Using Oshima splines to produce accurate numerical results and high quality graphical output

Setsuo Takato and José A. Vallejo

Abstract. We illustrate the use of Oshima splines in producing high-quality LaTeX output in two cases: first, the numerical computation of derivatives and integrals, and second, the display of silhouettes and wireframe surfaces, using the macros package KETCindy. Both cases are of particular interest for college and university teachers wanting to create handouts to be used by students, or drawing figures for a research paper. When dealing with numerical computations, KETCindy can make a call to the CAS Maxima to check for accuracy; in the case of surface graphics, it is particularly important to be able to detect intersections of projected curves, and we show how to do it in a seamlessly manner using Oshima splines in KETCindy. The C language can be called in this case to speed up computations.

Mathematics Subject Classification (2010). Primary 97U50; Secondary 97U60.

Keywords. KeTCindy, Cinderella, Maxima, Oshima spline.

1. Introduction

Cinderella is a dynamic geometry software (DGS) comprising two main components: CindyScreen, an interactive screen where geometric elements can be constructed like in any other DGS, and CindyScript, a scripting language which can manipulate not only geometric objects but more general constructions. The following figure shows both components of Cinderella, CindyScreen (left), and the CindyScript editor (right).

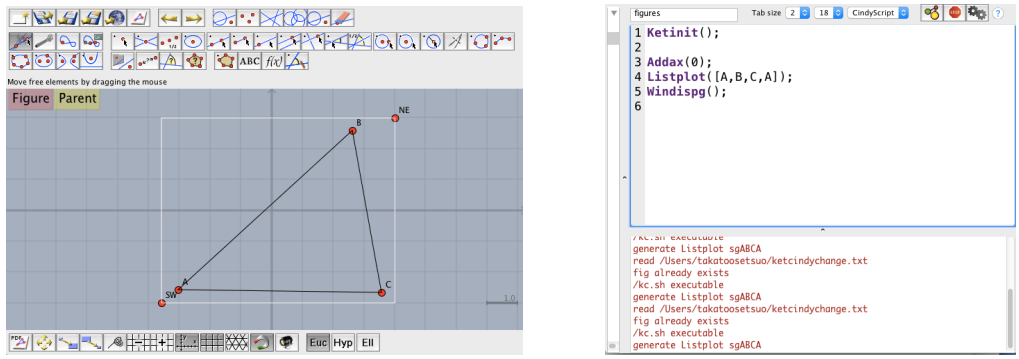


Fig.1 CindyScreen and CindyScript

KETCindy is a package of CindyScript macros designed to produce high-quality LaTeX figures and animations (see [5, 6]). It is well suited for particularly complex graphics such as the ones appearing in dynamical systems theory [9], and flexible enough to satisfy the demands of a wide

class of topics ranging from linear algebra to the calculus of variations, including the preprocessing of data for 3D printing [10]–[15]. It can be accessed from the Comprehensive TEX Archive Network (CTAN) repository [4], requiring `Cinderella` [1] and `R` [2] as dependences. In this text we will also require the optional Computer Algebra System (CAS) `Maxima` [3]. The package comes with its own documentation explaining the installation process.

As an example of its use, we now show how to draw a freehand smooth curve. For this task, KETCindy uses Bézier curves and has several commands to create splines from them: `Bezier`, `CRspline`, `Ospline`, `Bspline` and `Mkbezierptcrv` are already implemented. Here we use `CRspline`, the command to draw a Catmull-Rom spline. The steps to generate a plotting data file suitable for being included in a LATEX document are as follows:

1. Open a template in the work folder of ketcindyfolder or any KETCindy file. The rectangle in the screen shows the drawing range for the `picture` environment in LATEX.
2. Change points SW, NE to fix the drawing area and add some other points A, B, C, D that will serve as control points for the spline.

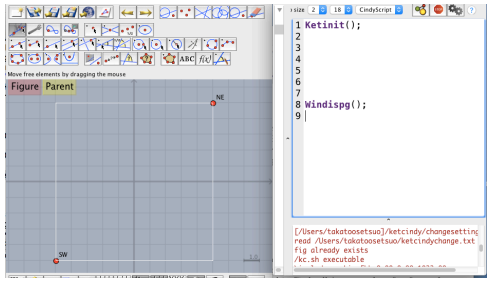


Fig.2 Initial screen

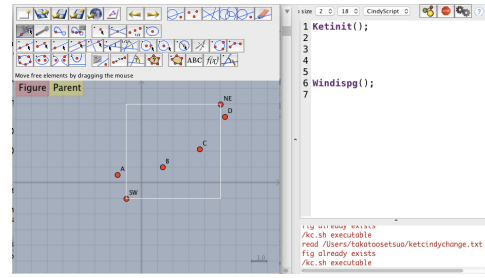


Fig.3 Adding points

3. In the script editor, write the command `CRspline("1",[A,B,C,D])`: the curve is displayed in the screen. One can change the shape of the curve by moving any of the control points A,B,C,D in the CindyScreen.
4. Pressing the button `Figure` in the screen will generate the plotting data for LATEX . The output is represented in Figure 5 below.

