Collaborative Use of KeTCindy and Free Computer Algebra Systems

Setsuo Takato, Alasdair McAndrew, José A. Vallejo and Masataka Kaneko

Abstract. In mathematics research and education, harmonious use of high-quality mathematical expressions and mathematically precise artwork is desirable. To produce them, an environment in which several kinds of software (including editing tools, computing tools, and high-quality graphics generators) interact in a collaborative way, is needed. To accomplish this, we have developed KeTCindy, a plug-in for the popular dynamic geometry software Cinderella, to help mathematicians generate high-quality graphical images. KeTCindy also enables us to bind the high-performance graphics capability of Cinderella with the powerful computing capability of various computer algebra systems. In this paper, we will show some mathematical artwork to illustrate how the collaborative use of LaTeX, Cinderella, and computer algebra systems via KeTCindy can enhance the power and convenience of each other. Since all the related software are freely accessible, KeTCindy should be especially valuable for educational purposes.

Keywords. Dynamic Geometry Software, Computer Algebra System, Class Materials.

1. Introduction

In the scientific field, the TeX typesetting system and its many variants and extensions (such as LaTeX, XeLaTeX or pdfLaTeX) have become the standard in which (almost) every document is written. Its high-quality output, careful design, and the availability of hundreds of packages makes the system a very powerful one, suitable for all needs. However, this strength comes at a price: writing the code for even the most simple graphics requires a suitable package (say, `pict2e` or `TikZ`), and an understanding of its ins and outs. This can be a time-consuming task (particularly considering that frequently there is little resemblance between one package and another), if all one wants is to occasionally draw some graphics.

It is often said that the `TikZ` package offers the best compromise between advanced capabilities and ease of use. To get a grasp of what we are talking about, here is the code for a simple graphics created with it:

```
\begin{tikzpicture}
  \draw[-] (-7,0)--(7,0)node[right]{$x$};
  \draw[-] (0,-2.7)--(0,2.7)node[above]{$y$};
  \draw[domain=-7:7,thick,samples=80]plot(\x,{sin(deg(\x))});
  \draw[domain=-2.7:2.7,thick,samples=80]plot(\x,{\x})node[right]{$y=x$};
  \node[] (O) at (0.3,-0.3){$\mathrm{O}$};
  \node[] (sine) at (4,0.7){$y=\sin x$};
\end{tikzpicture}
```

Toho University,Victoria University, Universidad Autónoma de San Luis Potosí,Toho University.

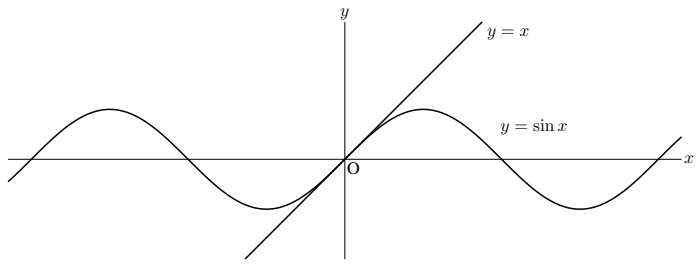


FIGURE 1. A simple example in TikZ.

Though fairly intuitive, the code above requires a non trivial knowledge of TikZ and some previous preparation regarding the geometry of the figure and the placement of labels. On the other hand, consider how easy it is to get the same figure in a dynamic geometry software, such as Cinderella, GeoGebra, Cabri, and the like. In Figure 3, we show how to get the plots of $y = \sin(x)$ and $y = x$ in Ciderella 2: the user just clicks on the 'Define function' button (the one with the $f(x)$ symbol), clicking on the empty canvas a box appears, and by introducing the expression to be plotted (simply `sin(x)` and `x` in our case), the two functions are plotted. The user can move the labels by clicking on them and dragging them wherever she wants. It would be of great help for teachers who simply want to include geometric drawings in their classroom notes, or researchers who need a high-quality plot to illustrate a certain calculation in a journal paper, to be able of obtaining these graphics without the need for in depth knowledge of the typesetting language, as in the Cinderella example above.

