SAGE MATH CLOUD: MATHEMATICS IN THE CLOUD

James Quinlan
University of Southern Maine
james.quinlan@maine.edu

Abstract

SAGE is a free open-source (GPL licensed) mathematics software system. It is a broker to many existing software packages. SageMathCloud, (SMC), is an online service that provides a unified technological framework in which to do mathematical work individually or collaboratively. It also integrates LaTeX for mathematical communication. Additionally, the SMC contains course management functionality that can be used to assign, collaborate, edit, and collect mathematics documents. This article introduces basic features of SAGE/SMC and provides brief overview of the syntax for basic arithmetic, number theory operations, algebra, calculus, differential equations, and linear algebra.

1. Introduction

SAGE (*System for Algebra and Geometry Experimentation*) [3] is a mathematical software integrates many specialized mathematics software packages into a web interface. SageMathCloud, (SMC) is a powerful resource for doing mathematics that interfaces with several existing software. With SMC there is no need to download and install software, including a virtual machine for use of SAGE on Windows, or maintain SAGE software (e.g., maintaining updates). Any modern browser (i.e., not IE) and an internet connection is all that is needed.

This article reflects content covered in a workshop presented at the 2016 International Conference on Technology in Collegiate Mathematics (ICTCM). Section 2 introduces basic arithmetic, order of operations, provides a list of common mathematical constants, basic number theory functions, logarithms, roots and radicals, and numerical approximation. Section 3 covers algebra, in particular solving equations, factoring, and simplifying. Section 4 describes how to define single and multi-variable functions. Section 5 covers 2-d and 3-d plotting including polar plots and parametric plots. Section 6 reports on the constructs of calculus and Section 7 shows how to solve a first and second order ordinary differential equation (ODE). Section 8 provides a few basics from Python (loops and conditionals). Finally, Section **??** reviews some features of SMC's course management capabilities. An index is included for quick reference.

2. Getting Started

Start by creating a new SAGE worksheet (.sagews). Commands are entered into *cells* and cells are executed using SHIFT + ENTER. Help with a command can be found within SMC by entering the command followed by a question mark (i.e., command_name?. **Comments** are inserted using hashtag, (i.e., #). For complete documentation see [4].

2.1 Assignment, Arithmetic, & Comparisons

SAGE uses = for assignment (e.g., x=2), and ==, <=, >=, < and > for the comparisons *equal, less than or equal, greater than or equal, less than*, and *greater than*, respectively. In SAGE you can add, subtract multiply, divide, and raise a number (or variable) to a power.

```
5+2;  # Add
5-2;  # Subtract
5*2;  # Multiply
5/2;  # Divide
5^2   # Exponentiate (or 5**2)
```

2.2 Order of Operations

SAGE follows the standard *Arithmetical binary operator precedence* (See Table 1), where the operations are listed here in decreasing order of precedence.

Order	Operation	Description
1	^, **	exponentiation
2	*, /, %	multiplication, division, remainder
3	+, -	addition, subtraction
4	>,<=,>,>=,==,!=	relational operators
5	in, not in	membership
6	not	logical
7	and	logical
8	or	logical

Table 1: *Arithmetical binary operator precedence; listed highest to lowest precedence*

2.3 *Mathematical Constants*

There are several constants defined in SAGE. Additionally, SAGE allows coercing into GAP, PARI/GP, KASH, Maple, etc. A few of the standard mathematical constants defined in SAGE are:

```
pi              # ratio of Circumference of circle to diameter
e               # base of natural logarithm
golden_ratio    # limit of sucessive terms of Fibonacci sequence
NaN             # Not a Number
oo              # Infinity
khinchin        # Khinchin's constant
catalan         # Catalan constant
euler_gamma     # Euler's gamma constant
```

2.4 *Number Theory*

SAGE contains most basic number theory functions such as the greatest common divisor, least common function, remainder , factorial, prime range, prime counting function, divisors, prime divisors , Euler phi function , and many more. Examples with proper syntax listed below.

```
gcd(4,6);            # great common divisor
lcm(4,6);            # least common multiple
factor(24);          # or, 24.factor()
mod(15,4);           # or, 15 % 4
divisors(24)         # lists divisors of 24. Use also 24.divisors();
prime_divisors(24)   # list prime divisors of 24. (i.e., 2,3
prime_range(10)      # list of primes <= 10 (i.e., 2, 3, 5, 7)
prime_pi(25);        # number of primes < 25 (i.e., 9)
euler_phi(10);       # number relative prime to 10
factorial(5);        # 5! = 5*4*3*2*1.  Use also 5.factorial();
```

2.5 *Logarithms*

SMC can calculate logarithms to any base. By default, $b = e$. Results are given in exact form unless a numeric approximation is requested. The syntax for $\log_b(x)$ is:

```
log(x,b)             # log base b of x
```