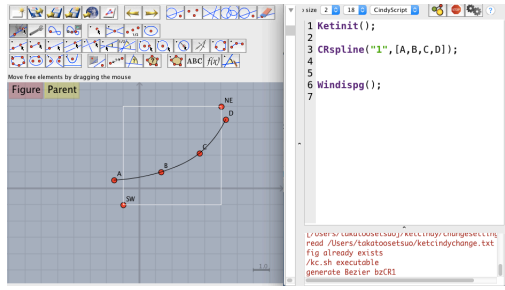


Fig.4 Drawing CRspline curve

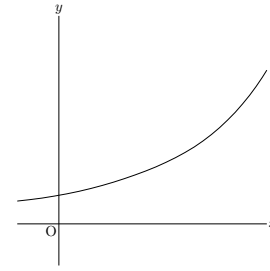


Fig.5 Plotting LATEX data

5. Further embellishment of the figure is, of course, possible. Le us complete it to illustrate the notion of differences quotient. It suffices to write the following commands in the CindyScript editor:

```
CRspline("1",[A,B,C,D]);
pt=[C.x,B.y];
Listplot("1",[B,pt,C,B]);
Bowdata("1",[B,pt],["do","E=\Delta x"]);
```

```
Bowdata("2",[pt,C],[1.5,"do","E=\Delta y"]);
Letter(B,''n'',''P'');
Htickmark([B.x,"x",C.x,"\xi"]);
Vtickmark([B.y,"f(x)",C.y,"f(\xi)"]);
```

Again, pressing `Figure` will generate the LaTeX code and the corresponding `pdf` file.

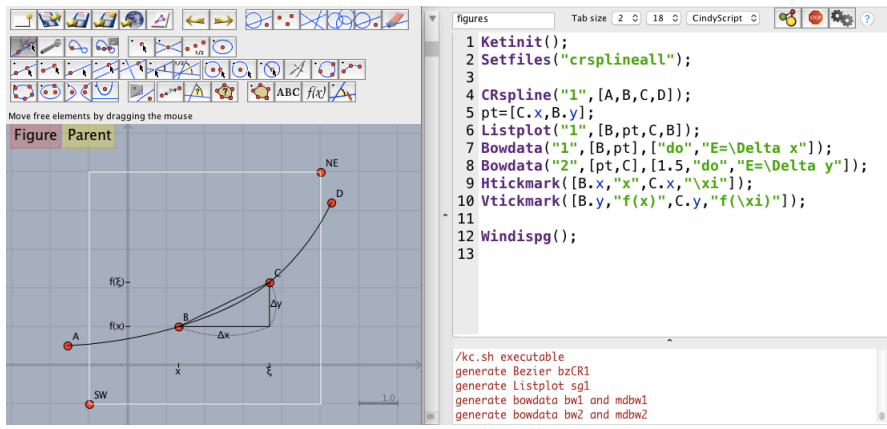


Fig.6 Adding other components to the figure

As a side remark, sometimes more flexibility than that provided by LaTeX is required to insert figures in a document. This is particularly the case when creating handouts to be distributed to students, quizzes, cheat sheets, and the like. To deal with these cases, the `layer` environment, which is defined in `ketlayer.sty`, can be useful. If `graph1.tex` is the result of steps $1-5$ above, the code

```
\begin{layer}{140}{50}
\putnotes{115}{2}{$\frac{\varDelta y}{\varDelta x}\to f'(x)$}
\putnotese{80}{5}{\inputf{fig/graph1.tex}}
\end{layer}
```

when inserted in a LaTeX document, will produce the following:

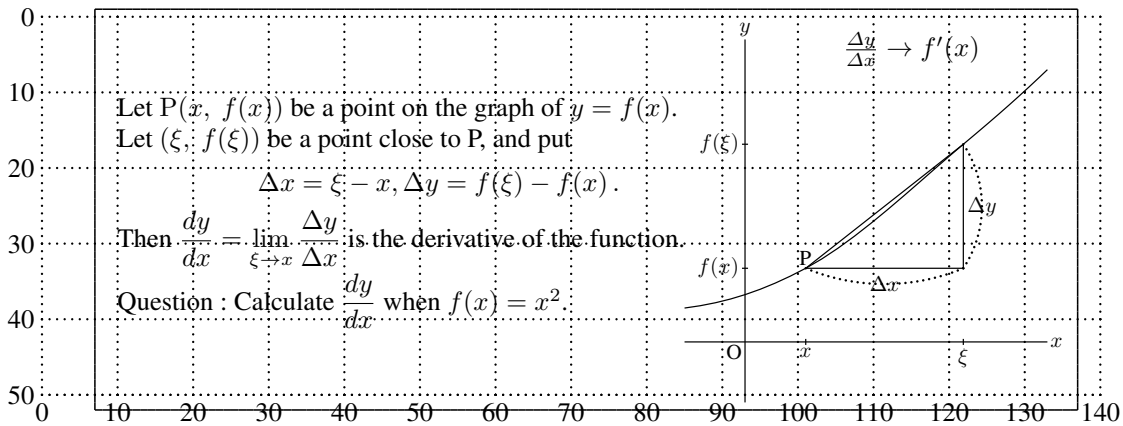


Fig.7 Example of a material for use in teaching

Change the arguments of `\putnote` to move each component. If they are placed properly, set the second argument of `layer` to 0, then grids will disappeared without moving the position of all components.

```
\begin{layer}{140}{0}
```

```
\putnotese{80}{5}{\inputf{fig/graph1.tex}}
\end{layer}
```

