the Wolfram Language

Pick an Exploration

## Basics of Mathematica

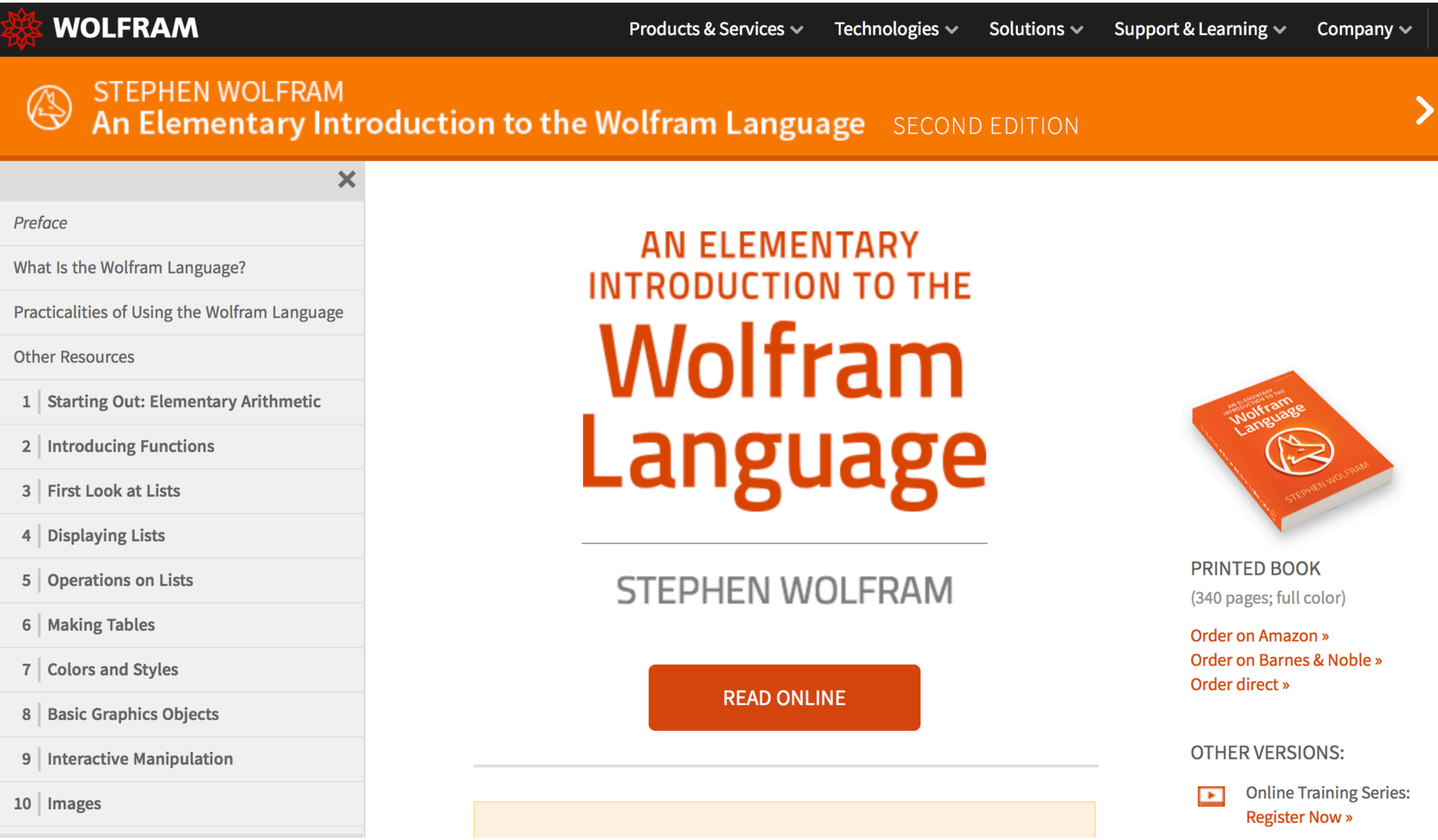
STEPHEN WOLFRAM

An Elementary Introduction to the Wolfram Language SECOND EDITION

http://www.wolfram.com/language/elementary-introduction/2nd-ed/

v. anche il corso del Prof. Marco Budinich

https://wwwusers.ts.infn.it/~mbh/Prog_2019-2020_1.pdf



WOLFRAM

Products & Services | Technologies | Solutions | Support & Learning | Company

STEPHEN WOLFRAM
An Elementary Introduction to the Wolfram Language SECOND EDITION