In particular, to express $\ln 7$, $\log_{10} 100$, and $\log_{12} 192$ in SAGE

```
log(7)             # log base e of 7
log(100,10)        # log base 10 of 100
log(192, 12)       # log base 12 of 192
```

2.6 Roots and Radicals

The square root can be evaluated using `sqrt()` command. Cube roots, fourth roots, etc. are evaluated using fractional exponents.

```
sqrt(2)           # Square root of 2
2^(1/3)           # Cube root of 2
2^(1/4)           # Fourth root of 2
```

2.7 Numerical Evaluation

To display the numeric value of an expression (such as a constant), use `<expression>.n()`. The number of digits or precision can be specified as an option when evaluating. The default is 52 bits of precision, which is about 15 digits.

```
pi.n()               # returns numerical approximation of pi
pi.n(digits=100)     # returns 100 digits of pi
pi.n(prec=100)       # returns 100 bits of precision
```

3. Algebra

SMC uses Maxima as it Computer Algebra System (CAS).

3.1 Solving Equations

With SMC you can symbolically and numerically solve equations. The `solve` function solves equations in SAGE. To use the `solve` function, first define all necessary variables, then specify the equation (or system of equations) to solve as the first argument of the `solve` function, together with the variables for which to solve.

```
x = var('x');
solve(x^2 - 5*x + 6==0, x)
```

If no right-hand side is specified, it defaults to zero. In particular, using `solve(x^2 - 5*x + 6, x)` will produce same results.

It is possible solve equations for one variable in terms of others variables. For example, solve $ax^2 + bx + c = 0$ as follows.

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

It is also possible to solve a system of equations for multiple variables. Here is an example from Jason Grout.

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

3.2 Factor, Expand, and Simplify

To factor, expand, or simplify an expression, use `factor(<expression>)`, `expand(<expression>)`, or `simplify(<expr>)`, respectively. Trigonometric functions are simplified using `trig_simplify()`.

```
factor(x^2-1);
expand((x-1)^2);
simplify(x^2+6*x^2);
```

4. Mathematical Functions

Mathematical functions can be defined using familiar notation.

```
f(x) = <expression>
```

For example, $f(x) = x^2$, $g(x) = \sin(x)$, and $h(x) = \sqrt{x} + 3x - e^x$ are defined in SAGE as

```
f(x) = x^2
g(x) = sin(x)
h(x) = sqrt(x) + 3*x - e^x
```

A function can be easily evaluated using the standard notation. To evaluate $f(5)$ when $f(x) = x^2$, enter:

```
f(x)=x^2;
f(5);
```

4.1 Special Functions

There are many special functions. Here we list a few including: `heaviside(x), dirac_delta(x), un`

4.2 Multivariable Functions

After declaring all variables, use standard (mathematical) notation to express a multivariable function. Remember to specify multiplication operator instead of juxtaposition.

```
x,y,t=var('x y t')
f(x,y)=x^2+y^3;
g(x,y)=e^(x*y)+sin(x)-2*x*y^2;
h(x,y,t)= 4*x*y-3*t^2;
```

5. Plotting

Sage can handle several types of two-dimensional and three-dimensional plots including polar plots, parametric plots, implicit plots, density plot, and contour plots. For full list of plots see plot sectoin of the SAGE documentation [4].

5.1 2-d Plots

Basic two-dimensional plots are produced using the following command:

```
plot(<expression>, <options separated by commas>);
```

Use plotting options (See Table 2) to change the color, opacity, domain, y axis range, linestyle, etc., separated by commas. The default domain is $x \in [-1, 1]$.

Option	**Description**
`figsize`	size of bounding box, (default = 8)
`aspect_ratio`	height to width, (default = 1)
`ymin` & `ymax`	bottom and top of bounding box
`thickness`	line thickness (default 1)
`linestyle`	solid (-, default), : dotted, .- dot-dash
`color`	default=blue,
`alpha`	opacity, default=1
`exclude[]`	Excludes certain $x-$values listed, e.g., `exclude=[0]`

Table 2: *Plotting options*

5.2 *Polar Plots*

```
polar_plot(<polar expression>, <angle domain>, <other options>);
```

Examples: Polar plots of $r = 2 - 2\sin(\theta)$, $r = \sin(3\theta)$, and $r = 2 + 2\cos(\theta)$.

```
theta=var('theta')
x=var('x')
polar_plot(2 - 2*sin(theta), (theta, 0, 2*pi));
polar_plot(sin(3*theta), (theta, 0, pi));
polar_plot(2 + 2*cos(x), (x, 0, 2*pi), color='red', thickness=4);
```