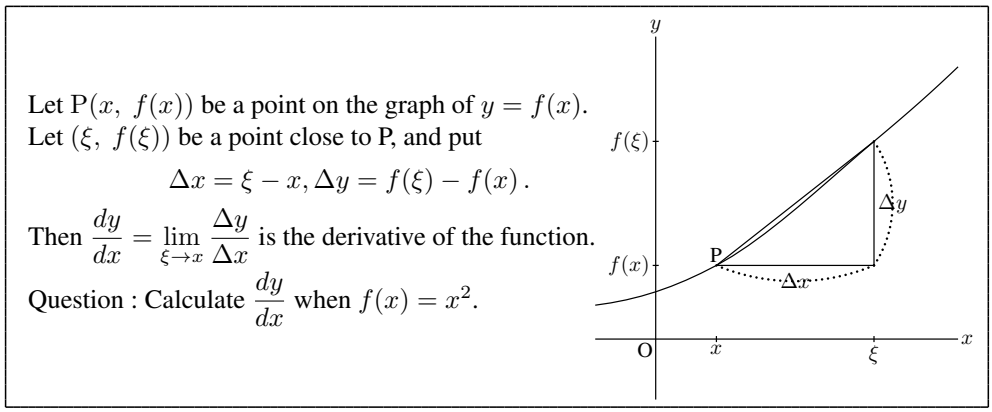


Fig.8 Example of a material for use in teaching

`\includegraphics` of `graphicx` cannot do the same thing by itself. Moreover, `\input` might be more convienient when inputting the file cosist of graphical codes such as `pict2e` and `tikz`.

The Catmull-Rom spline which we used in this example is well known and often used to draw freehand shapes or to make a nonlinear interpolation. However, as with any other class of splines, there are many cases in which the results it gives are not as good as one desires. This is particularly true for conic sections, which can not be exactly generated using Bézier curves: if we take four points A, B, C, D on an ellipse and use the Catmull-Rom spline to approximate it, the corresponding code is

```
Paramplot("1","[3*cos(t),2*sin(t)]","t=[0,2*pi]",["da"]);
  // shows the ellipse with a dashed line.
A=[3,0]; B=[0,2]; C=[-3,2]; D=[0,-2];
CRspline("1",[A,B,C,D,A]);
```

and the resulting figure is the following:

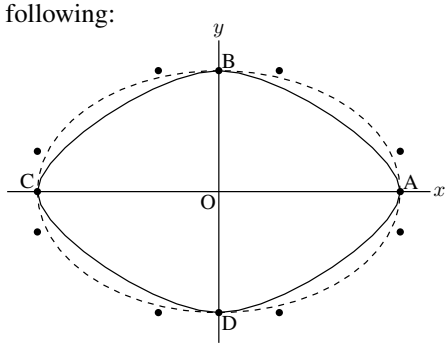


Fig.9 Catmull-Rom spline for an ellipse

Even recognizing the impossibility of exactly reproducing conic sections, this result is less than sub-optimal. Due to the shortcomings of Catmull-Rom splines, Oshima developed a new idea to determine the spline control points from the given points on the curve [7], in such a way that the interpolation is nearly optimal in the case of conics (compare Figure 8 with Figure 11 below). This opens the possibility of constructing numerical schemes for derivation and integration based on this spline, as well as a promising technique to attack the problem of determining the intersection

points of surfaces and curves, a basic problem in Computer Aided Geometric Design (or CAGD) that must be solved in order to construct wireframes. Thus, our contribution in this paper is to describe these techniques, and illustrate both its practical implementation and its application to some typical problems. In this regard, we have used the set of macros KETCindy because it allows us to exploit the very precise numerical computation of wireframes resulting from Oshima techniques (when those computations are done through an external C compiler), and as a result, we can produce high-quality graphical output, as shown in the examples below. The resulting framework, integrating the DGS Cinderella, the CAS Maxima and a C compiler with KETCindy as the interface, has proven to be very flexible and powerful, allowing the user to carry on the numerical tasks as well as the generation of graphics in an intuitive and unified environment.

2. Oshima spline curve

Let points $\mathrm{P}_{j-1}\,,\mathrm{P}_{j}\,,\mathrm{P}_{j+1}\,,\mathrm{P}_{j+2}$ be on a certain curve which we want to approximate, and $\mathrm{Q}_j,\ \mathrm{R}_j$ be the control points corresponding to an interval $\mathrm{P}_j\mathrm{P}_{j+1}$. In the case of the Catmull-Rom spline curve, these control points are defined solely by $\mathrm{P}_j, \mathrm{P}_{j+1}$ using the conditions

$$\overrightarrow{\mathrm{P_j Q_j}} = \frac{1}{6}\overrightarrow{\mathrm{P_{j-1}P_{j+1}}}, \quad \overrightarrow{\mathrm{P_{j+1}R_j}} = \frac{1}{6}\overrightarrow{\mathrm{P_{j+2}P_j}}\,.$$

Note that the coefficients appearing here are contants. Figure 10, shows the shape of the resulting spline, where it is to be noticed that the curve bends rapidly.

