2 Analysis and Algebra

This chapter uses simple examples to describe the useful basic functions in analysis and algebra. Students will be able to replace *pen and paper* by *keyboard and screen* while keeping the same intellectual challenge of understanding mathematics.

This presentation of the main calculus commands with Sage should be accessible to young students; some parts marked with an asterisk are reserved for higher-level students. More details are available in the other chapters.

2.1 Symbolic Expressions and Simplification

2.1.1 Symbolic Expressions

Sage allows a wide range of analytic computations on *symbolic expressions* formed with numbers, symbolic variables, the four basic operations, and usual functions like `sqrt`, `exp`, `log`, `sin`, `cos`, etc. A symbolic expression can be seen as a tree like in Figure 2.1. It is important to understand that a symbolic expression is a *formula* and not a value or a mathematical function. Thus, Sage does not recognise the two following expressions as equal[1]:

```
sage: bool(arctan(1+abs(x)) == pi/2 - arctan(1/(1+abs(x))))
False
```

Thanks to the commands presented in this chapter, the user can transform expressions into the desired form.

[1]The equality test `==` is not only a syntactic comparison: for example, the expressions `arctan(sqrt(2))` and `pi/2-arctan(1/sqrt(2))` are considered equal. In fact, when one compares two expressions with `bool(x==y)`, Sage tries to prove that their difference is zero, and returns `True` if that succeeds.

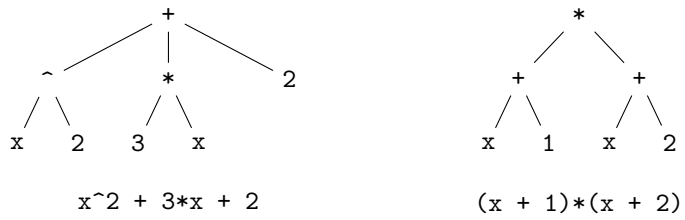

FIGURE 2.1 – Two symbolic expressions representing the same mathematical object.

The most common operation consists of *evaluating* an expression by giving a value to some of its parameters. The `subs` method — which can be made implicit — performs this transformation:

```
sage: a, x = var('a, x'); y = cos(x+a) * (x+1); y
(x + 1)*cos(a + x)
sage: y.subs(a=-x); y.subs(x=pi/2, a=pi/3); y.subs(x=0.5, a=2.3)
x + 1
-1/4*sqrt(3)*(pi + 2)
-1.41333351100299
sage: y(a=-x); y(x=pi/2, a=pi/3); y(x=0.5, a=2.3)
x + 1
-1/4*sqrt(3)*(pi + 2)
-1.41333351100299
```

Compared to the usual mathematical notation $x \mapsto f(x)$, the variable which is substituted must be explicitly given. The substitution of several parameters is done in parallel, while successive substitutions are performed in sequence, as shown by the two examples below:

```
sage: x, y, z = var('x, y, z') ; q = x*y + y*z + z*x
sage: bool(q(x=y, y=z, z=x) == q), bool(q(z=y)(y=x) == 3*x^2)
(True, True)
```

To replace an expression more complex than a single variable, the `substitute` method is available:

```
sage: y, z = var('y, z'); f = x^3 + y^2 + z
sage: f.substitute(x^3 == y^2, z==1)
2*y^2 + 1
```

2.1.2 Transforming Expressions

The simplest non-constant expressions are polynomials and rational functions of one or more variables. The functions allowing to rewrite expressions in several forms or to put them in normal form are summarised in Table 2.1. For example, the `expand` method is useful to expand polynomials:

```
sage: x, y = SR.var('x,y')
```

Symbolic Functions

Sage allows also to define *symbolic functions* to manipulate expressions:

```
sage: f(x)=(2*x+1)^3 ; f(-3)
-125
sage: f.expand()
x |--> 8*x^3 + 12*x^2 + 6*x + 1
```

A symbolic function is just like an expression that we can call like a command and where the order of variables is fixed. To convert a symbolic expression into a symbolic function, we use either `f(x) = ...`, or the `function` method:

```
sage: y = var('y'); u = sin(x) + x*cos(y)
sage: v = u.function(x, y); v
(x, y) |--> x*cos(y) + sin(x)
sage: w(x, y) = u; w
(x, y) |--> x*cos(y) + sin(x)
```

Symbolic functions are useful to represent mathematical functions. They differ from Python functions or *procedures*, which are programming constructions described in Chapter 3. The difference between symbolic functions and Python functions is similar to the difference between symbolic variables and Python variables, described in §1.2.5.

A symbolic function can be used like an expression, which is not the case for Python functions; for example, the `expand` method does not exist for the latter.

```
sage: p = (x+y)*(x+1)^2
sage: p2 = p.expand(); p2
x^3 + x^2*y + 2*x^2 + 2*x*y + x + y
```

whereas the `collect` method groups terms together according to the powers of a given variable:

```
sage: p2.collect(x)
x^3 + x^2*(y + 2) + x*(2*y + 1) + y
```

Those methods do not only apply to polynomials in symbolic variables, but also to polynomials in more complex sub-expressions like $\sin x$:

```
sage: ((x+y+sin(x))^2).expand().collect(sin(x))
x^2 + 2*x*y + y^2 + 2*(x + y)*sin(x) + sin(x)^2
```

For rational functions, the `combine` method enables us to group together terms with common denominator; the `partial_fraction` method performs the partial fraction decomposition over $\mathbb{Q}$. (To specify a different decomposition field, we refer the reader to §7.4.)

