Mathomatic User Guide

George Gesslein II

Mathomatic User Guide

Table of Contents

1. Introduction
2. History
3. Developer Information
4. Startup
5. Equations and Expressions
    a. Constants
    b. Variables
    c. Operators
        1. Order of Operations example
    d. Complex Numbers
6. Commands
7. Documentation License

Mathomatic Command Reference

Introduction

Mathomatic is an easy to use and colorful algebra calculator that can symbolically:

- combine and solve equations containing many variables,
- completely simplify and compare mathematical expressions and equations,
- do simple calculus transformations and series,
- perform quick real number, complex number, and polynomial arithmetic,
- generate efficient C, Java, or Python language code from simplified equations,
- plot expressions with gnuplot in two or three dimensions,
- … and much more.

The name "Mathomatic" is a portmanteau of "math" and "automatic". It is a unique computer algebra system (CAS), in that all constants are one or more floating point values. All numeric arithmetic is IEEE standard floating point arithmetic, which most computers do very quickly. Mathomatic is written entirely in C, which is like a CAS written in assembly language, running as fast as the computer allows without any high-level language overhead.

Mathomatic is exceptionally good at solving, differentiating, simplifying, and calculating elementary algebra. It is designed with a simple command-line interface (CLI) that tries to be helpful. All input and output is line at a time ASCII text. By default, input is standard input and output is standard output.

History

Mathomatic development started in the year 1986, originally using the Microsoft C compiler for MS-DOS. Versions 1 and 2 were published by Dynacomp of Rochester, New York in 1987 and 1988 as a scientific software product for DOS. Afterwards it was released as shareware and emailware with a 2D equation graphing program written in Microsoft C for DOS. At the turn of the century, Mathomatic was ported to the GNU C Compiler (GCC) under Linux and became free and open source software by publishing under the GNU Lesser General Public License. The graphing program was discontinued; 2D/3D graphing of equations is now accomplished with gnuplot.

Mathomatic is currently developed and maintained on a Ubuntu Linux x86-64-bit computer.

Developer Information

Building Mathomatic from source requires a C compiler with the standard C libraries. If compiled with the GCC C compiler for a Unix-like operating system, no changes need to be made to the source code. See the file *README.txt* for compilation instructions.

Mathomatic can easily be ported to any computer with at least 1 megabyte of free RAM. In the standard distribution, found on the Mathomatic home page, the maximum memory usage defaults to 400 megabytes (the version command tells this). Maximum memory usage is not reached unless all equation spaces are filled. The default maximum memory usage should be less than the amount of free RAM, and is adjusted by changing the `DEFAULT_N_TOKENS` define in the C include file *am.h* and recompiling. Memory usage can also be adjusted at startup with the **-m** option.

The Mathomatic source code can also be compiled as a symbolic math library that is callable from any C compatible program. See the file *lib/README.txt* for more developer information and how to include Mathomatic in your program.

Very little disk space (a few megabytes) is required to compile, install, and run the Mathomatic application. A readline library must be installed to compile-in and use readline (package called "libreadline-dev"), which allows editing and history recall of all Mathomatic line input by pressing the cursor keys.

Startup

SYNOPSIS

> **mathomatic** [options] [input_files or input]
> **rmath** [input_files]

To start the compiled, interactive Mathomatic application, run a terminal emulator which opens a shell window, and type "mathomatic" at the shell prompt (without double quotes). If m4 (macro) Mathomatic was installed, you may type "rmath" instead, to use Mathomatic with input of functions like `sin(x)` and `sqrt(x)` allowed and automatically expanded to equivalent algebraic expressions. Logarithm function input is currently not available, because the logarithm function has not yet been implemented in the Mathomatic symbolic math engine.

If you are wondering what to try first in Mathomatic, type "help examples" at the Mathomatic prompt.

Color mode is toggled by the **-c** option on the shell command line, like this:

```
$ mathomatic -c
```

ANSI color mode outputs ANSI terminal escape sequences to make each level of parentheses a different color, improving readability. If ANSI color mode is on, an ANSI compatible terminal emulator is required. If the colors are hard to see, use the **-b** option (bold colors) instead, to enable color mode and increase the brightness.

The other options are described in the Unix/Linux man page for Mathomatic. After any options, text files may be specified on the shell command line that will be automatically read in with the read command, unless the **-e** option is specified, in which case mathematical expressions and Mathomatic commands are expected, separated by unquoted space characters.

It is recommended that the name *mathomatic* be shortened to *am* and *e* for quicker and easier access from the shell command line. This can be done in the Bash shell by adding the following two lines to your *~/.bashrc* file:

```
alias am=mathomatic
alias e="mathomatic -e --"
```

Then just typing "am" at the shell prompt will run Mathomatic as an interactive application. Typing "e" followed by a quoted mathematical expression at the shell prompt will quickly and silently bring up Mathomatic and calculate and display the result. "am" stands for "algebraic manipulator", and "e" stands for "evaluate".

Equations and Expressions

Mathematical equations and expressions are entered into **equation spaces** by typing, pasting, or reading them in. The maximum number and size of available equation spaces is displayed every time Mathomatic starts up. When an expression grows larger than half the equation space size, processing stops and the "Expression too large" message is displayed, returning you to the main prompt.

Each equation space is successively numbered with an **equation number** (starting at 1). The main prompt "`1-> `" contains the equation number of the current equation space. The current equation can be changed by typing a valid equation number at the main prompt, or by entering another equation or expression, which becomes the current equation.

To enter an equation into the first available equation space and make it the current equation, simply type it in at the main prompt. Each equation space consists of two equation sides, called the Left-Hand Side (LHS) and the Right-Hand Side (RHS), separated by an equals sign (**=**). Each equation side consists of a mathematical expression, which is a mix of constants, variables, and operators, mostly in standard infix notation.

Parentheses are used to override operator precedence and group things together. Valid parentheses characters are **()** and **{}**.

Note that the equals sign does not make an assignment to any variables, it only signifies equality (sameness) between the results of evaluating the LHS and RHS. Shown here is a valid equation with its parts labeled:

```
       equation
-----------------------
| variables   constant|
|--------------     | |
||      |      |    | |
 a  =  b  -  (c  +  2)
| |   |   |      |    |
| |   |   --------    |
| |   |   operators   |
---   -----------------
LHS          RHS
```

In the above equation, the variable **a** is called the **dependent** variable because its value depends on the **independent** variables **b** and **c**. In Mathomatic, any variable can be made the dependent variable by simply typing the variable name in at the prompt. This will solve the current equation for that variable and, if successful, make that variable the LHS.

Here is the above equation entered into Mathomatic and solved for **b**, then calculated for the values **a=1** and **c=1**:

```
1—> a=b-(c+2)

#1: a = b - c - 2

1—> b

#1: b = 2 + c + a

1—> calculate
Enter a: 1
Enter c: 1

 b = 4

1—>
```

The "`#1:`" listed in front of each displayed equation always indicates the equation space number it is stored in.

Mathomatic automatically does both symbolic and numeric mathematics computations during any manipulations. This means that it can handle algebraic formulas, as well as numbers. What follows is a simple example of the result of both types of computations working together during equation simplification and solving:

```
1—> 3*(x-1) + 1 = 2x + 1

#1: (3·(x - 1)) + 1 = (2·x) + 1

1—> simplify
```

```
#1: (3·x) - 2 = (2·x) + 1

1-> solve verify x

#1: x = 3

Solution verified.
1->
```

The "solve verify" command, used above, solves the current equation and then verifies the result by plugging the result into the original equation and simplifying. If an identity results (the LHS is identical to the RHS), a "Solution verified" message is displayed, otherwise "Solution might be incorrect" is displayed.

Non-equations, that is any mathematical expression without an equals sign, may be entered into equation spaces too. However, if the expression entered at the main prompt contains no variables, it will only be calculated and displayed, unless the autocalc or auto option is turned off. Non-equations cannot be solved.

Constants

In Mathomatic, numeric arithmetic is double precision floating point with about 14 decimal digits accuracy. Many results will be exact, because symbolic math is an exact math, and because multiple floating point numbers can be combined for a single mathematical value; for example: **2^(1/3)**, which is the cube root of 2 exactly.

Constants are approximated real numbers stored internally as IEEE 754 standard 64-bit (8 bytes) double precision floating point values. They may be entered as decimal (base 10) numbers in normal notation or in scientific notation (also called exponential notation). Constants may also be entered in hexadecimal (base 16) by prepending them with "0x".

Constants are displayed in decimal (base 10, rounded to 14 digits) using either normal or scientific notation, whichever is shortest. Results are usually accurate from 12 to 14 digits, due to accumulated round-off error, because all constants are stored internally as double precision (rounded to 15 decimal digits) floats. And the amount of round-off error is not tracked, making Mathomatic unsuitable for applications requiring high precision, like astronomical calculations.

