Collaborative Use of KeTCindy and Free CASs for Making Materials

Setsuo Takato[1], Alasdair McAndrew[2], Masataka Kaneko[3]

[1] *Toho University, Japan, takato@phar.toho-u.ac.jp*
[2] *Victoria University, Australia, Alasdair.McAndrew@vu.edu.au*
[3] *Toho University, Japan, masataka.kaneko@phar.toho-u.ac.jp*

1 Introduction

Many mathematics teachers at collegiate level use LaTeX to write materials for distribution to their classes. As is well known, LaTeX can typeset complex mathematical formulas. On the other hand, it has poor ability to create figures. One possibility might be to use a third-party package, such as TiKZ. However, TiKZ coding is complicated and is not easy to read even for the following simple figure.

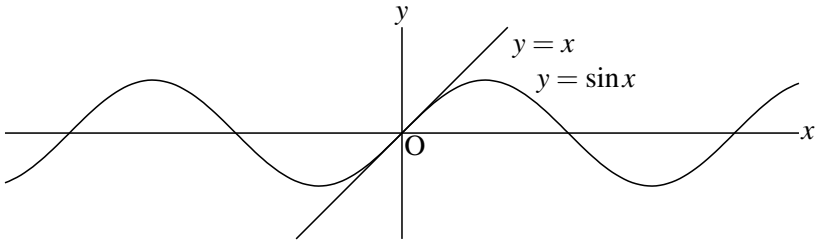


Figure 1 A simple example

TiKZ can in fact produce figures of great complexity, but its power comes at the cost of a steep learning curve. In order to provide a system for easy creation of publication quality figures, the first author has developed KeTpic, the first version of which was released in 2006. KeTpic is a macro package of mathematical software such as Maple, Mathematica, Scilab and R. The recent version uses Scilab mainly and R secondarily.

The flow of generating and inserting graphs with KeTpic is as follows.

1. KeTpic and Scilab commands are listed in a Scilab editor and executed by Scilab.
2. Scilab generates a LaTeX file composed of codes for drawing figures.
3. That file can be inserted into a LaTeX document with `\input` command.
4. The document can be compiled to produce a pdf file.

Scripts for Figure 1 are as follows.

```
Setwindow([-7.5,7.5],[-2,2]);
A=[2.5,1]; B=[2.0,1.5];
Setax(7,"se");
gr1=Plotdata("sin(x)","x");
gr2=Plotdata("x","x","Num=1");
Openfile("figsin");
  Drwline(gr1,gr2);
  Expr(A,"e","y=\sin x",B,"e","y=x");
Closefile('1');
```

Scripts of KETpic are far more readable than those of TiKZ, and the simplicity of the scripts provides for a simple method of inserting figures into LaTeX.

However, it is cumbersome to write all scripts in the Scilab editor first. In particular, mathematics teachers who are not used to mathematical software may find KETpic hard to use. One of the long-standing plans for KETpic has been to write a graphical user interface(GUI) for it.

2 Development of KETCindy

Cinderella[1] is a dynamic geometry software(DGS), and was developed by Gebert and Kortenkamp. The first author had been searching for the possibility of collaborating KETpic and Cinderella with Kortenkamp, and the first version of KETCindy was released in September, 2014. Cinderella works as GUI of KETCindy. It has two screens for the display of figures and for the editor of CindyScript, which is a programming language of Cinderella. To draw a figure,

1. Put geometric components such as points and segments on the display.
2. Describe scripts for drawing a figure and generating of LaTeX graphic codes.
3. Press two buttons on the display to execute batch processing of Scilab, LaTeX compiling and viewing PDF sequentially.

Then you can get the objective figure of LaTeX.