The more useful representations are the expanded form for a polynomial, and the reduced form P/Q with P and Q expanded in the case of a fraction. When

Polynomial $p = zx^2 + x^2 - (x^2 + y^2)(ax - 2by) + zy^2 + y^2$	
`p.expand().collect(x)`	$-ax^3 - axy^2 + 2\,by^3 + (2\,by + z + 1)x^2 + y^2z + y^2$
`p.collect(x).collect(y)`	$2\,bx^2y + 2\,by^3 - (ax - z - 1)x^2 - (ax - z - 1)y^2$
`p.expand()`	$-ax^3 + 2\,bx^2y - axy^2 + 2\,by^3 + x^2z + y^2z + x^2 + y^2$
`p.factor()`	$-(ax - 2\,by - z - 1)\left(x^2 + y^2\right)$
`p.factor_list()`	$\left[(ax - 2\,by - z - 1, 1)\,, \left(x^2 + y^2, 1\right), (-1, 1)\right]$

Fraction $r = \frac{x^3+x^2y+3\,x^2+3\,xy+2\,x+2\,y}{x^3+2\,x^2+xy+2\,y}$	
`r.simplify_rational()`	$\frac{x^2+(x+1)y+x}{x^2+y}$
`r.factor()`	$\frac{(x+y)(x+1)}{x^2+y}$
`r.factor().expand()`	$\frac{x^2}{x^2+y} + \frac{xy}{x^2+y} + \frac{x}{x^2+y} + \frac{y}{x^2+y}$

Fraction $r = \frac{(x-1)x}{x^2-7} + \frac{y^2}{x^2-7} + \frac{b}{a} + \frac{c}{a} + \frac{1}{x+1}$	
`r.combine()`	$\frac{(x-1)x+y^2}{x^2-7} + \frac{b+c}{a} + \frac{1}{x+1}$

Fraction $r = \frac{1}{(x^3+1)y^2}$	
`r.partial_fraction(x)`	$\frac{-(x-2)}{3\left(x^2-x+1\right)y^2} + \frac{1}{3\,(x+1)y^2}$

TABLE 2.1 – Polynomials and fractions.

two polynomials or fractions are written in this form, it suffices to compare their coefficients to decide if they are equal: we say they are in *normal form*.

2.1.3 Usual Mathematical Functions

Most mathematical functions are known to Sage, in particular the trigonometric functions, the logarithm and the exponential: they are summarised in Table 2.2.

Knowing how to transform such functions is crucial. To simplify an expression or a symbolic function, the `simplify` method is available:

```
sage: (x^x/x).simplify()
x^(x - 1)
```

However, for more subtle simplifications, the desired kind of simplification should be explicit:

```
sage: f = (e^x-1) / (1+e^(x/2)); f.canonicalize_radical()
e^(1/2*x) - 1
```

For example, to simplify trigonometric expressions, the `simplify_trig` method should be used:

```
sage: f = cos(x)^6 + sin(x)^6 + 3 * sin(x)^2 * cos(x)^2
sage: f.simplify_trig()
1
```

Usual mathematical functions	
Exponential and logarithm	`exp, log`
Logarithm in base a	`log(x, a)`
Trigonometric functions	`sin, cos, tan`
Inverse trigonometric functions	`arcsin, arccos, arctan`
Hyperbolic functions	`sinh, cosh, tanh`
Inverse hyperbolic functions	`arcsinh, arccosh, arctanh`
Integer part, etc.	`floor, ceil, trunc, round`
Square and n-th root	`sqrt, nth_root`
Rewriting trigonometric expressions	
Simplification	`simplify_trig`
Linearisation	`reduce_trig`
Anti-linearisation	`expand_trig`

TABLE 2.2 – Usual functions and simplification.

To linearise (resp. anti-linearise) a trigonometric expression, we use `reduce_trig` (resp. `expand_trig`):

```
sage: f = cos(x)^6; f.reduce_trig()
1/32*cos(6*x) + 3/16*cos(4*x) + 15/32*cos(2*x) + 5/16
sage: f = sin(5 * x); f.expand_trig()
5*cos(x)^4*sin(x) - 10*cos(x)^2*sin(x)^3 + sin(x)^5
```

Expressions containing factorials can also be simplified:

```
sage: n = var('n'); f = factorial(n+1)/factorial(n)
sage: f.simplify_factorial()
n + 1
```

The `simplify_rational` method tries to simplify a fraction; whereas to simplify square roots, logarithms or exponentials, the `canonicalize_radical` method is recommended:

```
sage: f = sqrt(abs(x)^2); f.canonicalize_radical()
abs(x)
sage: f = log(x*y); f.canonicalize_radical()
log(x) + log(y)
```

The `simplify_full` command applies the methods `simplify_factorial`, `simplify_rectform`, `simplify_trig`, `simplify_rational` and `expand_sum` (in that order).

All that is needed to determine the variation of a function (derivatives, asymptotes, extrema, localisation of zeroes and graph drawing) can be easily obtained using a computer algebra system. The main Sage operations applying to functions are presented in §2.3.

2.1.4 Assumptions

During a computation, the symbolic variables appearing in expressions are in general considered as taking potentially any value in the complex plane. This

might be a problem when a parameter represents a quantity in a restricted domain (for example, a positive real number).

A typical case is the simplification of the expression $\sqrt{x^2}$. The proper way consists of using the `assume` function, which enables us to define the properties of a variable, which can in turn be reverted by the `forget` instruction:

```
sage: assume(x > 0); bool(sqrt(x^2) == x)
True
sage: forget(x > 0); bool(sqrt(x^2) == x)
False
sage: n = var('n'); assume(n, 'integer'); sin(n*pi)
0
```

2.1.5 Some Pitfalls

The Simplification Problem

The examples of §2.1.5 demonstrate how important normal forms are, and in particular the *test of zero*.

Some families of expressions, like polynomials, have a decision procedure for the equality to zero. As a consequence, for those families, a program is able to decide whether a given expression is zero or not. In most cases, this test is done via the reduction to the normal form: the expression is zero if and only if its normal form is 0.

Unfortunately, not all classes of expressions have a normal form, moreover for some classes it is possible to show that no general method is able to decide in a finite amount of time whether an expression is zero. An example of such a class is made of the rational numbers, the constants π, $\log 2$ and a variable, together with additions, subtractions, multiplications, exponentials and the sine function. The repeated use of `numerical_approx`, while increasing the precision, succeeds in most cases to conjecture if a given expression is zero or not; however, it has been proven impossible to write a computer program taking as input an expression of this class, and returning true if this expression is zero, and false otherwise.

The simplification problem is much harder in those classes. Without any normal form, computer algebra systems can only provide some rewriting functions that the user must play with to obtain some result. Some hope is however possible, if we can identify sub-classes of expressions which have a normal form, and if we know which methods should be applied to compute those normal forms. The Sage approach to handle those issues is presented in more details in Chapter 5.