Excepting constants with a name (like "inf" for the infinity constant), constants always start with a decimal digit (0..9) or a period.

Examples of equivalent constants follow:

Normal Notation (base 10)	Scientific Notation (base 10)	Hexadecimal Notation (base 16)
10	1e1 (1.0 times 10^{1})	0xa
.125	1.25e-1 (1.25 times 10^{-1})	0x.2
255	2.55e2 (2.55 times 10^{2})	0xff

The exact syntax to enter constants as above may be found by looking up the C library function `strtod(3)`. In the Unix shell, "`man strtod`" will do that.

- The largest valid constant is **±1.797693e+308** (slightly less than 2^1024).
- The smallest valid constant is **±2.225074e-308** or 0.

The infinity constant is entered by typing "inf". Positive and negative infinity are distinct and understood, however division by zero produces one infinity value, not the two-valued ±infinity which would be more correct. Also, floating point overflow produces either positive or negative infinity.

```
1-> 1/0
Warning: Division by zero.

 answer = inf

1-> 0/0
Warning: Division by zero.

 answer = nan

1->
```

nan or *NaN* stands for **Not a Number** and it means an invalid or indeterminate floating point arithmetic result. *NaN* cannot be directly entered into Mathomatic. The appearance of the constant *NaN* in an expression means the expression is unusable.

Fractions (such as **100/101**) are preserved if the numerator and denominator are not large. Fractions are always presented in fully reduced form; for example, **6/9** is converted to the irreducible fraction **2/3**. Constants which are exactly equal to a fraction are converted and displayed as fully reduced fractions; for example, **0.5** converts to **1/2**. Mathomatic internally converts a fraction to a single floating point value, then may convert it back to a fraction for display after all floating point arithmetic has been done, if the result is equal to a fraction.

Irrational numbers, such as the square root of two (**2^(1/2)**) and **pi**, are preserved and simplified for exactness, unless explicitly approximated.

Denominators of fractions are usually rationalized in Mathomatic; for example, **1/(2^(1/2))** becomes the equivalent **(2^(1/2))/2** upon simplification. This can be turned off with the command "set no rationalize_denominators".

Variables

Variables are what Mathomatic is all about. That is where the term "symbolic" comes from, because variables are symbolic in nature. They are symbols that can represent known or unknown values, or any expression. Variables need not be defined in Mathomatic, just entering the variable name is enough.

Variable names consist of any combination of letters (a..z), digits (0..9), and underscores (_). They never start with a digit. By using the "set special_variable_characters" command, you can add to the allowed variable name characters. By default, letters in variable names are case sensitive, meaning the alphabetic case of each letter in the variable name is important. For example, variables named "A1" and "a1" represent two different variables, unless "set no case_sensitive" is entered beforehand.

The following variables are predefined and are not normal variables:

e or **e#** - the universal constant e (2.718281828…)
pi or **pi#** - the universal constant pi (3.1415926…)
i or **i#** - the imaginary unit (square root of -1)
sign, **sign1**, **sign2**, … - may only be +1 or -1
integer, **integer1**, … - may be any integer value

The above can be used anywhere variables are required.

To automatically enter multiplication by a unique, two-valued "sign" variable, precede any expression with "+/–".

Operators

Mathomatic implements the standard rules of algebra for addition (**+**), subtraction (**–**), multiplication (*****), division (**/**), modulo division (**%**), and all forms of exponentiation (**^** or ******). An example of a rule of algebra is **2*x + 3*x** being simplified to **5*x**.

All available operators are at least numerically capable and have precedence decreasing as indicated:

```
! factorial (gamma function)
** or ^ power (exponentiation)
* multiply      / divide        % modulus       // integral divide
+ add           - subtract or negate
= equate (lowest precedence)
```

Higher precedence operators are grouped (or evaluated) first, then multiple operators of the same precedence level are grouped left to right. This is called the "order of operations".

The default operator is multiply (*****). If an expression (operand) is entered when an operator is expected, a multiply operator is automatically inserted. For example, entering **2x**, **2(x)**, and **(2)(x)** all result in the expression **2*x**.

The modulus operator **a % b** (spoken as "a modulo b") gives the remainder of the division **a / b**. The sign of the result depends on the "set modulus_mode" option. Using "integer" variables allows further simplification here. An integer variable is specified by using a variable name that starts with "integer", like "integer1", "integer_x", etc.

The integral divide operator **a // b** divides **a** by **b** and then truncates by zeroing the fractional part to make the result an integer. For example, **8 // 3** results in 2, which is useful when doing integer arithmetic. This operator currently implements no rules of algebra, and will not evaluate if an operand is a complex number.

Factorials **x!** use the gamma function **gamma(x+1)**, so that the factorial operator works with any real number, not just the positive integers. The factorial operator currently implements no rules of algebra, and will not evaluate for complex numbers or if an overflow happens.

Absolute value notation is allowed, **|x|** is converted to **(x^2)^.5**. This is not the same as standard absolute value where the real and imaginary parts of complex numbers are separated and then squared, but it works the same when given real number values with no imaginary units. The absolute value operation **|x|** results in a positive value for any **x** value; that is, if -1 is a factor, it is removed.

Order of Operations example

The following example shows why operator precedence is important. Given the numerical expression:

```
64/(-2)^4 + 6*(3+1)
```

Mathomatic will parenthesize the highest precedence operators first (power, then times and divide; from left to right, but that doesn't matter in this case). Addition and subtraction are the lowest precedence, so no need to parenthesize them. The result will be:

```
(64/((-2)^4)) + (6*(3+1))
```

This is evaluated by combining constants from left to right on the same level of parentheses, deepest levels first. So the calculations are performed in the following order:

```
(64/16) + (6*4)   Combine deepest level parentheses first.
4 + 24            Divided 64 by 16 and multiplied 6 by 4.
28                Added 24 to 4.
```

If the calculations were performed in a different order, the result would be different.

Complex Numbers

Mathomatic automatically performs complex number addition, subtraction, multiplication, division, and exponentiation. It can also approximate roots of real and complex numbers, giving a single result; when multiple results are possible, the first real number result is chosen.

Complex numbers are in the form:

```
a + b*i
```

where **a** is the real part (a real number) and **b** is the imaginary part (an imaginary number). **i** represents the square root of -1 ("`(-1)^.5`" in Mathomatic notation).

The imaginary unit **i** may appear anywhere within an expression, as many times as you want, Mathomatic will handle and simplify it properly.

As an example of imaginary numbers being produced, **(-2)^.5** will be converted to **(2^.5)*i**.

Roots of complex numbers, such as **i^.5** and **.5^i**, will be approximated, and only a single root will be produced, even though there may be many roots (see the roots command). That single root is called the "principal value", which may be unexpected and will often be inexact.

Conjugation of all complex numbers in the current equation is accomplished by typing the following command:

```
replace i with -i
```

Commands

Mathomatic has about 43 simple English commands that may be typed at the main prompt. Please consult the Mathomatic Command Reference, for detailed information on all commands.

Up to the documentation index www.mathomatic.org

Mathomatic Command Reference

Topics

1. Introduction
2. Selecting Expressions
3. Solving Equations
4. Documentation License

Commands

Approximate	Divide	Help	Pause	Replace	Taylor
Calculate	Echo	Imaginary	Plot	Roots	Unfactor
Clear	Edit	Integrate	Product	Save	Variables
Code	Eliminate	Laplace	Push	Set	Version
Compare	Extrema	Limit	Quit	Simplify	
Copy	Factor	List	Read	Solve	
Derivative	For	NIntegrate	Real	Sum	
Display	Fraction	Optimize	Repeat	Tally	

Mathomatic User Guide

Introduction

LHS is shorthand for the Left-Hand Side of an equation. Similarly, **RHS** used here always means the Right-Hand Side.

In this document, text enclosed by straight brackets **[like this]** means it is optional and may be omitted. The word "expression" (without double quotes) always means a mathematical expression or formula.

At the Mathomatic main prompt, you may enter:

- a numerical expression, which is instantly evaluated and displayed with the calculate command (autocalc),
- an algebraic expression or equation, which is stored and made the current equation,
- a variable to solve the current equation for (autosolve),
- an equation number to select as the current equation (autoselect),
- a Mathomatic command (listed in this document),
- a question mark (**?**) for quick, short help (same as the help command),
- a semicolon (**;**) followed by a comment (everything on a line after a semicolon is ignored),
- or an exclamation point (**!**) followed by a shell or system command. "**!**" by itself invokes the default shell.