It was with this aim in mind that the first author developed KETpic, a set of macro packages to produce the LaTeX output for graphics, using the drawing engines of Scilab (primarily) and **R** (secondarily). The first version of KETpic was released in 2006, and a flowchart describing its functioning is given in Figure 2 below.

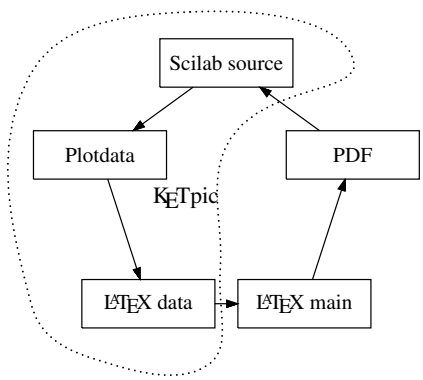


FIGURE 2. The KETpic flowchart.

In words:

1. KETpic and Scilab commands are written in the Scilab editor and executed by Scilab.
2. Scilab generates a LaTeX file containing the code for drawing the graphics.
3. That file can be inserted into a LaTeX document with the `\input` command.
4. The document can be compiled to produce a PDF file.

For instance, the KETpic script corresponding to Figure 1 would be

```
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');
```

This first attempt proved successful in that it provided the user with a simple way to instruct Scilab to do the dirty job of creating the LaTeX code for the figure through a unified scripting language, but still had the same problem as TikZ: it was necessary to know the underlying language. The next step was to create a GUI (Graphical User Interface) for generating those scripts automatically, without writing much code. And here is where KeTCindy enters into play. It provides an interface between Scilab and the dynamic software Cinderella, and/or the computer algebra systems (CAS) Maxima, FriCAS, and Risa/Asir: those programs are used to generate geometric constructions (Cinderella) or a combination of symbolic computations and graphic representations (CASs), which are processed through KeTCindy. It also creates the script to be passed to Scilab, which then in turn generates the final LaTeX output.

Although the whole procedure looks overly complicated, it is precisely the intervention of many individual components what gives KeTCindy its strength. We could say that the Unix philosophy is followed here: if you already have a set of tools that each carry out a definite task, use them in conjunction with each other. The remaining sections of the paper are devoted to illustrating this principle by way of showing some examples of the interaction between KeTCindy and their counterparts Cinderella and Maxima (there are also comments on the use of FriCAS and Risa/Asir).

2. Development of KeTCindy

2.1. Drawing with KeTCindy

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

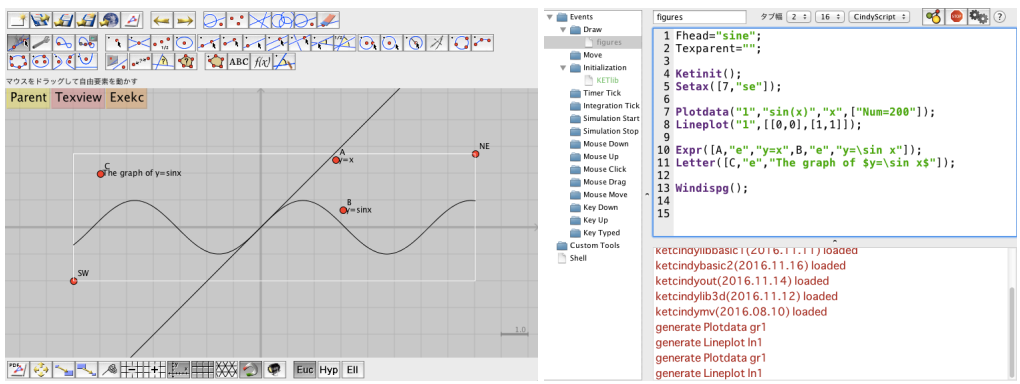


FIGURE 3. Screens of Cinderella in action.