Let c be a slightly complex expression:

```
sage: a = var('a')
sage: c = (a+1)^2 - (a^2+2*a+1)
```

where we want to solve the equation $cx = 0$ in the variable x:

```
sage: eq =  c * x == 0
```

One might be tempted to divide out this equation by c before solving it:

```
sage: eq2 = eq / c; eq2
x == 0
sage: solve(eq2, x)
[x == 0]
```

Fortunately, Sage avoids this mistake:

```
sage: solve(eq, x)
[x == x]
```

Sage was able to correctly solve the equation because the coefficient c is a polynomial expression. It is thus easy to check whether c is zero, by expanding it:

```
sage: expand(c)
0
```

and use the fact that two mathematically identical polynomials share the same expanded form, or said otherwise, that the expanded form is a normal form for polynomials.

However, on a slightly more complex example, Sage does not avoid the pitfall:

```
sage: c = cos(a)^2 + sin(a)^2 - 1
sage: eq = c*x == 0
sage: solve(eq, x)
[x == 0]
```

even if Sage is able to correctly simplify and test to zero this expression:

```
sage: c.simplify_trig()
0
sage: c.is_zero()
True
```

2.2 Equations

We now deal with equations and how to solve them; the main functions are summarised in Table 2.3.

2.2.1 Explicit Solving

Let us consider the following equation, with unknown z and parameter φ:

$$z^2 - \frac{2}{\cos\varphi} z + \frac{5}{\cos^2\varphi} - 4 = 0, \quad \text{with } \varphi \in \left]-\frac{\pi}{2}, \frac{\pi}{2}\right[.$$

It is written in Sage:

Scalar equations	
Symbolic solution	`solve`
Roots (with multiplicities)	`roots`
Numerical solving	`find_root`

Vector and functional equations	
Solving linear equations	`solve_right, solve_left`
Solving differential equations	`desolve`
Solving recurrences	`rsolve`

TABLE 2.3 – Solving equations.

```
sage: z, phi = var('z, phi')
sage: eq =  z**2 - 2/cos(phi)*z + 5/cos(phi)**2 - 4 == 0; eq
z^2 - 2*z/cos(phi) + 5/cos(phi)^2 - 4 == 0
```

We can extract the left-hand (resp. right-hand) side with the `lhs` (resp. `rhs`) method:

```
sage: eq.lhs()
```
$z^2 - \frac{2\,z}{\cos(\varphi)} + \frac{5}{\cos(\varphi)^2} - 4$
```
sage: eq.rhs()
```
0

then solve it with `solve`:

```
sage: solve(eq, z)
```
$\left[z = -\frac{2\sqrt{\cos(\varphi)^2-1}-1}{\cos(\varphi)}, z = \frac{2\sqrt{\cos(\varphi)^2-1}+1}{\cos(\varphi)}\right]$

Let us now solve the equation $y^7 = y$.

```
sage: y = var('y'); solve(y^7==y, y)
[y == 1/2*I*sqrt(3) + 1/2, y == 1/2*I*sqrt(3) - 1/2, y == -1,
 y == -1/2*I*sqrt(3) - 1/2, y == -1/2*I*sqrt(3) + 1/2, y == 1, y == 0]
```

The roots of the equation can be returned as an object of type *dictionary* (cf. §3.3.9):

```
sage: solve(x^2-1, x, solution_dict=True)
[{x: -1}, {x: 1}]
```

The `solve` command can also solve systems of equations:

```
sage: solve([x+y == 3, 2*x+2*y == 6], x, y)
[[x == -r1 + 3, y == r1]]
```

This linear system being underdetermined, the variable allowing to parametrise the set of solutions is a real number named `r1`, `r2`, etc. If this parameter is known to be an integer, it is named `z1`, `z2`, etc. (below, `z...` stands for `z36`, `z60`, or similar, according to the Sage version):

```
sage: solve([cos(x)*sin(x) == 1/2, x+y == 0], x, y)
[[x == 1/4*pi + pi*z..., y == -1/4*pi - pi*z...]]
```

The `solve` command can also solve inequalities:

```
sage: solve(x^2+x-1 > 0, x)
[[x < -1/2*sqrt(5) - 1/2], [x > 1/2*sqrt(5) - 1/2]]
```

Sometimes, `solve` returns the solutions of a system as floating-point numbers. For example, let us solve in $\mathbb{C}^3$ the following system:

$$\begin{cases} x^2yz &= 18, \\ xy^3z &= 24, \\ xyz^4 &= 6. \end{cases}$$

```
sage: x, y, z = var('x, y, z')
sage: solve([x^2 * y * z == 18, x * y^3 * z == 24,\
....:        x * y * z^4 == 6], x, y, z)
[[x == 3, y == 2, z == 1],
 [x == (1.337215067329613 - 2.685489874065195*I),
  y == (-1.700434271459228 + 1.052864325754712*I),
  z == (0.9324722294043555 - 0.3612416661871523*I)], ...]
```

Sage returns here 17 tuples, among which 16 are approximate complex solutions. To obtain a fully symbolic solution, we refer to Chapter 9.

To solve equations numerically, the `find_root` function takes as input a function of one variable or a symbolic equality, and the bounds of the interval in which to search. Sage does not find any symbolic solution to this equation:

```
sage: expr = sin(x) + sin(2 * x) + sin(3 * x)
sage: solve(expr, x)
[sin(3*x) == -sin(2*x) - sin(x)]
```

Two choices are then possible: either a numerical solution,

```
sage: find_root(expr, 0.1, pi)
2.0943951023931957
```

or first rewrite the expression:

```
sage: f = expr.simplify_trig(); f
2*(2*cos(x)^2 + cos(x))*sin(x)
sage: solve(f, x)
[x == 0, x == 2/3*pi, x == 1/2*pi]
```

Last but not least, the `roots` function gives the roots of an equation with their multiplicity. The ring in which solutions are looked for can be given; with `RR` $\approx \mathbb{R}$ or `CC` $\approx \mathbb{C}$, we obtain floating-point roots. The solving method is specific to the given equation, contrary to `find_roots` which uses a generic method.

