SAGE for Undergraduates: How to Get Started

Gregory V. Bard

January 7, 2012

Chapter 1

Welcome to SAGE!

As the open-source and FREE competitor to expensive software like Maple, Mathematica, Magma and Matlab, SAGE offers anyone with access to a web-browser the ability to use cutting-edge mathematical software, and display one's results for others. This document is going to share with you several common mathematical tasks that are extremely easy, and which may serve as your starting point into SAGE. I'm sure that you will find SAGE far easier to use than a graphing calculator, and vastly more powerful.

Introduction

There is no need to read this entire document. If you are handed this guide as part of a class, then your professor will tell you which section numbers correspond with what you need to know. If you are just curious, then I recommend you just read the first 11 lessons, and then start playing around on your own. Using the find feature of your pdf viewer, you can always search for whatever command you would like to use. I recommend that you never try to read more than 3 entire lessons in one day. Otherwise there is too much for your brain to absorb while still keeping the experience fun and new.

I assume you do not already have an account on the SAGE notebook server. If you have one, then log in now, but if not, then see Appendix A, which will tell you how to get an account. In Appendix D are the directions for installing it on your personal computer, but this a very long and difficult process, and it is not recommended to beginners.

1.1 Using SAGE as a calculator

First off, you can always use SAGE as a simple calculator. For example, if you type `2+3` and click "evaluate" then you will learn the answer is 5.

The expressions can become as complicated as you like. To calculate the amount on a simple interest loan at 6% per year for 90 days, and principal \$ 900, we all know the formula to be $A = P(1 + rt)$ so we would just type in

```
900*(1+0.06*(90/365))
```

and click “evaluate”. You will learn that the answer is 913.315068493151, or \$ 913.32 after rounding to the nearest penny.

Notice that the symbol for addition is the plus sign, and the symbol for subtraction is the hyphen. The symbol for multiplication is the asterisk, and the symbol for division is the forward slash (that’s the one found with the question mark, not the backslash.)

For compound interest, we’d need to take exponents. The symbol for exponents is the caret, found above the number six on most keyboards. It looks like this “ˆ”.

Let’s consider 12% compounded annually on a signature loan for 3 years, and a principal of \$ 11,000. We all know the formula to be $A = P(1 + i)^n$, and so we would just type in

```
11,000*(1+0.12)^3
```

and click “evaluate”. You will learn that the answer is 15454.2080000000, or \$ 15,454.21 after rounding to the nearest penny. By the way, instead of clicking “evaluate,” you can just press shift-enter on the keyboard, and it will do the same thing.

For those who don’t like the caret, you can use two asterisks in a row.

```
11,000*(1+0.12)**3
```

which is actually a throw-back to the historical programming language `FORTRAN` which was most popular[1] during the 1970s and 1980s.

Warning: It is very important not to enter a comma in large numbers when using SAGE. You have to type `11000` and not `11,000`

What if you make a mistake? Let’s say that I really meant to say \$ 13,000 and not \$ 11,000. Click on the mistake, and correct it using the keyboard, and change the 11 into 13. Now click “evaluate” again, and all is well.

There’s also an “Undo” button and it does exactly what you think it does. However, if you’ve done a lot of work, you have to scroll the window all the way to the top of the document to find it, as this button sits at the top of the sheet.

The Red Line: You may have noticed a vertical red line next to the box after you had begun to change the formula, but before you had clicked “evaluate.” After you have clicked “evaluate,” the red line vanished. The purpose of this red line is to tell you that the information in the box had changed since the last time “evaluate” had been clicked, and so you need to hit “evaluate” again to get an accurate answer.

The Green Rectangle: Also, you may have noticed (if you have a slow internet connection in particular) a green rectangle that is very thin under the box where you just typed, and next to the red line. This is to indicate that your web-browser is connecting to the SAGE server, and has made its request, and waiting for the answer. Do not hit “evaluate” again. On fast connections, the green rectangle will appear and disappear very rapidly, and you probably will not see it.

Note, if you have a very slow connection, then see “The Moving Slash” on Page 70.

Saving or Printing your Work: We will go into more detail about this later, but for now notice the “Save” and “Save & Quit” buttons, as well as the “Discard & Quit” button. These do exactly what you think they do. You can also use the “Print” button to print your work.

Like the undo button, if you’ve done a lot of work, you’ll have to scroll the window all the way to the top of the document to find it, as this button sits at the top of the sheet.

[1] An irrelevant historical aside: Because rewriting software is a painful and time-consuming process, many important pieces of scientific software remained written in `FORTRAN` until a few years ago, and likewise the same is true of business software in `COBOL`. Each new programming language incorporates features of the previous generation of languages, to make it easier to learn, and accordingly one sees some truly ancient seeds of dead languages once in a while.

Shortcut: While I mentioned it before, if you are tired of clicking "evaluate" all the time, you can just press Shift and Enter at the same time.

The Underscore: blah: Coming Soon! (I can't believe I forgot to mention this!!!!)

Grouping Symbols When there are multiple sets of parentheses in a formula, sometimes mathematicians use brackets as a type of "super parentheses." As it turns out, SAGE needs the brackets for other things, like lists, and so you have to always use parentheses for grouping inside of formulas.

For example, let's say you need to evaluate

$$550\frac{\left[1-(1+0.05)^{-30}\right]}{0.05}$$

So you should not type

```
550 [ 1 - (1+0.05)^(-30) ]/0.05
```

but rather

```
550 ( 1 - (1+0.05)^(-30) )/0.05
```

where the brackets have become parentheses.

Some very old math books use braces { and } as a sort of auxiliary also to the parentheses and brackets. These too, if they are for grouping in a formula, must become parentheses. As it turns out, SAGE, as well as modern mathematical books, use the braces { and } to denote sets.

By the way, the above formula was not artificial. It is the value of a loan at 5% compounded annually for 30 years, with an annual payment of $ 550. The formula is called "the present value of an annuity."

1.2 Using SAGE with Common Functions

Now I'll discuss how SAGE works with square roots, logarithms, exponentials, and so forth.

Square Roots: The standard "high school" functions are also built-in. For example, type

```
sqrt(144)
```

then click "evaluate" and you'll learn that the answer is 12. From now on, I'm not going to say "click evaluate", because repeating that might get tiresome; I'll assume that you know you have to do that. Since SAGE likes exact answers, try

```
sqrt(8)
```

and get `2*sqrt(2)` as your answer. If you really need a decimal, try instead

```
N( sqrt(8) )
```

and obtain

```
2.82842712475.
```

which is a decimal approximation.

The `N()` function, which can also be written `n()`, will convert what is inside the parentheses to a real number, if possible. Usually that is a decimal expansion, and so unless what is inside the parentheses is an integer[2] then it will be, necessarily, an approximation. SAGE will assume that all decimals are mere approximations, so for example

```
sqrt(3.4)
```

will evaluate to `1.84390889145858`, without need of using `N()`.

[2]To be mathematically correct, I should say "an integer or a fraction with denominator writable as a product of a power of 5 and a power of 2." For example, 25ths, 16ths, and 80ths can be written exactly with decimals, where as 3rds, 15ths, and 14th cannot. Observe that $25 = 5^2$ and $16 = 2^4$ as well as $80 = 2^4 \times 5$. Those denominators have only 2s and 5s in their prime factorization. Meanwhile, $3 = 3$ and $3 = 5 \times 3$ while $14 = 7 \times 2$. As you can see, those denominators have primes other than 2 and 5 in their prime factorization. If you find this interesting, you should read "Using SAGE to work with Integers" on Page 47.

Higher Order Roots: Higher order roots can be calculated like exponents are. For example, to find the sixth root of 64, do

```
64^(1/6)
```

and obtain that the answer is 2.

Getting Both Square Roots: In the spirit of absolute pomposity, you may know that $(-2)^2 = 4$ as well as $2^2 = 4$. So if you want *all* the square roots of a number you can do

```
sqrt(4,all=True)
```

to obtain

```
[2, -2]
```

where the square brackets indicate a list.

A Taste of the Complex Numbers If you know about complex numbers, you can do

```
sqrt(-4,all=True)
```

to obtain

```
[2*I, -2*I]
```

where the capital letter I represents $\sqrt{-1}$, the imaginary constant. You might be interested to know that you can use the lowercase `i` instead of the capital one, if you prefer. While SAGE is quite good at complex analysis, and can produce some rather lovely plots of functions of a complex variable, we will not go into those details in this document.

Special Constants Pi and e: We've learned that the imaginary constant is capital "I". It turns out that π is built in as "pi" (both letters lower case) and e is built in as "e" (again, lower case).

You can do things like

```
numerical_approx(pi, prec=200)
```

to get many digits of pi, or likewise

```
numerical_approx(sqrt(2), prec=200)
```

to get a high-accuracy expansion of $\sqrt{2}$. These expansions are to 200 bits, and basically if you want 100 digits, you can do 332 bits; 200 digits[3] is 664 bits, and so on. You can also abbreviate with

```
n(sqrt(2), prec=200)
```

as we did earlier. That's what the `N()` or `n()` function (they are the same) is an abbreviation of, namely "numerical approx."

Is SAGE Case-Sensitive? You've probably had a situation in life where you entered your password correctly, but discovered that it was rejected because you had the wrong capitalization. For example, perhaps you've left the "CAPS-LOCK" key on. In any event, SAGE does think of `Pi` as different from `pi` and `Sin` as different from `sin`. The only exception is that you can represent $\sqrt{-1}$ as either `i` or `I` and either way SAGE will know what you meant. But that leniency only applies to $\sqrt{-1}$ and in all other cases, capitalization matters.

Tab Completion: Now would be a good time to explain a neat feature of SAGE. If you are typing a long command, like "numerical_approx" and part way through you forget the exact ending (is it approximation? approximations? appr?) then you can hit the tab button. If only one command in the entire library has that prefix, then SAGE will fill in the rest for you. If it does not, then you have to enter a few more letters. On a slow internet connection, the pause can be very noticeable, but it is still a useful feature when you cannot remember exact commands.

[3]This turns out to be because $10^{200} \approx 2^{664}$, if you are curious.

Exponentials: Just as

```
2^3
```

gives you 2^3 and likewise

```
3^3
```

gives you 3^3, if you want to say e^3 then just say

```
e^3
```

and that's fine. Or for a decimal approximation you can do

```
N(e^3)
```

Also, it is worth mentioning sometimes books will write

$$exp(5 \cdot 11 + 8) \text{ instead of } e^{5 \cdot 11+8}$$

and SAGE thinks that's just fine. For example,

```
exp(5*11+8) - e^(5*11+8)
```

evaluates to 0. Don't forget the asterisk between the 5 and the 11.

Logarithms Of course, SAGE knows about logarithms. But there's a problem. There are several types of logarithm, including the common logarithm, the natural logarithm, the binary logarithm, and the logarithm to any other base you feel like using. In high school and middle school, "log" refers to the common logarithm, and "ln" to the natural logarithm. However, during and after calculus, "log" refers to the natural logarithm. Since SAGE was mainly meant for university-and-higher level work, then it is only natural they chose to use the natural logarithm for "log."

So to find the natural logarithm of 100 type

```
N( log(100) )
```

for the common logarithm of 100 type

```
N( log(100,10) )
```

and for the binary logarithm of 100 type

```
N( log(100,2) )
```

which of course generalizes. To find the logarithm of 100 base 42, type

```
N( log(100,42) )
```

Note that SAGE is quite good at getting exact answers. Try

```
log ( sqrt(100^3) ,10)
```

and you will obtain 3, the exact answer.

Chemistry Example: Coming Soon! An example of common log with pH

Finance Example: Coming Soon! An example of natural log for getting the number of compounding periods or length of time.

1.3 Using SAGE for trigonometry.

For trigonometry, SAGE works in radians. So if you want to know the sine of $\pi/3$, you should type

```
sin(pi/3)
```

and you will get the answer `1/2*sqrt(3)`. This is an exact answer, rather than a mere decimal approximation. You will find that SAGE is very oriented toward exact rather than approximate answers. Sometimes this is irritating, because if you ask for the cosine of $\pi/12$, then you would type

```
cos(pi/12)
```

and obtain `1/12*(sqrt(3) + 3)*sqrt(6)` which is especially irritating if you want a decimal. Instead, if you type

```
N(cos(pi/12))
```

then you will obtain `0.965925826289`, a rather good decimal approximation.

You will discover that SAGE is fairly savvy when it comes to knowing when functions will go wrong. In particular, just try evaluating tangent at one of its asymptotes. For example,

```
tan(pi/2)
```

will produce the helpful answer "Infinity." The "rare" or "reciprocal" trigonometric functions of cotangent, secant, and cosecant, which are important in calculus but annoying on hand-held calculators, are built into SAGE. They are identified as `cot`, `sec`, and `csc`.

