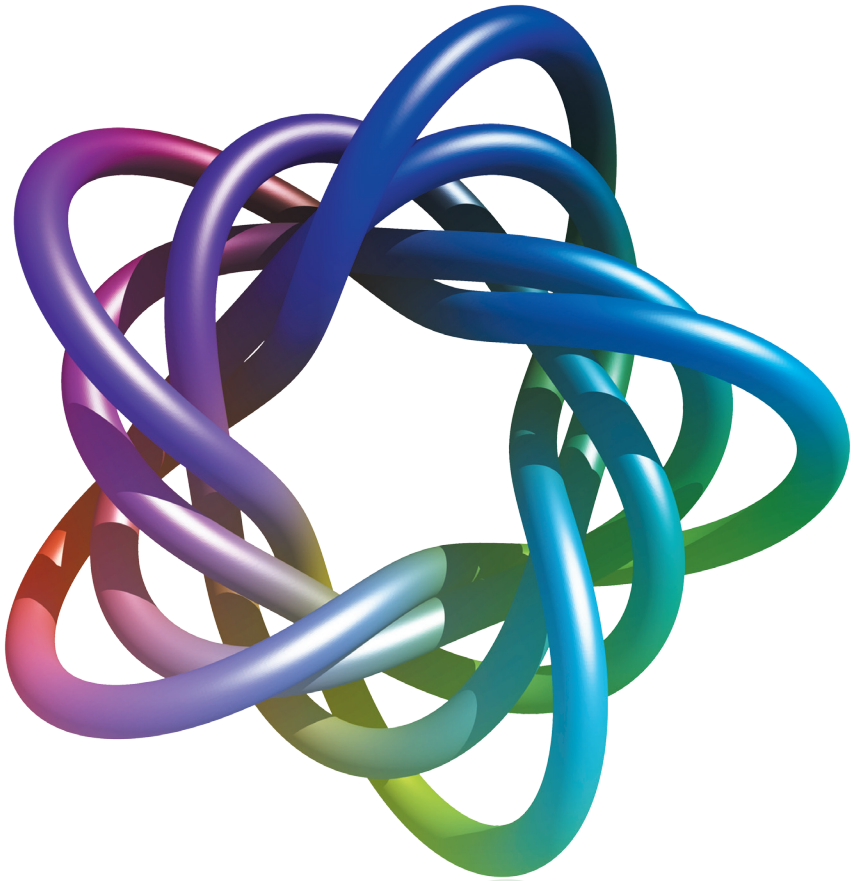


How Maple™ Compares to Mathematica®

Maplesoft™

Mathematics • Modeling • Simulation

A Cybernet Group Company

How Maple™ Compares to Mathematica®

Choosing between Maple™ and Mathematica®? On the surface, they appear to be very similar products. However, in the pages that follow you'll see numerous technical comparisons that show that Maple is much easier to use, has superior symbolic technology, and gives you better performance.

These product differences are very important, but perhaps just as important are the differences between companies. At Maplesoft™, we believe that given great tools, people can do great things. We see it as our job to give you the best tools possible, by maintaining relationships with the research community, hiring talented people, leveraging the best available technology even if we didn't write it ourselves, and listening to our customers.

Here are some key differences to keep in mind:

- **Maplesoft has a philosophy of openness and community which permeates everything we do.** Unlike Mathematica, Maple's mathematical engine has always been developed by both talented company employees and by experts in research labs around the world. This collaborative approach allows Maplesoft to offer cutting-edge mathematical algorithms solidly integrated into the most natural user interface available. This openness is also apparent in many other ways, such as an eagerness to form partnerships with other organizations, an adherence to international standards, connectivity to other software tools, and the visibility of the vast majority of Maple's source code.

- **Maplesoft offers a solution for all your academic needs,** including advanced tools for mathematics, engineering modeling, distance learning, and testing and assessment. By contrast, Wolfram Research has nothing to offer for automated testing and assessment, an area of vital importance to academic life. Maple T.A.™, our testing and assessment system, is based on Maple. Even if you aren't looking at assessment now, choosing Maple today will give you and your institution a valuable head-start later on.

Finally, we are frequently told that we are a better company to work with. We don't believe we have the solution to every problem inside our own company walls, we don't expect that you will use our products exclusively, and we don't believe our company will radically transform every branch of science. Individuals, companies, educational institutions, and publishers choose our technology because we are more flexible, more responsive, and more focused on creating great tools for our customers.

Now on to the technical stuff!

Laurent Bernardin
Executive Vice-President & Chief Scientist
Maplesoft

Interface

A good interface is almost invisible. If your software looks and works the way you are expecting, you and your students can simply get your work done, without worrying about the mechanics of the tool you are using. Maplesoft has always been a pioneer in math software usability, and continually strives to ensure that new and occasional users are immediately productive while experienced users have the tools and flexibility they need to work efficiently. Our aim is to allow our customers to work in an environment that feels as natural as possible for teaching, learning, and doing mathematics. This includes using standard mathematical notation everywhere, respecting standard software conventions, and providing an environment where users can create the same sort of documents they see in their textbooks, on their whiteboards, and in their notebooks.

Standard Math Notation

Entering mathematical expressions that look like mathematical expressions is very easy in Maple. The equation editor automatically formats fractions and exponents as you type. You can enter the expression the same way you would write it down, and it appears in Maple as it would when written in your textbook. This makes the mathematics easy to enter and easy to read. Mathematica, however, uses some non-standard notation which requires the user to translate back and forth between standard mathematics and Mathematica syntax.

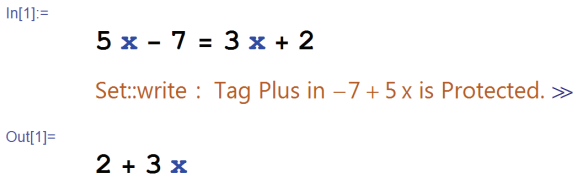
Here are examples of expressions entered using the default settings in both systems.