Mathomatic has about 43 commands that may be typed at the main prompt. Most commands operate on the current expression or equation by default. Commands are simple English words and are described below, in alphabetical order. If the command is longer than 4 letters, you only need to type in the first 4 letters for Mathomatic to recognize the command. Option words and arguments for commands follow the command name and are separated by spaces. Commands are not executed until you press the Enter key, and any missing command line arguments that don't have a default are prompted for. For example, the Mathomatic command

```
help calculate
```

gives short help and usage information for the calculate command. "help" is the command, "calculate" is the argument (which is also a command, in this case).

Many commands have an optional **equation number range** argument, which specifies the equation spaces that the command is to operate on. An **equation number range** may be a single equation number, or a range of equation numbers separated by a dash (like "2-7", which means every equation space between and including equations 2 and 7), or the word "all", which specifies all equation spaces. If omitted, the current expression or equation is assumed. If pluralized as "**equation-number-ranges**" in the command syntax, that means multiple equation numbers and ranges may be specified for that command, separated by spaces.

A greater-than character (**>**) may be appended to the end of any command line, followed by a file name. This will redirect the output of the command to that file. If the file already exists, it will be overwritten without asking. Note that any debugging output will be redirected, too. Two greater-than characters (**>>**) next to each other will **append** command output to a file, like the Unix shell does. For example, the Mathomatic command

```
list export all >filename
```

will output all stored expressions and equations to a file in exportable, single-line per equation format, so they can be read in by a different math program. "list" is the command, "export" is the option word, and "all" is the **equation number range**.

Command option words, such as "export" in the above list command line, always come immediately after the command name and before anything else on the command line. These words direct the command to do a different, but related, task.

If Mathomatic becomes unresponsive (a rare occurrence), pressing Control-C once will usually safely abort the current operation and return you to the main prompt. If not, pressing Control-C three times in a row will exit Mathomatic, with a warning displayed the second time.

Selecting Expressions

Syntax: **#["+" or "−"]equation-number**

To change the current equation at the main prompt, type a pound sign (**#**) followed by the equation space number you wish to select. The **equation number** may be preceded by plus (**+**) or minus (**−**), to select an equation relative to the current equation. This syntax also works when prompted for an expression, the RHS or the expression at that equation number is substituted.

New feature: Selecting an equation space to make it the current equation is now conveniently done by typing only the **equation number** at the main prompt. This is called the "autoselect" option.

Solving Equations

Syntax: **variable or "0"**

Mathomatic can solve symbolic equations for any **variable** or for **zero**. Solving is accomplished internally by applying identical mathematical operations to both sides of the equation and simplifying, or by plugging the general coefficients of the solve variable into the quadratic formula. The Mathomatic solve algorithm is the best possible for general algebra, however the result is not verified by plugging the solutions into the original equation unless the "solve verify" command is used for solving.

To automatically solve the current equation for a variable, type the **variable** name at the main prompt. Mathomatic will proceed to manipulate the current equation until all of the solutions for the specified **variable** are determined. If successful, the current equation is replaced with the solutions and then displayed.

Automatic cubic (third degree), quintic (fifth degree), and higher degree polynomial equation solving is not supported. Some cubic and quartic polynomial equations can be manually solved with the general equations in files "*tests/cubic.in*" and "*tests/quartic.in*". Quartic (fourth degree) polynomial equations can be automatically solved if they are biquadratic; that is, containing only degree four, degree two, and degree zero terms of the solve variable. Biquadratic polynomial equations of any degree can be generally solved by Mathomatic because they can be plugged into the quadratic formula.

Note that running the simplify command is a good idea after solving. The solve routine only unfactors the equation as needed to solve it and the result is not completely simplified.

To solve for zero, type in "0" at the main prompt. Zero solving is a special solve that will always be successful and will transform most divide operators into mathematically equivalent multiplications and subtractions, so that the result will more likely be a valid polynomial equation:

```
1—> a=b+1/b ; this is actually a quadratic equation

            1
#1: a = b + -
            b

1—> 0 ; solve for zero

#1: 0 = (b*(b - a)) + 1

1—> unfactor ; expand, showing that this is a quadratic polynomial equation in "b"

#1: 0 = (b^2) - (b*a) + 1

1—> b ; solve for "b"
Equation was quadratic.

                       1
        (((((a^2) - 4)^-)*sign1) + a)
```

```
                       2
#1: b = -----------------------------
                      2

1-> a ; solve for "a", to check the answer
Raising both sides to the power of 2 and unfactoring...

        ((b^2) + 1)
#1: a = -----------
             b

1-> simplify

            1
#1: a = b + -
            b

1->
```

You can prefix the solve **variable** name with "=" to solve and swap equation sides, putting the solve variable on the right-hand side. Typing "=" by itself will only swap sides of the current equation and display.

To see all of the steps performed during a solve operation, type "set debug 2" before solving.

Approximate command

Syntax: **approximate [equation-number-ranges]**

This command operates on the current or specified equation spaces. It substitutes the special universal constants **pi** and **e** with their respective floating point values and approximates all constants, roots, and surds. This allows them to combine with other constants and may help with simplification and comparisons.

Calculate command

Syntax: **calculate ["factor"] [feedback-variable number-of-iterations]**

This command is the formula calculator that prompts for the value of each normal variable in the current expression or the RHS of the current equation. It then temporarily substitutes any entered values, and simplifies, approximates, and substitutes all "sign" variables with all possible combinations of values (+1 and -1), displaying each solution as it does so. If all variables are supplied with constant values, then each solution will be a constant, otherwise the result will contain the variables you didn't enter values for. Nothing is modified by this command.

This command is used to temporarily plug in values and approximate expressions and expand "sign" variables. When an expression with only numbers is entered at the main prompt, this calculate command is automatically invoked on it, displaying the calculated result without storing.

If there are any simple numerical fractions in the result, that is displayed alongside the result.

To only simplify and expand "sign" variables in stored expressions without approximating, use the "simplify sign" command instead.

If a **feedback variable** and **number of iterations** are specified on the calculate command line, you will be prompted for the initial value of the **feedback variable**, and the calculation will be iterated, with the simplified result repeatedly plugged back into the **feedback variable**. This will be done until convergence (the output equals the input) or when the specified **number of iterations** have been performed (if non-zero), whichever comes first. To see all of the intermediate values, type "set debug 1" before this.

"calculate **factor**" factors all integers and variables before display.

Examples of using the calculate command:

```
1—> y=x^2+x

#1: y = (x^2) + x

1—> x ; solve for x
Equation is a degree 2 polynomial in x.
Equation was quadratic.

                                1
        -1·(1 + (((1 + (4·y))^—)·sign))
                                2
#1: x = ———————————————————————————————
                       2

1—> calculate
Enter y: 0
There are 2 solutions.

Solution number 1 with sign = 1:
 x = -1

Solution number 2 with sign = -1:
 x = 0

1—>
```

An example of iteration:

```
1—> x_new=(x_old+(y/x_old))/2 ; iterative formula for calculating the square root of y

                       y
            (x_old + -----)
                     x_old
#2: x_new = ---------------
                   2

2—> calculate x_old 1000 ; iterate up to 1000 times to calculate the square root of 100
Enter y: 100
Enter initial x_old: 1
Convergence reached after 9 iterations.
```

```
 x_new = 10

2->
```

Clear command

Syntax: **clear [equation-number-ranges]**

This command clears the specified equation spaces so that they are empty and can be reused. They are deleted from RAM only.

"clear all" quickly clears all equation spaces and restarts Mathomatic, without losing your settings.

Code command

Syntax: **code ["c" or "java" or "python" or "integer"] [equation-number-ranges]**

This command outputs the current or specified equations as floating point or integer assignment statements in C, Java, or Python programming language code. The default is C double precision floating point code. The output from this command should compile correctly and emulate the equation from Mathomatic, if no warnings are given.

With "code integer", integer arithmetic is assumed, otherwise double precision floating point arithmetic is assumed. "code integer" is more generic and should work with any language.

To represent factorials, the user supplied function `fact()` is called, since there is no equivalent function or operator in these languages. `fact()` functions for several languages are supplied in the directory *examples* in the Mathomatic source distribution.

For the most efficient code, use the simplify and optimize commands on your equations before running this code command.

The C and Java languages require that all variables be defined before use. The variables command is provided for this. The output of the variables command should be prepended to the output of the code command before compiling.

Compare command

Syntax: **compare ["symbolic"] ["approximate"] equation-number ["with" equation-number]**

This command compares two equation spaces to see if they are mathematically the same (equal). If only one **equation number** is supplied, the comparison is between the current equation and the specified equation. The comparison will be faster and more accurate if both equations are previously solved for the same variable.

The simplify command is automatically used on both expressions if needed. If this compare command says the equations or expressions are identical, then they are definitely identical. If this command says the equations or expressions differ, then they might be identical if they are too hard for Mathomatic to simplify completely.