The inverse trigonometric functions are also available. They are used just like the trigonometric functions. For example, if you type

```
arcsin(1/2)
```

you will obtain

```
1/6*pi
```

as expected. Likewise

```
arccos(1/2)
```

produces

```
1/3*pi
```

The usual abbreviations are all known by and used by SAGE. Here is a complete list:

Math Notation	Long-form SAGE command	Short-form SAGE command
$\sin^{-1} x$	`arcsin(x)`	`asin(x)`
$\cos^{-1} x$	`arcsin(x)`	`acos(x)`
$\tan^{-1} x$	`arcsin(x)`	`atan(x)`
$\cot^{-1} x$	`arccot(x)`	`acot(x)`
$\sec^{-1} x$	`arcsec(x)`	`asec(x)`
$\csc^{-1} x$	`arccsc(x)`	`acsc(x)`

You can also use SAGE to graph the trigonometric functions. We'll do that in the section entitled "Using SAGE to graph 2-dimensionally," on Page 9

Converting to Degrees, and Back: Remember, to covert radians to degrees, just multiply by $180/\pi$, and to multiply from degrees to radians, just multiply by $\pi/180$. Here's what you type, to convert $\pi/3$ to degrees:

```
(pi/3) * (180/pi)
```

produces

```
60
```

while

```
60 * (pi/180)
```

produces

```
1/3*pi
```

The way I like to remember this is that one protractor is 180 degrees, and protractor begins with "p," while π is the Greek "p." For example, 90 degrees is half a protractor, and 60 degrees is a quarter protractor.

That's why I can remember that they are $\pi/2$ and $\pi/2$. Likewise, if I see $pi/6$, then I know that this is one-sixth of a protractor or 30 degrees.

There is actually an elaborate package in SAGE for converting among different units, and it can be quite useful. I hope to add it to this document in the near future.

1.4 Using SAGE to graph 2-dimensionally.

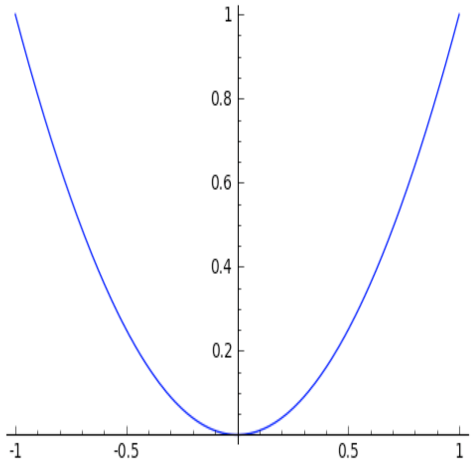
My favorite shape is the parabola, so let's start there. Type

```
plot(x^2)
```

and you get a lovely plot of a parabola going in the range

$$-1 < x < 1$$

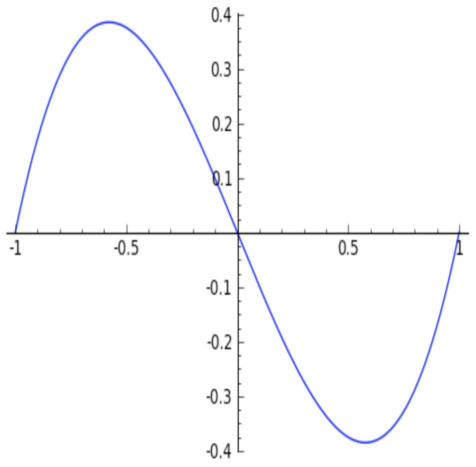
which is the default range. It will bound the y-values to whatever is needed to show those x-values. Here's what you will get:



Likewise you can do

```
plot(x^3-x)
```

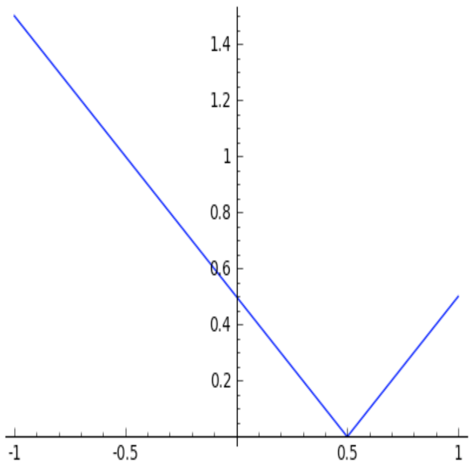
which is nice and visually appealing, as you can see:



For $|x - 1/2|$ you can do

```
plot(abs(x-1/2))
```

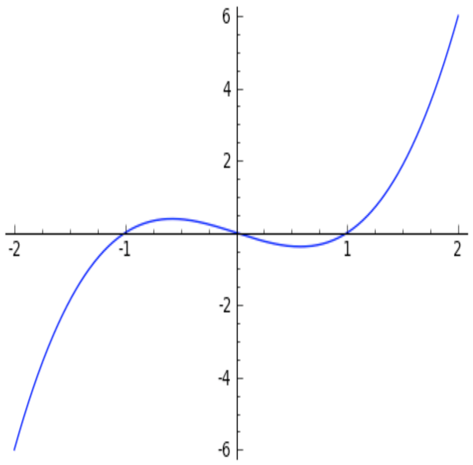
which can come up from time to time. This produces



But what if you wanted a different x range? For example, to graph in $-2 < x < 2$ you would type

```
plot(x^3-x, -2, 2)
```

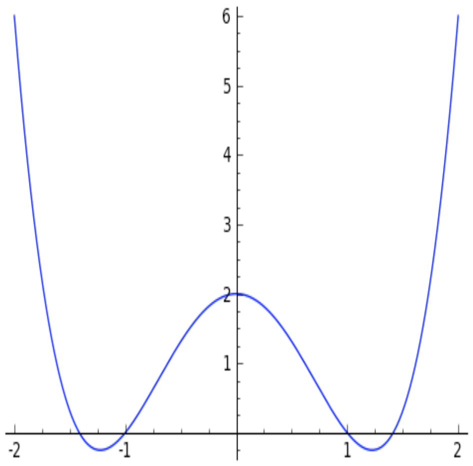
and you get the desired graph, namely:



A very cool graph is

```
plot(x^4 - 3*x^2+2,-2,2)
```

but notice the asterisk between 3 and x in " `3*x` ". You will get an error if you leave that out! The plot is



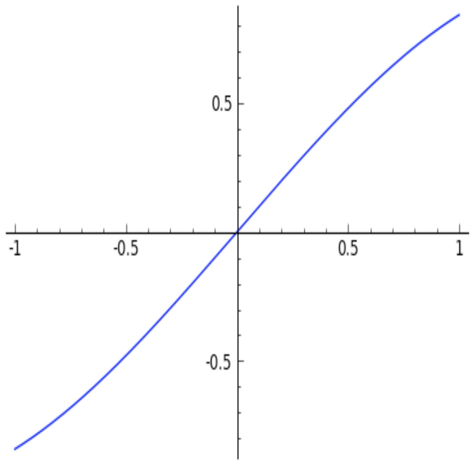
Another minor sticking point is that for $y = \sin(t)$ you have to say

`plot(sin(x))`

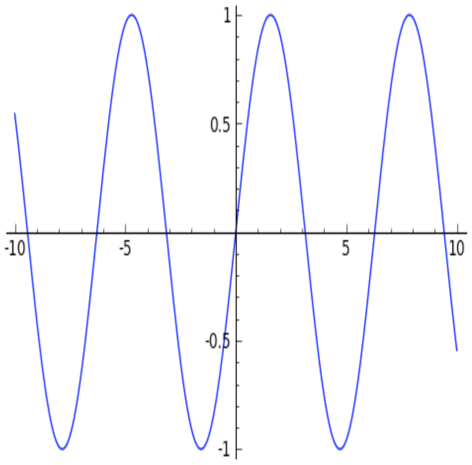
or better yet

`plot(sin(x),-10,10)`

because plot is not expecting t, it expects x. The images that you get are



`plot(sin(x))`



`plot(sin(x),-10,10)`

In fact if you were to type

`plot(sin(t))`

you would see

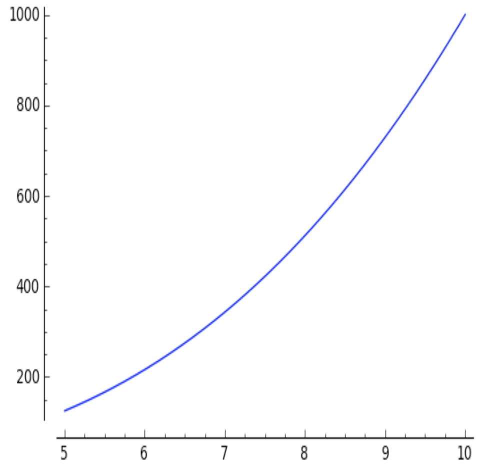
```
Traceback (click to the left of this block for traceback)
...
NameError: name 't' is not defined
```

because in this case, SAGE does not know what 't' means. An alternative way of handling this is given on Page 26.

Going "Off the Scale": Consider the plot of x^3 from $x = 5$ to $x = 10$, which is given by

```
 plot(x^3, 5, 10)
```

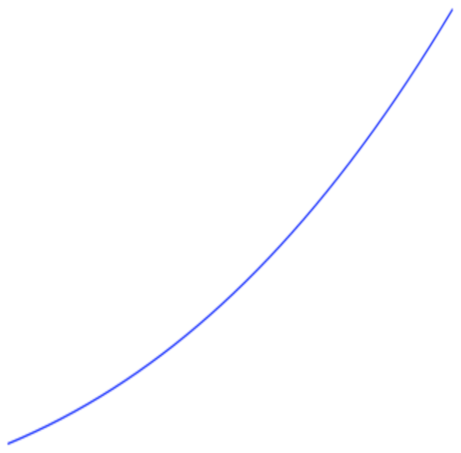
and as you can imagine, the function goes from $5^3 = 125$ up to $10^3 = 1000$, and so the origin should appear very far below the graph. This is the plot:



Your hint that the location of the x-axis in the display is not where it would be normally is that the axes do not intersect. This is to tell you that the origin is far away. When the axes do intersect on the screen, then the origin is (both in truth and on the screen) where they intersect. Also, if you want to, you can hide the axes with

```
plot(x^3, 5, 10, axes=false)
```

and that produces

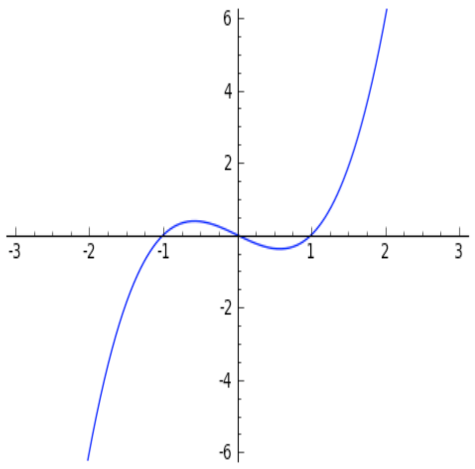


which is considerably less informative.

To Force the Y-Range of a Graph: If, for some reason, you want to force the y-range of a graph to be constrained between two values, you can do that. For example, to keep $-6 < y < 6$, we can do

```
plot( x^3-x, -3, 3, ymin = -6, ymax = 6)
```

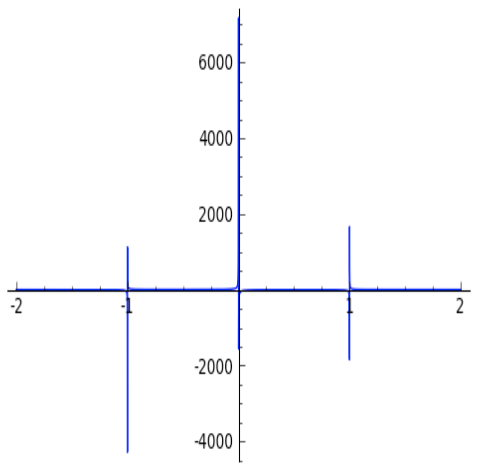
in order to obtain



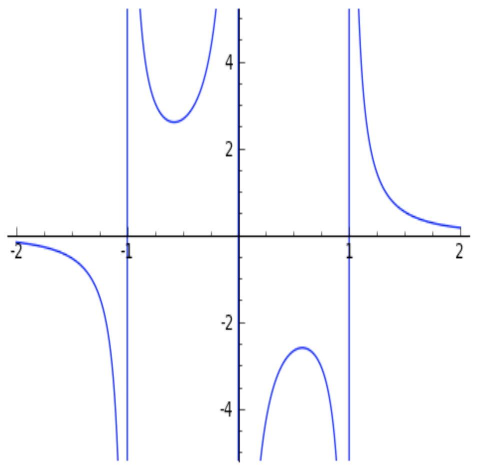
This is important, because normally SAGE wants to show you the entire graph. Therefore, it will make sure that the y-axis is tall enough to include every point, from the maximum to the minimum. For some functions, either the maximum or the minimum or both, could be huge.

Plots of Functions with Asymptotes: The way that SAGE (and all computer algebra tools) computes the plot of a function is by generating a very large number of points in the interval of the xs, and evaluating the function at each of those points. To be precise, if you want to graph $f(x) = 1/x^2$ between $x = -4$ and $x = 4$, the computer might pick 10,000 random values of x between -4 and 4 find the y values by plugging them into $f(x)$ and then finally drawing the dots in the appropriate spots on the graph.