Maple	VS	Mathematica
$\sin(2\,x)$	See notes 1 and 2.	`Sin[2 x]`
$5\,x - 7 = 3\,x + 2$	See notes 3.	`5 x - 7 == 3 x + 2`
$2\,x^2 + \cos\left(\frac{x}{2}\right)$	See notes 4.	`2 x^2 + Cos[x / 2]`
$\lim_{x \to 0} \frac{\sin(x)}{x}$	See notes 5.	`Limit[Sin[x] / x, x → 0]`

Interface

Note that

1. In Maple, as in standard mathematical notation, round brackets are used for functions: *f(x)*. In Mathematica, square brackets are used: f[x].
2. In Maple, common functions are written using standard notation, with the initial letter in lower case: *log(x)*, *sin(x)*, *cos(x)*. In Mathematica, these functions all require an initial capital letter: Log[x], Sin[x], Cos[x]
3. In Maple, mathematical equality is denoted by '='. In Mathematica, the equal sign is reserved for variable assignment, and a double equal sign is used for equality. Using a single equal sign results in an error:



4. "2-D" formatting of fractions and exponents is applied automatically by default in Maple. For instance, when you type '/', Maple inserts the horizontal fraction bar and the next thing you type appears in the denominator. In Mathematica, fractions and exponents are not formatted by default. They can be formatted during typing by using alternative keystrokes to standard entry (e.g. using Ctrl+/ for a fraction instead of /) or by first entering the expression in Mathematica syntax and then converting it to traditional form afterwards using a menu operation.
5. In Maple, by default palettes use standard math formatting. For example, integrals look like integrals. In Mathematica, palettes insert the command. A menu operation must be applied afterwards to convert the command to standard mathematical notation.

Preferences vary, so traditional syntax entry is also an option for Maple users. In that case, Maple supports familiar calculator-style syntax: 2*x^2+cos(x/2). Users can choose which style they wish to use, and even switch between them in the same document.

Enter vs. Shift Enter

In Maple, once you have entered your problem, you press the <Enter> key to tell Maple to perform the computation and give you the result. Typing "2+2 <Enter>" results in 4.

In Mathematica, typing "2+2 <Enter>" moves the cursor to the next line, without calculating anything. To ask Mathematica to perform the computation, you must press <Shift>+<Enter>. This non-standard interaction requires users to adapt their normal behavior.

Typesetting

Maple's mathematics looks like it would appear in a textbook, making it very easy to read, understand, and verify. In Mathematica, even when traditional formatting is applied, the results still do not follow textbook standards. For example, the variable names are not italicized. While the expressions are understandable, they can be harder to comprehend at a glance.

Here are some samples from both Maple and Mathematica. In some of the Maple examples, Maple's in-line results feature is used to put both the input and the result on the same line for more compact presentation. Mathematica does not have this ability.

Maple	VS	Mathematica
$\sum_{i=1}^{3} a_i = a_1 + a_2 + a_3$		In[2]:= $\sum_{i=1}^{3} a_i$ Out[2]= $a_1 + a_2 + a_3$
$\prod_{i=m}^{n} x_i$		$\prod_{i=m}^{n} x_i$
$\frac{d}{dx}\sqrt{x} = \frac{1}{2\sqrt{x}}$		In[3]:= $\partial_x \sqrt{x}$ Out[3]= $\frac{1}{2\sqrt{x}}$
$\int_a^b f(x)\,dx$		$\int_a^b f(x)\,dx$

> "Maple goes out of the way to make the learning curve as short as possible. Compared to other tools, this is Maple's biggest advantage. Another feature I like in Maple is its ability to combine good looking mathematics with interactivity. With other tools, you get one or the other; to get them both in one is difficult. But with Maple, I can create sophisticated documents with attractive mathematical expressions that have interactive features. The mathematical expressions show up exactly like in textbooks, and that's exactly how students should see it."
>
> Dr. Joshua Holden, Mathematics Professor, Rose-Hulman Institute of Technology

Interface

Combining Text and Results

In Maple, it is very easy to combine text and mathematics in the same sentence. You can even have calculated results appear in the middle of a sentence, so that the sentence changes automatically if the results are updated.

$$\text{The function } \frac{1}{(x-1)^2} \text{ has an essential discontinuity at } x = 1.$$

By changing the definition of the function and re-executing the document, the new discontinuity is found and the statement is updated appropriately:

$$\text{The function } \frac{1}{(x-3)^2} \text{ has an essential discontinuity at } x = 3.$$