Let us consider the degree-3 equation $x^3 + 2\,x + 1 = 0$. This equation has a negative discriminant, thus it has a real root and two complex roots, which are given by the `roots` method:

```
sage: (x^3+2*x+1).roots(x)
```

$$\left[\left(-\frac{1}{2}\left(I\sqrt{3}+1\right)\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}+\frac{-\left(I\sqrt{3}-1\right)}{3\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}},1\right),\right.$$
$$\left(-\frac{1}{2}\left(-I\sqrt{3}+1\right)\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}+\frac{-\left(-I\sqrt{3}-1\right)}{3\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}},1\right),$$
$$\left.\left(\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}+\frac{-2}{3\left(\frac{1}{18}\sqrt{3}\sqrt{59}-\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}},1\right)\right]$$

```
sage: (x^3+2*x+1).roots(x, ring=RR)
```

$[(-0.453397651516404, 1)]$

```
sage: (x^3+2*x+1).roots(x, ring=CC)
```

$[(-0.453397651516404, 1), (0.226698825758202 - 1.46771150871022 * I, 1),$
$(0.226698825758202 + 1.46771150871022 * I, 1)]$

2.2.2 Equations with no Explicit Solution

In most cases, as soon as the equation or system becomes too complex, no explicit solution can be found:

```
sage: solve(x^(1/x)==(1/x)^x, x)
[(1/x)^x == x^(1/x)]
```

However, this is not necessarily a limitation! Indeed, a specificity of computer algebra is the ability to manipulate objects defined by equations, and in particular to compute their properties, without solving them explicitly. Even better: in some cases, the equation defining a mathematical object is the best algorithmic representation for it.

For example, a function given by a linear differential equation and initial conditions is perfectly defined. The set of solutions of linear differential equations is closed under sum and product (among other operations), and thus forms an important class where equality to zero can be decided. However, if we explicitly solve such an equation, the obtained solution might be part of a much larger class where very few questions are decidable.

```
sage: y = function('y')(x)
sage: desolve(diff(y,x,x) + x*diff(y,x) + y == 0, y, [0,0,1])
-1/2*I*sqrt(2)*sqrt(pi)*erf(1/2*I*sqrt(2)*x)*e^(-1/2*x^2)
```

We will go back to this in more detail in Chapter 14 and in §15.1.2.

2.3 Analysis

This section is a quick introduction of useful functions in real analysis. For more advanced usage or more details, we refer to the following chapters, in particular about numerical integration (Chapter 14), non-linear equations (Chapter 12), and differential equations (Chapter 10).

2.3.1 Sums

The `sum` function computes symbolic sums. Let us obtain for example the sum of the n first positive integers:

```
sage: k, n = var('k, n')
sage: sum(k, k, 1, n).factor()
```
$\frac{1}{2}\,(n+1)n$

The `sum` function allows simplifications of a binomial expansion:

```
sage: n, k, y = var('n, k, y')
sage: sum(binomial(n,k) * x^k * y^(n-k), k, 0, n)
```
$(x+y)^n$

Here are more examples, among them the sum of the cardinalities of all parts of a set of n elements:

```
sage: k, n = var('k, n')
sage: sum(binomial(n,k), k, 0, n),\
....: sum(k * binomial(n, k), k, 0, n),\
....: sum((-1)^k*binomial(n,k), k, 0, n)
```
$\left(2^n, 2^{n-1}n, 0\right)$

Finally, some examples of geometric sums:

```
sage: a, q, k, n = var('a, q, k, n')
sage: sum(a*q^k, k, 0, n)
```
$\frac{aq^{n+1}-a}{q-1}$

To compute the corresponding power series, we should tell Sage that the modulus[2] of q is smaller than 1:

```
sage: assume(abs(q) < 1)
sage: sum(a*q^k, k, 0, infinity)
```
$-\frac{a}{q-1}$

```
sage: forget(); assume(q > 1); sum(a*q^k, k, 0, infinity)
Traceback (most recent call last):
...
ValueError: Sum is divergent.
```

[2]Remember that by default, symbolic variables represent complex values.

Exercise 2 (Computing a sum by recurrence)**.** Compute, without using the `sum` command, the sum of p-powers of integers from 0 to n, for $p = 1, ..., 4$:

$$S_n(p) = \sum_{k=0}^{n} k^p.$$

The following recurrence can be useful to compute this sum:

$$S_n(p) = \frac{1}{p+1}\left((n+1)^{p+1} - \sum_{j=0}^{p-1}\binom{p+1}{j} S_n(j)\right).$$

This recurrence is easily obtained when computing by two different methods the telescopic sum $\sum_{0\leq k\leq n} (k+1)^{p+1} - k^{p+1}$.

2.3.2 Limits

To determine a limit, we use the `limit` command or its alias `lim`. Let us compute the following limits:

a) $\lim_{x\to 8} \dfrac{\sqrt[3]{x} - 2}{\sqrt[3]{x+19} - 3}$;

b) $\lim_{x\to \frac{\pi}{4}} \dfrac{\cos\left(\frac{\pi}{4} - x\right) - \tan x}{1 - \sin\left(\frac{\pi}{4} + x\right)}$.

```
sage: limit((x**(1/3) - 2) / ((x + 19)**(1/3) - 3), x = 8)
9/4
sage: f(x) = (cos(pi/4-x)-tan(x))/(1-sin(pi/4 + x))
sage: limit(f(x), x = pi/4)
Infinity
```

The last output says that one of the limits to the left or to the right is infinite. To know more about this, we study the limits to the left (`minus`) and to the right (`plus`), with the `dir` option:

```
sage: limit(f(x), x = pi/4, dir='minus')
+Infinity
sage: limit(f(x), x = pi/4, dir='plus')
-Infinity
```

2.3.3 Sequences

The above functions enable us to study sequences of numbers. We illustrate this by comparing the growth of an exponential sequence and a geometric sequence.

EXAMPLE. (*A sequence study*) Let us consider the sequence $u_n = \frac{n^{100}}{100^n}$. Compute the first 10 terms. How does the sequence vary? What is the sequence limit? From which value of n does $u_n \in \left]0, 10^{-8}\right[$ hold?