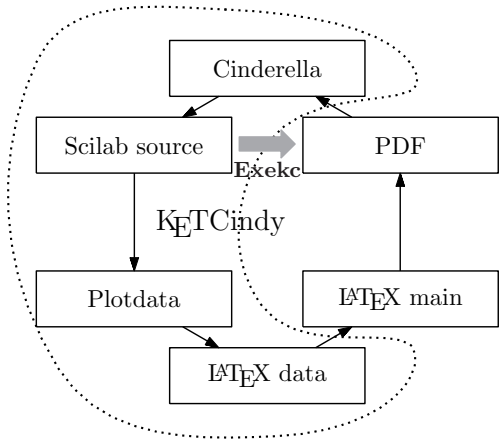


FIGURE 4. The KETCindy flowchart.

Figure 4 shows the flowchart for producing a figure with KETCindy. To draw a figure, one follows these steps:

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

KETCindy can be downloaded from `http://ketpic.com/?lang=english`.

2.2. Examples

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

2.2.1. Geometric Figures. Producing geometric figures in the plane is easy. Moreover, we can hatch some areas, which is better than shading for monochrome printing. The following are the main parts of the script for Figure 5.

```
Listplot([A,B,C,A]);
Circledata([D,E]);
Bowdata([B,A],[1,0.5,"Expr=c","da"]);
Bowdata([C,B],[1,0.5,"Expr=a","da"]);
Bowdata([A,C],[1,0.5,"Expr=b","da"]);
Hatchdata("2",["oi"],[["crDE"],["sgABCA"]],["dr,0.7",""]);
Pointdata("I",D,["size=4"]);
Letter([A,"sw","A",B,"ne","B",C,"se","C",D,"se","I"]);
```

2.2.2. Graphs of Functions. KETCindy can produce graphs of functions with

```
Plotdata("1","x^2","x");
```

or parametrically with

```
Paramplot("1","[2*cos(t),sin(t)]", "t=[0,2*pi]");
```

Here we give an example of the solution curve of a differential equation (see 6). The script is:

```
Deqplot("1","y``=-L.x*y`-G.x*y","t=[0,XMAX]",0,[C.y,0]);
// the equation is y''=-ay'-by (a=L.x, b=G.x).
// C.y,0 are initial values of y and y' at t=0.
Expr(M,"e","\displaystyle\frac{d^2 x}{dt^2}+"
       +"+L.x+"\frac{dx}{dt}+"+G.x+"x=0");
```

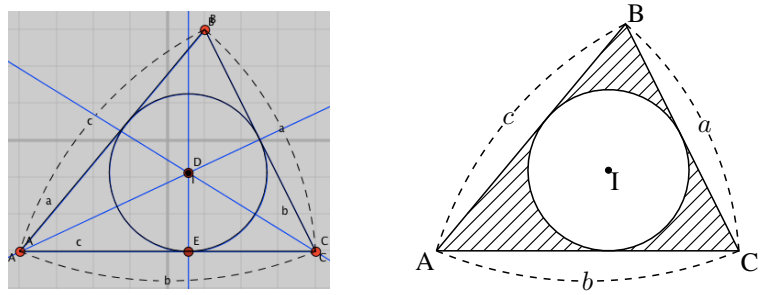


FIGURE 5. Hatching a geometric figure.

Note that points C, G, L on segments AB, EF, HK are movable, and are used to decide the coefficients and the initial value as you can see in the above scripts.

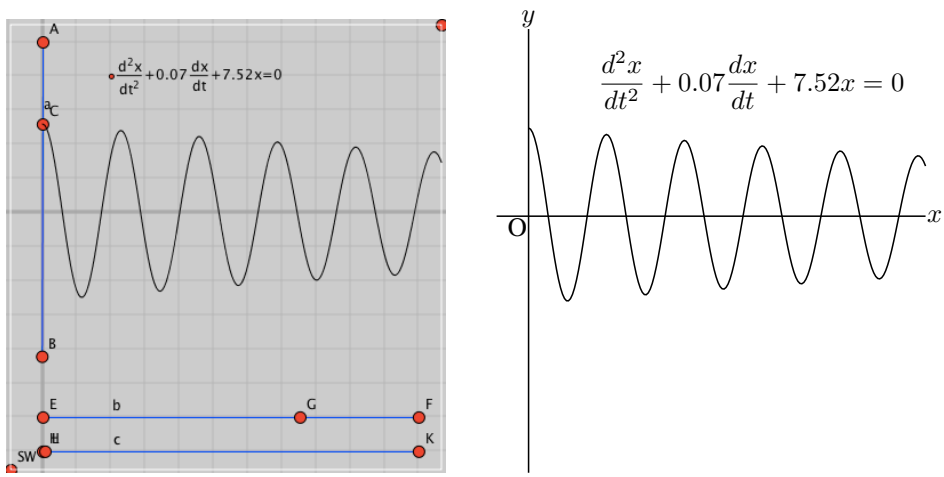


FIGURE 6. Solution of a differential equation.