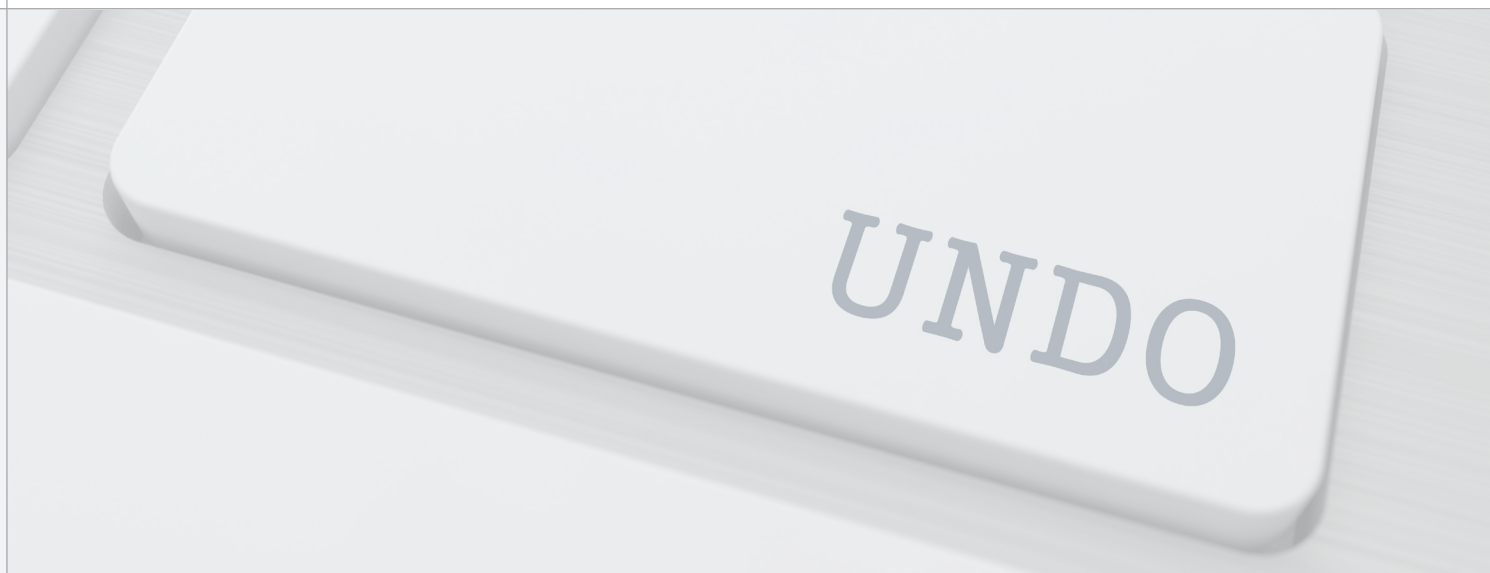
In Mathematica, it is not possible to combine text and mathematics results in this way. You can combine text and static math in the same cell, but you cannot display calculated results. If your results change, you must edit your statement by hand.

Undo

"Oops, I wish I hadn't done that." It happens. You make a mistake, and then wish you could undo a previous action, or sometimes even a sequence of previous actions.

In Maple, you can undo and redo text, plot, and other visual changes to your document, including the visual effects of a math calculation or plot operation. You can undo many steps, not just the last one.

In Mathematica, you can only undo your last basic editing command (e.g. cut, copy, typing text). It is not possible to redo an action you have undone, and it is not possible to undo more than the last action. As per the Mathematica Undo documentation, "The only way to undo several actions at once is to go back to an earlier version of your file."



Clickable Math™

From the introduction of context-sensitive menus for mathematical operations in 1998 to today's extensive collection of tutors, task templates, Math Apps, Smart Popups, and more, Maple's Clickable Math approach has revolutionized how mathematics is taught, learned, and done.

Example: Drag-to-Solve™

The following example uses Drag-to-Solve in Maple to solve this linear equation in the way students are taught to do it, moving terms around and performing operations on both sides of the equation. To move a term from one side of the equal sign to the other, the student simply drags the term across, and Maple understands what the student means by the action.

Drag 3*x* from the right side of the equal sign to the left side. A Smart Popup window is displayed, previewing the results of this manipulation.	$5x - 7 = 3x + 2$ Subtract $2x - 7 = 2$
Next, in the resulting output equation, drag -7 from the left side of the equation to the right side.	$2x - 7 = 2$ Add $2x = 9$
Drag and drop the factor of 2, in front of *x*, to the right side of the equal sign.	$2x = 9$ Divide $x = \frac{9}{2}$
The result of the above steps is a fully worked out solution.	$5x - 7 = 3x + 2$ subtract 3*x from both sides → $2x - 7 = 2$ add 7 to both sides → $2x = 9$ divide both sides by 2 → $x = \frac{9}{2}$

Interface

The student also has the choice of going directly to the solution using the context-sensitive menu in Maple:

$$5x - 7 = 3x + 2 \xrightarrow{\text{solve}} \left\{x = \frac{9}{2}\right\}$$

In Mathematica, it is not possible to solve a problem like this step-by-step dragging the terms of an expression around to perform an operation. To solve this equation, the student enters it, remembering to use the double equal sign syntax, and then chooses a menu option that goes directly to the solution:

```
In[1]:= 5 x - 7 == 3 x + 2
Out[1]= -7 + 5 x == 2 + 3 x
In[2]:= Solve[-7 + 5 x == 2 + 3 x, {x}]
Out[2]= {{x → 9/2}}
```

Example: Menu Operations on Subexpressions

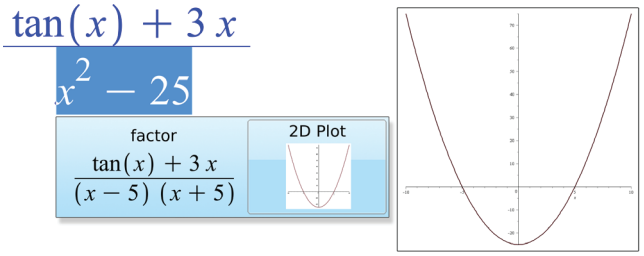
In Maple, mathematical operations can be done on the full expression or on a subexpression. This allows you to rewrite part of your expression in a different form, as one step in your solution. For example, in Maple, you can use menus to apply a trigonometric identity to *sec(x)* in:

$$\sec(x)\tan(x) = \frac{\sin(x)}{\cos(x)^2}$$

reciprocal function identity: sec(x)=1/cos(x) →

$$\frac{\tan(x)}{\cos(x)} = \frac{\sin(x)}{\cos(x)^2}$$

You can even ask for a plot of a subexpression. For instance, if you want to remind your students that the denominator in your expression can sometimes be 0, you can use the Smart Popup preview feature to show the a quick preview of a plot of just the denominator.



It is not possible to apply menu operations to subexpressions in Mathematica.

> "Maple is so easy to use; its *Clickable Math* interface has features that make common mathematical operations as simple as pointing and clicking. This also makes it a very easy-to-teach program. Students effortlessly learn the fundamentals of mathematics using the same tools that the industry is using, and this early introduction will help them in the long run."
>
> Dr. Christopher Chin, Senior Lecturer, Australian Maritime College

Mathematics

Unlike Mathematica, Maple's mathematical engine has always been developed by both talented company employees and by experts in research labs around the world. This collaborative approach allows Maplesoft to offer cutting-edge mathematical algorithms in a wide variety of fields.