1. To define the term of order n, we use a symbolic function. We then compute the first 10 terms by hand (loops will be introduced in Chapter 3):

```
sage: u(n) = n^100 / 100^n
sage: u(1.);u(2.);u(3.);u(4.);u(5.);u(6.);u(7.);u(8.);u(9.);u(10.)
0.0100000000000000
1.26765060022823e26
5.15377520732011e41
1.60693804425899e52
7.88860905221012e59
6.53318623500071e65
3.23447650962476e70
2.03703597633449e74
2.65613988875875e77
1.00000000000000e80
```

   We could quickly conclude that u_n tends to infinity...

2. To get an idea of the variation of the sequence, we can draw the graph of the function $n \to u_n$ (cf. Figure 2.2).

```
sage: plot(u(x), x, 1, 40)
Graphics object consisting of 1 graphics primitive
```

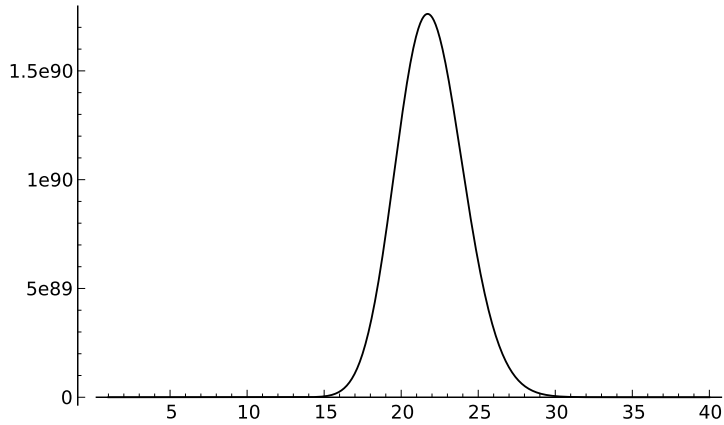

FIGURE 2.2 – Graph of $x \mapsto x^{100}/100^x$.

   We then conjecture that the sequence decreases from index 22 onwards.

```
sage: v(x) = diff(u(x), x); sol = solve(v(x) == 0, x); sol
[x == 100/log(100), x == 0]
sage: floor(sol[0].rhs())
21
```

   The sequence is thus increasing up to index 21, then decreasing after index 22.

Functions and operators	
Derivative	`diff(f(x), x)`
n-th derivative	`diff(f(x), x, n)`
Antiderivative	`integrate(f(x), x)`
Numerical integration	`integral_numerical(f(x), a, b)`
Symbolic summation	`sum(f(i), i, imin, imax)`
Limit	`limit(f(x), x=a)`
Taylor expansion	`taylor(f(x), x, a, n)`
Power series expansion	`f.series(x==a, n)`
Graph of a function	`plot(f(x), x, a, b)`

TABLE 2.4 – Useful functions in analysis.

3. We then compute the limit:

```
sage: limit(u(n), n=infinity)
0
sage: n0 = find_root(u(n) - 1e-8 == 0, 22, 1000); n0
105.07496210187252
```

Since the sequence decreases from index 22 onwards, we deduce that starting from index 106, the sequence always lies in the interval $\left]0, 10^{-8}\right[$.

2.3.4 Power Series Expansions (*)

To compute a power series expansion of order n at x_0, the command to use is `f(x).series(x==x0, n)`.

Let us determine the power series expansion of the following functions:

a) $(1+\arctan x)^{\frac{1}{x}}$ of order 3, at $x_0 = 0$;

b) $\ln(2\sin x)$ of order 3, at $x_0 = \frac{\pi}{6}$.

```
sage: taylor((1+arctan(x))**(1/x), x, 0, 3)
```
$\frac{1}{16}\,x^3e + \frac{1}{8}\,x^2e - \frac{1}{2}\,xe + e$

```
sage: (ln(2*sin(x))).series(x==pi/6, 3)
```
$(\sqrt{3})(-\frac{1}{6}\,\pi + x) + (-2)(-\frac{1}{6}\,\pi + x)^2 + \mathcal{O}\left(-\frac{1}{216}\,(\pi - 6\,x)^3\right)$

To extract the regular part of a power series expansion obtained by `series`, we call the `truncate` method:

```
sage: (ln(2*sin(x))).series(x==pi/6, 3).truncate()
```
$-\frac{1}{18}\,(\pi - 6\,x)^2 - \frac{1}{6}\,\sqrt{3}(\pi - 6\,x)$

The `taylor` command provides asymptotic expansions too. For example, let us see how the function $(x^3+x)^{\frac{1}{3}} - (x^3-x)^{\frac{1}{3}}$ behaves around $+\infty$:

```
sage: taylor((x**3+x)**(1/3) - (x**3-x)**(1/3), x, infinity, 2)
2/3/x
```

Exercise 3 (Computing a symbolic limit)**.** Let f be $\mathcal{C}^3$ around $a \in \mathbb{R}$. Compute

$$\lim_{h \to 0} \frac{1}{h^3} \left(f(a+3h) - 3f(a+2h) + 3f(a+h) - f(a) \right).$$

Generalisation?

EXAMPLE. (*Machin's formula*) Prove the following formula:

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}.$$

The astronomer John Machin (1680-1752) used this formula and the series expansion of arctan to compute hundred digits of π in 1706.

We first notice that $4 \arctan \frac{1}{5}$ and $\frac{\pi}{4} + \arctan \frac{1}{239}$ admit the same tangent:

```
sage: tan(4*arctan(1/5)).simplify_trig()
120/119
sage: tan(pi/4+arctan(1/239)).simplify_trig()
120/119
```

Since the real numbers $4 \arctan \frac{1}{5}$ and $\frac{\pi}{4} + \arctan \frac{1}{239}$ are both in the open interval $]0, \pi[$, they are equal. To obtain an approximation of π, we thus proceed as follows:

```
sage: f = arctan(x).series(x, 10); f
1*x + (-1/3)*x^3 + 1/5*x^5 + (-1/7)*x^7 + 1/9*x^9 + Order(x^10)
sage: (16*f.subs(x==1/5) - 4*f.subs(x==1/239)).n(); pi.n()
3.14159268240440
3.14159265358979
```

Exercise 4 (A formula due to Gauss)**.** The following formula required 20 pages of factorisation tables in the edition of Gauss' works (cf. *Werke*, ed. Königl. Ges. d. Wiss. Göttingen, vol. 2, p. 477-502):

$$\frac{\pi}{4} = 12 \arctan \frac{1}{38} + 20 \arctan \frac{1}{57} + 7 \arctan \frac{1}{239} + 24 \arctan \frac{1}{268}.$$

1. Define $\theta = 12 \arctan \frac{1}{38} + 20 \arctan \frac{1}{57} + 7 \arctan \frac{1}{239} + 24 \arctan \frac{1}{268}$. Verify with Sage that $\tan \theta = 1$.
2. Justify the inequality: $\forall x \geq 0, \arctan x \leq x$. Deduce Gauss' formula.
3. Approximate the arctan function by its Taylor expansion of order 21 at 0, and deduce a new approximation of π.