2.2.3. Drawing Tables. Writing the code for tables to be inserted into the TEX documents is sometimes troublesome. However, it is not a hard job for KETCindy (see the output in Figure 7).

```
xLst=apply(1..7,15);
yLst=[10,10,10,10,80];
rmvL=apply(1..6,"c"+text(#)+"r4r5");
rmvL=concat(rmvL,["r2c1c2","r3c1c2"]);
Tabledata("",xLst,yLst,rmvL);
Tlistplot(["c1r1","c2r4"]);
Tlistplot(["c2r1","c1r4"]);
Putrowexpr(1,"c",
    ["x","0","\cdots","e","\cdots","e\sqrt{e}","\cdots"]);
Putrowexpr(2,"c",["y`","","+","0","-","-","-"]);
Putrowexpr(3,"c",["y``","","-","-","-","0","+"]);
Putrowexpr(4,"c",["y","","","10/e","","15/e\sqrt{e}",""]);
Putcell("c0r4","c7r5","c","\input{fig/graph}");
```

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

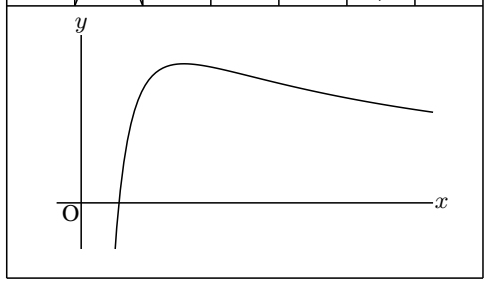


FIGURE 7. A table.

2.2.4. 3D Graphics. 3D figures like the following (Figure 8) 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"]);
Sfbdparadata("1",fd1);
Sfbdparadata("2",fd2,["nodisp"]);
Crvsfparadata("3","crvsf3d2","sfbd3d1",fd1);
```

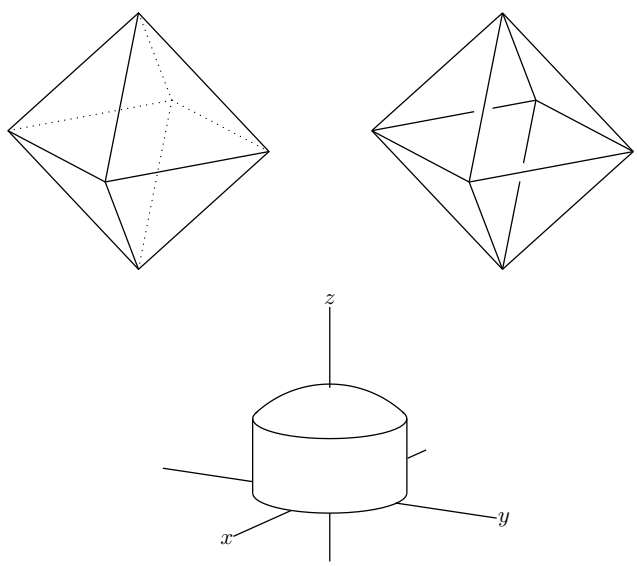


FIGURE 8. Polyhedra and a Cylinder.

3. Calling CASs from KETCindy

Recently, we added new features, allowing KETCindy to call CASs such as Maxima[2], FriCAS[3] and Risa/Asir[4]. In this section, we introduce these features and show some applications in mathematics education.

The steps are as follows.

1. Generate the shell file to call a CAS.
2. Execute the file.
3. Return the result as text.
4. Use the result in KETCindy.
5. Produce the PDF file.

And the flowchart is as follows:

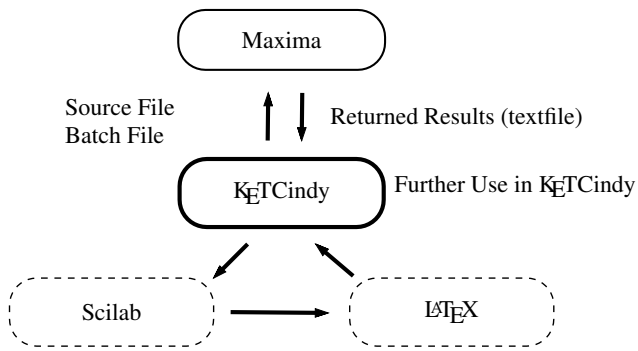


FIGURE 9. Flowchart of KETCindy.

When interfacing with Maxima, the command `Mxfun` is all we need to complete the task. Other commands such as `calcbyM` and `Mxtex` may be used for multi-step and code conversion to LaTeX, respectively. The output of the CAS is returned to KETCindy as a character string for further processing of the graphics. Used this way, KETCindy proves to be a powerful companion to CASs.

We give an example to draw the derivative and the integral of a function (see the output in Figure 10).