The "symbolic" option uses the "simplify symbolic" option when simplifying. This option sometimes simplifies more, but is not 100% mathematically correct.

The "approximate" option runs the approximate command on both expressions or equations. This makes the compare command more likely to succeed.

Copy command

Syntax: **copy [equation-number-ranges]**

This command simply duplicates the current or specified equation spaces. The new, duplicated expressions are stored in the next available equation spaces and displayed, along with their new equation numbers.

Derivative command

Syntax: **derivative ["nosimplify"] [variable or "all"] [order]**
Alternate command name: **differentiate**

This command computes the exact symbolic derivative of a function with respect to the specified **variable**, using the current expression or RHS of the current equation as the function. It does this by recursively applying the proper rule of differentiation for each operator encountered. The result is fully simplified with the simplify command, unless the "nosimplify" option is specified. If successful, the derivative is placed in the next available equation space, displayed, and becomes the current equation. The original equation is not modified.

Specifying "all" computes the derivative of the current expression with respect to all normal variables. It is equivalent to adding together the derivatives with respect to each variable.

Specifying the **order** allows you to repeatedly differentiate and simplify. The default is to differentiate once (**order**=1).

If differentiation fails, it is probably because symbolic logarithms are required. Symbolic logarithms are not implemented in Mathomatic, yet. Also, the factorial, modulus, and integral divide operators cannot be differentiated if they contain the specified **variable**. Because this command handles almost everything, a numerical differentiation command is not needed.

Some examples:

```
1-> x^3+x^2+x+1

#1: (x^3) + (x^2) + x + 1

1-> derivative ; no need to specify the variable if there is only one
Differentiating with respect to (x) and simplifying...

#2: (3*(x^2)) + (2*x) + 1

2-> a*x^n

#3: a*(x^n) ; show a general rule of differentiation

3-> derivative x
Differentiating with respect to (x) and simplifying...

#4: a*n*(x^(n - 1))

4-> integrate x ; undo the differentiation

#5: a*(x^n)

5->
```

Display command

Syntax: **display ["factor"] ["mixed"] [equation-number-ranges]**

This command displays stored expressions in nice looking multi-line 2D (two dimensional) fraction format, where division is displayed as a numerator over a fractional line (made up of dashes) over a denominator. If the width (number of columns) required for this 2D display exceeds the screen width, the expression is displayed instead in single-line format by the list command. The screen width is set automatically on startup, or by the "set columns" option.

Non-integer constants are converted to reduced fractions, if they are exactly equal to a simple fraction and it would improve readability.

The "factor" option causes all integers, less than or equal to 15 decimal digits long, to be factored into their prime factors before display, including the numerator and denominator of fractions. To always factor integers like this before display, use the "set factor_integers" option.

The "mixed" option displays mixed fractions when possible. A mixed fraction is an expression like **(2 + (1/4))**, rather than the simple fraction **9/4**. To always display mixed fractions, use the "set fractions mixed" option.

Divide command

Syntax: **divide [variable]**

This command is for doing and experimenting with polynomial and numerical division and Greatest Common Divisors (GCD). It simply prompts for two expressions and divides them, displaying the result and the GCD. Mathomatic has general symbolic polynomial division and GCD routines used by the simplify command which this divide command calls without any other processing if two polynomials are entered.

This command prompts for the dividend (the main polynomial) and then the divisor (what you want to divide the main polynomial by). The polynomial quotient, remainder, and GCD are displayed. The power of the highest power term in the dividend must be greater than or equal to the power of the highest power term in the divisor, otherwise the polynomial division will fail (as it should). In other words, the degree of the divisor polynomial must be less than or equal to the degree of the dividend polynomial. A **variable** may be specified on the command line as the base variable of the two polynomials, but is usually not necessary because a base variable is automatically selected.

If two numbers are entered instead of polynomials, the result of the numerical division, the GCD, and the Least Common Multiple (LCM) are displayed. The LCM of two numbers is the smallest positive number that can be evenly divided by both numbers separately, without remainder. The LCM is the same as the Lowest Common Denominator (LCD) of two fractions and is the two numbers multiplied together, divided by the GCD.

The Greatest Common Divisor of **a** and **b** is defined as the greatest positive number or polynomial that evenly divides both **a** and **b** without remainder. In Mathomatic, the GCD is not necessarily integer, unless both **a** and **b** are integers. The Euclidean GCD algorithm is used by Mathomatic to compute the GCD for numbers and polynomials.

The GCD is the best way to reduce any fraction to its simplest form. Just divide the numerator and denominator by their GCD, and replace them with the quotients (there will be no remainder), and the fraction is completely reduced. The GCD is also used when factoring polynomials and for simplifying.

The Euclidean GCD algorithm of successive divides is the best way to compute the GCD for numbers and polynomials. Multivariate polynomial GCD computation usually requires recursion of the GCD algorithm or other methods. Currently the polynomial GCD routine in Mathomatic is not recursive, making it univariate and simpler and faster. Because it is univariate, Mathomatic may be unable to find the GCD of polynomials with many variables.

The polynomial division algorithm in Mathomatic is generalized and able to handle any number of variables (multivariate), and division is always done with one selected base variable to be proper polynomial division. Being generalized, the coefficients of the polynomials may be any mathematical expression.

An example of polynomial division:

```
1—> divide
Enter dividend: (x^4) - (7*(x^3)) + (18*(x^2)) - (22*x) + 12
Enter divisor: (x^2) - (2*x) + 2

Polynomial division successful using variable (x).
The quotient is:
```

```
6 + (x^2) - (5*x)

The remainder is:
0

Polynomial Greatest Common Divisor (iterations = 1):
(x^2) - (2*x) + 2
1->
```

The number of iterations displayed is the number of polynomial divides done to compute the GCD with the Euclidean algorithm.

Echo command

Syntax: **echo [text]**

This command outputs a line of **text**, followed by a newline.

Edit command

Syntax: **edit [file-name]**

This command invokes the ASCII text editor specified by the EDITOR environment variable. By default, all equation spaces are edited. Access to shell (*/bin/sh*) is required for this command to work.

Type "edit" at the Mathomatic prompt to edit all expressions and equations you have entered for the current session. When you are done editing Mathomatic expressions and commands, save and exit the editor to have them automatically read in by Mathomatic. If Mathomatic gets an error reading in its new input, observe where the error is and continue, to automatically re-enter the editor.

To edit an existing file and have it read in, specify the **file name** on the edit command line.

Eliminate command

Syntax: **eliminate variables or "all" ["using" equation-number]**

This command is used to combine simultaneous equations, by automatically substituting variables in the current equation. It will scan the command line from left to right, replacing all occurrences of the specified **variables** in the current equation with the RHS of solved equations. The equation to solve can be specified with the "using" argument. If "using" is not specified, Mathomatic will search backwards, starting at the current equation minus one, for the first equation that contains the specified variable. The equation to solve is solved for the specified variable, then the RHS is inserted at every occurrence of the specified variable in the current equation. That effectively eliminates the specified variable from the current equation, resulting in one less unknown.

There is an advantage to eliminating multiple variables in one command: each equation will be used only once. If the same equation is solved and substituted into the current equation more than once, it will cancel out.

"eliminate all" is shorthand for specifying all normal variables on the command line. "repeat eliminate all" will eliminate all variables repeatedly until nothing more can be substituted, using each equation only once.

Here is a simple example of combining two equations:

```
1-> ; This arrives at the distance between two points in 3D space from the
1-> ; Pythagorean theorem (distance between two points on a 2D plane).
1-> ; The coordinate of point 1, 2D: (x1, y1), 3D: (x1, y1, z1).
1-> ; The coordinate of point 2, 2D: (x2, y2), 3D: (x2, y2, z2).
1->
1-> L^2=(x1-x2)^2+(y1-y2)^2 ; Distance formula for a 2D Cartesian plane.

#1: L^2 = ((x1 - x2)^2) + ((y1 - y2)^2)

1-> distance^2=L^2+(z1-z2)^2 ; Add another leg.

#2: distance^2 = (L^2) + ((z1 - z2)^2)

2-> eliminate L ; Combine the two equations.
Solving equation #1 for (L) and substituting into the current equation...

#2: distance^2 = ((x1 - x2)^2) + ((y1 - y2)^2) + ((z1 - z2)^2)

2-> distance ; Solve to get the distance formula for 3D space.

                                                           1
#2: distance = ((((x1 - x2)^2) + ((y1 - y2)^2) + ((z1 - z2)^2))^—)·sign2
                                                           2

Finished reading file "pyth3d.in".
2->
```

Extrema command

Syntax: **extrema [variable] [order]**

This command computes possible extrema (the minimums and maximums) of the current expression by default, or possible inflection points when **order** is 2. The result is placed in the next available equation space, displayed, and becomes the current equation. The original expression is not modified.