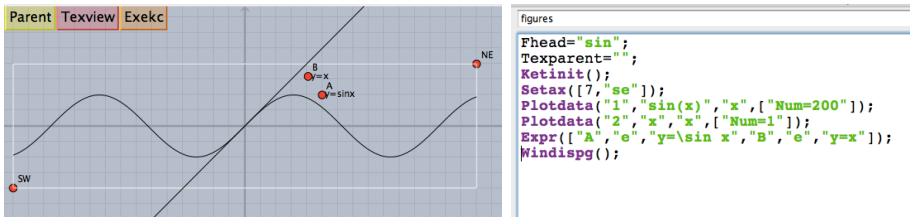


Figure 2 Screens of Cinderella

So far, KETCindy can generate various types of figures or tables as follows.

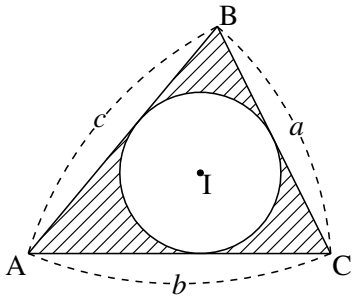


Figure 3 Geometric Figure

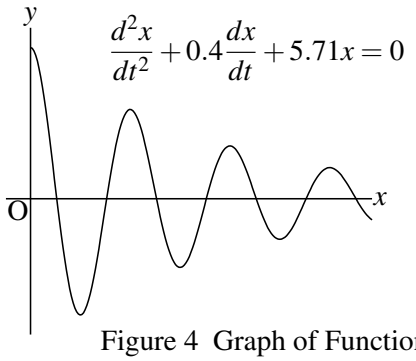


Figure 4 Graph of Function

x	0	$\cdots$	e	$\cdots$	$e\sqrt{e}$	$\cdots$
y'		+	0	−	−	−
y''		−	−	−	0	+
y			$\frac{10}{e}$		$\frac{15}{e\sqrt{e}}$	

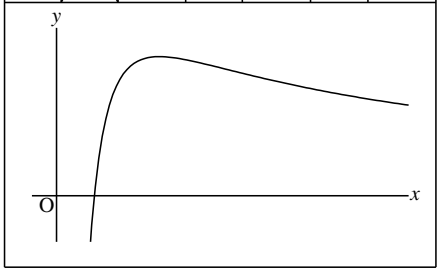
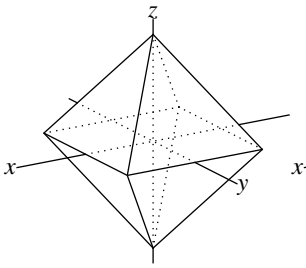


Figure 5 Table

Figure 6 Bézier Curve

3D figures such like the followings can be produced by KETCindy.

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-4,4]");
polydt=Readobj("r02.obj",["size=-3.5"]);
VertexEdgeFace("1",polydt,["Pt=fix","Edg=nogeo"]);
Nohiddenbyfaces("1","phf3d1",[],["do"]);
```



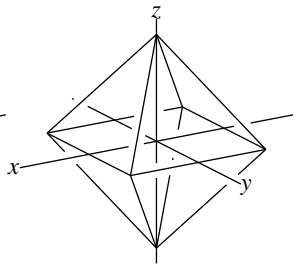


Figure 7 Polyhedron

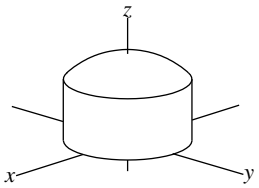


Figure 8 Surface

KETCindy is downloadable from `http://ketpic.com/?lang=english`.

3 Calling CASs from KETCindy

Recently, we implemented functionality to call CASs such as Maxima and FriCAS. In this section, we introduce the functionality and show some application in mathematics education.

The flow and chart are as follows.

1. To generate the shell file to call a CAS
2. To execute the file
3. To return the result as text
4. To use the result in KETCindy
5. To produce the PDF file

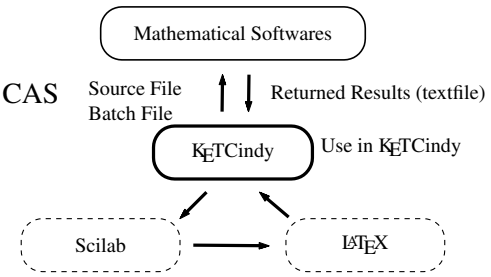


Figure 9 Chart of KETCindy

We give an example to find points of contact of two curves.