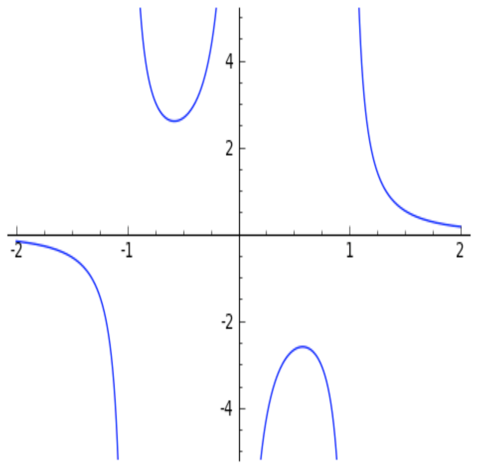
So if the graph has a vertical asymptote, then near that asymptote, the value will be huge. Because of this, when you graph a rational function, be sure to restrict the y-values. Compare:



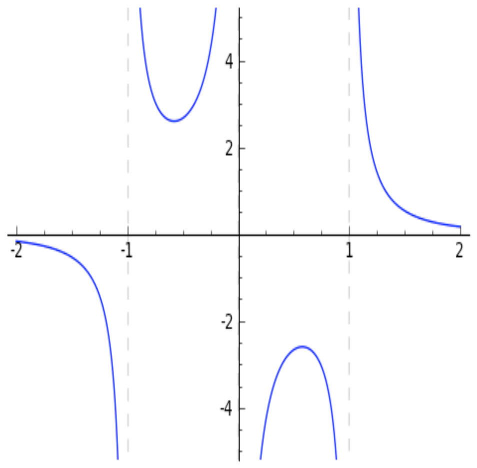
Plot 1



Plot 2



Plot 3



Plot 4

Plot 1: `plot(1/(x^3-x), -2, 2)`

Plot 2: `plot(1/(x^3-x), -2, 2, ymin = -5, ymax = 5)`

Plot 3: `plot(1/(x^3-x), (x,-2, 2),detect_poles=true, ymin = -5, ymax = 5)`

Plot 4: `plot(1/(x^3-x), (x,-2, 2),detect_poles='show', ymin = -5, ymax = 5)`

As you can see, the first is a disaster. The second one cuts off the very-high and very-low y-values, but it keeps trying to connect the various "limbs" of the graph. When you set "detect poles" to true, then it will figure out that the pieces are not connected.

One minor point. Did you notice in the four plots above, that `'show'` is in quotes, but `true` is not in quotes? That's because `true` and `false` occur so often in math, that they are built-in keywords in SAGE. However, the `'show'` occurs less often, and therefore is not built in, and we must put it in quotes, just as we do with colors while plotting.

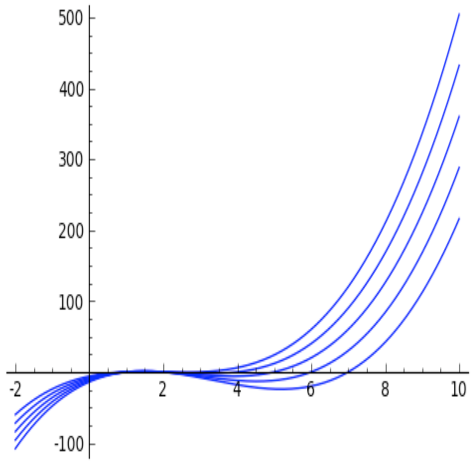
Multiple Graphs at Once: Now we're going to see how to superimpose plots on each other, using a plus sign. Suppose you were investigating polynomials, and you wanted to know the effect of varying the last numeral in

$$y = (x-1)(x-2)(x-5)$$

on its graph. Then you could consider looking at $(x-3)$ and $(x-4)$ as replacements for the $(x-5)$, as well as $(x-6)$ and $(x-7)$. This would be done by

```
 plot( (x-1)*(x-2)*(x-3), -2, 10) + plot( (x-1)*(x-2)*(x-4), -2, 10) + plot(
(x-1)*(x-2)*(x-5), -2, 10)
```

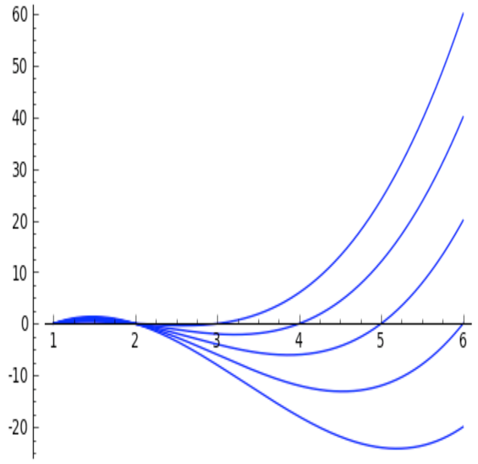
where I chose the domain, $x = -2$ until $x = 10$, arbitrarily. Now we obtain



It seems that I should zoom in a bit, to see more detail. So I type

```
 plot( (x-1)*(x-2)*(x-3), 1, 6) + plot( (x-1)*(x-2)*(x-4), 1, 6) + plot(
(x-1)*(x-2)*(x-5), 1, 6)
```

and obtain as a result



Now I can see better what that last numeral does. It controls the location, or x-coordinate, of the last time that the curve crosses the x-axis. As you can see, the plus sign among the `plot` commands means to superimpose them.

There is a technicality here. You must not break a newline among the plots and plus signs. The various plot commands and their plus signs must be strung together, like a paragraph, wrapping naturally from one line to the next. You do not use the enter key to put each plot on its own line. Thus, you cannot do:

```
 plot( (x-1)*(x-2)*(x-3), 1, 6)
+ plot( (x-1)*(x-2)*(x-4), 1, 6)
+ plot( (x-1)*(x-2)*(x-5), 1, 6)
+ plot( (x-1)*(x-2)*(x-6), 1, 6)
+ plot( (x-1)*(x-2)*(x-7), 1, 6)
```

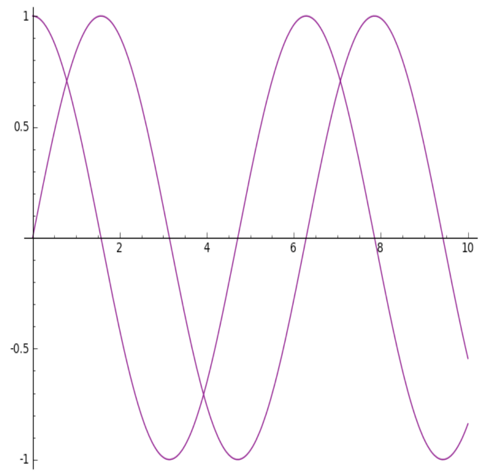
which SAGE thinks is five separate commands, instead of one big command.

Therefore, if you have problems getting those last two commands to work, then check to see that you have no accidental "line breaks" in the command. For SAGE, these commands must be entered without a line break, kind of like typing a long paragraph in an ordinary word processor—you can wrap around the end of the line, but you cannot insert a line break.

Another cool one is

```
plot([sin(x),cos(x)], 0, 10, color='purple')
```

producing



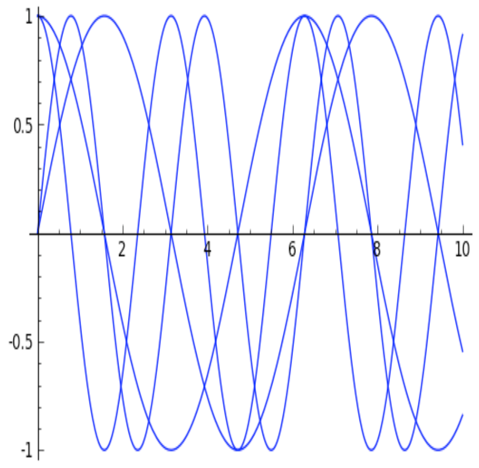
which looks like DNA to me. As you can see by that, you can plot multiple functions at the same time on the same graph. To do that, the list of functions should be separated by commas, and enclosed in brackets.

This is an example of a list in SAGE. You can enclose any data with `[` and `]`, and separate the entries with commas, to make a list. This is our first example, but we'll see another one, on Page 29, in solving equations; and in Number Theory on Page 48.

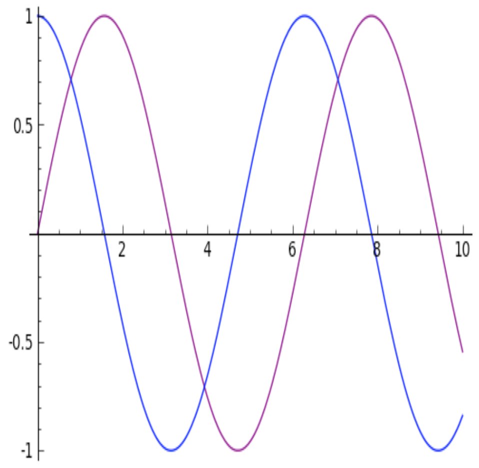
However, it is important not to get carried away. Rarely does it make sense for four functions to appear together in one graph. For example

```
plot([sin(x),cos(x),sin(2*x),cos(2*x)], 0, 10)
```

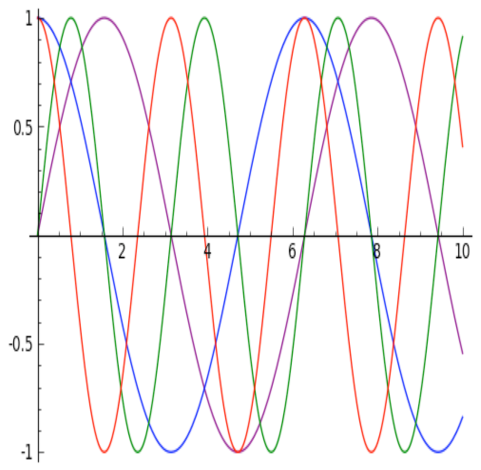
makes a total mess, as you can see



To do multiple functions in multiple colors, the command is actually to add the plots:
`plot(sin(x), 0, 10, color='purple') + plot(cos(x), 0, 10, color='blue')`
which produces something quite readable:

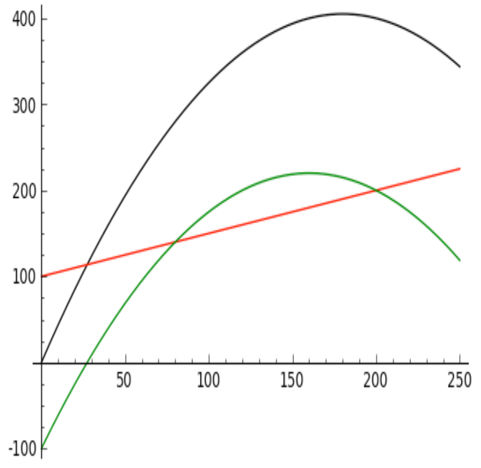


Or perhaps
```
plot(sin(x), 0, 10, color='purple') + plot(cos(x), 0, 10, color='blue') + plot(sin(2*x),
0, 10, color='green') + plot(cos(2*x), 0, 10, color='red')
```
which is not readable but certainly is very pretty, as you can see:



Which Color on Top? Rarely but occasionally one must be concerned with which color goes on top during a set of plots that are superimposed. The criterion in SAGE is simple. If two or more curves overlap at any point, the one given last (furthest to the right) in the sequence of plots goes on top. Sometimes this is important in getting the plot to look as you'd like it to.

A Micro-Economic Example Coming Soon! The ice cream problem, but with $ 100 in start-up costs. In any case, here's the graph:



Further Graphing & Plotting Topics The following additional topics on graphing and plotting can be found in the section "Advanced Topics in Plotting and Graphing" on Page 52. They include

- Labels and Legends.
- Grids and Graphing Calculator-Style Graphs.
- Labeling the Axes of Plots.
- Graphs of Hyperactive Functions.
- Odd Roots of Negative Numbers.
- Restricting the X-Range of Plots.

- Graphing Inequalities.

1.5 Making your Own Functions in SAGE

It is extremely easy to define your own functions in SAGE. To me, this is one of the most brilliant design features of SAGE, that the creators made this very easy for the user. You can make your own function using exactly the symbols that you would normally use in writing the function down with your pencil.

Defining your own Functions: If you want to define $f(x) = x^3 - x$ then you type

```
f(x) = x^3-x
```

and likewise if you wanted to define $g(x) = \sqrt{1-x^2}$ you would type

```
g(x) = sqrt( 1-x^2 )
```

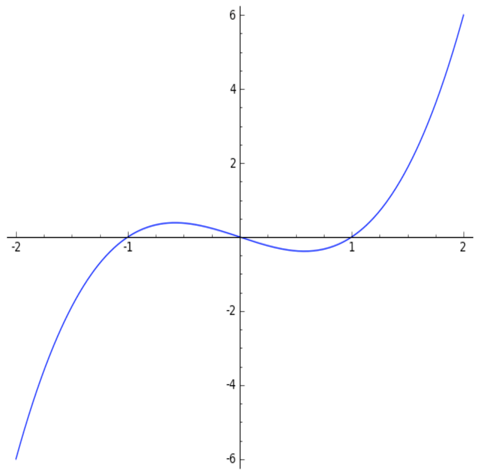
which can be very handy.