2.3.5 Series (*)

The commands introduced earlier can be used to perform computations on series. Let us give some examples.

EXAMPLE. (*Evaluation of the Riemann zeta function*)

```
sage: k = var('k')
sage: sum(1/k^2, k, 1, infinity),\
....: sum(1/k^4, k, 1, infinity),\
....: sum(1/k^5, k, 1, infinity)
```

$\left(\frac{1}{6}\,\pi^2, \frac{1}{90}\,\pi^4, \zeta(5)\right)$

EXAMPLE. (*A formula due to Ramanujan*) Using the first 12 terms of the following series, we give an approximation of π and we compare it with the value given by Sage.

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{+\infty} \frac{(4k)! \cdot (1103 + 26390\,k)}{(k!)^4 \cdot 396^{4k}}.$$

```
sage: s = 2*sqrt(2)/9801*(sum((factorial(4*k)) * (1103+26390*k) /
....:     ((factorial(k)) ^ 4 * 396 ^ (4 * k)) for k in (0..11)))
sage: (1/s).n(digits=100)
3.1415926535897932384626433832795028841971693993751058209 74...
sage: (pi-1/s).n(digits=100).n()
-4.36415445739398e-96
```

We notice that the partial sum of the first 12 terms already yields 95 significant digits of π!

EXAMPLE. (*Convergence of a series*) Let us study the convergence of the series

$$\sum_{n\geq 0} \sin\left(\pi\sqrt{4\,n^2+1}\right).$$

To get an asymptotic expansion of the general term, we use the 2π-periodicity of the sine function, so that the sine argument tends to 0:

$$u_n = \sin\left(\pi\sqrt{4\,n^2+1}\right) = \sin\left[\pi\left(\sqrt{4\,n^2+1} - 2n\right)\right].$$

We can then apply the `taylor` function to this new expression of the general term:

```
sage: n = var('n'); u = sin(pi*(sqrt(4*n^2+1)-2*n))
sage: taylor(u, n, infinity, 3)
```

$\frac{\pi}{4\,n} - \frac{6\,\pi+\pi^3}{384\,n^3}$

We deduce $u_n \sim \frac{\pi}{4\,n}$. Therefore, by comparison with the series defining the Riemann zeta function, the series $\sum_{n\geq 0} u_n$ diverges.

Exercise 5 (Asymptotic expansion of a sequence)**.** It is easy to show (for example, using a bijection) that for all $n \in \mathbb{N}$, the equation $\tan x = x$ has exactly one solution x_n in the interval $[n\pi, n\pi + \frac{\pi}{2}[$. Give an asymptotic expansion of x_n to order 6 in $+\infty$.

2.3.6 Derivatives

The `derivative` function (with alias `diff`) computes the derivative of a symbolic expression or function.

```
sage: diff(sin(x^2), x)
2*x*cos(x^2)
sage: function('f')(x); function('g')(x); diff(f(g(x)), x)
f(x)
g(x)
D[0](f)(g(x))*diff(g(x), x)
sage: diff(ln(f(x)), x)
diff(f(x), x)/f(x)
```

2.3.7 Partial Derivatives (*)

The `derivative` (or `diff`) command also computes iterated or partial derivatives.

```
sage: f(x,y) = x*y + sin(x^2) + e^(-x); derivative(f, x)
(x, y) |--> 2*x*cos(x^2) + y - e^(-x)
sage: derivative(f, y)
(x, y) |--> x
```

EXAMPLE. Let us check that the following function is harmonic[3]:
$$f(x,y) = \tfrac{1}{2}\ln(x^2+y^2) \text{ for all } (x,y) \neq (0,0).$$

```
sage: x, y = var('x, y'); f = ln(x**2+y**2) / 2
sage: delta = diff(f,x,2) + diff(f,y,2)
sage: delta.simplify_rational()
0
```

Exercise 6 (A counter-example due to Peano to Schwarz' theorem)**.** Let f be the function from $\mathbb{R}^2$ to $\mathbb{R}$ defined by:

$$f(x,y) = \begin{cases} xy\frac{x^2-y^2}{x^2+y^2} & \text{if } (x,y) \neq (0,0), \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$$

Does $\partial_1\partial_2 f(0,0) = \partial_2\partial_1 f(0,0)$ hold?

2.3.8 Integrals

To compute an indefinite or definite integral, we use `integrate` as a function or method (or its alias `integral`):

```
sage: sin(x).integral(x, 0, pi/2)
1
sage: integrate(1/(1+x^2), x)
arctan(x)
```

[3] A function f is said *harmonic* when its Laplacian $\Delta f = \partial_1^2 f + \partial_2^2 f$ is zero.

```
sage: integrate(1/(1+x^2), x, -infinity, infinity)
pi
sage: integrate(exp(-x**2), x, 0, infinity)
1/2*sqrt(pi)
```

```
sage: integrate(exp(-x), x, -infinity, infinity)
Traceback (most recent call last):
...
ValueError: Integral is divergent.
```

EXAMPLE. Let us compute, for $x \in \mathbb{R}$, the integral $\varphi(x) = \displaystyle\int_0^{+\infty} \frac{x \cos u}{u^2 + x^2} \, \mathrm{d}u$.

```
sage: u = var('u'); f = x * cos(u) / (u^2 + x^2)
sage: assume(x>0); f.integrate(u, 0, infinity)
1/2*pi*e^(-x)
sage: forget(); assume(x<0); f.integrate(u, 0, infinity)
-1/2*pi*e^x
```

We thus have: $\forall x \in \mathbb{R}^*, \quad \varphi(x) = \frac{\pi}{2} \cdot \operatorname{sgn}(x) \cdot e^{-|x|}$.