By default (**order**=1) this command computes stationary points. The stationary points of function **f(x)** are the values of **x** when the slope (derivative) equals zero. Stationary points are likely the local minimums and maximums of the function, unless the point is determined to be an inflection point.

For **y=f(x)**, where **f(x)** is the RHS and **x** is the specified **variable**, this command gives the values of **x** that make the minimums and maximums of **y**. This is computed by taking the derivative of **f(x)**, setting it equal to zero, and then solving for **x**.

The number of derivatives to take before solving can be specified by the **order** argument (default is 1). When **order** is 2, possible points of inflection are determined. A point of inflection is a point on a curve at which the second derivative changes sign from positive to negative or negative to positive.

```
1—> y=x^2

#1: y = x^2

1—> extrema x

#2: x = 0

2—>
```

This function is a parabola, with the minimum at **x=0**.

Factor command

Syntax: **factor ["number" [integers]] or ["power"] [equation-number-range] [variables]**
Alternate command name: **collect**

This command will factor manually entered **integers** when "**numbers**" is specified on the command line, otherwise this command will factor **variables** in expressions in the specified equation spaces. This command does not factor polynomials. To factor polynomials with repeated or symbolic factors, use the simplify command. To factor integers in equation spaces and display, use the "display factor" command.

"factor **number**" will prompt for an integer to factor, which may be up to 15 decimal digits. The plural "factor **numbers**" will repeatedly prompt for **integers** to factor. Multiple **integers**, or multiple expressions that evaluate to integers, can be specified on the same line and should be separated with spaces.

Without the "number" option, this command will factor out repeated sub-expressions in equation spaces. When factoring expressions, this command does some basic simplification and factors out any common (equal) sub-expressions it can, unless **variables** are specified on the command line, in which case only common sub-expressions containing those variables are factored out. This collects together terms involving those variables.

For example, with the following expression:

```
(b*c) + (b*d)
```

variable **b** factors out and the result of this command is:

```
b*(c + d)
```

If no variables are specified on the command line, this command factors even more: the bases of common (equal) bases raised to any power are factored out. This is called Horner factoring or Horner's rule.

For example:

```
1—> (2+3x)^2*(x+y)

#1: ((2 + (3·x))^2)·(x + y)

1—> unfactor ; expand

#1: (4·x) + (12·(x^2)) + (9·(x^3)) + (4·y) + (12·x·y) + (9·(x^2)·y)

1—> factor x ; collect terms containing x

#1: (x·(4 + (12·y))) + ((x^2)·(12 + (9·y))) + (9·(x^3)) + (4·y)

1—> x^3+2x^2+3x+4 ; enter another expression

#2: (x^3) + (2·(x^2)) + (3·x) + 4

2—> factor ; Horner factoring

#2: (x·((x·(x + 2)) + 3)) + 4

2—>
```

"factor **power**" does only power operator collecting; that is, **(a^n)*(a^m)*(b^n)*(b^m)** is transformed to **(a*b)^(n+m)**.

To undo any kind of factoring in selected equation spaces, use the unfactor command.

For command

Syntax: **for variable start end [step-size]**

This command only evaluates and displays the current expression for each value of the index **variable** as the index **variable** goes from **start** to **end** in steps of **step-size** (default 1). Nothing is modified.

The syntax of this command is the same as the sum and product commands. This command is not a programming construct, and only allows automatically plugging in sequential values of a **variable** in the current expression, displaying the results.

Fraction command

Syntax: **fraction [equation-number-range]**

This command reduces and converts expressions with any algebraic fractions in them into a single simple algebraic fraction (usually the ratio of two polynomials), similar to what Maxima's rat() function does. It does this by combining all terms added together with like and unlike denominators to a single simple fraction with a like denominator. Unlike denominators are combined by converting the terms to what they would be over like (common) denominators. Fractions are reduced by cancelling out the Greatest Common Divisor (GCD) of the numerator and denominator. The result of this command is mathematically equivalent to the original expression. Note that algebraic fractions added together with like denominators are automatically combined by almost any Mathomatic command.

Example:

```
1-> 1/x+1/y+1/z

    1   1   1
#1: - + - + -
    x   y   z

1-> fraction

     (((y + x)·z) + (x·y))
#1: ---------------------
           (x·y·z)

1-> unfactor

     ((y·z) + (x·z) + (x·y))
#1: -----------------------
            (x·y·z)

1-> unfactor fraction

    1   1   1
#1: - + - + -
    x   y   z

1->
```

"repeat fraction" repeatedly runs the fraction command until the result stabilizes to the smallest size expression.

If more simplification is needed, use the "simplify fraction" command instead.

Help command

Syntax: **help [topics or command-names]**

This command is provided as a quick reference while running Mathomatic. If the argument is a command name, a one line description and one line syntax of that command are displayed. Command names may be abbreviated.

Entering this command by itself will display a list of topics and commands. "help copyright" will display the copyright and license notice for the currently running version of Mathomatic.

To create a quick reference of all Mathomatic commands, type:

```
help all >quickref.txt
```

Imaginary command

Syntax: **imaginary [variable]**

This command copies the imaginary part of a complex number expression to the next available equation space. If the current expression or RHS of the current equation is not complex, the warning message "Expression is not a mix" will be displayed. A complex number expression contains both imaginary and real number parts. To copy only the real number part, see the real command.

The separation **variable** may be specified on the command line, the default is the imaginary unit **i**. **i** is really a mathematical constant equal to the square root of -1, but it can often be specified where variables are required in Mathomatic.

If successful, the result may contain the imaginary unit **i** or the specified separation **variable**.

```
1—> (a+b*i)/(c+d*i)

     (a + (b·i))
#1: -----------
     (c + (d·i))

1—> imaginary

    i·((b·c) - (a·d))
#2: -----------------
       (c^2 + d^2)

2—>
```

Integrate command

Syntax: **integrate ["constant" or "definite"] variable [order]**
Alternate command name: **integral**

This command computes the exact symbolic integral of a polynomial function with respect to the specified **variable**, using the current expression or RHS of the current equation as the function. If successful, the simplified integral is placed in the next available equation space, displayed, and becomes the current equation.

The default is to compute and display the **indefinite integral**, also known as the **antiderivative** or **primitive**. The antiderivative is the inverse transformation of the derivative.

"integrate constant" simply adds a uniquely named **constant of integration** ("C_1", "C_2", etc.) to each integration result. The constants of integration here are actually variables that may be set to any constant.

"integrate definite" also integrates, but prompts you for the lower and upper bounds for **definite integration**. If **g(x)** is the indefinite integral (antiderivative) of **f(x)**, the definite integral is:

```
g(upper_bound) - g(lower_bound)
```

Specifying the **order** allows you to repeatedly integrate. The default is to integrate once (**order**=1).

This command is only capable of integrating polynomials.

```
1-> x^3+x^2+x+1

#1: (x^3) + (x^2) + x + 1

1-> integrate x

     (x^4)    (x^3)    (x^2)
#2: ----- + ----- + ----- + x
       4        3        2

2-> derivative x ; check the result
Differentiating with respect to (x) and simplifying...

#3: (x^3) + (x^2) + x + 1

3-> compare 1
Comparing #1 with #3...
Expressions are identical.
3->
```

Laplace command

Syntax: **laplace ["inverse"] variable**

This command computes the Laplace transform of a polynomial function of the specified **variable**, using the current expression or RHS of the current equation as the function. If successful, the transformed function is placed in the next available equation space, displayed, and becomes the current equation.

This command only works with polynomials.

A Laplace transform can be undone by applying the **inverse** Laplace transform. That is accomplished by specifying the "inverse" option to this command.

```
1-> y=1

#1: y = 1

1-> laplace x ; compute the Laplace transform of 1

        1
#2: y = -
        x

2-> a*x^n ; a general polynomial term

#3: a*(x^n)

3-> laplace x

      a*(n!)
#4: -----------
    (x^(n + 1))

4-> laplace inverse x

#5: a*(x^n)

5->
```

Limit command

Syntax: **limit variable expression**

This command takes the limit of the current expression as **variable** goes to the specified **expression**. The result is always an equation and placed in the next available equation space and displayed.

L'Hopital's rule for taking limits is not used by this command. Instead the limit is taken by simplifying, solving, and substituting. This command is experimental and does not know about negative infinity and occasionally gives a wrong answer when dealing with infinities.

```
1-> 2x/(x+1)

      2·x
#1: -------
```

```
    (x + 1)

1-> limit x inf ; take the limit as x goes to infinity
Solving...

#2: limit = 2

1->
```

List command

Syntax: **list ["export"/"maxima"/"gnuplot"/"hexadecimal"] [equation-number-ranges]**

This command displays stored expressions in single-line (one dimensional) format. A single formatting option may be specified. With no option specified, expressions are displayed in decimal, Mathomatic format. The result can then be read back in by Mathomatic.