AN ELEMENTARY INTRODUCTION TO THE
Wolfram Language

STEPHEN WOLFRAM

READ ONLINE

PRINTED BOOK
(340 pages; full color)

Order on Amazon »
Order on Barnes & Noble »
Order direct »

OTHER VERSIONS:

Online Training Series:
Register Now »

## Mathematica easily solves 3rd degree algebraic equations ...

```
In[•]:= Solve[x ^ 3 + b * x ^ 2 + c * x + d == 0, x]
```

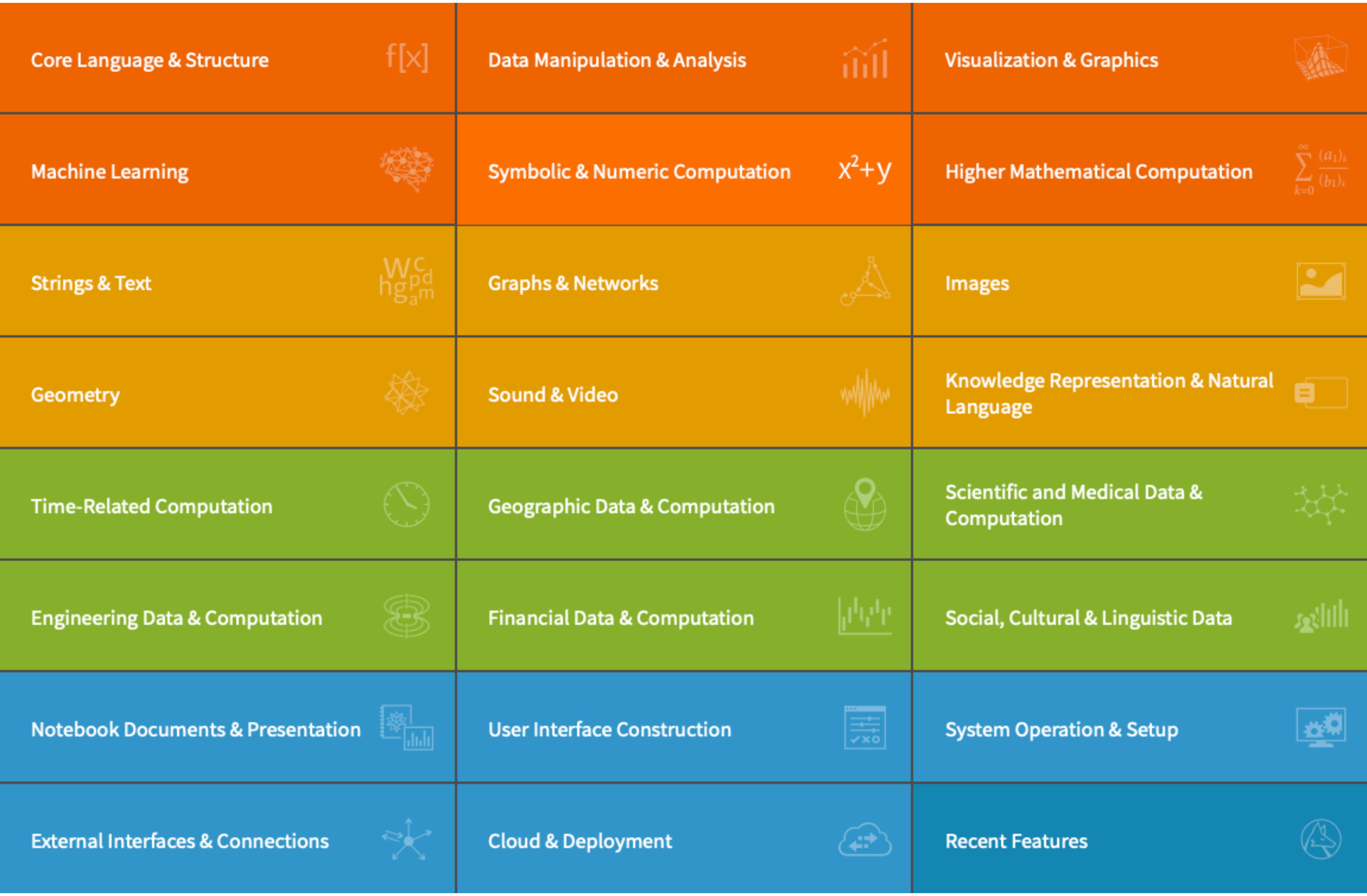
Out[•]=

$$\Bigg\{\Bigg\{x \to -\frac{b}{3} - \frac{2^{1/3}\left(-b^2+3c\right)}{3\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}} + \frac{\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}}{3\times 2^{1/3}}\Bigg\},$$

$$\Bigg\{x \to -\frac{b}{3} + \frac{\left(1+i\sqrt{3}\right)\left(-b^2+3c\right)}{3\times 2^{2/3}\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}} - \frac{\left(1-i\sqrt{3}\right)\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}}{6\times 2^{1/3}}\Bigg\},$$

$$\Bigg\{x \to -\frac{b}{3} + \frac{\left(1-i\sqrt{3}\right)\left(-b^2+3c\right)}{3\times 2^{2/3}\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}} - \frac{\left(1+i\sqrt{3}\right)\left(-2b^3+9bc-27d+3\sqrt{3}\sqrt{-b^2c^2+4c^3+4b^3d-18bcd+27d^2}\right)^{1/3}}{6\times 2^{1/3}}\Bigg\}\Bigg\}$$