Oshima's definition is

$$\overrightarrow{\mathrm{P_j Q_j}} = c\,\overrightarrow{\mathrm{P_{j-1}P_{j+1}}}, \quad \overrightarrow{\mathrm{P_{j+1}R_j}} = c\,\overrightarrow{\mathrm{P_{j+2}P_j}}\,,$$

where the value of the coefficient c is determined from $\mathrm{P}_{j-1}\,,\mathrm{P}_{j}\,,\mathrm{P}_{j+1}\,,\mathrm{P}_{j+2}$ as follows:

$$c = \frac{4\overrightarrow{\mathrm{P_j P_{j+1}}}}{3(\overrightarrow{\mathrm{P_{j-1}P_{j+1}}} + \overrightarrow{\mathrm{P_j P_{j+2}}})} \times \frac{1}{1+\sqrt{\frac{1}{2}\,(1+\cos\theta)}}\,,$$

where θ is the angle between $\overrightarrow{\mathrm{P_{j-1}P_{j+1}}}$ and $\overrightarrow{\mathrm{P_j P_{j+2}}}$. See Figure 11, and notice that this time the curve is smoother, as well as the different placement of the control points. Figure 12 represents the Oshima spline for the ellipse of the preceding section.

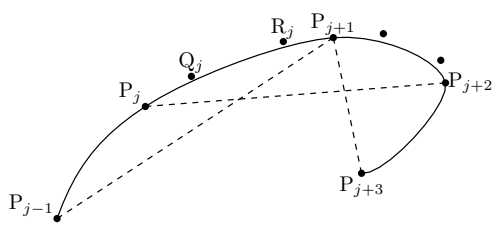


Fig.10 Case of Catmull-Rom spline

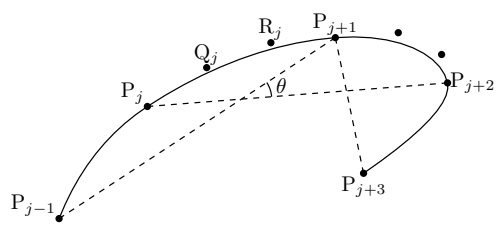


Fig.11 Case of Oshima spline

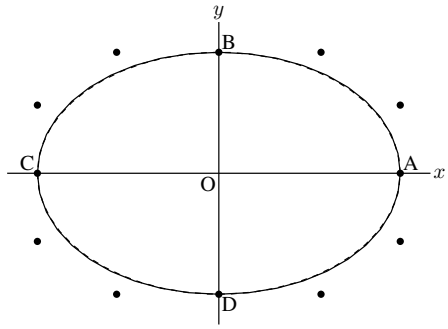


Fig.12 Oshima spline for an ellipse

Though we can not say that Oshima spline gives a better interpolation in all cases, in general it certainly does when it comes to drawing smooth freehand curves. As an application of this fact, we will apply next the Oshima technique to the problem of numerical differentiation and integration. KETCindy provides two commands (`Integrate`, `Derivative`) for these tasks.

2.1. Numerical integration

Let C be a Bézier curve determined by the control points P_j, P_{j+1}, P_{j+2}, P_{j+3}, and let (x_k, y_k) denote the coordinates of P_k. Since the parametric equations of C are

$$\begin{aligned}\mathrm{P} = {} & \mathrm{P}_j(1-t)^3 + 3\mathrm{P}_{j+1}(1-t)^2 t \\ & +3\mathrm{P}_{j+2}(1-t)t^2 + \mathrm{P}_{j+3}t^3 \qquad (1) \\ & (0 \leqq t \leqq 1),\end{aligned}$$

the definite integral $\displaystyle\int_{x_j}^{x_{j+3}} y\,\mathrm{d}x$ becomes

$$\int_0^1 y\frac{\mathrm{d}x}{\mathrm{d}t}\mathrm{d}t\,.$$

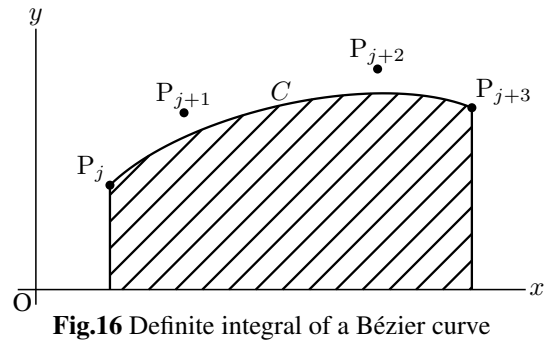


Fig.16 Definite integral of a Bézier curve

KETCindy can call Maxima using `Mxfun` to execute a single command and `CalcbyM` to execute several commands [6]. Thus, we can evaluate the preceding integral with the following CindyScript code:

```
cmdL=[
 "P:[x1,y1]*(1-t)^3+3*[x2,y2]*(1-t)^2*t
          +3*[x3,y3]*(1-t)*t^2+[x4,y4]*t^3",[],
 "f:P[2]*diff(P[1],t)",[],
 "ans:integrate",["f","t",0,1],
 "ans",[]
];
CalcbyM("ans",cmdL);
println(ans);
```

The output is displayed in the console as

```
((10*x4-6*x3-3*x2-x1)*y4+(6*x4-3*x2-3*x1)*y3
        +(3*x4+3*x3-6*x1)*y2+(x4+3*x3+6*x2-10*x1)*y1)/20.
```

This procedure is implemented in KETCindy's command `Integrate`. Here we illustrate its use through several examples.