```
fn="sin(x)^4";
Mxfun("1","diff",[fn,"x"],[""]);
Mxfun("1b","ratsimp",[mx1]);
Mxfun("2","integrate",[fn,"x"],[""]);
Mxfun("2b","ratsimp",[mx2]);
Plotdata("0",fn,"x",["Num=200","do"]);
Plotdata("1",mx1,"x",["Num=200","dr"]);
Plotdata("2",mx2,"x",["Num=200","da"]);
Mxtex("0",fn);
Mxtex("1",mx1);
Mxtex("2",mx2b);
Expr([A,"e",tx0,B,"e",tx1,C,"w",tx2]);
```

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, FriCAS, being open-source version of once commercial software: Macysma and IBM Axiom respectively, and Risa/Asir. Maxima is written in Lisp, and can run on

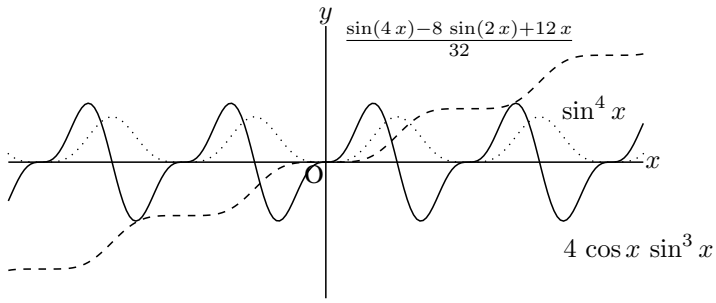


FIGURE 10. Derivative and Integral.

any system which supports that language. In particular, Maxima can run under MS Windows, Linux, and MacOS. There are even versions for 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 contrast to all other CASs today, FriCAS is *strongly typed*: each expression, variable, or any 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 rigor 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 (the computation of an integral):

```
fun="((1-cos(t))/(1/2-cos(t)))^(1/2)";
Mxfun("1","integrate",[fun,"t"]); //in Maxima (it fails)
Frifun("2","integrate",[fun,"t"]); // in FriCAS
```

The result given by Fricas is

$$\int \sqrt{\frac{1-\cos t}{\frac{1}{2}-\cos t}}\,\mathrm{d}t = -\arctan\left(\frac{4\,\cos t+1}{4\,\sin t}\sqrt{\frac{2\,\cos t-2}{2\,\cos t-1}}\right).$$

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

Risa/Asir is an open source general computer algebra system which was originally developed at Fujitsu Labs LTD, and now maintained (in its Kobe branch) by OpenXM developers. Toshio Oshima[13][14] has developed a mathematical library with Risa/Asir. Recently, he implemented various functions for numeric and symbolic integration. KETCindy can call his library from Cinderella as shown below.