Take a moment and see that $f(0)$ and $f(2)$ as well as $g(0)$ do exactly what you expect them to do.

Plotting Functions: Now that we've defined $f(x)$ and $g(x)$, we can plot them with

```
plot( f, -2, 2 )
```

you get the expected plot



Likewise if you type

```
plot( g, 0, 1 )
```

which is equivalent to

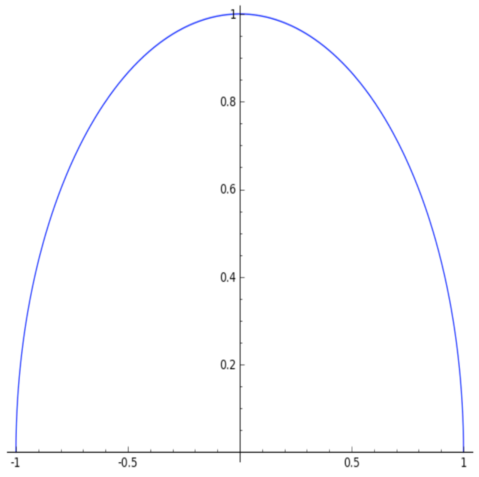
```
plot( g(x), 0, 1 )
```

then you get the graph of $g(x)$ on $0 \leq x \leq 1$.

Last but not least, if you type

```
plot( g )
```

which is rather abbreviated—omitting the interval of the x-axis that you want—then SAGE will assume that you want $-1 \leq x \leq 1$, by default. This produces the plot



Using Intermediate Variables: Sometimes you'll be using the same value a lot, and you want to store it somewhere. For example, if you type 355/113 a lot, then you can type

```
c=335/113
```

which silently stores the value 335/113 in the variable c. You can use it anytime, such as

```
2+c
```

which evaluates to 561/113. If you go back and change the value of c however, you have to make sure you use the "evaluate" button to re-evaluate all other boxes that make use of c. In general, it is easy to avoid this by not changing intermediate variables once you set them. You can also just tell SAGE to re-evaluate every single box, in order. This is described on Page 71.

If we wanted to find the relative error of c as an approximation of π we would remember the formula

$$\text{relative error} = \frac{\text{approximation} - \text{truth}}{\text{truth}}$$

and type

```
N((c-pi)/pi)
```

and learn that the relative error is around 84 parts per billion. Not bad. This approximation was found by the Chinese Astronomer Zu Chongzhi, also known as Tsu Ch'ung-Chih. We can compare 335/113 to 22/7 with

```
N((22/7-pi)/pi)
```

and learn that this much more common approximation of π as 22/7 has a relative error around 402 parts per million. Much worse!

The intermediate variables technique is great for things like the strength of gravity in physics, or the speed of light, which (so far as we know) do not change. For things that do change, try "substitution", which I'll tell you about in just a moment. In finance, you could use this for interest rates or the coefficients in a cost function.

1.6 Using SAGE to Manipulate Polynomials

Coming Soon!

1.7 Using SAGE to graph a 3-dimensional function, or an implicit function.

Three-dimensional images are not only visually stimulating, but they can help demonstrate a lot of important effects in the multivariate calculus. Instead of $f(x)$, we will have functions of the form $f(x, y)$. Before, we had only one variable x, and now we're adding in the variable y.

This means we need the variable y, so we first type

```
 y = var('y')
```

and now y is declared. The declaration of variables will be explained on Page 28, in case you are curious, but for now just type it. In general, you need to declare every variable that you use, except that x is so commonly used so it is built in.

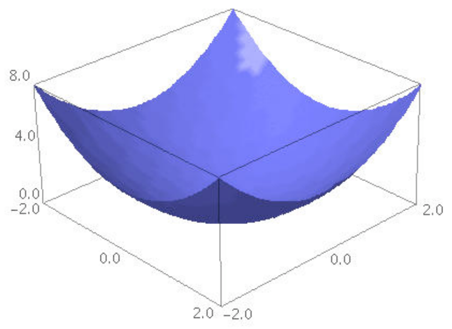
If we wanted to view the bottom part of the paraboloid

$$z = x^2 + y^2$$

with x and y both ranging from -2 to 2, we would type

```
 y=var('y')
plot3d( x^2 + y^2, (x, -2, 2), (y, -2, 2) )
```

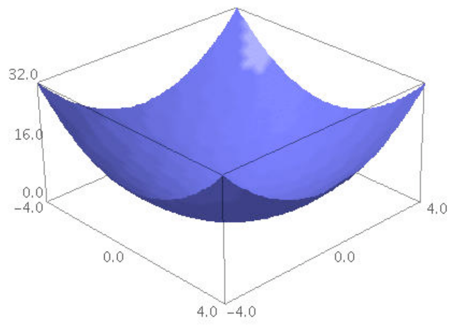
to obtain



Try that, and try dragging the plot around with the mouse. You can view the plot from all sorts of angles and directions. Now we can view things from -4 to 4 instead with

```
 plot3d( x^2 + y^2, (x, -4, 4), (y, -4, 4) )
```

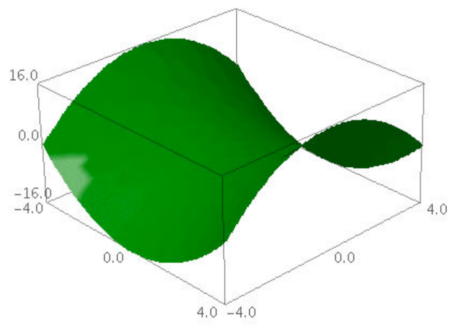
to obtain



For a hyperbola of one sheet, try

```
plot3d( x^2 - y^2, (x, -2, 2), (y, -2, 2), color='green' )
```

where you can see that we changed the color, just like we did in the previous section on 2-dimensional plots. This results in

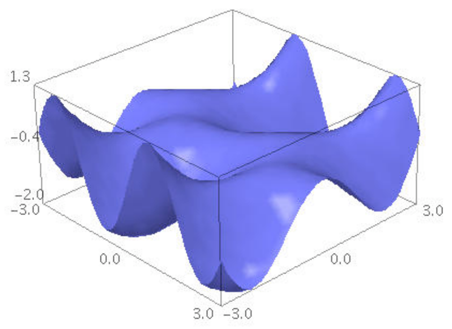


Note, that we do not type `y=var('y')` again, because you only have to declare each variable once. Once you have declared it, you do not have to declare it a second time.

And for a surface that is just plain weird, try

```
plot3d(sin(x-y)*y*cos(x),  (x,-3,3),  (y,-3,3)  )
```

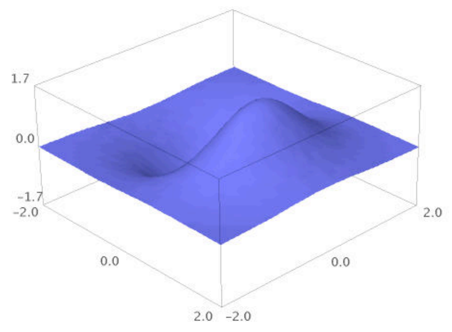
which creates



This is an example of a surface that you really do have to see from many angles to understand. Try dragging it around with the mouse to see what it looks like from many angles. Another neat one is

```
plot3d( 4*x*exp(-x^2-y^2), (x,-2,2), (y,-2,2))
```

which creates

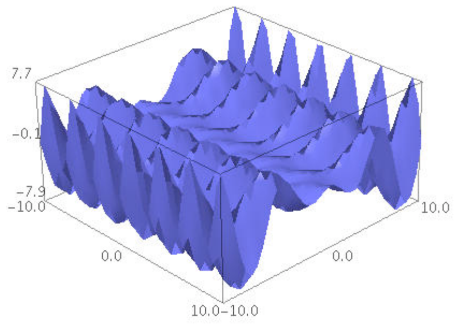


Clearly, this too is a surface that can only be understood by rotating the object and examining it from several angles.

Questions of Resolution Density If instead you were to type

```
plot3d(sin(x-y)*y*cos(x),  (x,-10,10),  (y,-10,10)  )
```

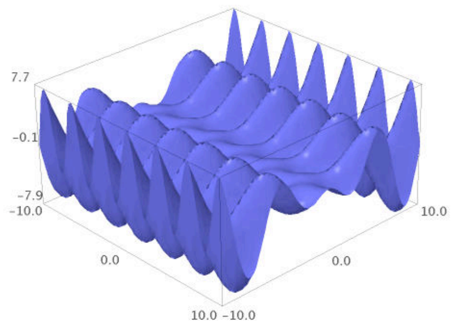
you get a much rougher, irregular surface that does not seem to properly resemble the previous. After all, we didn't change the function, and we just went from the 6×6 region about the origin to the 20×20 region about the origin. Have a look:



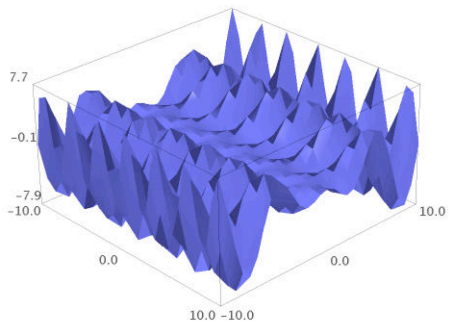
The issue comes down to resolution. The plot3d command by default chooses a certain number of points, by some method, and samples the function at those points. Whatever method it uses—which is not known to me—produced enough points for a good 6×6 picture, but not enough for a good 20×20 picture. The situation is much improved by typing

```
plot3d(sin(x-y)*y*cos(x),  (x,-10,10),  (y,-10,10),  plot_points=90 )
```

which forces SAGE to use 90 points on the x-axis, and 90 points on the y-axis, for a total of 8100 points. To exaggerate the effect for the purposes of explanation, consider the following images



with `plot_points=90`
(that's 8100 points)



with `plot_points=30`
(that's 900 points)

Perhaps you might be wondering why we do not always use "a large number" of points. Well, even with 90, you might find that your internet connection slows down, and your computer starts acting slowly, unless you have a large amount of memory. These 3D-plots require an enormous amount of computation power, especially if you rotate them.

Implicit Functions Sometimes, in calculus texts, a function is defined implicitly, instead of explicitly. What I mean is that instead of $z = f(x, y)$ one could have some polynomial, for instance, in x, y, and z equal to zero. An example might be

$$4x^2(x^2 + y^2 + z^2 + z) + y^2(y^2 + z^2 - 1) = 0$$

This should not be too surprising. After all, while most functions in the univariate calculus are defined as $y = f(x)$, on the other hand, we write a circle like $x^2 + y^2 = 4$, or alternatively $x^2 + y^2 - 4 = 0$ as a polynomial in x and y, set equal to zero.

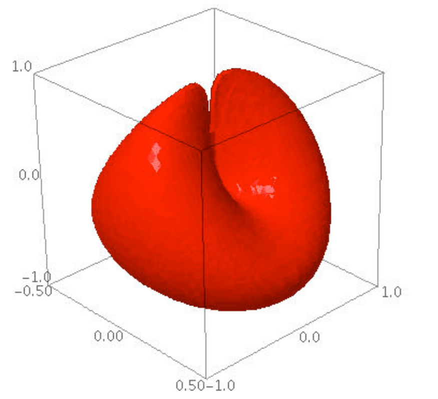
In any case, here's what you would do in SAGE. First, we define the function that we wish to draw:

```
f(x,y,z) = 4*x^2 *(x^2+y^2+z^2+z) + y^2*(y^2+z^2-1)
```

Now the following command will plot the set of points where $f(x,y,z) = 0$, and the variables range over the values $-1/2 < x < 1/2$ and $-1 < y < 1$ as well as $-1 < z < 1$.

```
implicit_plot3d( f, (x, -0.5, 0.5), (y, -1, 1), (z, -1, 1), color = 'red' )
```

That produces this:



Before we wrap up our discussion of 3D plots, it is important to point out that if you were to use `plot_points=100` in the above command, then the x-interval would be divided into 100 parts, as well as the y and z intervals, for a total of

$$100 \times 100 \times 100 = 1,000,000$$

plotting points. This is extremely unwise, and can crash your web browser.