Here are examples of some areas in which Maple excels:

- **Differential Equations.** Maple is the uncontested leader for computing symbolic solutions to differential equations. Maple computes symbolic solutions to 97.5% of the 1345 solvable linear and non-linear ODEs from the famous text, *Differentialgleichungen* by Kamke, and does so in 43 minutes. Mathematica only handles 79.8%, and takes 7 hours and 8 minutes to find those solutions.
- **Factoring Polynomials.** Factoring is an important, frequently-used method of expression simplification. For mathematical researchers, finding the irreducible factors of potentially large polynomials can provide significant insight into related mathematical structures. Maple employs numerous techniques found in recent research literature to successfully tackle classes of large, challenging polynomial factoring problems. The Zimmermann benchmark of *Polynomial Factorization Challenges* provides eight problems that are representative of types of polynomials that are difficult to factor. Maple can solve each of the eight problems in this suite. Five of them are solved in less than two seconds each, and the other three in under 80 seconds. By contrast, Mathematica finds three solutions in under two seconds each, two more took close to an hour each, and for each of the remaining three, the computation was stopped after an hour with no solution found.
- **Physics.** Maple has, by far, the most comprehensive support for undergraduate, graduate, and research level physics of any mathematical software package. It covers vector analysis, tensor analysis, quantum state vector calculus, classical and quantum field theories, and more, while providing pencil-and-paper style input and textbook-quality display of results. By contrast, Mathematica does not handle anticommutative and noncommutative variables. It also does not handle the vast majority of specialized rules and operators from physics, nor does it support conventional physics notation on input or on output.
- **Differential Geometry.** Maple's differential geometry support covers a wide range of topics, from jet calculus to the realm of the mathematics behind general relativity. It also includes a series of lessons in differential geometry covering both beginner and advanced topics. Mathematica does not provide any functionality for differential geometry.

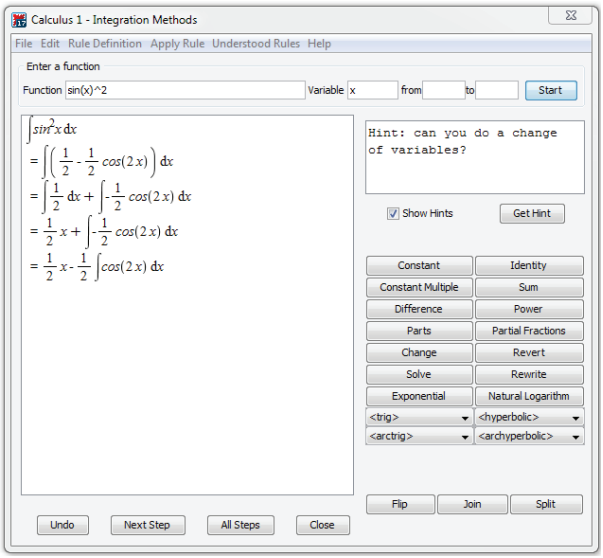
> "In my research I mostly use Maple for solving systems of differential equations and systems of polynomial equations, because Maple is faster and better than Mathematica."
>
> Dr. Sergio Parreiras, Associate Professor of Economics, University of North Carolina at Chapel Hill

Students

Students using a system like Maple or Mathematica often have different needs than non-students. They don't just need a final answer. They are still learning the mathematical concepts behind the problems they are trying to solve, and need an environment that allows them to explore the concepts and break problems down into smaller steps instead of jumping immediately to the solution.

In addition to features like Drag-to-Solve, context-sensitive menus, task templates, and Smart Popups, which allow students to solve problems step-by-step without commands, Maple also features a set of student packages. Student packages offer focused learning environments in which students can explore and reinforce fundamental concepts in the same way their instructor does in class.

For example, Maple includes step-by-step tutors that allow students to practice integration, differentiation, finding limits, and more. The Integration Tutor, shown below, lets a student compute an integral by selecting which rule to apply at each step. Maple will also offer hints or show the next step, if asked. The tutor doesn't only demonstrate how to obtain the result, but is truly designed for practicing and learning.



Mathematica has nothing like these step-by-step tutors. The closest Mathematica comes is "Show Steps" in Wolfram | Alpha®, which is not actually available in Mathematica at all, but must be accessed through a web browser. In addition, the "Show Steps" functionality is not interactive. It shows the final answer plus the steps needed to get there, but it does not allow the student to try the problem step-by-step on their own.

Another advantage of the Maple student package concept is that computations are performed using assumptions that are appropriate for the student's level. This avoids situations where a computation returns results that the student is unable to make sense of as the required background will be covered later in the curriculum. For example, students in a first year linear algebra course typically do not worry about complex numbers. Calculating the Euclidean norm for a vector using Maple's Student Linear Algebra package returns the expected result:

$$with(Student:\text{-}LinearAlgebra):$$
$$Norm\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}, 2\right) = \sqrt{a^2 + b^2 + c^2}$$

However, taking into account that symbolic quantities like *a*, *b* and *c* may represent complex numbers, Maple will give a more general result outside of the context of the Student package:

$$with(LinearAlgebra):$$
$$Norm\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}, 2\right) = \sqrt{|a|^2 + |b|^2 + |c|^2}$$

Mathematica does not have the ability to tailor its computational context to the knowledge of students at a particular level, and so it always gives the more general answer. As a result, instructors need to spend time explaining unexpected results to their students.

"My students found it overwhelming and very time consuming to learn different software tools as part of their course. Maple made a lot of difference to their approach to learning. It is easy to use and you can even write down equations with the mouse, similar to writing with a pen or pencil. You can access various expressions from the palette just by the click of a mouse – I can't imagine it getting any simpler! The context-sensitive menu is amazing; students learn a lot more by just experimenting with the various possibilities that Maple presents for each equation. For students who have their days crammed with course work, this came as a miracle."