Example 1. To compute the area surrounded by ellipse $\dfrac{x^2}{3^2}+\dfrac{y^2}{2^2}=1$ we would execute the script

```
Paramplot("1","[3*cos(t),2*sin(t)]","t=[0,pi]",["Num=25"]);
Paramplot("2","[3*cos(t),2*sin(t)]","t=[pi,2*pi]",["Num=25"]);
ans=Integrate("gp1",[-pi,pi])-Integrate("gp2",[-pi,pi]);
println(Sprintf(ans/(6*pi),6));
```

The result is 0.999937, which gives a good approximation to $\dfrac{S}{\pi ab}=1$.

Example 2. The definite integral of $y=x^2\sin x$ from 0 to π is computed by the script

```
Plotdata("1","x^2*sin(x)","x=[-pi,pi]",["Num=50"]);
ans=Integrate("gr1",[0,pi]);
println(Sprintf(ans,6));
```

The result is 5.869063. We can check this result with Maxima:

```
Mxfun("1","integrate",["x^2*sin(x)","x",0,"%pi"]);
Mxfun("2","float",[mx1]);
```

The output is `%pi^2-4`, whose numerical value in double precision is 5.869604401089358.

Notice that `Integrate` can be applied to a list of points. Actually, `gr1` and `gp1`, `gp2` in the preceding examples are lists of points.

Example 3. The definite integral of the implicit function $8x^2-4\sqrt{2}xy+y^2-3x-6\sqrt{2}y+2=0$ in the region $[-2,2]\times[-2,2.5]$ is performed by

```
Implicitplot("1","8*x^2-4*sqrt(2)*x*y+y^2-3*x-6*sqrt(2)*y+2=0",
        "x=[-2,2]","y=[-2,2.5]");
P=Ptstart("imp1");
Q=Ptend("imp1");
Letter([P,"n","P",Q,"n","Q"]);
ans=Integrate("imp1",[Q_1,P_1]);
println(Sprintf(ans,6));
```

The result is 1.699186.

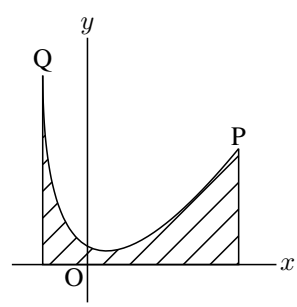


Fig.17 Integration of an implicit function

2.2. Numerical differentiation

We can also calculate the derivative of a function given by a list of points. To find the derivative of C at the point P_j in Figure 16, we differentiate (1) and put $t=0$, then

$$\frac{d\mathrm{P}}{dt}=-3\mathrm{P}_j+\mathrm{P}_{j+1}\,.$$

Using this procedure, we have improved the command `Derivative` and implemented `Tangentplot` to draw the tangent line at a point as an application. The following is an example of the use of `Tangentplot` with the implicit function of Example 3.

```
dx=(Q_1-P_1)/10;
forall(0..10,ii,
  v=P_1+ii*dx;
  Tangentplot(text(ii),"imp1","x="+format(v,6),["dr,0.2"]);
);
```

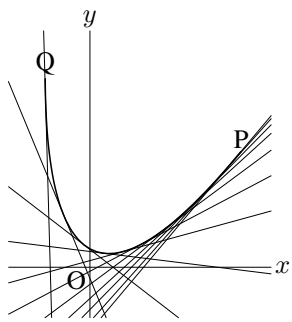


Fig.18 Tangent lines to an implicit function

3. Drawing 3D surfaces

3.1. Intersection of silhouette lines

Teaching materials displaying 3D figures are often required in mathematics courses. For such printed materials, figures presented as line drawings are better suited, because students can write their own remarks over them on the paper (Figure 19). KETCindy supports the line drawing of 3D figures as explained below, but let us first present a couple of examples:

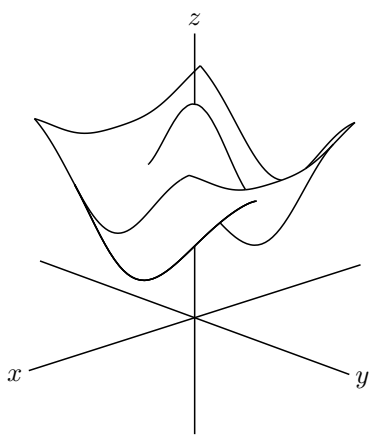


Fig.19 Line drawing of a surface

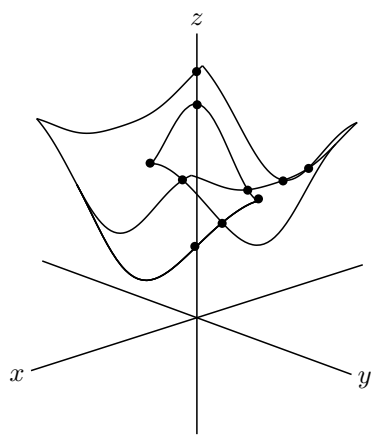


Fig.20 Segmenting curves

To produce these 3D figures, KETCindy follows the steps:

1. The silhouette lines of the surface are determined. To this end, numerical data are obtained from an implicit function of the form
$$J(u,v) = \frac{dX}{du}\frac{dY}{dv} - \frac{dX}{dv}\frac{dY}{du} = 0\,,$$
where $(X,Y) = \mathrm{Proj}(x,y,z)$ is the map onto the plane of projection.
2. The intersections of silhouette lines and projected curves (see Figure 20) are computed.
3. The curve is segmented by these intersections, and it is determined whether each segment is hidden by the surface or not.