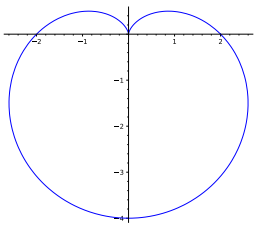

Figure 1: $r = 2 - 2\sin(\theta)$

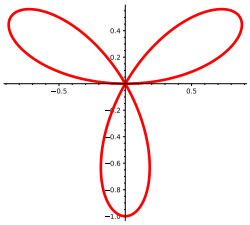

Figure 2: $r = \sin(3\theta)$

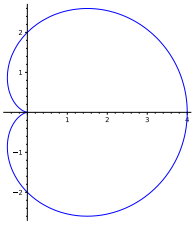

Figure 3: $2 + 2 * cos(x)$

5.3 *Parametric Plots*

Two-dimensional parametric plots use they syntax `parametric_plot( (x(t), y(t)), (t, min, max`

```
t = var('t')
parametric_plot( (cos(t), sin(t)), (t, 0, 2*pi))
```

5.4 *Implicit 2d plots*

```
y=var('y')            # recall: y not defined like x by default
implicit_plot(x^2+y^2-2==4, (x,-3,3), (y,-3,3));
```

5.5 *Line Plot*

```
line([(x0,y0), (x1, y1)]);   # plots line through the points (x0, y0) and (x1, y1
line([(0,0), (1, 1)]);  # Example
```

5.6 *Contour Plots*

```
x,y=var('x y');
contour_plot(f(x,y), (x, a, b), (y, c, d));
contour_plot(x^2+y^2, (x, -4, 4), (y, -4, 4),plot_points=120)   # Example
```

5.7 *Multiple Plots*

Multiple plots can be placed on a single axis using a list (array). However, it is better maintain independence among the plots and using the + operator. In particular,

```
plot([sin(x), cos(x)])        # plots sin(x) and cos(x) on same axis

# multiple plots that are more easily distinguishable
p1=plot(sin(x), -10, 10, color='blue');       # blue is default
p2=plot(cos(x), -10, 10, color='red');
p1+p2;
```

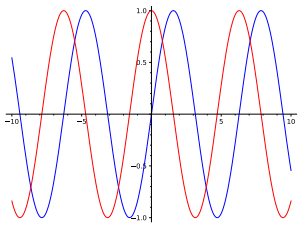

Figure 4: *Multiple Plots on one axis*

This method also works for plotting piecewise functions. For example,

```
p1 = plot(x^2, x, 0, 1)
p2 = plot(-x+2, x, 1, 2)
p3 = plot(x^2-3*x+2, x, 2, 3)
pt1 = point((0, 0), rgbcolor='black', pointsize=30)
pt2 = point((3, 2), rgbcolor='black', pointsize=30)
(p1+p2+p3+pt1+pt2).show(xmin=0, xmax=3, ymin=0, ymax=2)
```

5.8 *3d Plots*

Let $f(x,y) =$<expression in x and y>. To plot $z = f(x,y)$, use SAGE command plot3d as shown below.

```
plot3d(f(x,y), (x,a,b), (y,c,d));
```

Examples of the $3d$ plots for $x^2 - y^2$, $y\sin(x-y)\cos(x)$, and $y^3 - x$ are given below.

```
y=var('y');
plot3d(x^2 - y^2, (x, -2, 2), (y, -2, 2), color='red' );
plot3d(sin(x-y)*y*cos(x),  (x,-3,3),  (y,-3,3));
plot3d(y^3-x,(x,0,3),(y,-1,1));
```

6. Calculus

SAGE can perform various computations related to basic calculus: limits, derivatives, and integrals. Symbolic computations in SAGE interfaces with *Maxima*.

6.1 Limits

The command for limit in SAGE is, naturally, `limit`. The command to calculate $\lim_{x\to a} f(x)$ is:

$$\lim_{x\to a} f(x) \Longleftrightarrow \texttt{ limit(f(x), x=a)}$$

Example: The $\lim_{x\to 0} \sin(x)$ and $\lim_{x\to\infty} \frac{1}{x}$ are calculated in SAGE with the following commands.

```
limit(sin(x), x=0)
limit(1/x, x=oo)
```

6.1.1 One-sided Limits

One-sided limits use the optional parameter, `dir`. Left-limit are specified by either `'-', 'minus'` or `'left'` and right-limits using `'+', 'plus'` or `'right'`.
Example: The one-sided limits $\lim_{x\to 0^+} e^{-1/x}$ and $\lim_{x\to 0^-} e^{-1/x}$ are calculated in SAGE with the following commands.

```
limit(e^(-1/x), x=0, dir='+')
limit(e^(-1/x), x=0, dir='-')
```

6.2 Derivatives

Derivatives are found using either `derivative` or `diff` command. Additionally, either the dot notation or function notation can be used. In particular, the following are equivalent commands for the first derivative of $f(x)$.

```
f(x).derivative();
derivative(f(x));

f(x).diff();
diff(f(x));
```