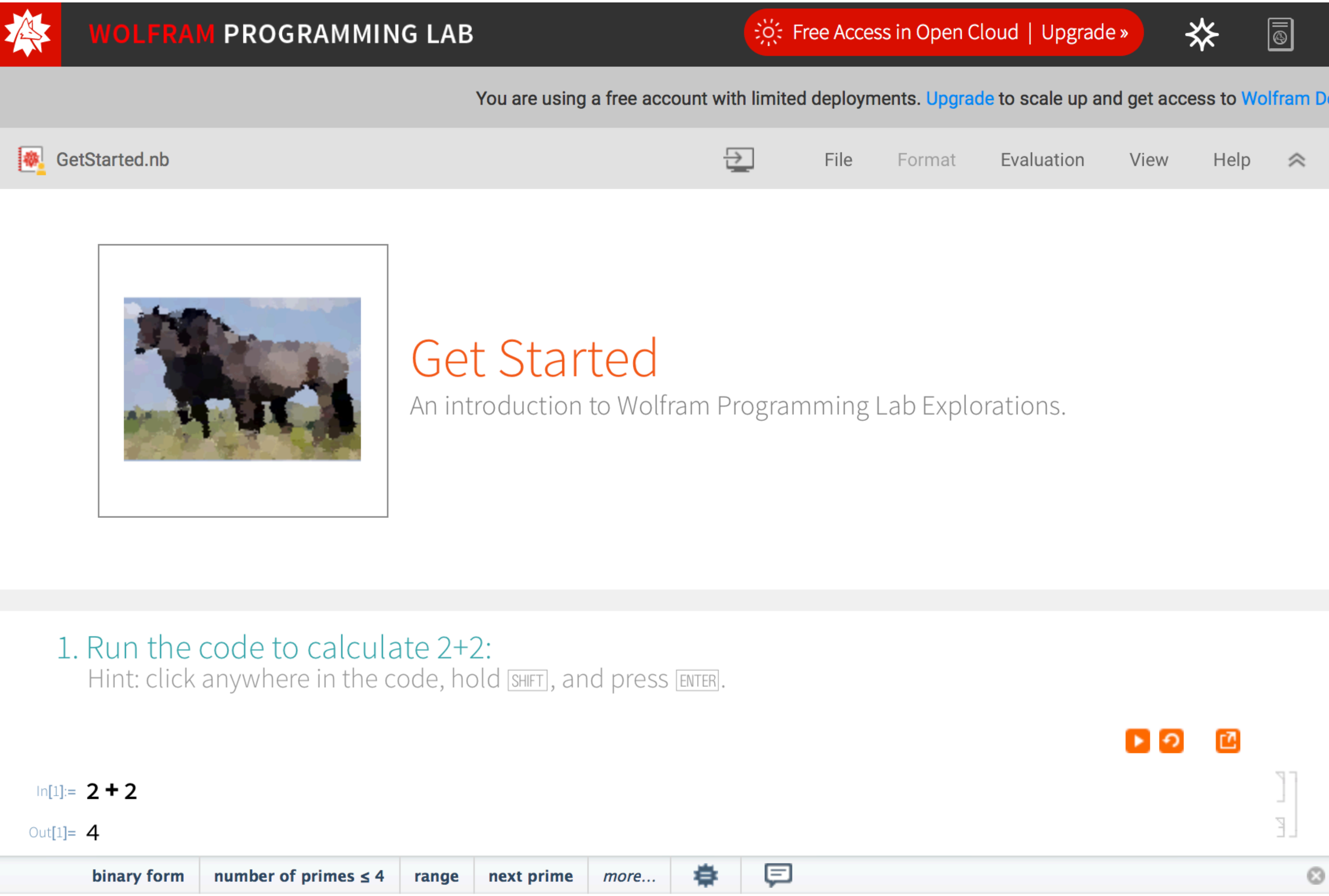
... and more symbolic algebra and analysis but it can also do much more



| | | |
|---|---|---|
| Core Language & Structure | Data Manipulation & Analysis | Visualization & Graphics |
| Machine Learning | Symbolic & Numeric Computation | Higher Mathematical Computation |
| Strings & Text | Graphs & Networks | Images |
| Geometry | Sound & Video | Knowledge Representation & Natural Language |
| Time-Related Computation | Geographic Data & Computation | Scientific and Medical Data & Computation |
| Engineering Data & Computation | Financial Data & Computation | Social, Cultural & Linguistic Data |
| Notebook Documents & Presentation | User Interface Construction | System Operation & Setup |
| External Interfaces & Connections | Cloud & Deployment | Recent Features |

- Mathematica has nearly 6,000 built-in functions covering all areas of technical computing

- Mathematica has the following features
  - Symbolics, with algebraic manipulation and mathematical computation
  - Numerics
  - Visualization
  - Number theory
  - Data analysis
  - Graph computation
  - Interactive computation
  - Image computation
  - Geometric computation
  - Import and export of many data formats



You are using a free account with limited deployments. Upgrade to scale up and get access to Wolfram D

## Get Started

An introduction to Wolfram Programming Lab Explorations.

### 1. Run the code to calculate 2+2:

Hint: click anywhere in the code, hold SHIFT, and press ENTER.

```
In[1]:= 2 + 2
Out[1]= 4
```

## The Euclidean algorithm

This is one of the oldest known algorithms. It finds the Greatest Common Divisor (GCD) *g* of two integer numbers.