Formatting options:

> "list **export**" outputs expressions in a generally exportable, single-line format. You can cut-and-paste the expressions or redirect them to a file, so they can be read in with a different math program.
>
> "list **maxima**" is for making expression output compatible with the free computer algebra system Maxima.
>
> "list **gnuplot**" is for making expression output that is compatible with the free graphing utility gnuplot.
>
> "list **hexadecimal**" displays as normal, except constants are displayed as hexadecimal values in exponential notation, where no more precision is lost, as it would be if converted to decimal. The result that is displayed can be read back in by Mathomatic exactly as it was.

This list command simply outputs expressions and equations as stored internally by Mathomatic, translating them to the requested output format. There is no simplification and nothing more is done.

To display equation spaces in better looking multi-line fraction format, use the display command.

NIntegrate command

Syntax: **nintegrate ["trapezoid"] variable [partitions]**

This is a numerical integrate command that will work with almost any expression and will not generally compute the exact symbolic integral except for the simplest of expressions. This command will prompt you for the lower and upper bounds to perform numerical definite integration on the current expression or the RHS of the current equation, with respect to the specified **variable**. These bounds may be any expression not containing infinity.

This command uses Simpson's rule to do the approximation. Accuracy varies widely, depending on the expression integrated, the interval between the lower and upper bounds, and the number of **partitions**. The default is to split the interval into 1000 partitions. Setting the number of partitions greater than 10000 seldom is helpful, because of accumulated floating point round-off error.

If "trapezoid" is specified on the command line, the trapezoid method is used instead, which is usually less accurate than Simpson's rule. The way the trapezoid method works is: the interval from the lower bound to the upper bound is divided into 1000 **partitions** to produce 1000 trapezoids, then the area of each trapezoid is added together to produce the result. This means that the accuracy usually decreases as the interval increases. Simpson's rule uses the same method, with quadratic curves bounding the top of each trapezoid, instead of straight lines, so that curves are approximated better.

If the integration fails, chances of success are greater if you reduce the number of variables involved in the integration.

If there are any singularities, such as division by zero, between the bounds of integration, the computed result will be wrong.

Here is an example of successful numerical integration:

```
1-> y=x^0.5/(1-x^3)

             1
           (x^-)
             2
#1: y = ---------
         (1 - x^3)

1-> nintegrate x
Warning: Singularity detected, result of numerical integration might be wrong.
Enter lower bound: 2
Enter upper bound: 4
Approximating the definite integral of the RHS
using Simpson's rule (1000 partitions)...
Numerical integration successful:

#2: y = -0.16256117185712

1->
```

This example avoids the singularity at **x=1** and is accurate to 12 digits.

Optimize command

Syntax: **optimize [equation-number-range]**

This command splits the specified equations into smaller, more efficient equations with no repeated expressions. Each repeated sub-expression becomes a new equation solved for a temporary variable (named "temp").

Note that the resulting assignment statements may be in the wrong order for inclusion in a computer program with the code command; the order and generated code should be visually checked before using. The source code statements may need to be manually put in the right order to work properly.

```
1-> y = (a+b+c+d)^(a+b+c+d)

#1: y = (a + b + c + d)^(a + b + c + d)

1-> optimize

#2: temp = a + b + c + d


#1: y = temp^temp

1-> eliminate temp ; undo the optimization
Solving equation #2 for (temp)...

#1: y = (a + b + c + d)^(a + b + c + d)

1->
```

Pause command

Syntax: **pause [text]**

This command waits for the user to press the Enter key. It is useful in text files (scripts) that are read in to Mathomatic. Optionally, a one line text message may be displayed.

Typing "quit" or "exit" before pressing the Enter key will fail this command and abort the current script.

This command is ignored during test mode and when input is not a terminal.

Plot command

Syntax: **plot [equation-number-ranges] [x-range [y-range]] [expressions,]** This command automatically plots the given mathematical **expressions** in 2D or 3D with the free graphing utility gnuplot. The specified equations are plotted at once, along with any comma separated expressions on the command line. Each expression should contain the variable **x** to be successfully plotted. If it also contains the variable **y**, a 3D surface plot is performed in Cartesian space, with the **x**, **y**, and **z** axes projected on your 2D display.

A gnuplot X range, Y range, and even a Z range may be specified on the plot command line. For example, "plot [-128:128]" will make the X axis go from -128 to 128, and "plot [-128:128] [-1:1]" will also make the Y axis go from -1 to 1. A pair of straight brackets **[]** must surround each range.

Plotting of an equation space with the imaginary number **i** in it is allowed, however plots are always plotted on real Cartesian space, showing only real number points. If there are no real number points, the plot will fail.

Gnuplot must be installed and accessible from shell by typing "gnuplot" for this command to work. All functions and operators of gnuplot can be used in the plot **expression**. If gnuplot fails, the gnuplot command line is displayed to show what failed. To always show the gnuplot command line, enter "set debug 1" beforehand, to set debug level 1 for the current session.

Plots may be customized. Typing "set **plot_prefix**" at the Mathomatic main prompt, followed by a string of 8-bit characters, will prepend the string to the gnuplot plot string, when using the Mathomatic plot command. For example, if you type "set plot set polar\;" at the Mathomatic main prompt, 2D polar plots will be performed with subsequent plot commands, using variable "t" instead of "x".

Product command

Syntax: **product variable start end [step-size]**

This command performs a mathematical product ( ) of the current expression or the RHS of the current equation as the index **variable** goes from **start** to **end** in steps of **step-size** (default 1). The result is stored and displayed. The current equation is not changed.

```
1—> y=a*x

#1: y = a*x

1—> product
Enter variable: x
x = 1
To: 10

#2: y = 3628800*(a^10)

1—> 10!
 Answer = 3628800
1—>
```

To see all of the intermediate results, type "set debug 1" before this.

Push command

Syntax: **push [equation-number-ranges]**

This command pushes the current or specified equation spaces into the readline history, for easy editing and re-entry by using the cursor keys. The equation spaces are not modified. After this command, the pushed expressions are accessed by pressing the cursor UP (up arrow) key.

This command only exists if Mathomatic was compiled with readline support.

Quit command

Syntax: **quit [exit-value]**
Alternate command name: **exit**

Type in this command to exit Mathomatic. All expressions in memory are discarded. To save all your expressions stored in equation spaces, use the save command before quitting.

The optional decimal **exit value** argument is the exit status returned to the operating system. The default is 0, meaning OK.

Another way to quickly exit Mathomatic is to enter your operating system's End-Of-File (EOF) character at the beginning of an input line. The EOF character for Unix-like operating systems is Control-D.

Read command

Syntax: **read file-name**

This command reads in a text file as if you typed the text of the file in at the main prompt. The text file (also known as a script) should contain Mathomatic expressions and commands. Read commands may be nested; that is, the file read in may contain further read commands. If any command or operation returns with an error, the read operation is aborted.

Expressions saved with the save command are restored using this read command.

This command is automatically executed when you start up Mathomatic with file names on the shell command line.

The default file name extension (suffix) for Mathomatic input files is ".in". A file name extension is not required.

Real command

Syntax: **real [variable]**

This command copies the real part of a complex number expression to the next available equation space. If the current expression or RHS of the current equation is not complex, the warning message "Expression is not a mix" will be displayed. A complex number expression contains both imaginary and real number parts. To copy only the imaginary number part, see the imaginary command.

The separation **variable** may be specified on the command line, the default is the imaginary unit **i**.

There will be no imaginary numbers in the result.

```
1—> (a+b*i)/(c+d*i)

    (a + (b·i))
#1: -----------
    (c + (d·i))

1—> real

    ((a·c) + (b·d))
#2: ---------------
      (c^2 + d^2)

2—>
```

Repeat command

Syntax: **repeat command-line**

Any command may be preceded by "repeat", which sets the repeat flag for that command. Most commands ignore the repeat flag. Currently the calculate, divide, eliminate, fraction, roots, and simplify commands are repeatable.

Replace command

Syntax: **replace [variables ["with" expression]]**

By default, this command prompts you for a replacement expression for each variable in the current expression or equation. If an empty line is entered for a variable, that variable remains unchanged. The result is placed in the current expression or equation and displayed.

This command is very useful for renaming or substituting variables. It is smart enough to do variable interchange.

If **variables** are specified on the command line, you will be prompted for those variables only and all other variables will be left unchanged.