Dr. Nick Zorka, Lawrence Tech University

Programming

Functional vs. Procedural Programming

Maple allows you to write your own scripts and programs with a procedural programming language that will be very familiar to users of C, Java, Fortran, Visual Basic, and other procedural languages. It also includes many elements from functional and object-oriented programming, allowing you to choose the approach that is most suited to your problem and programming style.

Mathematica also offers constructs to support different programming styles, but typically encourages a functional programming approach. Even their procedural programming elements are implemented in a functional programming style, so that constructs like if-statements and loops are written as function calls. Functional programs are often opaque; most people, even experienced programmers, find functional-style programs to be significantly harder to write, read, and debug.

As an example, the web page http://rosettacode.org/wiki/Hailstone_sequence offers sample code for calculating the Hailstone sequence using more than 90 different languages including Maple and Mathematica. The Hailstone sequence of numbers can be generated from a starting positive integer *n* by:

- If *n* is 1 then the sequence ends.
- If *n* is even then the next number of the sequence = *n*/2
- If *n* is odd then the next number of the sequence = 3*n* + 1

At the time of writing, Rosetta Code contributors provided the following implementations for Mathematica and Maple, illustrating the differing styles favored for each language.

Maple VS **Mathematica**

```
hailstone := proc( N )
    local n := N, HS := Array();
    HS(1) := n;
    while n > 1 do
        if type(n,even) then
            n := n/2;
        else
            n := 3*n+1;
        end if;
        HS(numelems(HS)+1) := n;
    end do;
    HS;
end proc;
```

```
HailstoneFP[n_] :=
   Drop[
      FixedPointList[
         If[# != 1,
            Which[Mod[#, 2] == 0,
               #/2,
               True,
               ( 3*# + 1)
            ],
            1
         ] &, n
      ], -1
   ]
```

Detecting Mistakes

Programmers make mistakes, the most common of which are syntax errors. The job of the compiler or interpreter is to catch these errors and make it as easy as possible for the programmer to correct them.

Consider this simple Maple procedure. The programmer has mistakenly used *n* in the if-statement instead of the parameter *N*. The variable *n* does not have a value, so it is impossible for Maple to evaluate the if-statement.

```
MyProc := proc( N )
    if 3 < n then
        print(1);
    else
        print(2);
    end if;
    return 1;
end;
```

Calling MyProc(4) in Maple results in an appropriate error, stating that Myproc "*cannot determine if this expression is true or false: 3 < n*". This helps to quickly identify the mistake.

In Mathematica, the procedure looks like this:

```
MyModule[N_] := Module[{},
    If[ 3 < n,
        Print[1],
        Print[2]
    ];
    Return[ 1 ];
]
```

Calling this procedure in Mathematica produces no warning. The code simply does not act as expected, with no explanation.

Now consider what happens to the same procedure when the user accidentally omits the semicolon after the if-statement:

Maple	VS	Mathematica

Code fragment missing semicolon:
`if 3 < n then`
`  print(1);`
`else`
`  print(2);`
`end if`
`return 1;`
		Code fragment missing semicolon:
		`If[ 3 < n,`
		`  Print[1],`
		`  Print[2]`
		`]`
		`Return[ 1 ];`

Result: Maple gives a syntax error, and places the cursor at the point of failure.		Result: No error message is given. Mathematica interprets the code to mean `If[] * Return[]`, and so returns incorrect results when the procedure is called.

This problem is compounded by the fact that, unlike in Maple, you usually have to omit the semicolon when exploring interactively in a live Mathematica session. If you put semi-colons at the end of statements in an interactive session, output will be suppressed. When you prototype the steps of your solution in an interactive session, you have to add semi-colons afterwards when you convert those steps to a procedure, and you will not receive a warning if you forget one.

Programming

Other Programming Considerations

- **Debugging.** Both products include debuggers that allow you to set breakpoints and step through code. In Maple, you can evaluate arbitrary expressions in the context of the stopped procedure so you can evaluate, inspect, and debug your code. In Mathematica, it is not possible to evaluate expressions referring to locally scoped variables. You can inspect the values of the local variables, but you cannot perform computations involving them, such as asking for the size of a local array, looking deeper into the structure of an individual element in a local list, or even multiplying two local variables together. In addition, in Maple you can use the debugger to step through built-in library routines as well as user code, but in Mathematica, you can only step through user-defined routines.

- **Parallel Programming.** Maple is the only technical computing system that allows you to take advantage of multithreading in your own programs. The Maple programming language offers direct access to launching and controlling threads, while a task-based programming model simplifies thread management. Writing parallel algorithms using the Task Programming Model in Maple reduces and removes many of the difficulties associated with standard threaded programming, making it possible for even moderately experienced programmers to take full advantage of the processing power of their computer. Parallel programming in Mathematica involves spawning new math kernels, which requires extra memory and resources as well as complicated message-passing between nodes to communicate context and state. This style of parallel programming takes longer to develop and debug, and generally requires an experienced programmer. In addition, Maple's multi-threading and multi-process parallelism have no restrictions on the number of cores you can access. If you have 16 cores, you can use all 16 cores out-of-the-box with your default license. Mathematica limits the number of processes you can have running to 4 unless you buy additional licenses.

- **Writing scripts.** Both Mathematica and Maple provide a non-GUI based script interpreter, which is often useful for programmers who prefer to use their favorite editor for code development. In this environment, error messages are displayed in Maple but sometimes suppressed in Mathematica, so it is much harder for the programmer to debug the code.

> "I find modules and records to be an incredibly powerful feature set in Maple, allowing me to do object-oriented programming "lite" in addition to procedural programming. It gives me the flexibility to choose the programming style most suited for my work. In fact, in a recent project I was able to build a system in Maple with design optimizations I would not have attempted otherwise. Unbeatable!"
>
> Uli Wienands, Senior Staff Scientist, SLAC National Accelerator Laboratory

Numerics

Maple's numerical model is derived from the IEEE/754 floating point standard. This standard is then extended consistently in Maple to cover computations involving arbitrarily high precision and complex floats. This numerical model is the accepted standard in the computer hardware and software industry, and it is thoroughly understood and documented. Results from Maple can be compared in a consistent way with results from other systems which use the same international standard. For example, an algorithm coded in MATLAB® can be imported into Maple and it will give the same results. In addition, the rationale for the choices Maple makes can always be determined by consulting the standards documentation.