Backwards Compatibility The following syntax makes the `plot` command look like the `plot3d` command's syntax:

```
plot( x^2, (x, -2, 2) )
```

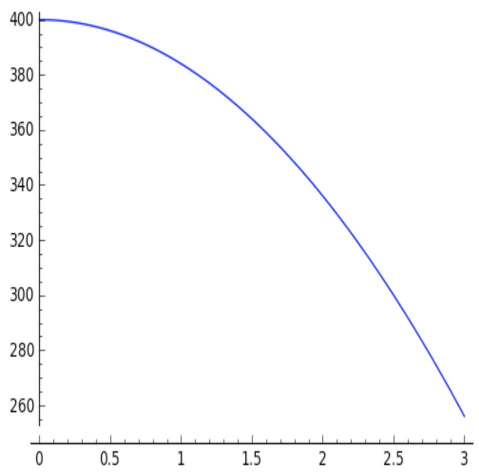
where as we would normally have typed

```
plot( x^2, -2, 2 )
```

as in the last section. This useful for when you have to plot things in terms of t and not in terms of x. For example:

```
 t=var('t')
plot( -16*t^2+400, (t, 0, 3) )
```

is the height of a bowling ball dropped off of a 400 foot building, t seconds after release. Here is the plot:

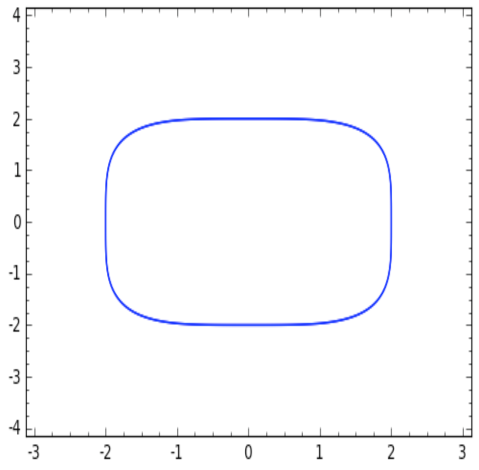


Two-Dimensional Implicit Plotting Once in a while, you have to plot a curve implicitly, even in two dimensions. The syntax is basically identical to three-dimensional plots. For example,

```
g(x,y)=x^4+y^4-16

implicit_plot(g, (x,-3,3), (y,-4,4) )
```

is a quartic curve favored by furniture designers for making conference room tables. As you can see, its mathematical formula is $x^4 + y^4 = 16$. Here is the plot:



1.8 Using SAGE to solve problems symbolically.

When we say that a computer has solved a problem for us, that can come out to be in one of two flavors: symbolically, or numerically. When we solve numerically, we get a decimal expansion for a (very good) approximation of the answer. When we solve symbolically, we get an exact answer, often in terms of radicals or other complicated functions. It is a matter of saying if the solutions to

$$\frac{x^2}{2} - x - 2 = 0$$

are $1+\sqrt{5}$ and $1-\sqrt{5}$ or saying that they are

$$-1.23606797749979 \text{ and } 3.23606797749979$$

The former is an example of a symbolic solution, and the latter of a numerical solution. If the answer is a formula, then symbolic solution is the only way to go. This section is about symbolic solution, and then the next one is about numerical solution.

Univariate Formulae First, let's try solving some single-variable problems. We can type

```
solve(x^2 + 3*x+2, x)
```

and then we obtain

```
[x == -2, x == -1]
```

But don't forget the asterisk! To be precise, to type

```
solve(x^2 + 3x+2, x)
```

would be wrong, because of the absence of the asterisk between the "3" and the "x." Another, more illustrative of a symbolic computation is

```
solve(x^2+9*x+15==0,x)
```

which gives

```
[x == -1/2*sqrt(21) - 9/2, x == 1/2*sqrt(21) - 9/2]
```

One way to really see what symbolic versus exact is about is to compare

```
3+1/(7+1/(15+1/(1+1/(292+1/(1+1/(1+1/6))))))
```

which returns `1,354,394 / 431,117` with

```
n(3+1/(7+1/(15+1/(1+1/(292+1/(1+1/(1+1/6)))))))
```

which returns `3.14159265350241`. That could easily be mistaken for π. In fact it is really close to π with relative error 2.78×10^{-11}.

There is no restriction to polynomials. You can also

```
theta=var('theta')
solve(sin(theta)==1/2, theta)
```

to get

```
[theta == 1/6*pi]
```

You might be confused to see that `var` command. The idea is that you need to tell SAGE that this variable is not yet known. We've used variables before, but that's because we had assigned values to them. When you want a variable to represent some unknown quantity, you must use `var`. The variable x is always pre-declared, you do not need to declare it—SAGE assumes that x is an unknown.

Multivariate Formulae Now we're going to use SAGE to re-derive the quadratic formula. First we must declare our variables with:

```
 a, b, c = var('a b c')
```

and then we type

```
 solve( a*x^2 + b*x + c == 0, x )
```

and get instead back

```
[x == -1/2*(b + sqrt(-4*a*c + b^2))/a,
x == -1/2*(b - sqrt(-4*a*c + b^2))/a]
```

which has some terms in an unusual order but is undoubtedly correct. Note that the square brackets and comma indicate a list in SAGE, something which SAGE inherited from the computer language Python.

By the way

```
 a, b, c = var('a b c')
```

is merely an abbreviation for

```
a = var('a')
b = var('b')
c = var('c')
```

Linear Systems of Equations You can solve several equations simultaneously, whether they are linear or non-linear. An easy case might be to solve

$$\begin{aligned} x + b &= 6 \\ x - b &= 4 \end{aligned}$$

which would be done by typing

```
b=var('b')
solve( [x+b == 6, x-b == 4], x, b )
```

Again, note the useof square brackets and commas to form a list in SAGE. This is our second example of a list in SAGE. You can enclose any data with `[` and `]`, and separate the entries with commas, to make a list. Our first example, was in graphing multiple functions at once, on Page 16; we'll see another example in Number Theory on Page 48.

Also, it is important to point out that there is no harm in "declaring" b twice. The two `var` commands do no harm to each other. On the other hand, there is also no need to "declare" b twice, as once it is declared, SAGE will remember that it is a variable.

A more typical linear system might be

$$\begin{aligned} 9a + 3b + 1c &= 32 \\ 4a + 2b + 1c &= 15 \\ 1a + 1b + 1c &= 6 \end{aligned}$$

and to solve that we'd type

```
a=var('a')
b=var('b')
c=var('c')
solve( [9*a + 3*b + c == 32, 4*a + 2*b + c == 15, a + b + c == 6], a, b, c )
```

and SAGE gives the answer

```
 [[a == 4, b == -3, c == 5]]
```

which is correct. Of course, this is exactly the linear system of equations that you would use if someone asked you to find the parabola connecting the points $(3, 32)$ and $(2, 15)$ as well as $(1, 6)$. That would be $f(x) = 4x^2 - 3x + 5$.

Naturally, linear systems of equations can also be solved with matrices. In fact, SAGE is quite useful when working with matrices, and can remove much of the tedium normally associated with matrix algebra. That will be discussed in the matrix section on Page 44.

Non-Linear Systems of Equations While linear equations are very easy to solve via matrices, the non-linear case is usually much harder.

First, we will warm up with just one equation, but a highly-non-linear one. Let us try to solve

$$x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720 = 0$$

which can be done by

```
solve( x^6 - 21*x^5 + 175*x^4 - 735*x^3 + 1624*x^2 - 1764*x + 720 == 0, x)
```

which gives

```
[x == 5, x == 6, x == 4, x == 2, x == 3, x == 1]
```

That is a list of six answers, because that degree six polynomial has six roots. It was easy to read this time, but you can also access each answer one at a time. Instead, we type

```
answer = solve( x^6 - 21*x^5 + 175*x^4 - 735*x^3 + 1624*x^2 - 1764*x + 720 == 0, x)
```

and then we can type `print answer[0]` or `print answer[1]` to get the first or second entries. To get the fifth entry of that list, we'd type `print answer[4]`. This is because SAGE is built out of Python, and Python numbers its lists from 0 and not from 1. There are reasons for this, based on some very old computer languages.

Now consider this problem, suggested by Dr. Grout. To solve

$$\begin{aligned} p + q &= 9 \\ qy + px &= -6 \\ qy^2 + px^2 &= 24 \\ p &= 1 \end{aligned}$$

we would type

```
p, q, y = var('p q y')
eq1 = p+q == 9
eq2 = q*y + p*x == -6
eq3 = q*y^2 + p*x^2 == 24
eq4 = p == 1
solve( [eq1, eq2, eq3, eq4 ], p, q, x, y )
```

which produces

```
[[p == 1, q == 8, x == -4/3*sqrt(10) - 2/3, y == 1/6*sqrt(2)*sqrt(5) -
2/3], [p == 1, q == 8, x == 4/3*sqrt(10) - 2/3, y ==
-1/6*sqrt(2)*sqrt(5) - 2/3]]
```

As you can see, that is a list of lists, again using square brackets and commas. Clearly, that is a very hard to read mess. Using the technique that we learned when analyzing the degree six polynomial above, we replace the last line with

```
answer=solve( [eq1, eq2, eq3, eq4 ], p, q, x, y )
```

Now we can type

```
print answer[0]
print answer[1]
```

and we get

```
[p == 1, q == 8, x == -4/3*sqrt(10) - 2/3, y == 1/6*sqrt(2)*sqrt(5) -
2/3]
[p == 1, q == 8, x == 4/3*sqrt(10) - 2/3, y == -1/6*sqrt(2)*sqrt(5) -
2/3]
```

This is SAGE's way of telling you:
Solution 1: $p = 1, q = 8, x = -\frac{4}{3}\sqrt{10} - \frac{2}{3}, y = \frac{\sqrt{10}}{6} - \frac{2}{3}$

Solution 2: $p = 1, q = 8, x = \frac{4}{3}\sqrt{10} - \frac{2}{3}, y = -\frac{\sqrt{10}}{6} - \frac{2}{3}$

Since there are only two answers, we cannot ask for a third. In fact, if we type `print answer[2]` then we get

```
Traceback (click to the left of this block for traceback)
...
IndexError: list index out of range
```

which is SAGE's way of telling you that it has already given you all the answers for that problem—the list does not contain a 3rd element.

Now let's try to intersect the hyperbola $x^2 - y^2 = 1$ with the ellipse $\frac{x^2}{4} + \frac{y^2}{3} = 1$. We type

```
y=var('y')
solve([x^2-y^2==1,     (x^2)/4+(y^2)/3==1],x,y)
```

and obtain

```
[x == -4/7*sqrt(7), y == -3/7*sqrt(7)], [x == -4/7*sqrt(7), y ==
3/7*sqrt(7)], [x == 4/7*sqrt(7), y == -3/7*sqrt(7)], [x == 4/7*sqrt(7),
y == 3/7*sqrt(7)]]
```

which is unreadable. Once again, we change the command to be

```
answer=solve([x^2-y^2==1,     (x^2)/4+(y^2)/3==1],x,y)
```

and then type

```
print answer[0]
print answer[1]
print answer[2]
print answer[3]
print answer[4]
```

and that produces four answers and an error message

```
[x == -4/7*sqrt(7), y == -3/7*sqrt(7)]
[x == -4/7*sqrt(7), y == 3/7*sqrt(7)]
[x == 4/7*sqrt(7), y == -3/7*sqrt(7)]
[x == 4/7*sqrt(7), y == 3/7*sqrt(7)]
Traceback (click to the left of this block for traceback)
...
IndexError: list index out of range
```

which is SAGE's way of telling you

Solution 1: $x = \frac{4}{7}\sqrt{7}, y = \frac{3}{7}\sqrt{7}$

Solution 2: $x = \frac{4}{7}\sqrt{7}, y = -\frac{3}{7}\sqrt{7}$

Solution 3: $x = -\frac{4}{7}\sqrt{7}, y = \frac{3}{7}\sqrt{7}$

Solution 4: $x = -\frac{4}{7}\sqrt{7}, y = -\frac{3}{7}\sqrt{7}$

but that there is no "Solution 5."

Last but not least, note that we aren't limited to polynomials. We can do

```
 solve(sin(x+y)==0.5,x)
```

and obtain

```
 [x == 1/6*pi - y]
```

Complex Numbers: Normally, we expect an 8th degree polynomial to have 8 roots, over the complex plane, unless some are repeated roots. Therefore, if we ask "what are the roots of $x^8 + 1$?", surely we expect 8 roots. These are the 8 eigthth-roots of -1 in the complex numbers. We can find them with

```
solve(x^8==-1, x)
```

which produces

```
[x == (1/2*I + 1/2)*(-1)^(1/8)*sqrt(2), x == I*(-1)^(1/8), x == (1/2*I -
1/2)*(-1)^(1/8)*sqrt(2), x == -(-1)^(1/8), x == -(1/2*I +
1/2)*(-1)^(1/8)*sqrt(2), x == -I*(-1)^(1/8), x == -(1/2*I -
1/2)*(-1)^(1/8)*sqrt(2), x == (-1)^(1/8)]
```

giving us eight distinct complex numbers, all of which have -1 as their eighth power. This can be calculated (by hand) more slowly with DeMoivre's formula, but for SAGE, this is an easy problem.

Advanced Cases: To see why we don't ask undergraduates to memorize Cardano's cubic formula, try typing in:

```
 solve(x^3 + b*x + c==0, x)
```

That gives you Cardano's formula for a monic depressed cubic. A cubic polynomial is said to be depressed if the quadratic coefficient is zero, and monic means that the leading coefficient (here, the cubic coefficient) is zero. The full version of the cubic formula would be given by:

```
d = var('d')
solve(a*x^3 + b*x^2 + c*x + d==0, x)
```

which is just insanely complicated. Try it, and you'll see. In fact, it is so complicated that one would have to confess that it is not useable.