Of the above, the second item is of fundamental importance, but it represents a difficult task in the case of contacting curves because, numerically, they are polygonal lines (see Figure 21 for an example, where the small window in red shows the contact region that will be further discussed below).

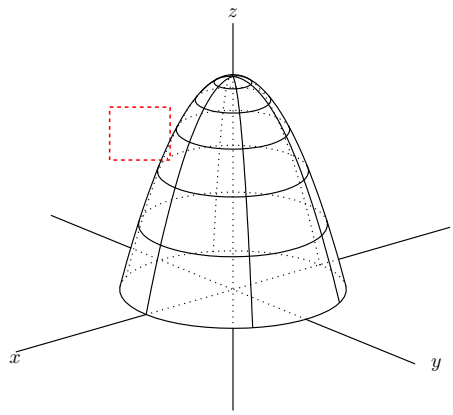


Fig.21 Case of contacting curves

The following graphical examples illustrate this setting: the right figure shows a magnification of the contact region in the left one. The parametric equations of the surface are $x = u\cos v$, $y = u\sin v$, and $z = 4 - u^2$.

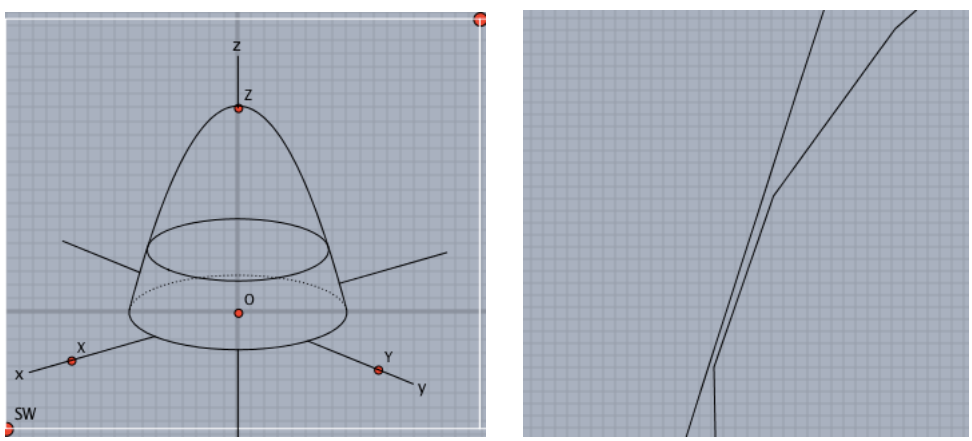


Fig.22 Magnifying around the contact point

To refine the calculation in item 2, we have adopted an interpolatory scheme using Oshima splines around the contact point.

In the figure below, the left pane shows a further magnification. The right one shows the Oshima splines of the silhouette line of the surface and the curve in red.

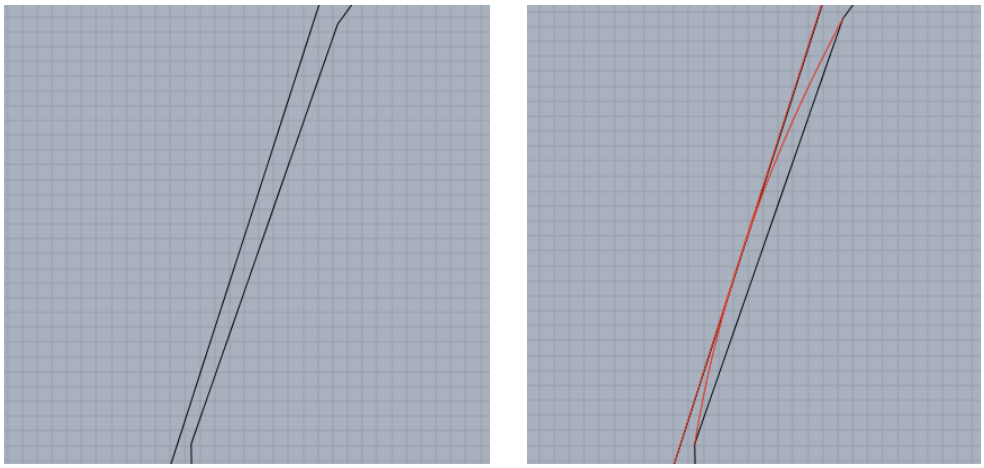


Fig.23 Use of Oshima spline curve

We can see that the intersection is still represented by a cluster of points, but a much narrow one. A randomly chosen point in this intersection has coordinates

$$\mathrm{P} = [-1.65827, 1.20578]\,. \tag{2}$$

The exact values for the coordinates of the intersecting point (in double precision) computed by Maxima are:

$$\mathrm{P} = [-1.656701299244927, 1.210755779027779]\,,$$

confirming that (2) is a good approximation to the contact point.