If "with" is specified, you won't be prompted and all **variables** specified will be replaced with the **expression** that follows.

Roots command

Syntax: **roots root real-part imaginary-part**

This command displays all complex number roots of a given positive integer **root** of a complex number. The number of the **root** equals the number of correct solutions. For example, "3" would give the 3 roots of the cube root. This command will also convert rectangular coordinates to polar coordinates.

The floating point **real part** (X coordinate) and **imaginary part** (Y coordinate) of the complex number are prompted for. Just enter an empty line if the value is zero. The polar coordinates of the given complex number are displayed first, which consist of an amplitude (distance from the origin) and an angle (direction). Then each root is calculated and displayed, along with an "Inverse check" value, which should equal the original complex number. The "Inverse check" is calculated by repeated complex number multiplication of the root times itself.

Since double precision floating point is used, the results are only accurate from 10 to 12 decimal digits.

```
1—> roots
Enter root (positive integer): 3
Enter real part (X): 8
Enter imaginary part (Y):

The polar coordinates before root taking are:
8 amplitude and 0 radians (0 degrees).

The 3 roots of (8)^(1/3) are:

2
Inverse check: 8

-1 +1.73205080757*i
Inverse check: 8

-1 -1.73205080757*i
Inverse check: 8

1—>
```

Save command

Syntax: **save file-name**

This command saves all expressions in all equation spaces into the specified text file. If the file exists, Mathomatic will ask you if you want to overwrite it. The saved expressions and equations can be reloaded (restored) at a later time by using the read command. You can edit the saved expressions with your favorite ASCII text editor.

Another way to save all equation spaces exactly as they are is to enter:

```
list hex all >file-name
```

into the main Mathomatic prompt, however this saves all constants in hexadecimal and it always overwrites "file-name". Because internally constants are binary, hexadecimal can represent them exactly. Reading in the result of this command should result in exactly the same expressions in the same equation spaces.

Set command

Syntax: **set [["no"] option] …**

This command sets various options listed below, for the current session. They remain in effect until you exit Mathomatic. Typing "set" without arguments shows all current option settings.

The specified **option** is turned on, unless it is preceded by "no", which turns it off. Some options can be followed by a number and some options can be followed by text, setting that option to the following value.

To permanently change the default settings of Mathomatic, set options can be put in the file *~/.mathomaticrc* (for Microsoft Windows: *mathomatic.rc* in the same directory as the Mathomatic executable or in the $HOME directory). It should be a text file with one set option per line, without the word "set". That file is loaded every time Mathomatic starts up, when not in test mode. The command "set **save**" conveniently saves all current session options in that file, making them permanent. "set no **save**" removes that file, so then Mathomatic will start up with all options set to the defaults.

Options:

"set **precision**" followed by an integer less than or equal to 15 sets the display precision in number of decimal digits for most numerical output. All arithmetic in Mathomatic is double precision floating point, so it is not useful to set this higher than 15 digits. Display output is rounded to the precision set by this option, though internally all constants are rounded to fit in a double precision float data type of about 15 decimal digits precision. The default for this display "precision" set variable is 14 digits.

"set no **autosolve**" will turn off solving by typing the solve variable at the main prompt, unless an equals sign (=) is included. This allows entry of single variable expressions into equation spaces. Solving is always allowed using the solve command.

"set no **autocalc**" will turn off automatic approximation and display with the calculate command of purely numerical input entered at the main prompt.

"set no **autoselect**" will turn off selecting of equation spaces by just typing in the equation number. Selecting is still possible using the **#** operator.

"set **auto**" and "set no **auto**" turn on and off all three of the above options at once. If turned off, all expressions entered at the main prompt will be entered into equation spaces, so they can be operated on by Mathomatic commands.

"set **debug**" followed by an integer sets the debug level number. The initial debug level is 0, for no debugging. If the level number is 2 ("set debug=2"), Mathomatic will show you how it solves equations. Level 4 debugs the simplify command and its polynomial routines. Levels 5 and 6 show all intermediate expressions. Set the debug level to -1 for suppression of warnings and helpful messages.

"set **case_sensitive**" will set alphabetic case sensitive mode, while "set no case" will set case insensitive mode (all alphabetic characters will be converted to lower case). "set case" is the default.

"set **color**" enables color mode. When color mode is on, ANSI color escape sequences are output to make expressions easier to read. Requires a terminal emulator that supports ANSI color escape sequences. Put the line "no color" in your *~/.mathomaticrc* file to always startup Mathomatic with color mode disabled, unless the **-c** or **-b** option is given.

"set **bold_colors**" enables highlighting in color mode. It makes all output brighter. Use this if any colors are difficult to see. This command can be shortened to "set bold". The **-b** option also sets this.

"set **columns**" followed by a positive integer sets the expected number of character columns (width) on a terminal screen with line wrap. When an expression is to be displayed in multi-line fraction format (two dimensional) and it is wider than this number of screen columns, single-line format is used instead, because otherwise the expression would not display properly due to wrap-around. "set no columns" or "set columns=0" does no checking for screen size and always displays in fraction format, which is useful for a terminal that doesn't wrap lines. In most cases, this value is set automatically to be the correct width on startup, or by typing "set columns". This value only affects 2D expression output.

"set **wide**" sets the number of screen columns (like "set columns=0" above does) and screen rows to 0, so that no checking for screen size is done, forcing 2D display of expressions that are too wide to display properly on a terminal with line wrap. Setting this option is useful if output is going to a file.

"set no **display2d**" will set the expression display mode to single-line format (one dimensional) using the list command, instead of the default fraction format (two dimensional) using the display command. Single-line format is useful when feeding Mathomatic output into another program.

fractions_display_mode is a new set option that allows controlling whether or not to display numerical fractions. It also can set the preference of simple or mixed fractions. "set no **fractions_display**" sets the mode to 0, disabling the automatic conversion of non-integer constants like .5 to **1/2** for display. "set **fractions**=1" means display some constants like .5 and 2.25 as their simple fraction equivalents: **1/2** and **9/4**. "set **fractions**=2" means display some constants as mixed or simple fractions, for example, **9/4** is displayed as **(2+(1/4))**, which is a mixed fraction.

"set no **prompt**" turns off Mathomatic prompt output, exactly like the **-q** (quiet mode) option does.

"set **preserve**" will set "preserve_surds" mode, which suppresses approximation of real roots of real rational numbers, if the result will be irrational. A surd is a quantity which can not be expressed by rational numbers. For example, **2^.5** (the square root of two, which is irrational and a surd) will remain **2^.5** unless explicitly approximated or "set no preserve" is entered. This option is turned on by default ("set preserve_surds"), allowing exact arithmetic and simplification of surds. Surds can always be manually approximated with the approximate and calculate commands.

"set **rationalize**" will set the "rationalize_denominators" option, which attempts to move radicals from the denominator of fractions to the numerator during simplification. This is the default.

"set **modulus_mode**" requires an integer from 0 to 2. When modulo arithmetic is performed with the modulus (%) operator, mode 0 returns a result that is the same sign as the dividend (same as C's % operator gives), mode 1 returns a result the same sign as the divisor (same as Python's % operator gives), mode 2 returns an always positive or zero result. Mode 2 is the default and is 100% mathematically correct and the type of modulo operation that can be generally simplified. Mode 0 is the remainder modulus used by the C and Java computer languages. This mode only affects modulo (**%**) operator numeric calculations. All modulus simplification rules are enabled, regardless of this mode.

"set **finance**" sets finance mode (fixed point display), which displays all constants with 2 digits after the decimal point (for example: "2.00") and negative numbers are always parenthesized (for example: "(-2.00)"). Displayed constants are rounded to the nearest cent, though internally there is no loss of accuracy. The number of digits to display after the decimal point may be specified with "set finance=number". This is not truly fixed point arithmetic, it is floating point displayed as fixed point. With double precision floating point, only the most significant 15 decimal digits will ever be correct. The default is no fixed point display (finance=0).

"set **factor_integers**" sets automatic factoring of integers for all displayed expressions. When set, all integers of up to 15 decimal digits are factored into their prime factors before the result of any command is displayed. This command can be shortened to "set factor".

"set **right_associative_power**" associates power operators from right to left in the absence of parentheses, so that **x^a^b** is interpreted as **x^(a^b)**. Other math programs typically associate power operators from right to left. The default is "set no right", which associates power operators the same as all other operators in Mathomatic, from left to right, resulting in

(x^a)^b.

"set **plot_prefix**" followed by a string of 8-bit characters will prepend the string to the gnuplot plot string, when using the Mathomatic plot command. For example, "set plot set polar\;" typed at the Mathomatic main prompt will allow 2D polar plots with subsequent plot commands, using variable "t" instead of "x".