$$g = \mathrm{GCD}(a,b) \quad \Rightarrow \quad a = ng;\ b = mg;\ n, m \text{ coprimes}$$

obviously, *g* divides the difference as well $a - b = (n - m)g$

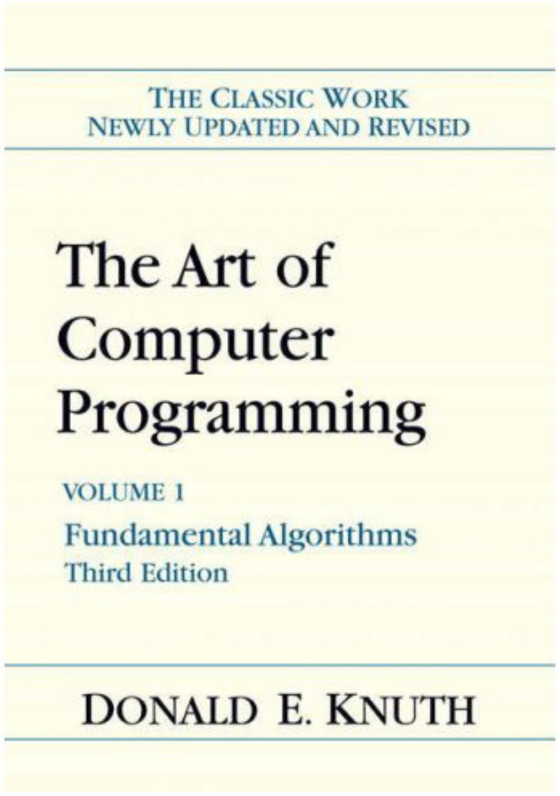
$$\boxed{(a,b) \rightarrow (\max(a-b, b-a), \min(a,b))}$$

we subtract from the largest of the two and we obtain a new pair.

We continue like this until both numbers are equal, so that *a* = *b* = *g*.

"[The Euclidean algorithm] is the granddaddy of all algorithms, because it is the oldest nontrivial algorithm that has survived to the present day.”

Donald Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 2nd edition (1981), p. 318.



```
In[•]:= GCD[27876, 32706]

Out[•]= 138
```

The Euclidean algorithm is built in Mathematica

```
In[•]:= a = 27876; b = 32706;

        While[a ≠ b,
           If[a > b, a -= b, b -= a];
         ];

        Print[a]

        138
```

However, we can write and execute a short program for the Euclidean algorithm

```
In[•]:= Simplify[27876 / 32706]

          202
Out[•]=  -----
          237

In[•]:= 27876 / 138

Out[•]= 202

In[•]:= 32706 / 138

Out[•]= 237
```

The Simplify instruction in Mathematica uses the Euclidean algorithm

```
In[•]:= a = 27876; b = 32706;

        While[a ≠ b,
          If[a > b, a -= b, b -= a];
         ];

        Print[a]

        138
```

Can you produce an improved version of this little program?

```
In[•]:= a = 27 876;  b = 32 706;

        While[a ≠ 0 && b ≠ 0,
          If[a > b, a -= b * Floor[a / b], b -= a * Floor[b / a]];
         ];

        Print[a];
        Print[b];
        Print[Max[a, b]];

        138

        0

        138
```

Is the second program really more efficient?

```
In[∘]:= a = 27 876; b = 32 706;

        k = 0;
        While[a ≠ b,
          If[a > b, a -= b, b -= a];
          k ++;
         ];
        Print["\n"];
        Print["GCD: ", a];
        Print["Number of iterations: ", k];
```

```
GCD: 138

Number of iterations: 14
```

```
In[∘]:= a = 27 876; b = 32 706;

        k = 0;
        While[a ≠ 0 && b ≠ 0,
          If[a > b, a -= b * Floor[a / b], b -= a * Floor[b / a]];
          k ++;
         ];

        Print["\n"];
        Print["GCD: ", Max[a, b]];
        Print["Number of iterations: ", k];
```

```
GCD: 138

Number of iterations: 7
```

YES! It is more efficient

# Euclidean division algorithm

Let

$$a \geq 0; \quad b > 0$$

then we can write the following algorithm

```
In[•]:= a = 45 900; b = 2787;

        q = 0; r = a;

        While[r ≥ b,
          q += 1;
          r = a - b * q;
         ];

        Print["q= ", q, "; r= ", r, "; a= ", q * b + r];

        q= 16; r= 1308; a= 45 900
```

The four integers that appear here have been given names: *a* is called the **dividend**, *b* is called the **divisor**, *q* is called the **quotient** and *r* is called the **remainder**.