Once the lines representing the surface and the curves on it have been determined, the next step is to hide those portions that lie behind the surface from the perspective of the observer. However, it takes a long time to apply the algorithm for the elimination of those hidden parts if only CindyScript is used. To speed up computations, KETCindy can call `gcc`(Gnu C Compiler). The following is the CindyScript code used to generate Figure 21

```
fd=[
  "z=4-(x^2+y^2)",
  "x=R*cos(T)","y=R*sin(T)",
  "R=[0,2]","T=[0,2*pi]","e"
];
Startsurf();
Sfbdparadata("1",fd);
Crvsfparadata("1","ax3d","sfbd3d1",fd);
Wireparadata("1","sfbd3d1",fd,5,2*pi/6*(0..5));
ExeccmdC("1",[""],[]);
```

Its execution takes only a few seconds in a standard desktop computer.

3.2. 3D animations

KETCindy can produce both, a flip movie which displays slides step by step, and LaTeX animations suitable to be included in `pdf` documents (or translated into graphics formats such as `APNG`), through the use of `animate.sty`. The use of the `gcc` compiler makes it feasible to animate 3D surfaces. We show an example of an animation illustrating the construction of the Möbius band. A snapshot of the corresponding CindyScreen appears in Figure 24, and the CindyScript code is the following:

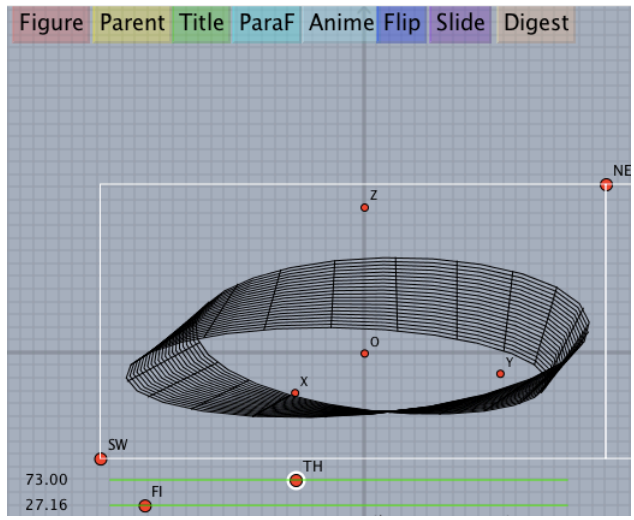
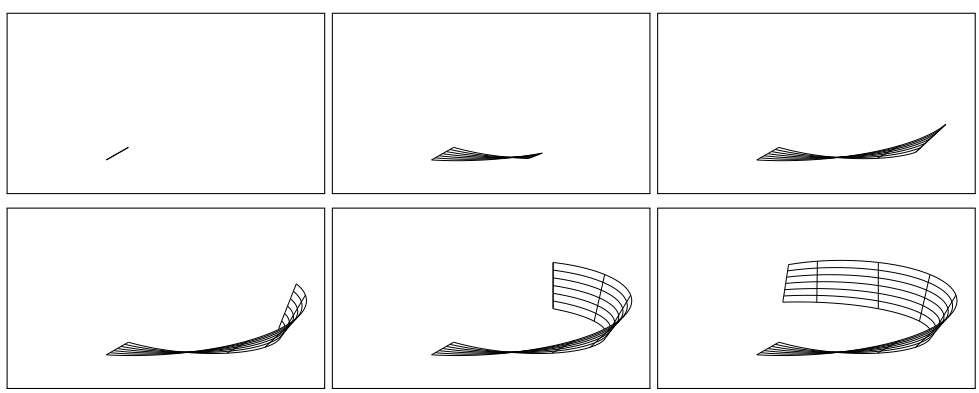


Fig.24 Screen for 3D movies

```
Start3d([A,B,C,S,Sl,Sr]);
mf(s):=(
  regional(tmp);
  Startsurf();
  fd=["p",
   "x=2*cos(t)*(2+r*cos(t/2))",
   "y=2*sin(t)*(2+r*cos(t/2))",
   "z=2*r*sin(t/2)","r=[-0.4,0.4]","t=[0,"+text(s)+"]","nsew"
  ];
  Sfbdparadata("1",fd);
  tmp=select((1..12)/12*2*pi,#<=s);
  Wireparadata("1","sfbd3d1",fd,5,tmp);
  ExeccmdC("",[""],["nodisp"]);
);
Setpara("mobius","mf(s)","s=[0,2*pi]",["Div=24"]);
```

Pressing the button `Flip` generates the flip movie, while `Anime` generates the animation. The separate slides of the flip movie are shown below.