```
fun="((1-cos(t))/(1/2-cos(t)))^(1/2)";
AsirfunO("1","integrate",[fun,"t|dumb=-1"]);
```

```
AsirfunO("2","fint",
        ["exp(1/z)/2",-96,"[[cos(t),sin(t)],[0,2*@pi]]"],[""]);
```

The results are $2\arctan\left(\frac{1}{2}\sqrt{3\,\tan^2\left(\frac{t}{2}\right)-1}\right)$ and $-4.81078\cdot 10^{-16}+3.14159\,i$, respectively.

4. Examples of CAS interfacing

4.1. Conic Sections

As an example using `calcbyM`, we show how to draw conic sections. Given a point A on the $z-$axis and a vector $\boldsymbol{n}$, consider a plane through A and perpendicular to $\boldsymbol{n}$. We first need to find an orthonormal basis $\{\boldsymbol{e}_1,\ \boldsymbol{e}_2\}$ on the plane. Then the equation of the plane is represented by

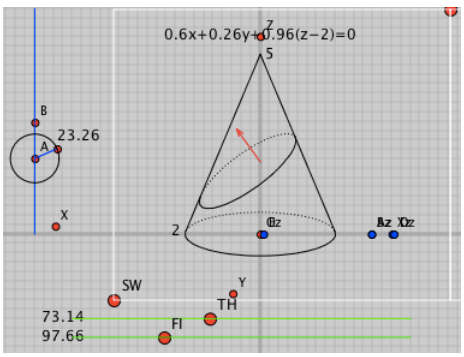
$$(x,\ y,z) = r\ \cos\theta\boldsymbol{e}_1 + r\ \sin\theta\boldsymbol{e}_2\,. \tag{4.1}$$

Let the equation of the cone be

$$z = h\big(k-\sqrt{x^2+y^2}\big)\,. \tag{4.2}$$

Solving simultaneous equations of (4.1), (4.2) with respect to r, we obtain the equation of the section. The main parts of the scripts for computing Figure 11 are as follows.

```
cmdL=[
  "n:[n1,n2,n3]",[],
  "b:[-n[2],n[1],0]",[],
  "c1:n[2]*b[3]-n[3]*b[2]",[],
  "c2:n[3]*b[1]-n[1]*b[3]",[],
  "c3:n[1]*b[2]-n[2]*b[1]",[],
  "c:[c1,c2,c3]",[],
  "nb:sqrt(b.b)",[],
  "nc:sqrt(expand(c.c))",[],
  "p:[0,0,zA]+r*cos(T)*b/nb+r*sin(T)*c/nc",[],
  "eq:x^2+y^2-(k-z/h)^2",[],
  "eq2:ev(eq,[x=p[1],y=p[2],z=p[3]])",[],
  "sol:solve(eq2=0,r)",[],
  "p1:ev(p,sol[1])",[],
  "p2:ev(p,sol[2])",[],
  "nv:[diff(eq,x),diff(eq,y),diff(eq,z)]",[],
  "p1::p2::nv::eq2",[]];
CalcbyM("ans",cmdL,[""]);
```



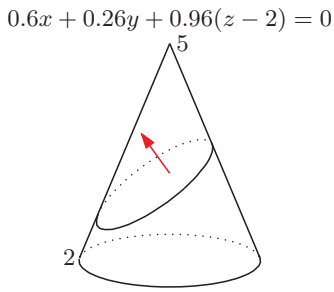


FIGURE 11. Drawing a Conic Section.

4.2. Fourier Series

The third author has developed a package for computing the Fourier series of piecewise defined functions in Maxima, called `fourier_sec`, and the first author has added two functions, `Periodfun` and `fourierseries`, to draw the graph of the target function and Fourier series, respectively. `Periodfun` also returns the definition of the target function in Maxima. For example, from

```
defL=["1",[-3,-2],1,"0",[-2,-1],1,"-x",[-1,1],1,
          "0",[1,2],1,"-1",[2,3],1];
```

we can get

```
fun="if (-3<=x and x<-2) then 1
      elseif (-2<=x and x<-1) then 0
       elseif (-1<=x and x<1) then -x
        elseif (1<=x and x<2) then 0
         elseif (2<=x and x<3) then -1";
```

Below we show the code needed to draw Figure 12. Here, the number of terms is obtained interactively from the position of point C on the slider AB.