Mathematica uses a proprietary numeric model derived from something called "significance arithmetic", not the international standard, and the details are not published. Algorithms written in another system, when implemented in Mathematica, can give different results and those differences are not predictable. While each system has its strengths and weaknesses due to the inherent nature of floating point computations, Maple's model is well understood and subject to mathematical investigation of algorithmic behavior in identifiable problematic cases. By contrast, the closed, proprietary model of Mathematica means that incorrect results are not always predictable or detectable.

Consider this example, first published by Richard Fateman from the University of California, Berkeley. A recursively defined sequence is defined as:

$$s_i = 2\,s_{i-1} - 3\,s_{i-1}^2 \quad \text{with} \quad s_0 = 0.3$$

Mathematically, this sequence converges to ⅓ very quickly. In Maple, and other systems that follow the IEEE standard, this is what happens:

```
s := proc(i)
      option cache;
      2 s(i - 1) - 3 s(i - 1)^2
    end proc:
s(0) := .3:
Digits := 20:

[s(1), s(2), s(10), s(25), s(40)]
[0.33, 0.3333, 0.33333333333333333333,
 0.33333333333333333333, 0.33333333333333333333]
```

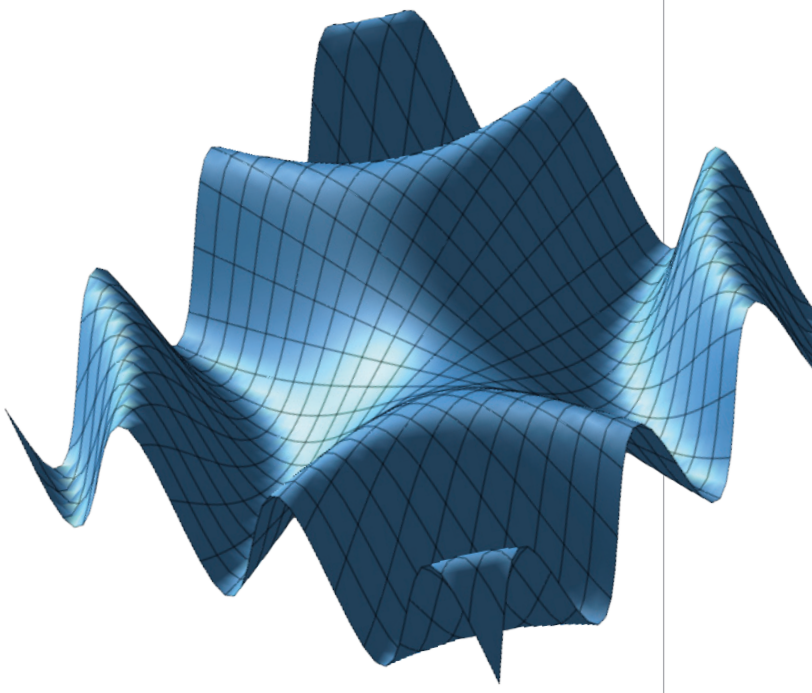
The same algorithm in Mathematica gives different results:

```
In[1]:=
s[i_] := s[i] = 2 * s[i - 1] - 3 * s[i - 1]^2;

In[2]:=
s[0] = SetAccuracy[3 / 10, 20]

Out[2]=
0.30000000000000000000

In[3]:=
{s[1], s[2], s[10], s[25], s[40]}

Out[3]=
{0.33000000000000000000, 0.33330000000000000000,
 0.33333333333333, 0.3333, 0. × 10^62}
```

The last term in the output says that $s_{40}=0.\times10^{62}$, which is not a good approximation of ⅓.

There is nothing in the computation to warn the user that the results may not be reliable at every step. For example, there is no accumulation of round-off error, which mathematicians and engineers are used to seeing as a warning sign that they may be getting problematic results.

The unexpected results of this computation are not caused by a bug, but are rather due to the floating point model used by Mathematica.

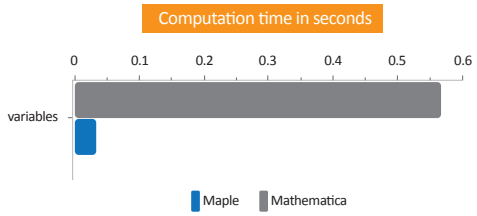


Performance

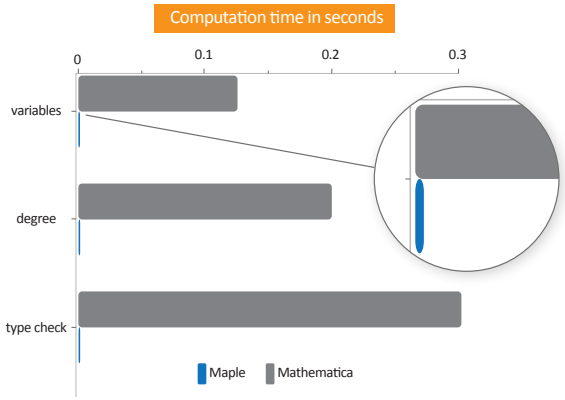
Maple is the world's fastest symbolic computation engine. The ability to perform mathematical computations symbolically lies at the heart of both Maple and Mathematica, and is what distinguishes them from most other computational software. Benchmarks comparing both products show that Maple is significantly faster when it comes to fundamental operations such as working with polynomials, with further improvements in every release. While polynomial operations may not seem like they will occur in many day-to-day applications, in fact they are at the core of almost all symbolic algorithms, such as solving equations exactly or computing an integral.