An even uglier formula would be the general formula for a quartic, or degree four, polynomial. Surely any quartic can be written

$$a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$$

and so we can ask SAGE to solve it with

```
a0, a1, a2, a3, a4 = var('a0 a1 a2 a3 a4')
solve( a4*x^4 + a3*x^3 + a2*x^2 + a1*x + a0 == 0, x)
```

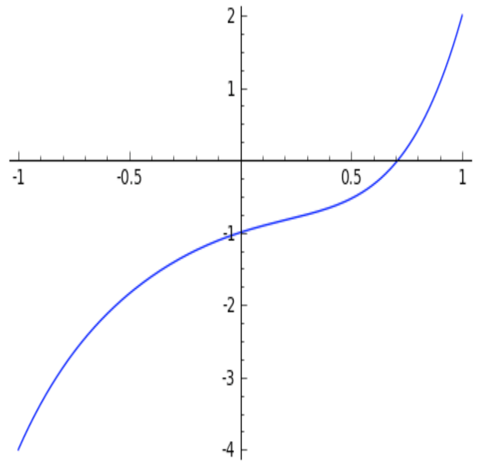
and we get an answer that is a formula of horrific proportions.

1.9 Using SAGE as a numerical solver.

Let's suppose you need to know the value of a root of

$$f(x) = x^5 + x^4 + x^3 - x^2 + x - 1$$

whose graph is



Furthermore, you are interested in a root between -1 and 1. This could come about because you graph it (with or without SAGE), and see that there is a root in that region; or it could come about because you see that $f(-1) = -4$ but $f(1) = 2$, and so clearly f crosses 0 at some point between -1 and 1, though we don't know where yet. Perhaps the problem in the textbook simply tells you that the root is between -1 and 1.

Okay, to solve the problem, you need only type this

```
 find_root(x^5+x^4+x^3-x^2+x-1,-1,1)
```

and SAGE tells you that

$$0.710434255787221398$$

is the root. This is a numerical approximation, because quintic equations have a very special property—if you don't know what the special property is, just ask any math teacher.

While it is an approximation, the computer does several billion instructions per second, and so it is an approximation that is trustworthy to around an accuracy of 10^{-16}, which is one tenth of a quadrillionth. So it is a good approximation.

Meanwhile, it turns out there's no root between -1 and 0. Perhaps you are told that, or perhaps you graph it (with or without SAGE). If you type

```
find_root(x^5+x^4+x^3-x^2+x-1,-1,0)
```

then SAGE tells you

```
RuntimeError: f appears to have no zero on the interval
```

which is perfectly honest, because there's no root in the interval for SAGE to go and find! If you have no idea where the root is, you can type

```
find_root(x^5+x^4+x^3-x^2+x-1,-10^12,10^12)
```

which is asking SAGE to find a root that is between plus/minus one trillion.

Extremely sophisticated problems can be asked about. Some favorites of mine include if someone asks you about $x^x = 5$, then you could try to find where $x^x - 5 = 0$. You would type

```
find_root(x^x-5,1,10)
```

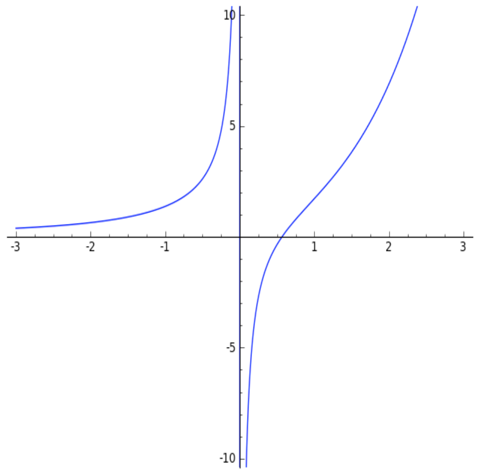
or equivalently

```
find_root(x^x==5,1,10)
```

Another one is to find where $e^x = 1/x$ and you'd do that by finding a root of $e^x - 1/x = 0$. You could graph that with

```
plot( e^x - 1/x, -3, 3, ymin=-10, ymax=10)
```

obtaining the plot



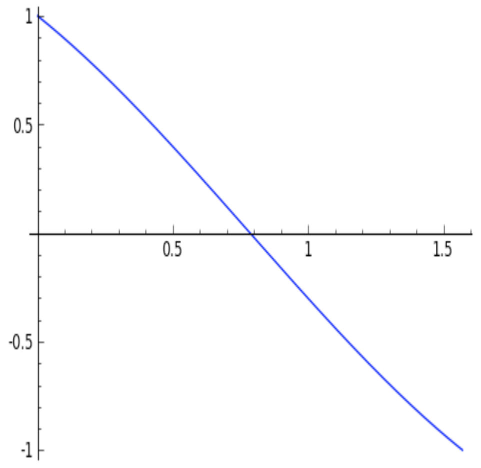
and then you can see the root is between -1 and 1. This means that we should type

```
find_root(e^x-1/x,-1, 1)
```

will result in finding out that the root is at

$$0.56714329040979539$$

Last but not least, another favorite is to find $\sin x = \cos x$. It happens to be between 0 and $\pi/2$, which you can see by graphing. The graph is



Now here you could ask for a root of $(\sin x) - (\cos x)$. Alternatively, you can also type

```
find_root( cos(x) == sin(x), 0, pi/2 )
```

which is just easier notation.

1.10 Using SAGE to take derivatives

The command for the derivative is just `diff`. For example, to take the derivative of $f(x) = x^3 - x$ you would type:

```
diff( x^3 - x, x)
```

to learn the answer

```
 3*x^2 - 1
```

Or for $f(x) = sin(x^2)$ you would type:

```
diff( sin(x^2), x )
```

to receive the answer

```
2*x*cos(x^2)
```

Sometimes you might have earlier defined your own function by

```
g(x)=e^(-10*x)
```

earlier, and so you can then type

```
diff(g(x), x)
```

to get the derivative, which is naturally

```
-10*e^(-10*x)
```

but you can also type

```
gprime(x) = diff(g, x)
```

which is silent, but then you can type `gprime(2)` or `gprime(3)` and it does what you expect. It tells you the values of $g'(2)$ and $g'(3)$.

Another 100% equivalent form is

```
g(x).derivative()
```

SAGE can also do very difficult derivatives:

```
diff( x^x, x)
```

which provides the answer

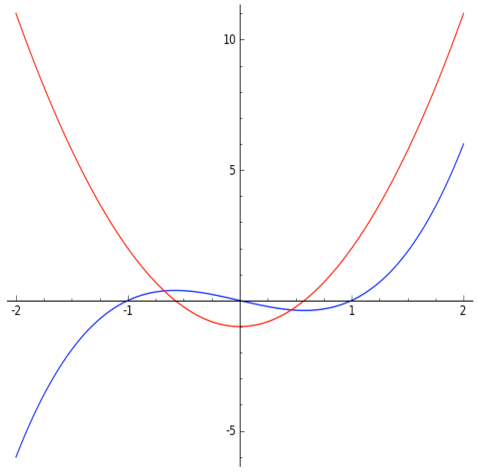
```
(log(x) + 1)*x^x
```

Now if you're interested in giving your calculus skills a workout, but possibly a very intense one, you can try to see how you would calculate the derivative of x^x with your pencil. In fact, if you happen to know Newton's Method, you can use that derivative and Newton's Method to calculate the self-logarithm of a number. Of course, many calculus teachers omit Newton's Method, so do not feel bad if you haven't been taught it yet.

Plotting $f(x)$ and $f'(x)$ Together One of my favorite bits of code in SAGE is

```
f(x)=x^3-x
fprime(x)=f(x).derivative()
plot( f(x), -2, 2, color='blue') + plot( fprime(x), -2, 2, color='red')
```

and I use this quite frequently. This plots $f(x)$ and $f'(x)$ on the same graph, with $f(x)$ in blue, and $f'(x)$ in red. You can very easily change what function you are analyzing, but just changing that first line. Here we are analyzing $f(x) = x^3 - x$, in blue, and its derivative, $f(x) = 3x^2 - 1$, in red. The plot is as follows:



Higher-Order Derivatives While you can find the second derivative by using `diff` twice, you can jump directly to the second derivative with

```
diff( x^3-x, x, 2)
```

to get the answer

```
6*x
```

Third or higher derivatives are also no problem. For example, the third derivative can be found with

```
diff( x^3-x, x, 3)
```

For those who don't mind typing more, you can do

```
derivative( x^3-x, x, 2)
```

which is more readable than writing `diff`.

1.11 Derivatives and Gradients in Multivariate Calculus

In the previous section, we learned about derivatives of single-variable functions. Now we'll learn about multivariate derivatives.

Partial Derivatives You're probably not surprised to learn that SAGE can do partial derivatives very easily. For example, if you have

$$g(x,y) = xy + sin(x^2) + e^{-x}$$

you can type

```
g(x,y) = x*y + sin(x^2) + e^(-x)
diff(g(x,y), x)
```

and get

```
 2*x*cos(x^2) + y - e^(-x)
```

which is $\partial g/\partial x$, and you can type

```
derivative(g(x,y), y)
```

to learn that $\partial g/\partial y$ is just x. Note here this really shows how `diff` and `derivative` are synonyms.

You can even find

$$\frac{\partial^2}{\partial x \partial y} f$$

by typing `derivative( g(x,y), x, y)`.

Gradients If you want ∇f then you type

```
g.derivative()
```

and SAGE will correctly calculate the gradient. In fact, SAGE displays

```
(x, y) |--> (2*x*cos(x^2) + y - e^(-x), x)
```

which is to remind you that this is a map from a point (x, y) to a pair of real numbers, and not an ordinary function. Interestingly, the gradient of a univariate function, done this way, is just the ordinary first derivative, which is quite correct according to the laws of calculus.

An Extended Example: Marginal Cost Coming Soon!

1.12 Using SAGE to calculate integrals, both numerically and symbolically.

The only thing you need to understand in order to do integrals in SAGE is that there are really three types of integrals that can be done: a numerical approximate integral, an exact definite integral, and an indefinite integral. The first two types result in numbers, but which will be found approximately or exactly, respectively. The third one is the antiderivative. If this is confusing, then let us consider a simple example:

Background: What integral might you want to consider, in order to find the area between the x-axis and $f(x) = x^2 + 1$ on the interval $3 \le x \le 6$? Well, that integral would be

$$\int_3^6 x^2 + 1\, dx = \left(\frac{1}{3}x^3 + x\right)\Bigg|_3^6 = \left(\frac{1}{3}6^3 + 6\right) - \left(\frac{1}{3}3^3 + 3\right) = 72 + 6 - 9 - 3 = 66$$

and that is a number. Here we've done it analytically, using calculus. This is an exact definite integral. It is definite, because it ends with a number. It is exact, because we did no approximations.

On the other hand,

$$\int x^2 + 1\, dx = \frac{1}{3}x^3 + x + C$$

is an indefinite integral. It produces a function, and not a number. Of course, we calculate an indefinite integral on the way to calculating a definite one, but they are two distinct categories. If you want a function, you should use the indefinite, and if you want a number, then you should use the definite.

Some integrals are famously impossible. The two most famous are the Fresnel

$$\int_0^x \sin \frac{\pi t^2}{2}\, dt$$

which is important in optics, and the Gaussian

$$\int \frac{1}{\sqrt{2\pi}} e^{(-x^2)/2}$$

which is pivotally important in probability. There is no function, built up of addition, subtraction, multiplication, division, roots, exponents, logarithms, trigonometric functions and inverse trigonometric functions (as well as their hyperbolic cousins, which you might or might not have been taught about) which will have, as its derivative, either the Fresnel or the Gaussian.

Yet, in applied mathematics, we need to calculate these integrals. In particular, the Gaussian is superbly important in statistics. So what is done, is that the definite integral can be computed by dividing the interval into many, many tiny regions. Lots of infinitesimal rectangles and trapezoids can be used to approximate the curve. Each of these has a small amount of error, but when you add up all the areas, the hope is that the area cancels out a bit. You probably did this during your calculus class at some point, or will at some future point. Even more sophisticated methods include using polynomials instead of trapezoids, and SAGE uses extraordinarily intricate techniques to make the best approximation possible. These approximations are an old and deep subject, going back to Simpson's Rule, where Simpson used tiny parabolas instead of trapezoids.

The above might have been a bit confusing, but the whole point of SAGE is that it will worry about these details for you, so long as you know what you want from SAGE. I hope this background was interesting and at least partially clear. If not, you should probably discuss it with your calculus instructor, before proceeding further.

The Indefinite Integral: The indefinite integral is the simplest, and for example if you wanted to know

$$\int x\sin(x^2)dx$$

you would simply type

```
integral( x*sin(x^2), x )
```

and then you'd learn the answer is

```
-1/2*cos(x^2)
```

Similarly, if you wanted to know

$$\int \frac{x}{x^2+1}dx$$

you would simply type

```
integral( x/(x^2 + 1), x )
```

and then you'd learn the answer is

```
1/2*log(x^2 + 1)
```

The Definite Integral: Alternatively, we could put a lower bound of 0 and an upper bound of 1 on that integral, and then get

$$\int_0^1 \frac{x}{x^2+1}dx$$

becomes

```
integral( x/(x^2 + 1), x, 0, 1 )
```

and so we discover

```
1/2*log(2)
```

is the answer, but remember that "log" means the natural logarithm, not the common logarithm.