```
tmp=Periodfun(defL,1,["dr,2"]);
fun=tmp_1; period=tmp_2;
cmdL=Concat(Mxbatch("fourier_sec"),[
  "Ffun(x):="+fun,[],
  "coef:fourier_sec_coeff",["Ffun(x)","x"],
  "coef[1]::coef[2]::coef[3]",[] ]);
CalcbyM("out",cmdL,[]);
Fourierseries("1",out,period,round(4*(C.x-A.x)),["Num=400"]);
Mxtex("2",out_3);
Expr([D,"e","s_n="+tx2,E,"e","n="+text(nterm)]);
```

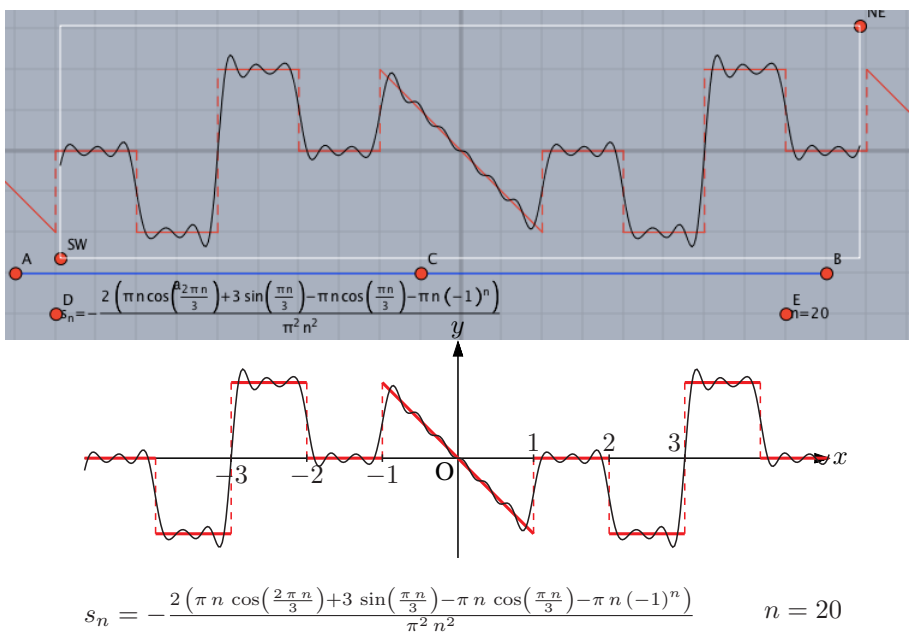


FIGURE 12. Fourier Series of a Piecewise Function.

4.3. Generating 3D Models

One of the appealing features of KETCindy is its capability of displaying 3D graphics (through .obj files). These can be viewed with programs such as Meshlab [5]. Once the basic 3D graphics have been coded, the surface can be fine-tuned using Maxima to give it the desired thickness (this is required by the 3D printing process), in addition to other features. The final code, along with its 3D printed version, is shown in Figure 13.