To compute numerically an integral on an interval, we have at our disposal the `integral_numerical` function, which returns a pair, whose first value is the approximation of the integral, while the second value is an estimate of the corresponding error.

```
sage: integral_numerical(sin(x)/x, 0, 1)
(0.946083070367183, 1.0503632079297087e-14)
sage: g = integrate(exp(-x**2), x, 0, infinity)
sage: g, g.n()
(1/2*sqrt(pi), 0.886226925452758)
sage: approx = integral_numerical(exp(-x**2), 0, infinity)
sage: approx
(0.8862269254527568, 1.714774436012769e-08)
sage: approx[0]-g.n()
-1.11022302462516e-15
```

Exercise 7 (The BBP formula (*))**.** Let us establish by a symbolic computation the BBP formula (or Bailey-Borwein-Plouffe formula); this formula directly gives the n-th digit of π in radix 2 (or 16) without computing the previous digits, and with very little memory usage and time. For $N \in \mathbb{N}$, let us define

$$S_N = \sum_{n=0}^{N} \left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6} \right) \left(\frac{1}{16} \right)^n .$$

1. Consider the function $f : t \longmapsto 4\sqrt{2} - 8t^3 - 4\sqrt{2}t^4 - 8t^5$. For $N \in \mathbb{N}$, express the following integral as a function of S_N:

$$I_N = \int_0^{1/\sqrt{2}} f(t) \left(\sum_{n=0}^{N} t^{8n} \right) \mathrm{d}t.$$

Usual functions on vectors	
Vector construction	`vector`
Cross product	`cross_product`
Scalar product	`dot_product`
Norm of a vector	`norm`

TABLE 2.5 – Vector computations.

2. For $N \in \mathbb{N}$, let us define $J = \displaystyle\int_0^{1/\sqrt{2}} \frac{f(t)}{1-t^8}\,\mathrm{d}t$. Prove $\lim\limits_{N\to+\infty} S_N = J$.
3. Prove the BBP formula:

$$\sum_{n=0}^{+\infty} \left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6}\right)\left(\frac{1}{16}\right)^n = \pi.$$

This fabulous formula was found on September 19, 1995 by Simon Plouffe, in collaboration with David Bailey and Peter Borwein. Thanks to computation derived from the BBP formula, the 4 000 000 000 000 000-th digit of π in radix 2 was computed in 2001.

2.4 Basic Linear Algebra (*)

In this section, we describe the basic useful functions in linear algebra: first operations on vectors, then on matrices. For more details, we refer the reader to Chapter 8 for symbolic linear algebra, and to Chapter 13 for numerical linear algebra.

2.4.1 Solving Linear Systems

To solve a linear system, we can use the `solve` function, already seen above.

Exercise 8 (Polynomial approximation of the sine function)**.** Determine the polynomial of degree at most 5 which approximates best, in the least squares sense, the sine function on the interval $[-\pi, \pi]$:

$$\alpha_5 = \min\left\{\int_{-\pi}^{\pi} |\sin x - P(x)|^2\,\mathrm{d}x \;\middle|\; P \in \mathbb{R}_5[x]\right\}.$$

2.4.2 Vector Computations

The basic functions for manipulating vectors are summarised in Table 2.5.

We can use those functions to deal with the following exercise.

Exercise 9 (Gauss' problem)**.** Consider a satellite in orbit around the Earth, and assume we know three points of its orbit: A_1, A_2 and A_3. Determine from these three points the orbit parameters of this satellite.

Let us denote O the centre of the Earth. The points O, A_1, A_2 and A_3 are clearly in the same plane, namely the plane defined by the satellite orbit. The satellite orbit is

an ellipse of which O is a focal point. We can choose as coordinate system $(O; \overrightarrow{\imath}, \overrightarrow{\jmath})$ in such a way that the ellipse equation in polar coordinates is $r = \frac{p}{1-e\cos\theta}$ where e is the ellipse eccentricity, and p its parameter. We will note $\overrightarrow{r_i} = \overrightarrow{OA_i}$ and $r_i = \|\overrightarrow{r_i}\|$ for $i \in \{1, 2, 3\}$. We then consider the three following vectors deduced from A_1, A_2 and A_3:

$$\begin{aligned}
\overrightarrow{D} &= \overrightarrow{r_1} \wedge \overrightarrow{r_2} + \overrightarrow{r_2} \wedge \overrightarrow{r_3} + \overrightarrow{r_3} \wedge \overrightarrow{r_1}, \\
\overrightarrow{S} &= (r_1 - r_3) \cdot \overrightarrow{r_2} + (r_3 - r_2) \cdot \overrightarrow{r_1} + (r_2 - r_1) \cdot \overrightarrow{r_3}, \\
\overrightarrow{N} &= r_3 \cdot (\overrightarrow{r_1} \wedge \overrightarrow{r_2}) + r_1 \cdot (\overrightarrow{r_2} \wedge \overrightarrow{r_3}) + r_2 \cdot (\overrightarrow{r_3} \wedge \overrightarrow{r_1}).
\end{aligned}$$

1. Show that $\overrightarrow{\imath} \wedge \overrightarrow{D} = -\frac{1}{e}\overrightarrow{S}$ and deduce the ellipse eccentricity.
2. Show that $\overrightarrow{\imath}$ is colinear with the vector $\overrightarrow{S} \wedge \overrightarrow{D}$.
3. Show that $\overrightarrow{\imath} \wedge \overrightarrow{N} = -\frac{p}{e}\overrightarrow{S}$ and deduce the ellipse parameter p.
4. Compute the half major axis a of the ellipse in term of the parameter p and the eccentricity e.
5. *Numerical application:* in the plane with a Cartesian coordinate system, we consider the following points:

$$A_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad A_2 \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad A_3 \begin{pmatrix} 3.5 \\ 0 \end{pmatrix}, \quad O \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Determine numerically the characteristics of the unique ellipse having O as focal point and passing through the three points A_1, A_2 and A_3.