6.2.1 *Higher Order Derivatives*

For higher order derivatives, include the order inside the parameter parenthesis. Given $f(x)$, the following are equivalent ways to calculute $f'''(x)$.

```
diff(f(x),3);          # equivalently:  f(x).diff(3)
derivative(f(x),3);    # equivalently:  f(x).derivative(3);
```

It is good habit to include the variable with which to differentiate with respect to, although in one-variable case, it is not necessary. The above can be rewritten to specify which variable to differentiate with respect to.

```
diff(f(x),x,3);
derivative(f(x),x,3);
```

6.2.2 *Implicit Differentiation*

For implicit differentiation, y first has to be declared as a function of x.

```
y=function('y')(x)
temp=diff(x^2+y^2-1)
solve (temp,diff(y))
```

6.3 *Integration*

The indefinite integral $\int f(x)$ dx can be found multiple ways.

```
# integrate f with respect to x
integrate(f(x),x)  # equivalently: f(x).integral(x)
```

The definite integral $\int_a^b f(x)$ dx is calculated by including the limits of integration in the `integral` command.

```
integral(f(x), x, a, b)
```

Numerical integration of $\int_a^b f(x)$ dx is calculated with the command `numerical_integral` and returns a tuple with the approximation and error.

```
numerical_integral(f(x),a,b)
```

6.4 *Sums and Series*

To calculate the sum of an expression $f(k)$ use `sum(<expression>, var, start, end)`. For example, in SAGE $\sum_{k=1}^{\infty} \frac{1}{k^4}$ is

```
k=var('k')
sum(1/k^4, k, 1, oo)
```

The binomial theorem can be expressed as follows:

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

6.4.1 Taylor Series

Taylor series approximation of $f(x)$ of order n around the point $x = a$ is given by `f.taylor(x,a,n)`. For example, when $f(x) = e^x$, the fifth order approximation around $x_0 = 1$ is given by,

```
f=e^x;
g=f.taylor(x,1,5)
```

6.5 Multi-variable Calculus

```
f(x,y)=<function of two variables x and y>
```

Typical multi-variate operations are included in **SAGE**.

```
f(x,y)=<function of two variables x and y>
f.gradient([x,y])
f.derivative(x)    # Partial w.r.t x
f.derivative(y)    # Partial w.r.t y
jacobian(f(x,y), (x,y))
```

7. Ordinary Differential Equations

Use `desolve` to find the general solution to a first or second order differential equation typically found in a first semester differential equations course using Maxima. In `SAGE`, the general solution to the differential equation $y''(x) - y = x$ is found with the following command:

```
x = var('x')
sage: y = function('y')(x)
sage: de = diff(y,x,2) - y == x
sage: desolve(de, y)
```

8. Python Programming

SAGE can be embedded in Python code. Assignments in Python are made using =. Indentation level of statements is significant (i.e., conditional statements and loops). One of the most useful functions in Python is the range command. For example, range(3) is the four elements: $0, 1, 2, 3$ and will be used in loops. Comments in Python start with #.

8.1 List, sets, Dictionaries & Comprehensions

Lists (arrays), sets, dictionaries, and comprehensions are a few of the data types in Python.

```
[1, 2, 5, 6, 10]    # An (ordered) List
{7, 3, 2}           # Set

# set builder notation
f(x)=2*x;
X={-2, 0, 1, 2, 3, 4, 8}
Set([f(x) for x in X if x>0])

# Dictionary (set with key:value)
{'sage':'math', 3:7}

# Comprehensions
myList=[i^2 for i in range(10)];
```

8.2 IF-THEN-ELSE

The basic structure of conditional statements, if-then-else, is given below (note the indentation).

```
if <condition>:
    <statement(s)>
elif <condition 2>:
    <statement(s)>
else:
    <statement(s)>
```

8.3 For Loop

The basic structure of a for loop utilizing the range function is below (note the indentation).

```
for k in range(n):
    <statement(s)>
```

9. Conclusion

SMC is a powerful tool for doing mathematics as well as teaching and learning. SAGE has all the features of a scientific calculator, but is way more. SMC incorporates several technologies into a single web environment providing a "one-stop-shop." In addition to being well documented and supported (see [4]), there is no cost, thus making SMC a viable alternative to several expensive software. For further reading on SAGE see [1,2].

References

[1] Gregory V Bard, *Sage for undergraduates*, vol. 87, American Mathematical Soc., 2015.

[2] Craig Finch, *Sage beginner's guide*, Packt Publishing Ltd, 2011.

[3] William Stein and David Joyner, *Sage: System for algebra and geometry experimentation*, Communications in Computer Algebra (SIGSAM Bulletin)(July 2005), http://sage. sourceforge. net (2005).

[4] The Sage Development Team, *Sage documentation v7.0*, 2016.

Index