```
Mxfun("1","solve",["[y=x^2, x^2+(y-1)^2=3/4]","[x,y]"],[""]);
P=parse(mx1)_1; Q=parse(mx1)_2;
Plotdata("1","x^2","x");
Circledata("1",[A,P]);
Listplot("1",[P,A,Q]);
Letter([A,"ne","A",P,"sw","P",Q,"se","Q"]);
```

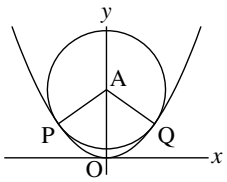


Figure 10

The use of a computer algebra system is required for use with KETCindy and in keeping with the open-source nature of the project, we choose an open-source system. Of the many available, we are most interested in Maxima and FriCAS, being open-source version of once commercial software: Macysma and IBM Axiom respectively. Maxima is written in Lisp and C, and can run on any system which supports those languages. In particular, Maxima can run under MS Windows, Linux, and MacOS. There are even versions of portable systems such as Android. Maxima, formally, is a "term-rewriting" system, where by a system of rules (which the user can add to or change) a term can be rewritten in another form. Thus there are formal rules for symbolic differentiation, for applying integral transforms, for linear algebra, and so on.

FriCAS is a fork of the CAS Axiom, which was released as open-source by IBM when it became clear it was losing its market share. In comparison to all other CAS's today, FriCAS is *strongly typed*: each expression, variable or other

object belongs to a particular type, or nest of types. The use of types allows for overwriting of operations, so that, for example, the outcome of a multiplication 'x * y' will depend on the types of x and y. The nesting of types means that you can define, for example, square matrices of polynomials over a finite field, and having defined that type you automatically have operations available: an inverse of such a matrix, for example, will produce an output object of the same type. The use of types is confusing for the beginner, but in fact provides immense power for the exploration of mathematical systems. In that sense FriCAS probably has greater depth than any other system, although it loses out on breadth.

For systems which are to be used as black boxes: question in, answer out, Maxima may be preferred because of its breadth. But for a system where mathematical precision and rigour are required, FriCAS is the best choice. FriCAS however works best under Linux, so its use in a Windows system requires either some Unix subsystem (such as Cygwin) or the use of a docker container.

Here we give an example of use of Maxima and FriCAS with KETCindy.

```
fun="((1-cos(t))/(1/2-cos(t)))^(1/2)";
Mxfun("1","integrate",[fun,"t"]); //by Maxima (inadequate)
Frifun("2","integrate",[fun,"t"]); // by Fricas
Mxtex("2",fri2);  // Maxima can generate LaTeX form
```

The result by Fricas is $\displaystyle\int \sqrt{\frac{1-\cos t}{\frac{1}{2}-\cos t}}\,dt = -\arctan\left(\frac{(4\cos t+1)\sqrt{\frac{2\cos t-2}{2\cos t-1}}}{4\sin t}\right)$.

Note that FriCAS implements a nearly complete version of the Risch algorithm for symbolic integration; thus its integration routines are amongst the strongest of any current CAS.

4 Conclusion and Future Work

The samples in this paper show that the symbolic computation capability of a CAS can enhance the graphics capability of the KETCindy system. The possibilities of their collaborative use in educational scene should be pursued.

Acknowledgment
This work was supported by JSPS KAKENHI Grant Numbers 25350370, 15K01037, 15K00944.

References

[1] Cinderella, http://www.cinderella.de/tiki-index.php