2.4.3 Matrix Computations

To construct a matrix, what you want is the `matrix` function, which allows to optionally specify the base ring (or field):

```
sage: A = matrix(QQ, [[1,2],[3,4]]); A
[1 2]
[3 4]
```

To find a particular solution of the matrix equation $Ax = b$ (resp. $xA = b$), we call the `solve_right` function (resp. `solve_left`). To find *all* the solutions, we should add to that particular solution the general solution of the associated homogeneous equation. To solve a homogeneous equation $Ax = 0$ (resp. $xA = 0$), we use the `right_kernel` (resp. `left_kernel`) function, as in the following exercise.

Exercise 10 (Basis of vector subspace)**.** 1. Determine a basis of the space of solutions of the linear homogeneous system corresponding to the matrix:

$$A = \begin{pmatrix} 2 & -3 & 2 & -12 & 33 \\ 6 & 1 & 26 & -16 & 69 \\ 10 & -29 & -18 & -53 & 32 \\ 2 & 0 & 8 & -18 & 84 \end{pmatrix}.$$

2. Determine a basis of the space F generated by the columns of A.
3. Characterise F by one or several equations.

Exercise 11 (A matrix equation)**.** Let us recall the factorisation lemma for linear maps. Let E, F, G be $\mathbb{K}$-vector spaces of finite dimension. Let $u \in \mathcal{L}(E, F)$ and $v \in \mathcal{L}(E, G)$. Then the following assertions are equivalent:

Usual functions on matrices	
Construction of a matrix	`matrix`
Solving a matrix equation	`solve_right, solve_left`
Right and left kernel	`right_kernel, left_kernel`
Row echelon form	`echelon_form`
Column-generated vector space	`column_space`
Row-generated vector space	`row_space`
Matrix concatenation	`matrix_block`

Matrix reduction	
Eigenvalues of a matrix	`eigenvalues`
Eigenvectors of a matrix	`eigenvectors_right`
Jordan normal form reduction	`jordan_form`
Minimal polynomial	`minimal_polynomial`
Characteristic polynomial	`characteristic_polynomial`

TABLE 2.6 – Matrix computations.

i) there exists $w \in \mathcal{L}(F,G)$ such that $v = w \circ u$,

ii) $\mathrm{Ker}(u) \subset \mathrm{Ker}(v)$.

We search all solutions to this problem in a concrete case. Let

$$A = \begin{pmatrix} -2 & 1 & 1 \\ 8 & 1 & -5 \\ 4 & 3 & -3 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 1 & 2 & -1 \\ 2 & -1 & -1 \\ -5 & 0 & 3 \end{pmatrix}.$$

Determine all solutions $B \in \mathcal{M}_3(\mathbb{R})$ of the equation $A = BC$.

2.4.4 Reduction of a Square Matrix

To study the eigenvalues and eigenvectors of a matrix, all functions from Table 2.6 are available.

Those functions will be detailed in Chapter 8. We only give here a few simple examples of their usage.

EXAMPLE. Is the matrix $A = \begin{pmatrix} 2 & 4 & 3 \\ -4 & -6 & -3 \\ 3 & 3 & 1 \end{pmatrix}$ diagonalisable? Triangularisable?

We start by defining the matrix A by giving the base field (`QQ`=$\mathbb{Q}$), then we determine its eigenvalues and eigenvectors.

```
sage: A = matrix(QQ, [[2,4,3],[-4,-6,-3],[3,3,1]])
sage: A.characteristic_polynomial()
x^3 + 3*x^2 - 4
sage: A.eigenvalues()
[1, -2, -2]
sage: A.minimal_polynomial().factor()
(x - 1) * (x + 2)^2
```

The minimal polynomial of A admits a simple root and a double root; thus A is not diagonalisable. However, its minimal polynomial being split into linear factors, A is triangularisable.

```
sage: A.eigenvectors_right()
```
$[(1, [(1, -1, 1)], 1), (-2, [(1, -1, 0)], 2)]$

```
sage: A.jordan_form(transformation=True)
```
$$\left(\left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -2 & 1 \\ 0 & 0 & -2 \end{array}\right), \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}\right)$$

EXAMPLE. Let us diagonalise the matrix $A = \begin{pmatrix} 1 & -1/2 \\ -1/2 & -1 \end{pmatrix}$. We first try the `jordan_form` function:

```
sage: A = matrix(QQ, [[1,-1/2],[-1/2,-1]])
sage: A.jordan_form()
Traceback (most recent call last):
...
RuntimeError: Some eigenvalue does not exist in Rational Field.
```

A small difficulty appears here: the eigenvalues are not all rational.

```
sage: A = matrix(QQ, [[1,-1/2],[-1/2,-1]])
sage: A.minimal_polynomial()
x^2 - 5/4
```

We therefore have to change the base field.

```
sage: R = QQ[sqrt(5)]
sage: A = A.change_ring(R)
sage: A.jordan_form(transformation=True, subdivide=False)
```
$$\left(\begin{pmatrix} \frac{1}{2}sqrt_5 & 0 \\ 0 & -\frac{1}{2}sqrt_5 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -sqrt_5 + 2 & sqrt_5 + 2 \end{pmatrix}\right)$$

This is to be interpreted as:

$$\left(\begin{pmatrix} \frac{1}{2}\sqrt{5} & 0 \\ 0 & -\frac{1}{2}\sqrt{5} \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -\sqrt{5} + 2 & \sqrt{5} + 2 \end{pmatrix}\right)$$

EXAMPLE. Let us diagonalise the matrix $A = \begin{pmatrix} 2 & \sqrt{6} & \sqrt{2} \\ \sqrt{6} & 3 & \sqrt{3} \\ \sqrt{2} & \sqrt{3} & 1 \end{pmatrix}$

Here, we have to work in an extension of degree 4 of the field $\mathbb{Q}$, for example as follows.

```
sage: K.<sqrt2> = NumberField(x^2 - 2)
sage: L.<sqrt3> = K.extension(x^2 - 3)
sage: A = matrix(L, [[2, sqrt2*sqrt3, sqrt2],  \
....:                [sqrt2*sqrt3, 3, sqrt3],  \
....:                [sqrt2, sqrt3, 1]])
sage: A.jordan_form(transformation=True)
```
$$\left(\begin{pmatrix} 6 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ \frac{1}{2}\sqrt{2}\sqrt{3} & 0 & 1 \\ \frac{1}{2}\sqrt{2} & -\sqrt{2} & -\sqrt{3} \end{pmatrix}\right)$$