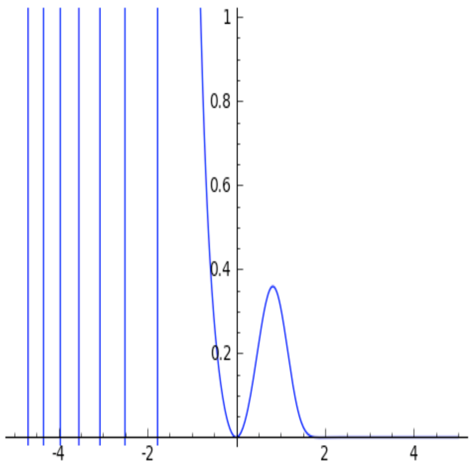
More Difficult Cases: Consider the following function f

$$f(x) = e^{-x^3}\sin(x^2)$$

which is a nice continuous function. We can find its plot with

```
plot( exp( -x^3 ) * sin(x^2), -5, 5, ymin = 0, ymax=1 )
```

which yields the plot:



The function is interesting, but the integral is quite frustrating. I cannot solve that integral with my pencil, and I am willing to wager that you cannot either. Try it if you like.

While it is the case that there is some function, somewhere, which has f as its derivative, which is a logical consequence of some advanced real analysis and the fact that f is continuous, the practicalities of the matter are that there is no way to write the integral. To be really precise, there is no function, built up of addition, subtraction, multiplication, division, roots, exponents, logarithms, trigonometric functions and inverse trigonometric functions (as well as their hyperbolic cousins, which you might or might not have been taught about) which will have, as its derivative,

In other words f is integrable, but the integral cannot be written in terms of common (and not so common) functions.

Thus since

$$\int e^{-x^3}\sin(x^2)dx$$

does not have a nice expression, then when you type

```
 integral( exp(-x^3)*sin(x^2),x)
```

the SAGE replies not with an answer, but instead with

```
 integrate(e^(-x^3)*sin(x^2), x)
```

which represents an admission of defeat. Have no fear, however, because we can calculate numerical answers approximately, with very high accuracy, and we'll learn about that momentarily.

A nefarious integral, famous among calculus teachers, is

$$\int t^{20}e^t dt$$

where the 20 could be any decently large number. This comes up in explaining what Euler's gamma function is, but we aren't too concerned with that right now. In any case, we could just type

```
 integral( (t^20)*(e^t), t )
```

but then we get back

```
(t^20 - 20*t^19 + 380*t^18 - 6840*t^17 + 116280*t^16 - 1860480*t^15 + 27907200*t^14
- 390700800*t^13 + 5079110400*t^12 - 60949324800*t^11 + 670442572800*t^10 -
6704425728000*t^9 + 60339831552000*t^8 - 482718652416000*t^7 +
3379030566912000*t^6 - 20274183401472000*t^5 + 101370917007360000*t^4
- 405483668029440000*t^3 + 1216451004088320000*t^2 - 2432902008176640000*t
+ 2432902008176640000)*e^t
```

which is large enough that it isn't clear how to get an idea of what that means. If instead, we had certain bounds, such as to calculate

$$\int_2^3 t^{20}e^t dt$$

then we would type

```
 integral( (t^20)*(e^t), t, 2, 3 )
```

and get instead

```
 -3292574823636008896*e^2 + 1211270590514628881*e^3
```

which is getting toward an answer! On the other hand, if you really want a numerical answer then you should use our old friend $n()$ and type

```
n( integral( (t^20)*(e^t), t, 2, 3 ) )
```

which would return the number

```
8.79797452800000e9
```

and that means $8.79797452800000 \times 10^9$. This highlights the difference and similarities of the various types of answer.

A Theoretical Side Note: That number which we just calculated looks like an integer, because it ends with 528 when you carry out the 10^9 part—but when we look at the exact answer, we can see that no, it is not an integer, because e^2 and e^3 are involved, and e is an irrational number. It is quite possible that you haven't yet learned what irrational numbers are, but if not, then ask any math teacher They will be happy to discuss such an advanced topic with you. Whenever dealing with computations, it is critical to remember what is an approximation, and what is exact. Irrational number can be expressed exactly on occasion (like we just did here, using e^2 and e^3) but that is the exception and not the rule. In any case, this paragraph has nothing to do with SAGE, and so we return to the topic at hand.

Numerical Integrals: Speaking of approximations, you can also jump to the numerical integral very rapidly. I'm sure we can both calculate

$$\int_0^1 x^3 - x\,dx = \left(\frac{1}{4}x^4 - \frac{1}{2}x^2\right)\Big|_0^1 = \left(\frac{1}{4}1^4 - \frac{1}{2}1^2\right) - \left(\frac{1}{4}0^4 - \frac{1}{2}0^2\right) = \frac{1}{4} - \frac{1}{2} = -\frac{1}{4}$$

but if you wanted to ask SAGE to calculate this numerically, then you should type

```
numerical_integral( x^3 - x, 0, 1)
```

which would output

```
(-0.24999999999999997, 2.775557561562891e-15)
```

where the first number is the best guess SAGE has for the answer, while the second number is the uncertainty. In this case, the uncertainty is 2.77 quadrillionths—which is very impressive.

Of course, SAGE can do this integral exactly as well, using the commands we learned moments ago. What is interesting about numerical integration, are the cases when you cannot do the indefinite integral, because like the Fresnel above it cannot be written, but you can find good numerical estimates. Consider again

$$\int_1^3 e^{-x^3}\sin(x^2)dx$$

but now with bounds, so that it can be done numerically. We would type

```
numerical_integral( exp(-x^3)*sin(x^2),1,3)
```

and that returns back

```
(0.077997011262087815, 8.6840496851995037e-16)
```

with (as before) the first number being the answer, and the second the accuracy. In this case, the uncertainty is 868 quintillionths—which is very impressive. Another way to say it is that the uncertainty is (1/1152) trillionths.

Partial Fractions One thing that is very important in problems that arise from *Differential Equations* and even *Calculus II* is the question of integration by partial fractions. If you want to know the partial fraction break-down of

$$\frac{x^3-x}{x^2+5x+6}$$

then you must do

```
f(x)=(x^3-x)/(x^2+5*x+6)
```

followed by

```
 f.partial_fraction()
```

and get the answer

```
x - 6/(x + 2) + 24/(x + 3) - 5
```

which is correct. In fact, it means

$$x-\frac{6}{x+2}+\frac{24}{x+3}-5$$

However, you can also do the shortcut and ask instead

```
integral( (x^3-x)/(x^2+5*x+6), x )
```

which will result in

```
1/2*x^2 - 5*x - 6*log(x + 2) + 24*log(x + 3)
```

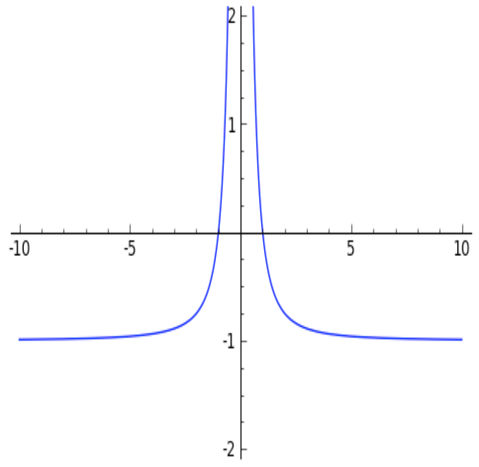
Improper Integrals While the following integral is mathematically valid

$$\int -1+\frac{1}{x^2}\,dx = -x-\frac{1}{x}+C$$

and the right-hand side can be evaluated at $x=-1$ and $x=1$, there is no finite answer to

$$\int_{-1}^{-1} -1+\frac{1}{x^2}\,dx$$

because this improper integral is divergent. The graph will reveal why that is the case. Here is the plot,



generated by

```
plot( -1+1/x^2, -10, 10, ymin=-2, ymax=2 )
```

Yet, SAGE knows this and will say

```
-x - 1/x
```

in response to

```
integral(-1+1/x^2,x)
```

but instead will respond

```
ValueError: Integral is divergent.
```

to the inquiry

```
integral(-1+1/x^2,x,-1,1)
```

Another type of improper integral is

$$\int_2^\infty \frac{1}{x^2} dx$$

and the way to write that in SAGE is to type

```
integral(1/x^2,x,2,oo)
```

to which SAGE responds 1/2, or another example is

$$\int_{-\infty}^{\infty} e^{-x^2} dx$$

which in SAGE is

```
integral(exp(-x^2),x,-oo,oo)
```

and it gives the correct response of $\sqrt{\pi}$.

The idea is simply that `oo` is a special code for "infinity." It is two letter o's, and the idea is that if you squint, it looks like an infinity symbol.

The Function erf and Integrals: Among mathematicians and especially among statisticians, the integral

$$\int_0^y \frac{2}{\sqrt{\pi}} e^{(-x^2)}\, dx = \mathrm{erf}(y)$$

which is just a slight variant of the one that I told you earlier cannot be found using the elementary functions of mathematics, is so important that they gave it a name, and that name is `erf`. While being friendly and pronounceable, `erf` stands for the "Error Function."

Thus if you type into SAGE

```
integral( (2/sqrt(pi)) * exp(-x^2), x, -oo, 2)
```

it will respond

```
(sqrt(pi)*erf(2) + sqrt(pi))/sqrt(pi)
```

which is correct because

$$\int_{-\infty}^{2} \frac{2}{\sqrt{\pi}} e^{(-x^2)}\, dx = \int_{-\infty}^{0} \frac{2}{\sqrt{\pi}} e^{(-x^2)}\, dx + \int_{0}^{2} \frac{2}{\sqrt{\pi}} e^{(-x^2)}\, dx = 1 + \text{erf}(2) = \frac{\sqrt{\pi}}{\sqrt{\pi}}(1 + \text{erf}(2)) = \frac{\sqrt{\pi} + \sqrt{\pi}\text{erf}(2)}{\sqrt{\pi}}$$

Perhaps you might find that $1 + \text{erf}(2)$ is a more compact and simpler answer. I would be inclined to agree. I'm not quite sure why SAGE does not simply its final answer further.

In any case, the point is that there are some functions out there which do not have integrals writable using elementary functions, but, which SAGE calculates anyway, using the `erf` function. Not all calculus textbooks are anticipating that, and there can be some unexpected surprises when you compute an answer to a textbook problem, and look in the back of the book, only to find that your textbook tells you that the problem is unsolvable. It really is a matter of whether or not you accept `erf` as a legitimate or illegitimate member of the family of functions.

1.13 Using SAGE to Reduce, Multiply and Invert Matrices, and Take Determinants

Coming soon!!

1.14 Some Extended Examples

Here are to be found three extended examples of using SAGE to explore a non-trivial problem in an area outside of computer algebra. We present one example each from Pure Math, Finance, and Physics. Few students will be interested in all three of these topics, so I just suggest that you pick one of the three that suits your background, and go from there.

1.14.1 The Pure Math Example:

Coming Soon! The quintic, and finding the correct window for graphing, by Prof. Ben Jones.

1.14.2 The Finance Example:

This example assumes you know how to do mortgage calculations. If you don't, then you might want to read the physics example.

Let us say that you want to sell some mortgage products, and you want to compute the cost-per-thousand for several loans at interest rate r and compounded monthly. If the length is for t years, then the number of compoundings is $n = mt = 12t$. The interest-rate-per-compounding-period would be $i = r/m = r/12$ and value would be $V = 1000$. So we would start with the basic formula

$$PMT = \frac{Vi}{1 - (1+i)^{-n}}$$

and plug in all that data. Since $V = 1000$ is not going to change, we can assign that one with

```
V=1000
```

to make it a long term constant. Note, SAGE accepts your constant quietly, it does not give a response. Then, we could define our own function, as described earlier

```
CostPerThousand (i, n) = ( V * i ) / (1 - (1+i)^(-n) )
```

and so for a 30 year mortgage at 6.5% compounded monthly, we can do

```
CostPerThousand ( 0.065/12, 30*12)
```

and then if the rate rises to 7% we can do

```
CostPerThousand ( 0.07/12, 30*12)
```

and so on, and so forth.

Notice that the function name has to be only one word, but by varying the capitalization that way (which programmers call "Camel Case"), we could express an idea that was more than one word. Surely it is much easier to read `CostPerThousand` than `costperthousand` or `COSTPERTHOUSAND`.

While this above function is nice, (for example it will save us from agonizing each time over the placement of the parentheses,) it also isn't quite perfect. Our data will be given in terms of the number of years of the loan t, not the number of payments $n = mt = 12t$. Likewise, the rate will be the nominal, published rate r not the per-period rate $i = r/m = r/12$. One student once described this to me by saying that the n and i are math-friendly, whereas the t and r are human-friendly.

This can be fixed by doing

```
CostPerThousand1 (r, t)=CostPerThousand( i=r/12, n=12*t )
```

and now we have made a new formula, which uses the rate in terms of r as we liked, and the number of years t, not the number of compounding periods. And now, for a 30 year mortgage at 6.5% compounded monthly, we can do

```
CostPerThousand1 ( 0.065, 30)
```

and get a more user-friendly input and output.