In the examples that follow, computations are performed on a 64-bit quad core Intel™ Core i7 920 2.66 GHz with Maple 17 and Mathematica 9.

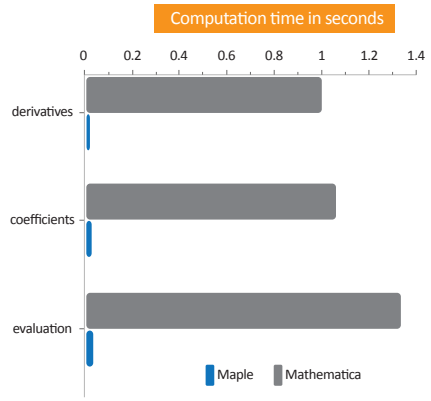
Expanding a polynomial with 170544 terms in Maple takes 31 milliseconds. In Mathematica, this takes 564 milliseconds.



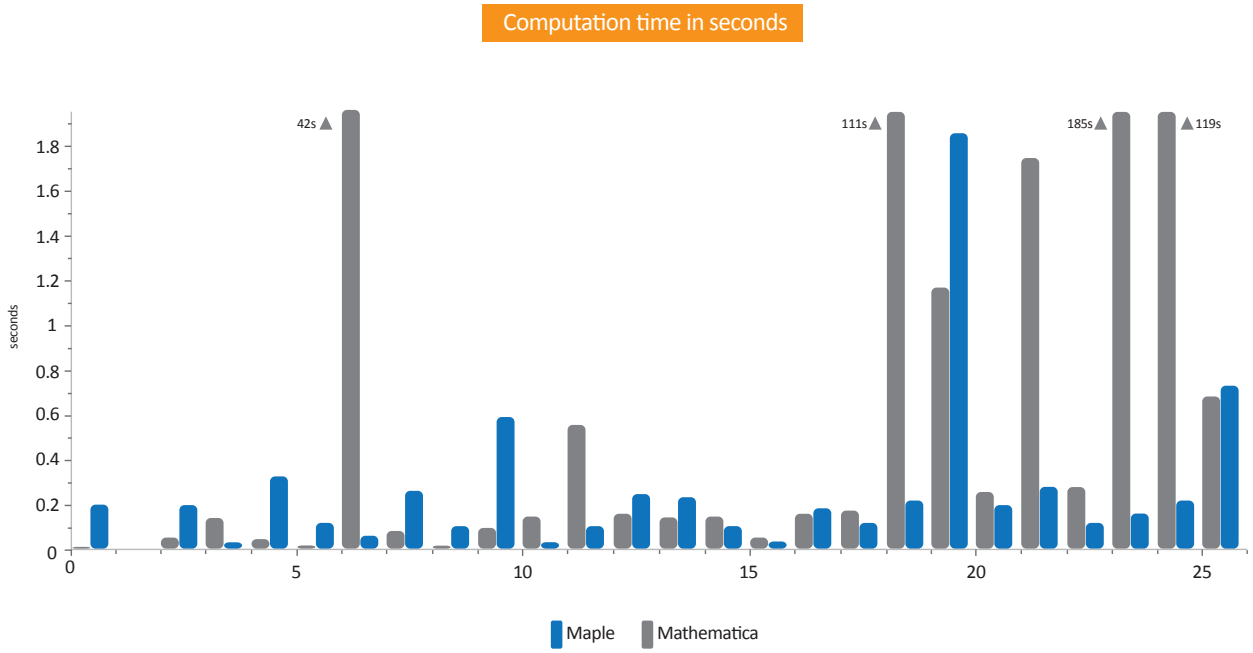
Common queries, such as finding the variables, computing the total degree, and testing for a polynomial, run much faster in Maple than in Mathematica.



Maple has developed high performance algorithms for the Maple kernel that speed up many core operations, including differentiation, extracting a coefficient, and evaluating a polynomial.



One situation in which faster fundamental operations makes a difference is solving equations symbolically. For example, this next example uses 50 small-sized polynomial systems from the polynomial system solving literature. Maple finds solutions to all 50 in less than 50 seconds, using its 'solve' command. Mathematica is able to solve only 26 of these systems, and was usually slower, sometimes much slower:



For each of the other 24 systems, the Mathematica 'Solve' command did not return a solution within 5 minutes and was halted.

Connectivity

Maple and Mathematica are both very powerful tools, but there are many reasons why they may not be the only tool you are using.

Maple provides connectivity with a wide variety of standard tools and languages, so you can take advantage of Maple's powerful mathematical environment no matter what other tools you use. The following describes some of the differences in connectivity options in the two products.

Code Generation

Code generation features let you convert expressions and programs to different programming languages. In this way, you can use Maple or Mathematica to develop the original solutions or algorithms and then export them to another language so you can use them as part of a larger project.

Maple can generate C, C#, Fortran, Java™, MATLAB, and Visual Basic® code. Options include automatic type deduction, automatic type coercion, reduction analysis of equations, and optimization of code. Maple's goal is to generate code that is easily integrated into your code base. For example, for maximum compatibility, the C code generated by Maple adheres to the ANSII C standard.

Mathematica has code generation for C and Fortran. The results cannot be used immediately, because the results are not standard C or Fortran code.

Consider the following example:

Maple VS **Mathematica**

$$CodeGeneration[C]\left(\sin(x) + \sqrt{\frac{x}{y}}\right)$$

```
cg0 = sin(x) + sqrt(x / y);
```

```
In[1]:=
CForm[Sin[x] + Sqrt[x/y]]
Out[1]//CForm=
Sqrt(x/y) + Sin(x)
```

Mathematica's CForm[] has converted the square brackets in Sin[] to round brackets, but retains the capitalization of the sine and square root functions. The C code generated by Mathematica then has to be compiled against a macro file (mdefs.h) to convert these Mathematica functions to their C equivalent. Even this will not always be sufficient, however, as can be seen in the next example.