```
fd=["p","x=U*cos(V)","y=U*sin(V)","z=cos(V)^2-sin(V)^2",
  "U=[0,2]","V=[0,2*pi]","e"];
tmp=Mkobjnrm("1",fd);
cmdL=["assume",["U>0"],"a:trigsimp",[tmp],"a",[]];
CalcbyM("ans",cmdL);
Mxfun("1","solve",[den,"C"]);
cmdL=["assume",["U>0"],
  "a1:limit",[norm1,"V","0"],...];
CalcbyM("lim1",cmdL,[""]);
norm="if or(V==%p/2*[0,1,2,3,4]) then Out=[0,0,1];"
   +"else Out="+norm+";end";
cmd=Mkobjthickcmd("1",fd,norm,[0.05,"+n+s-e-w+"]);
Mkviewobj("ds",cmd),["m","v");
```

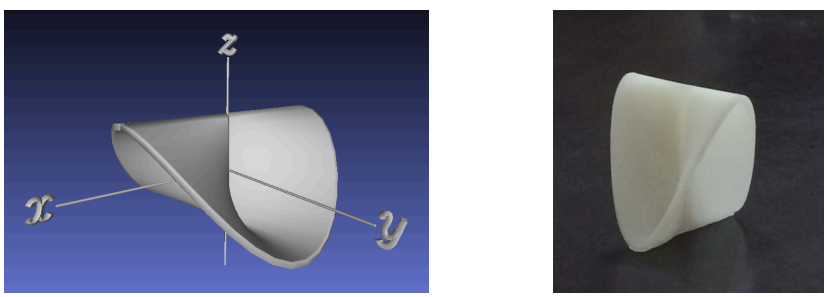


FIGURE 13. Graph of $z = \frac{x^2-y^2}{x^2+y^2}$.

5. Conclusion and Future Work

Printed materials are often distributed in mathematics classes at the university level, and high quality figures are indispensable in such materials. KETCindy makes it easy to produce and insert them into LATEX documents. The samples in this paper show that the symbolic computation capability of a CAS can enhance the graphical output of the KETCindy system. CASs are also useful to produce examples and problem solutions which can be embedded into teaching (or research) materials. Finally, an application to the creation of 3D models has been shown. Other possibilities of their collaborative use in educational matters should be pursued.

References

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

[2] Maxima, http://maxima.sourceforge.net

[3] Fricas, http://fricas.sourceforge.net

[4] Risa/Asir, http://www.math.kobe-u.ac.jp/Asir/asir.html

[5] Meshlab, http://meshlab.sourceforge.net

[6] Kaneko M., Yamashita S., Kitahara K., Maeda Y., Nakamura Y., Kortenkamp U, Takato S., KETCindy--Collaboration of Cinderella and KETpic, Reports on CADGME 2014 Conference Working Group, The International Journal for Technology in Mathematics Education, **22**(4), 179–185, 2015.

[7] Takato S., Hamaguchi N., Sarafian H., Generating Data of Mathematical Figures for 3D Printers with KETpic and Educational Impact of the Printed Models, ICMS 2014, LNCS **8592**, 629–634, Springer, Heidelberg, 2014.

[8] Kaneko, M., Maeda, Y., Hamaguchi, N., Nozawa, T., Takato, S., A scheme for demonstrating and improving the effect of CAS use in mathematics education, Proc. ICCSA, 62–71, IEEE, 2013.

[9] Kaneko M, Takato S., The effective use of LaTeX drawing in linear algebra, The Electronic Journal of Mathematics and Technology **5**(2), 1–20, 2011.

[10] Kaneko, M., Takato, S., A CAS macro package as LaTeX graphical command generator and its applications, Proc. ICCSA, 72–81, IEEE, 2011.

[11] Kaneko M., Abe T., Sekiguchi M., Tadokoro Y., Fukazawa K., Yamashita S., Takato S., CAS–aided Visualization in LaTeX Documents for Mathematical Education, Teaching Mathematics and Computer Science VIII-I, 1–18, 2010.

[12] Kortenkamp U., Interoperable interactive geometry for Europe, The Electronic Journal of Mathematics and Technology, **5**(1), 1–14, 2011.

[13] Oshima T., Drawing curves, Mathematical Progress in Expressive Image Synthesis III, to appear in Springer Japan.

[14] Oshima, T., Drawing curves, Symposium MEIS2015: Mathematical Progress in Expressive Image Synthesis, MI Lecture Notes 2015 **64**, 117–120, Kyushu University, 2015

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers 25350370, 15K01037, 15K00944. The third author was partially supported by the Mexican Consejo Nacional de Ciencia y Tecnología, Project CB-2012 179115.

Setsuo Takato
2-2-1, Miyama
Funabashi
Japan
e-mail: takato@phar.toho-u.ac.jp

Alasdair McAndrew
PO Box 14428
Melbourne, Vic 8001
Australia
e-mail: Alasdair.McAndrew@vu.edu.au

José A. Vallejo
Av. Salvador Nava s/n
78290 San Luis Potos (SLP)
MEXICO

e-mail: jvallejo@fc.uaslp.mx

Masataka Kaneko
2-2-1, Miyama
Funabashi
Japan
e-mail: masataka.kaneko@phar.toho-u.ac.jp