1.14.3 The Physics Example:

Let's say you're analyzing a satellite in its orbit around the earth. Newton's formula for gravity is

$$F = \frac{GM_eM_s}{r_s^2}$$

where $G = 6.77 \times 10^{-11} Nm^2/kg^2$ is the universal constant for gravitation; $M_e = 5.9742 \times 10^{24}$ kg is the mass of the earth; next r_s is the distance from the satellite to the center of the earth in meters; and finally M_s kg is the mass of the satellite. Surely G and M_e will not change, so we can set those with

```
 G= 6.77 * 10^(-11)
Me = 5.9742 * 10^(24)
```

just like we set $c = 355/113$ earlier.

Unlike the mass of the earth, or the universal constant of gravitation, the distance to the satellite might change quite often, and the mass of the satellite could change also (particularly if it has fuel aboard, which will be burned during maneuvers). So for those, we should make a function

```
Force( Ms, rs ) = (G * Me * Ms)/(rs^2)
```

and you could do things like

```
Force( 1000, 10 * 10^6 )
```

to get answers. You could even use this while plotting, which we haven't told you about yet. This is great if you have a large number of points that you're interested in, and want to save a lot of typing.

Suppose now that you really wanted the function to work in terms of altitude, as measured from the earth's surface, and not from the center of the earth. Well it is easy to realize that you just have to add the radius of the earth. First, you put in the radius of the earth, which is 6378.1 km, but of course we should switch it to meters to avoid unit conflicts.

```
re = 6378100
```

Then you can define

```
Force1( Ms, alt ) = Force( Ms, alt + re )
```

and then you can just go ahead and use `Force1`. This idea of building functions on top of functions is very powerful. I have often written large suites of functions in this way—one function on top of the other function, on top of the next—to model complex phenomena in SAGE. For example,

```
Force1( 1000, 3621900 )
```

is roughly equivalent to what we had before. I suppose it depends upon if the problem you are given is in terms of altitude, or distance from the center of the earth.

The Grouping Symbols: Just a reminder, while a mathematician might write

$$PV = PMT[1 - (1+i)^{-n}]/i$$

for the present value formula, using the [and] as a sort of auxiliary (and), in SAGE, the only grouping operators are parentheses. So that formula is

```
PV = PMT * (1 - (1+i)^(-n))/i
```

with additional explanation found on Page 3.

Chapter 2

Advanced Features of SAGE

2.1 Using SAGE to work with integers. (gcd, lcm, factorization, sigma, tau, etc. . .)

The study of problems that deal with the integers, in pure mathematics, is called "number theory", and in certain types of applied mathematics, called "integer programming." This subject can be started at a very young age (e.g. "what is a prime number?") but can be very advanced too. Many Ph.D. dissertations are written every year with new results in number theory. Mostly, number theory is about writing proofs, and so you'd imagine that SAGE is not very useful for that. Actually, SAGE is very good at working with the integers to help you work out specific examples on what the instruments of number theory do to specific integers. This can help you understand new concepts more intimately, and I find it can form a bridge between the lecture on a topic and the time when you are really ready to write serious proofs about a topic.

For example, you can type `factor(-2007)` to find the factorization of that integer, which is

```
-1 * 3^2 * 223
```

and likewise

```
factor(2008)
```

is

```
2^3 * 251
```

To find the next prime number after one million, you can do

```
next_prime( 10^6 )
```

and discover the answer is 1,000,003.

The gcd and the lcm To find the "greatest common denominator" or "gcd" of two numbers, perhaps 120 and 64, just type

```
gcd ( 120, 64 )
```

and you will learn that the answer is 8. This makes sense because $120/8 = 15$ while $64/8 = 8$, and as you can see, 15 and 8 share no common factors. More explicitly, we can look at the prime factorizations $15 = 5 \times 3$ and $8 = 2 \times 2 \times 2$. Another way of looking at that is

$$120 = 2^3 \times 3 \times 5 \text{ while } 64 = 2^6$$

and so all that 120 and 64 have in common is just 2^3, which is 8.

Likewise, you can find the "least common multiple" or lcm. This is done by typing

```
lcm( 120, 64 )
```

from which you learn that the lcm is 960. This makes sense because $960/120 = 8$ and also $960/64 = 15$, and again 8 shares no factors with 15. A mathematician would say "8 is coprime to 15." Some authors say "mutually prime" instead of "coprime."

Another way to look at it is that the lcm should be

$$2^6 \times 3^1 \times 5^1 = 960$$

because the 2 appears 6 times in 64 and 3 times in 120 (thus a maximum of 6), while the 3 and 5 both appear 0 times in 64 and 1 time in 120 (thus a maximum of 1 each). Raising each of those primes (2, 3, and 5) to those powers (6, 1, and 1) gives the lcm.

Before we continue, one more example is helpful. Let's work with slightly larger numbers, perhaps 3600 and 1000. First, we do the prime factorizations

$$3600 = 2^4 \times 3^2 \times 5^2 \text{ while } 1000 = 2^3 \times 5^3$$

and we can compute the lcm by taking the maximum exponent each time

$$2^4 \times 3^2 \times 5^3 = 18,000$$

and the gcd by taking the minimum exponent each time

$$2^3 \times 3^0 \times 5^2 = 200$$

Take a moment to verify that 200 goes into both 1000 and 3600, as well as the fact that no higher number will. Next, take another moment to verify that both 1000 and 3600 go into 18,000, but there is no lower number for which that's true. And of course, the product of the numbers divided by the gcd is the lcm, and the product of the numbers divided by the lcm is the gcd.

The key here is to realize that the prime factorization of a number communicates an enormous amount of information about that number.

Several gcds or lcms at Once If you want to take the gcd of many numbers at once (for example 120, 55, 25, and 35), there's no need to do

```
gcd( 120, gcd(55, gcd(25, 35 ) ) )
```

because you can do instead

```
gcd( [ 120, 55, 25, 35 ] )
```

to get 5 and likewise

```
lcm( [ 120, 55, 25, 35 ] )
```

to get 46,200. This is another example of a list in SAGE. You can enclose any data with `[` and `]`, and separate the entries with commas, to make a list. We saw this in graphing multiple functions at once, on Page 16, and in solving equations exactly, on Page 29.

And you can even verify the theorem that says the product of all the numbers in the list, divided by their gcd raised to the power of one less than the length of the list, equals the lcm. Since we have four numbers in our list, we would divide by the cube of the gcd. This is done by

```
120*55*25*35 / 5^3
```

More about Prime Numbers One neat trick is that if you wanted to find the 54,321th prime number, you could type

```
nth_prime(54321)
```

and discover that it is 670177. You can double check this with

```
factor(670177)
```

which returns itself, confirming that the number is prime.

Phi Sometimes it is useful to know how many integers are coprime to x, from the range 1 to x. For example, how many numbers from 1 to 100 share a factor with 10, and how many are coprime with 10. Well, the prime factors of 100 are 2 and 5, and so any number composed only of twos and fives will work. This would be

$$\{2, 4, 5, 6, 8, 10\}$$

and so the other numbers

$$\{1, 3, 7, 9\}$$

are coprime with 10. In particular, if you know what modular arithmetic is, then these are the numbers that have "multiplicative inverses" or "reciprocals" mod 10. Since 4 numbers are coprime to 10, we will write $\phi(10) = 4$.

Even if you don't know what modular arithmetic is, consider 7. Each of

$$\{1, 2, 3, 4, 5, 6\}$$

is coprime to 7, because the only prime that divides 7 is 7, and 7 does not divide any of those. So, 6 out of the 6 numbers are coprime to 7, and we would then write $\phi(7) = 6$. It will always be the case for a prime number that $\phi(p) = p - 1$. On the other hand, consider 12. Then among

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$$

we see that both 2 and 12 are divisible by 2; both 3 and 12 are divisible by 3; both 4 and 12 are divisible by 2; both 6 and 12 are divisible by either 2 or 3; both 8 and 12 are divisible by 2; both 9 and 12 are divisible by 3; both 10 and 12 are divisible by 5. Finally, this means that only 1, 5, 7, and 11 are coprime to 12, and so $\phi(12) = 4$.

It would be very annoying to calculate ϕ of a 100-digit number, such as when working with cryptography, the science of codes. All you need to do in SAGE is type

```
euler_phi(12)
```

and then you learn that the answer is 4. It is called "`euler phi`" because it was discovered by Leonhard Euler.

The Divisors of a Number Meanwhile, consider the set of divisors of a number. If you want to know the set of divisors of 12, which means the set of positive integers dividing 12, you would type

```
divisors(12)
```

and get then

```
 [1, 2, 5, 10, 25, 50, 125, 250]
```

and likewise

```
divisors(312500)
```

gives you

```
 [1, 2, 4, 5, 10, 20, 25, 50, 100, 125, 250, 500, 625, 1250, 2500,
3125, 6250, 12500, 15625, 31250, 62500, 78125, 156250, 312500]
```

but remember, it is very important to not have any commas separating the thousands, when inputting large numbers into SAGE.

Sometimes in number theory, we want to calculate the sums of the squares or cubes of the divisors. I can't think of an example why at the moment, but I'm sure there's a good one. Consider that 45 has as its divisors $\{1, 3, 5, 9, 15, 45\}$. Then we have sigma coming to 78, because

$$1 + 3 + 5 + 9 + 15 + 45 = 78$$

and then the sum of squares would be

$$1^2 + 3^2 + 5^2 + 9^2 + 15^2 + 45^2 = 2366$$

and with cubes

$$1^3 + 3^3 + 5^3 + 9^3 + 15^3 + 45^3 = 95,382$$

but in SAGE the commands for that are

```
 sigma( 45, 2 )
```

for squares and

```
 sigma( 45, 3 )
```

for cubes.

Likewise

```
 sigma( 45, 1 )
```

is the same as sigma.

Now there's a fun game that the Arabs used to play in the Medieval Period. Consider two particular numbers 220 and 284. The divisors are

$$\text{divisors}(220) = \{1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110, 220\}$$

and then they would calculate

$$1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$$

but on the other hand

$$\text{divisors}(284) = \{1, 2, 4, 71, 142, 284\}$$

which yields

$$1 + 2 + 4 + 71 + 142 + 284 = 220$$

and so we see 220 and 284 are intimately related.

The Arabs found this so moving that they inscribed these numbers on jewelry that they would give to their spouses and so forth. These are called Amicable Pairs—220 representing the young lady, and 284 representing the young gentleman. You can find this in SAGE via

```
 sigma( 220 )-220
```

and

```
 sigma( 284 )-284
```

Tau As you know, some fractions like 1/4 and 1/2 as well as 1/25 can be expressed as decimal fractions which terminate. Others, like 1/3 or 1/9 will repeat. It is pretty easy to see why a fraction can be expressed as an exact terminating decimal fraction of n decimal places (or fewer) if and only if the denominator divides 10^n. Take a moment to convince yourself of this, or alternatively, consider the following example. If we want to know what denominators result in a two-decimal place (or fewer) exact terminating decimal, we would just need to see what positive integers divide 100. In SAGE we'd do

```
divisors(100)
```

and then get

```
[1, 2, 4, 5, 10, 20, 25, 50, 100]
```

but in binary, the analogous rule would be 2^n. It is obvious that the only positive integers which divide 2^n are the numbers $1, 2, 4, 8, \ldots, 2^n$. Therefore, we can use this criterion for seeing which number bases are convenient for fractions, and which are not.

Now, we'll consider the Babylonian numbering system which is base 60. Two symbols would be sufficient to describe any fraction with the following denominators:

```
[1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 30, 36,
40, 45, 48, 50, 60, 72, 75, 80, 90, 100, 120, 144, 150, 180, 200, 225, 240,
300, 360, 400, 450, 600, 720, 900, 1200, 1800, 3600]
```

Did you notice how in decimal, there were 9 denominators, but in base 60, there were 45 denominators? Or if you do not count one, then 8 versus 44? What we really wanted to do was count the divisors. The function for counting the number of (positive integer) divisors of an integer is τ, and in SAGE is given by

```
sigma(3600, 0)
```

which just tabulates 1 for each divisor. In other words, each divisor is raised to the 0th power, thus becomes 1, and then if you add those up, you learn how many divisors there are.

Modular Arithmetic: To find out what 1939 is mod 37, you can type

```
1939 % 37
```

but the % operator takes the modular reduction only of the nearest number. This means that to evalulate

$$(3^2)4 + 2 \bmod 5$$

you must not type

```
3^2*4 + 2%5
```

which evaluates to 38, but rather

```
(3^2*4 + 2) %5
```

which gives 3, the correct residue.

The opposite of the "mod" operator is the integer quotient. To find the integer quotient of a number, use

```
25 // 4
```

to get 6. This is the quotient rounded down to the nearest integer.

You can define

```
def is_even(n): return n%2 == 0
```

which will return `True` for even integers, and `False` for odd ones. This can then be used to wrap up into other `if` statements. We'll talk much more about `if` statements in the chapter on programming.