Maple VS **Mathematica**

$$CodeGeneration[C](\operatorname{arctanh}(x))$$

```
cg1 = (log(0.1e1 + x) - log
(0.1e1 - x)) / 0.2e1;
```

```
In[2]:=
CForm[ArcTanh[x]]
Out[2]//CForm=
ArcTanh(x)
```

The function *arctanh(x)* is not part of the standard C math library. Maple automatically converts the *arctanh()* command to an equivalent form that can be evaluated in C. Mathematica leaves it as *arctanh()*, and so the Mathematica code will not run properly, even after compiling the code against the Mathematica macro file.

Generating Fortran code works the same way. Maple generates code that can be used immediately. Mathematica-generated code needs to be compiled against a separate definitions file and will not always produce working code. In addition, Maple takes into account restrictions imposed by Fortran compilers on line formatting and the length of variable names.

Maple VS **Mathematica**

$expr := longVariableName^2 + \sin(short) :$
$CodeGeneration:\text{-}Fortran(expr)$

```
Warning, the following variable name
replacements were made:
longVariableName -> cg0
      cg = cg0 ** 2 + sin(short)
```

Fortran variable name lengths are limited.

```
In[1]:=
        expr := longVariableName^2 + Sin[short];
In[2]:=
        FortranForm[expr]
Out[2]//FortranForm=
                  longVariableName**2 +
              -   Sin(short)
```

Fortran variable name lengths are not limited.

Because Mathematica does not take these factors into account, the code requires post-processing before it can be incorporated into your program.

MATLAB Connectivity

Maple offers a technical computing solution that is tightly integrated with MATLAB, providing direct access to all the commands, variables, and functions of each product while working in either environment. You can also translate MATLAB code to Maple and generate MATLAB code from Maple expressions and procedures.

Mathematica does not have any built-in connectivity to MATLAB. Third-party tools, not supported by Wolfram Research, offer the ability to call MATLAB functions from within Mathematica and do some MATLAB code generation. The code generation tool has not been updated in over 10 years. Two-way communication and code translation are not available.

CAD Connectivity

Maple provides a parametric two-way link into SolidWorks®, AutoDesk Inventor®, and NX® CAD systems, allowing you to retrieve parameters from a CAD drawing, perform analysis and optimizations, and send new values back into the design. Both an interactive assistant and a programming API are available to support active experimentation and the development of specialized tools for part reconfiguration and optimization.

Mathematica offers tools to export 3-D objects to CAD formats, but does not provide a live link between the two products. There is no way to retrieve parameters from a CAD diagram dynamically, and there is no way to push new parameter values directly into the CAD design.

Openness

The mathematical engines of both Maple and Mathematica have a similar architecture: a kernel written in C or C++ and a large library of predefined functions written in the Maple or the Mathematica programming language.

About 95% of Maple's functionality is written in the Maple programming language, and every Maple user can freely inspect the source code for any of these predefined Maple library routines. This is useful in determining what happens "under the hood" in Maple, which algorithms are being used, and for profiling purposes. The Maple debugger also allows a user to step through library routines for an in-depth look at exactly how the routine behaves for a given set of input. In fact, not only can Maple users see and step through library routines, but they can even modify or extend an existing Maple library routine in order to customize its functionality.

In Mathematica, the source code for all the predefined library routines written in the Mathematica programming language is hidden from the user. The source code is stored in .mx files in a proprietary binary format. Users cannot inspect the source code and they cannot step through such routines in the Mathematica debugger. Since the source code is not accessible, it is also impossible to customize the Mathematica library routines.

Comparisons in this document were made using Maple 17 and Mathematica 9.

> **"The open source aspect of Maple makes it much more useful when compared to other tools like Mathematica, and makes it easy to see many examples of advanced Maple programming. Maple typically covers a broader range of mathematics than Mathematica and uses traditional mathematical notation."**
>
> David Mazziotti - Professor, Department of Chemistry, James Franck Institute, The University of Chicago

Case Study

William Fox

Professor, Naval Post Graduate School

I teach a three course sequence in mathematical modeling in the Department of Defense Analysis at the Naval Post Graduate School. Maple helps me equip my students with problem-solving skills that are so essential for their professional lives, whichever career they choose.

I wasn't always a Maple user. I was introduced to Maple at the United States Military Academy, where I created and taught a nonlinear optimization course. I was using Mathematica initially, but I was not happy with it. Its functions were not robust enough and the system was, in my opinion, not user friendly. In addition, my students were struggling with Mathematica, and I felt that they were spending more time trying to understand the tool rather than the concepts. Other mathematics faculty were facing similar issues, and, as a department, we decided to change to Maple. Maple was an excellent program for our students. It was a rewarding decision because Maple is very easy to use, and my students loved it. I rewrote the entire lab using Maple, and that was very easy to do. Maple's friendly user interface and powerful features made the task effortless. I have been hooked on Maple ever since.

While I was the Chair of the Mathematics department at Francis Marion University, I created a mathematics computer lab and developed and taught a nonlinear optimization course. For this computer lab and course, I needed a solid mathematical software system that students could use to understand the topics deeply and get more involved in exploring mathematical concepts. I chose Maple! I worked with our computational physics professors to have them also use Maple.

I ended up running all of my course labs in Maple. These courses included mathematical modeling, linear algebra, differential equations, and both linear and nonlinear optimization. The success of these labs ultimately encouraged other faculty to start using Maple as well. In fact, I had the university purchase a Maple site license so Maple would be available to all our students.

I have seen Maple grow over the years, continuing to develop its powerful computation abilities and extremely intuitive, user-friendly interface. Today I go straight to Maple for most of my mathematical computations and analysis. I can explore the math more deeply and develop a more in-depth understanding. A most wonderful thing is that I can then pass on this magic of discovery to my students, with ease.

www.maplesoft.com

www.maplesoft.com I info@maplesoft.com
Toll-free: (US & Canada) 1-800-267-6583 | Direct:1-519-747-2373