The previous algorithm does not work as such for arbitrary integers.

**How can the algorithm be modified to take into account the possibilities**

$$a < 0;\ b > 0;$$

$$a > 0;\ b < 0;$$

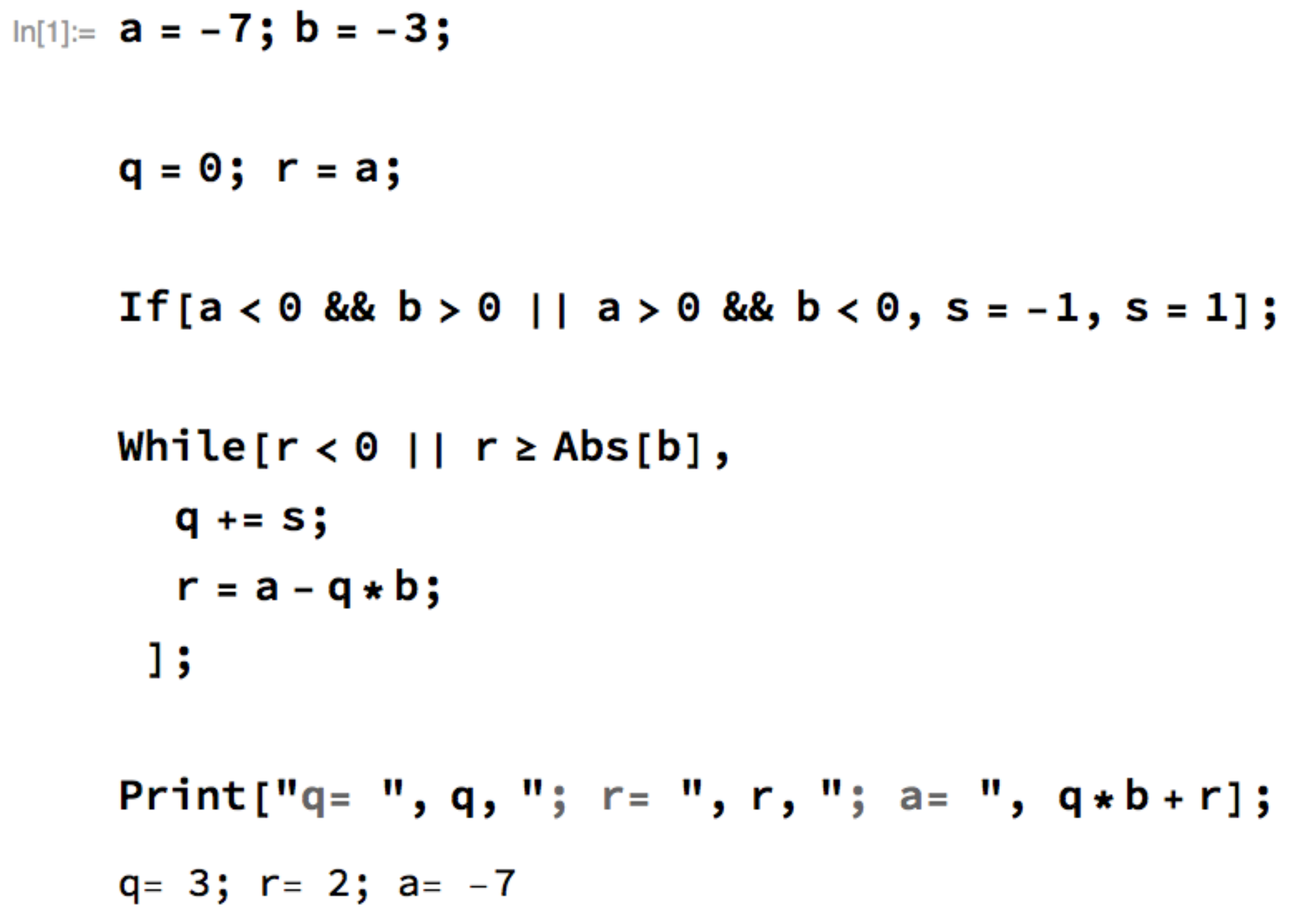
$$a < 0;\ b < 0;$$

with

$$0 \leq r < |b|$$

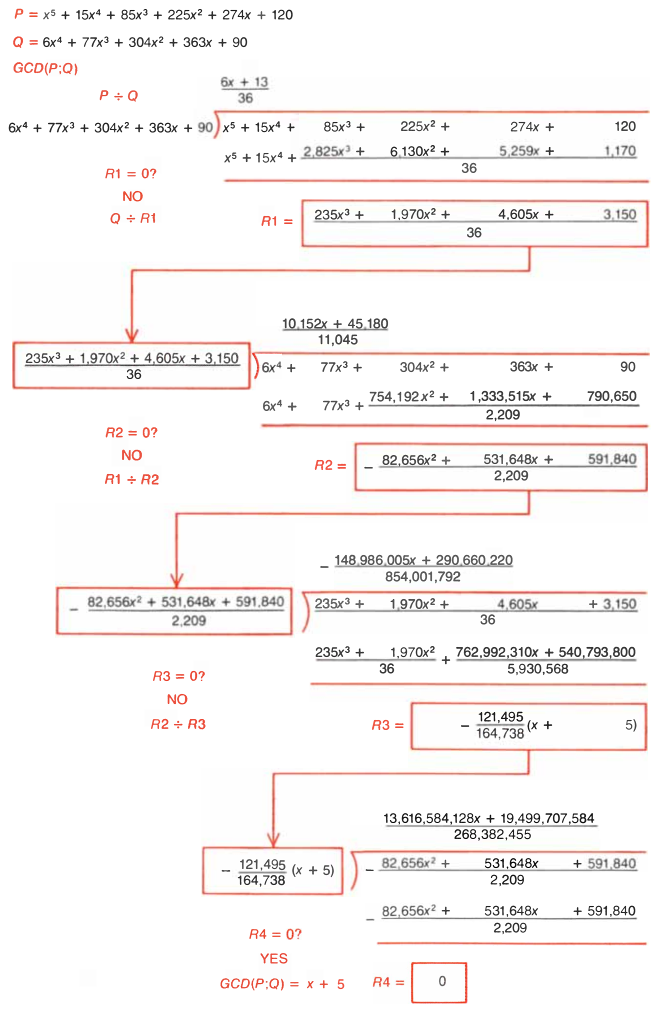
??????

Answer:



```
In[1]:= a = -7; b = -3;

        q = 0;  r = a;

        If[a < 0 && b > 0 || a > 0 && b < 0, s = -1, s = 1];

        While[r < 0 || r ≥ Abs[b],
          q += s;
          r = a - q * b;
         ];

        Print["q= ", q, "; r= ", r, "; a= ", q * b + r];

        q= 3; r= 2; a= -7
```