"set **special_variable_characters**" followed by a string of 8-bit characters will allow Mathomatic to use those characters in variable names, in addition to the normal variable name characters, which are the alphanumeric characters and underline (_). For example, "set special $" will allow variable names like "$a" and "a$", and "set special []" will allow entry of array elements like "a[3]" for simulated array arithmetic. All non-alphanumeric characters in variable names are converted to underline characters (_) when exporting to a programming language or to a different program.

"set **directory**" followed by a directory name will change the current working directory to that directory. Not specifying a directory name defaults to your home directory. This command can be shortened to "set dir".

Simplify command

Syntax: **simplify ["sign"] ["symbolic"] ["quick[est]"] ["fraction"] [equation-number-range]**

This command fully simplifies expressions in selected equation spaces. The result is usually the smallest possible, easily readable expression that is mathematically equivalent to the original expression.

Use this command whenever you think an expression is not completely simplified or if you don't like the way an expression is factored. Sometimes simplifying more than once or using the "symbolic" option simplifies even more.

More than one option may be specified at a time.

Options:

"simplify **sign**" conveniently expands all "sign" variables by substituting them with all possible combinations of values (+1 and -1), storing the unique results into new equation spaces and fully simplifying. This will effectively create one simplified equation for each solution.

The "**symbolic**" option indicates **(a^n)^m** should always be reduced to **a^(n*m)**. This often simplifies more and removes any absolute value operations: (**(a^2)^.5 = a^(2*.5) = a^1 = a**). Try this "symbolic" option if the simplify command doesn't simplify well, it often helps with powers raised to powers, though it is sometimes not 100% mathematically correct.

The "**quick**" option skips expanding sums raised to the power of 2 or higher, like **(x+1)^5**. Also, algebraic fractions will be simpler (less fractions within fractions) with this option.

The "**quickest**" option very basically simplifies without any unfactoring nor factoring. Running the simplify command with this option makes it complete almost instantaneously.

"simplify **fraction**" fully simplifies any expression with division in it down to the ratio of two polynomials or expressions, like Maxima's ratsimp() function does. The result will be a single simple algebraic fraction, like the fraction command produces, the difference here being it will be completely simplified. This is accomplished by full simplification without doing "unfactor fraction" and without doing polynomial or smart (algebraic) division on the divide operators.

This simplify command applies many algebraic transformations and their inverses (for example, unfactor and then factor) and then tries to combine and reduce algebraic fractions and rationalize their denominators. Complex fractions are converted to simple fractions by making the denominators of fractions added together the same, combining and simplifying. Polynomials with repeated or symbolic factors are factored next. Then smart (heuristic) and polynomial division are tried on any divides, possibly making complex fractions if it reduces the expression size. Lastly, the expressions are nicely factored and displayed.

Smart division is a symbolic division like polynomial division, but it tries every term in the dividend, instead of only the term with the base variable raised to the highest power, to make the expression smaller.

```
1—> (x+2^.5)^3 ; an irrational polynomial with repeated factors

               1
#1: (x + (2^-))^3
               2

1—> unfactor ; multiply it out

                              1                   1
#1: (x^3) + (3*(x^2)*(2^-)) + (6*x) + (2*(2^-))
                              2                   2

1—> simplify ; put it back together, since factored is its simplest form

               1
#1: (x + (2^-))^3
               2

1—>
```

"repeat simplify" repeatedly runs the simplify command until the result stabilizes to the smallest size expression.

Solve command

Syntax: **solve ["verify"] ["for"] variable or "0"**

Options:

This command automatically solves the current equation for the specified **variable** or for zero. The current equation is replaced with the result, and the result is displayed. See the section on Solving Equations for details.

Solving for **variable^2** or **0^2** will isolate the square root of the largest expression containing the specified variable, and then square both sides of the equation. This is a new feature for properly squaring, cubing, etc. both sides of an equation, and it works for any power and variable with any equation with roots.

The "**verify**" option checks the result of solving for a variable by plugging all solutions into the original equation and then simplifying and comparing. If the resulting equation sides are identical (an identity), a "Solutions verified" message is displayed, meaning that all of the solutions are correct. Otherwise "Solution may be incorrect" is displayed, meaning at least one of the solutions is incorrect or unverifiable. The "verify" option only works when solving for a single variable.

The "**for**" option has no additional effect and is to make entering this solve command more natural.

Sum command

Syntax: **sum variable start end [step-size]**

This command performs a mathematical summation (∑) of the current expression or the RHS of the current equation as the index **variable** goes from **start** to **end** in steps of **step-size** (default 1). The result is stored and displayed. The current equation is not changed.

```
1—> y=a*x

#1: y = a*x

1—> sum
Enter variable: x
x = 1
To: 10

#2: y = 55*a

1—>
```

To see all of the intermediate results, type "set debug 1" before this.

Tally command

Syntax: **tally ["average"]**

This command prompts for a value, adds it to a running (grand) total, simplifies and displays the running total and optional average, and repeats. The average is the arithmetic mean, that is the running total divided by the number of entries. No equation spaces are used.

It is a convenient way of adding, subtracting, and averaging many numbers and/or variables. Enter a minus sign (–) before each value you wish to subtract. Enter an empty line to end.

Taylor command

Syntax: **taylor ["nosimplify"] variable order point**

This command computes the Taylor series expansion of the current expression or RHS of the current equation, with respect to the specified **variable**. The Taylor series uses differentiation and is often used to approximate expressions near the specified **point**.

The Taylor series of **f(x)** at **point a** is the power series:

```
                                            2                    3
       f'(a) (x - a)   f''(a) (x - a)    f'''(a) (x - a)
f(a) + ------------- + --------------- + ---------------- + …
            1!                2!                 3!
```

where **f'(x)** is the first derivative of **f(x)** with respect to **x**, **f''(x)** is the second derivative, etc.

This command prompts you for the **point** of expansion, which is usually a variable or 0, but may be any expression. Typically 0 is used to generate Maclaurin polynomials.

Then it prompts you for the **order** of the series, which is an integer indicating how many derivatives to take in the expansion. The default is a large number, stopping when the derivative reaches 0.

The result is simplified unless the "nosimplify" option is specified, and placed in the next available equation space, displayed, and becomes the current equation. The original expression is not modified.

```
1-> e^x

#1: e^x

1-> taylor x
Taylor series expansion around x = point.
Enter point: 0
Enter order (number of derivatives to take): 8
Computing the Taylor series and simplifying...
8 derivatives applied.

            x^2   x^3   x^4   x^5   x^6   x^7     x^8
#2: 1 + x + --- + --- + --- + --- + --- + ---- + -----
             2     6    24    120   720   5040   40320

2->
```

Unfactor command

Syntax: **unfactor ["fraction"] ["quick"] ["power"] [equation-number-range]**
Alternate command name: **expand**

This command algebraically expands expressions in selected equation spaces by multiplying out products of sums and exponentiated sums and then simplifying a little. One or more options may be specified.

To illustrate what unfactoring does, suppose you have the following equations:

```
1—> a=b*(c+d)

#1: a = b*(c + d)

1—> z=(x+y)^2

#2: z = (x + y)^2

2—> unfactor all

#1: a = (b*c) + (b*d)


#2: z = (x^2) + (2*x*y) + (y^2)

2—>
```

(x+y)^2 is called an exponentiated sum and is converted to **(x+y)*(x+y)** and then multiplied out, unless the "**quick**" option is given. Because this is a general but inefficient expansion method, exponentiated sums usually fail expansion when the power is greater than 10, growing larger than will fit in an equation space. "unfactor **quick**" only expands products of sums, and not exponentiated sums.

The opposite of unfactoring is factoring. Careful and neat factoring is always done by the simplify command.

"unfactor **fraction**" by itself expands algebraic fractions by also expanding division of sums, multiplying out each fraction with a sum in the numerator into the sum of smaller fractions with the same denominator for each term in numerator. See the example under the fraction command.

"unfactor **power**" does only power operator expansion; that is, **(a*b)^(n+m)** is transformed to **(a^n)*(a^m)*(b^n)*(b^m)**.

Variables command

Syntax: **variables ["c" or "java" or "integer"] [equation-number-ranges]**

Show all variable names used within the specified expressions, from most frequent to least frequently occurring. The programming language options output the variable definitions required to make code from the specified equations. This does not initialize any variables, it only defines them as needed for a C or Java compiler. This command is not necessary for generating Python code.

Version command

Syntax: **version**

Shows the version number, compilation options used, maximum possible memory usage, and license summary, for the currently running version of Mathomatic. The maximum memory usage displayed is the amount of RAM used when all equation spaces have been filled. It does not include stack size (which varies) or executable (code) size.

Up to the documentation index www.mathomatic.org

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant

Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.