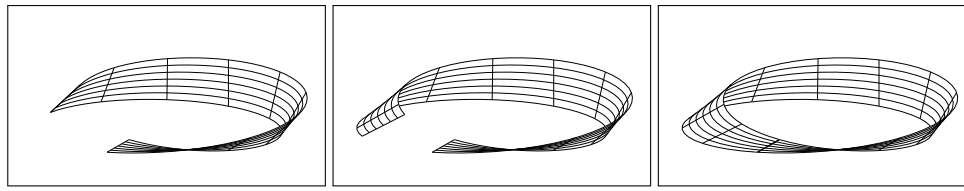


Fig.25 Slides for the animated Möbius band

4. Conclusions and future work

It is quite common that teachers, and researchers as well, need added functionalities in their software of choice for producing high-quality graphical output or accurate numerical computations. It is very important that these functionalities do not come at the cost of a significant increase in the time required to learn their use. CindyScript is a scripting language that allows to build them easily and, moreover, one that can interact with other software components such as Maxima and the `gcc` compiler. This opens a whole new world of possibilities. For example, consider how easy it is to write the following code to interactively compute the area surrounded by a closed curve:

```
Findarea(pd):=(
 regional(p0,p1,p2,p3,s);
 if(isstring(pdstr),pd=parse(pdstr),pd=pdstr);
 s=0;
 forall(1..(length(pd)-1),
   p1=pd_#;
   p2=pd_(#+1);
   if(#==1,p0=pd_(length(pd)-1),p0=pd_(#-1));
   if(#==length(pd)-1,p3=pd_2,p3=pd_(#+2));
   s=s+IntegrateO(p0,p1,p2,p3);
);
```

It is then possible to put a slider on the screen, and the results obtained by sliding the control point are shown below.

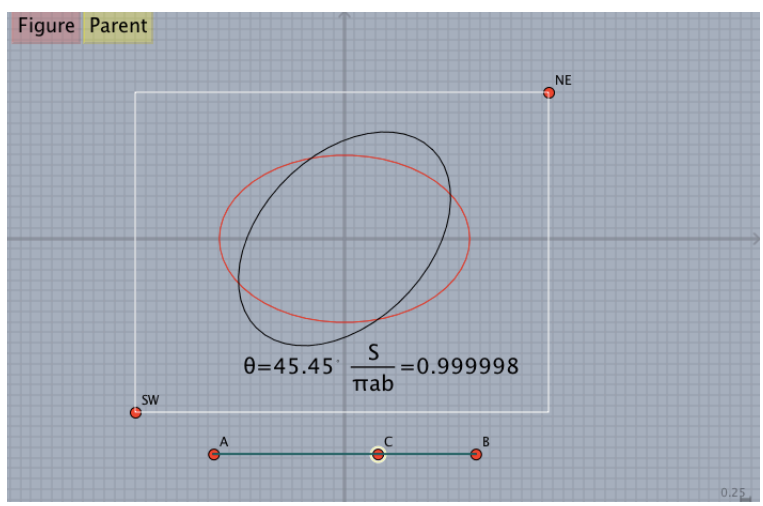


Fig.26 Adding a command with CindyScript

Programming is an important factor when creating more appealing materials, and DGSs such as Cinderella make it easy to visualize the output and modify it as necessary. Free CASs such as Maxima are also useful for symbolic computations, and it is remarkable that KETCindy can work as their user interface. Moreover, **C** compiler is very efficient in speeding up the calculations required for drawing 3D figures with KETCindy. We could say that the combined use of KETCindy, Cinderella, Maxima and C is a powerful tool to develop programs. Finally, let us comment that KETCindy also has built-in commands to generate files in `obj` format, suitable to print 3D models. As a future work, we will develop a **C** library to speed up the process.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 16K0115,18K02948,18K02872.

References

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

[2] R, `https://www.r-project.org`

[3] Maxima, `http://maxima.sourceforge.net`

[4] CTAN, `https://ctan.org`

[5] Takato S., What is and how to Use KeTCindy – Linkage Between Dynamic Geometry Software and Collaborative Use of KetCindy and Free Computer Algebra Systems and LaTeX Graphics Capabilities –, Mathematical Software –ICMS 2016, LNCS **9725**, 371–379, Springer, 2016.

[6] Takato S, McAndrew, Vallejo J, Kaneko M., Collaborative use of KeTCindy and free Computer Algebra Systems, Mathematics in Computer Science 11 3-4 , 503-514, 2017.

[7] Oshima T., Drawing Curves, Mathematical Progress in Expressive Image Synthesis III, edited by Y. Dobashi and H. Ochiai, Mathematics for Industry, 24, 95–106, Springer, 2016 ISBN : 9789811010750.

[8] Takato S, Vallejo J, Interfacing Free Computer Algebra Systems and C with KeTCindy, Computer Algebra Systems in Teaching and Research, Siedlce University of Natural Sciences and Humanities Volume 6, 172–185, (2017)

[9] Takato S, Vallejo J, Hamiltonian dynamical systems: symbolical, numerical and graphical study, to appear in Mathematics in Computer Science,2018.

[10] 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.

[11] 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, vol. 8592, pp. 629-634, Springer, Heidelberg, 2014.

[12] 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, pp.62-71, IEEE, 2013.

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

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

[15] Kortenkamp U., Interoperable interactive geometry for Europe, The Electronic Journal of Mathmatics and Technology, 5(1), pp.1-14, 2011.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers 16K01152, 18K02948, 18K02872.

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

José A. Vallejo
Universidad Autónoma de San Luis Potosí
Av. Salvador Nava s/n
78290 San Luis Potosí (SLP)
México
e-mail: jvallejo@fc.uaslp.mx