# Polynomial division (similar to Euclidean division)


P = x⁵ + 15x⁴ + 85x³ + 225x² + 274x + 120
Q = 6x⁴ + 77x³ + 304x² + 363x + 90
GCD(P;Q)
P ÷ Q
6x + 13 / 36
6x⁴ + 77x³ + 304x² + 363x + 90 ) x⁵ + 15x⁴ + 85x³ + 225x² + 274x + 120
x⁵ + 15x⁴ + (2,825x³ + 6,130x² + 5,259x + 1,170) / 36
R1 = 0?
NO
Q ÷ R1
R1 = (235x³ + 1,970x² + 4,605x + 3,150) / 36
(10,152x + 45,180) / 11,045
(235x³ + 1,970x² + 4,605x + 3,150) / 36 ) 6x⁴ + 77x³ + 304x² + 363x + 90
6x⁴ + 77x³ + (754,192x² + 1,333,515x + 790,650) / 2,209
R2 = 0?
NO
R1 ÷ R2
R2 = − (82,656x² + 531,648x + 591,840) / 2,209
− (148,986,005x + 290,660,220) / 854,001,792
− (82,656x² + 531,648x + 591,840) / 2,209 ) (235x³ + 1,970x² + 4,605x + 3,150) / 36
(235x³ + 1,970x²) / 36 + (762,992,310x + 540,793,800) / 5,930,568
R3 = 0?
NO
R2 ÷ R3
R3 = − 121,495 / 164,738 (x + 5)
(13,616,584,128x + 19,499,707,584) / 268,382,455
− 121,495 / 164,738 (x + 5) ) − (82,656x² + 531,648x + 591,840) / 2,209
− (82,656x² + 531,648x + 591,840) / 2,209
R4 = 0?
YES
GCD(P;Q) = x + 5
R4 = 0

**EUCLIDEAN ALGORITHM** applied to polynomials determines their greatest common divisor by repeated divisions, in the same way as it determines the greatest common divisor of two integers. The algorithm can give rise to huge intermediate numerical results that cannot be rounded off. For this reason the arithmetic operations in a computer-algebra system must have unlimited precision. Ordinarily a number stored in a computer is allotted a fixed amount of space in the computer memory, but in a computer-algebra system no fixed allocation can be made. Mathematicians who study the properties of large numbers have been attracted to this feature of computer-algebra systems. Colored arrows and the statements printed in color correspond to the steps of the algorithm diagrammed in the illustration on pages 144 and 145.

```
a = ExpandAll[(10 * x ^ 3 + 12 * x ^ 2 + 23 * x) * (23 * x + 1) + 3]; (* test polynomials *)
b = 23 * x + 1;

Print["a= ", a]; (* printout initial values *)
Print["b= ", b];
Print["\n"];

q = 0; r = a; (* initialize algorithm *)

d = Exponent[b, x];
c = Coefficient[b, x ^ d];

(* begin loop *)
k = 1; (* step counter *)
While[Exponent[r, x] ≥ d, (* while deg(r)≥deg(b) *)
  lc = Coefficient[r, x ^ Exponent[r, x] ]; (* find the leading coefficient *)
  s = (lc / c) * x ^ (Exponent[r, x] - d); (* monomial in q that corresponds to the leading term in r *)
  q = q + s; (* update q *)
  r = ExpandAll[r - s * b]; (* update r *)
  Print["Step ", k, "\nq= ", q, "; s= ", s, "; r= ", r, "\na = q*b+r = ", ExpandAll[q * b + r]];
  k ++; (* update counter *)
 ];
(* final printout *)
Print["\nq= ", q, "; r= ", r, "\na = q*b+r = ", ExpandAll[q * b + r]]
```

a= $3+23x+541x^2+286x^3+230x^4$

b= $1+23x$

Step 1

q= $10x^3$; s= $10x^3$; r= $3+23x+541x^2+276x^3$

a = q*b+r = $3+23x+541x^2+286x^3+230x^4$

Step 2

q= $12x^2+10x^3$; s= $12x^2$; r= $3+23x+529x^2$

a = q*b+r = $3+23x+541x^2+286x^3+230x^4$

Step 3

q= $23x+12x^2+10x^3$; s= $23x$; r= $3$

a = q*b+r = $3+23x+541x^2+286x^3+230x^4$

q= $23x+12x^2+10x^3$; r= $3$

a = q*b+r = $3+23x+541x^2+286x^3+230x^4$

## Homework

Division requires a long loop (how long?)

Find a numerical algorithm to divide floats using only the +, -, x operators