Using Maplesoft Maple to Create Professional Quality Images for Use in Teaching

Davorin Dujmovic
Suffolk County Community College-Grant
1001 Crooked Hill Rd, Mathematics Department, Brentwood, NY 11717
dujmovd@sunysuffolk.edu

Introduction

Maplesoft Maple is a computer algebra system used also in numeric computation since 1982. The current stable release is Maple 2018 from March 2018. Most of the functionality of this software is provided by libraries or the so-called packages that are available together with the main program. The software's ability of developing images of professional quality is usually not utilized as the main application of this software. We are demonstrating this capacity both in 2-D and 3-D images that are of quality that would meet desktop publishing standards and as such that can be used in teaching, in textbook publications, and online. Good reference for most of the instructions we are using in the examples can be found online [1], in Maplesoft reference manual [2], and in more recent good introductory book on Maple [3].

Modularity

The most important aspect of developing the so-called worksheets with the ultimate result of professional quality graphics is modularity. By this we mean multiple worksheets that are compiled within the same kernel or the so-called engine. This is accomplished by selecting the "Options" from the tab "Tools" with the selections chosen as in the figure 1 below. Note that we have changed the first selection to "Share one engine among all documents" and that we are also using "High Quality" print option as well as "Apply to Session". There are other options in "Display" tab most notably separate window display for plot if needed. The important option of anti-aliasing plots is turned on by default. It is important to do all this before the worksheets have been loaded into the current session.

Once the worksheets are separated one has to be aware that variables will be compiled together within the same kernel (engine) of the program which means that one needs to be able to track them down with descriptive names to avoid errors during the compilation. Every time the error is made the plot output would revert to default which results in change of plot outputs and that could be undesirable in saving the whole work. The newer versions of the software preserve the plot output format and size in every new compilation that has not resulted in error which is a vast improvement as it allows experimentation with addition of new plot options, image processing instructions, and drawing tools without losing already created optimal plot format. Of course every desirable plot output should be saved together with the whole worksheet. Images can be saved directly by right-clicking on the plot output and then using "Export" option.

2-D Example

This is example is about creating image for the statistical topic of Hypothesis Testing which demonstrates critical precision of graphical display of the statistical test result. Our first worksheet is used to load libraries, to define display of (numerical) precision, and to create custom coordinate grid since the default Maple coordinate grid is not graphically attractive. The following is the sequence of Maple instructions from the worksheet.

Packages loaded

```
>with(Statistics):
>with(plots):
>with(Typesetting):
```

Precision definition through interface(displayprecision) rather than Digits

```
>#Digits:=6:
>interface(displayprecision=3):
```

Grid definition

```
>k1:=-4: k2:=4: l1:=-0.1: l2:=0.5: stepx:=1: stepy:=0.1:
>mygrid1:= seq(implicitplot(x=k,x=k1..k2,y=l1..l2,color=red,linestyle=dot), k=k1..k2,
stepx):
>mygrid2:= seq(implicitplot(y=k,x=k1..k2,y=l1..l2, color=blue,linestyle=dot), k=l1..l2,
stepy):
>mygrid3:= implicitplot([x=0,y=0],x=k1..k2,y=l1..l2,color=black):
```

The second worksheet is about defining colors and display of points and texts within the image. The following is the sequence of instructions from the worksheet.

Text elements and points for critical values

```
>textcolor1:=ColorTools:-Color([1,0,1]):
>textcolor2:=ColorTools:-Color([0,0,1]):
>textcolor3:=ColorTools:-Color([1,0,0]):
>text1:=textplot([x1,l1, typeset("Critical Value:", x1),font=[TIMES,Bold,12]], align =
above,color=textcolor1):
>text2:=textplot([x2,l1, typeset("Critical Value:", x2),font=[TIMES,Bold,12]], align =
above,color=textcolor1):
>points1:=plot({[x1,0],[x2,0]},x=k1..k2,y=l1..l2,symbol=solidbox,style=point,thickness=
2, symbolsize=10,color=textcolor1):
>text3:=textplot([-1.265,l1+0.13, typeset("Test Value:", -1.265),font=[TIMES,Bold,12]],
align = above,color=textcolor2):
>points2:=plot({[-1.265,0]},x=k1..k2,y=l1..l2,symbol=solidbox,style=point,thickness=2,
symbolsize=10,color=textcolor2):
>text4:=textplot([0.0,0.25, typeset("Confidence: 0.95"),font=[TIMES,Bold,12]], align =
above):
```

```
>text5A:=textplot([-3,0.1, typeset("Rejection Region,"),font=[TIMES,Bold,12]], align = 
above,color=textcolor3):
>text5B:=textplot([-3,0.05, typeset("Probability 0.025"),font=[TIMES,Bold,12]], align = 
above,color=textcolor3):
>text6A:=textplot([3,0.1, typeset("Rejection Region,"),font=[TIMES,Bold,12]], align = 
above,color=textcolor3):
>text6B:=textplot([3,0.05, typeset("Probability 0.025"),font=[TIMES,Bold,12]], align = 
above,color=textcolor3):
>text:=display({text1,text2,text3,text4,text5A, text5B,text6A, text6B}):
```

Note that in the last program instruction we have used definition of plot display of all of the text elements which allowed us to have less cluttered final plot display instruction in the third worksheet. This is a typical strategy that allows for better readability of the final worksheet.

The final worksheet is with the definitions of graphical elements and the final display output. Sometimes it is a good idea to keep certain instructions, that one expects to use in other projects, commented out as we did with the first instruction below. The following is the sequence of instructions from the worksheet.

Definition of normal distribution

```
>#X:=RandomVariable(Normal(mu,sigma)):
>X:=RandomVariable(Normal(0,1)):
>normalpdf:=x->PDF(X,x):
>normalcdf:=x->CDF(X,x):
>inversenormalcdf:=y->Quantile(X,y):
```

Critical values

```
>x1:=inversenormalcdf(0.025): x2:=inversenormalcdf(0.975):
```

Graph definition

```
>BaseGraph:=plot([normalpdf(x)],x=k1..k2,color=[blue],discont=true,thickness=[2]):
>StripRed:=(c,d)->plot([normalpdf(x)],x=c..d,thickness=1,labels=["x",P( X <= 
d)],title="",labelfont=[TIMES,BOLD,14],titlefont=[TIMES,REGULAR,14],color=COL
OR(RGB,1,0,0),discont=true,thickness=[2],filled=true,transparency=0.5):
>StripYellow:=(c,d)->plot([normalpdf(x)],x=c..d,thickness=1,labels=["x",P( X <= 
d)],title="Hypothesis Test 
Example",labelfont=[TIMES,BOLD,14],titlefont=[TIMES,REGULAR,14],color=COLO
R(RGB,1,1,0),discont=true,thickness=[2],filled=true,transparency=0.5):
```

Display alltogether

```
>display({mygrid1,mygrid2,mygrid3, 
BaseGraph,StripRed(k1,x1),StripYellow(x1,x2),StripRed(x2,k2), text, points1, 
points2},labels=[``,``]);
```

The final graphical output is displayed in the figure 2. Note that we used image processing option “transparency” which improves the image quality whenever the transparency is desirable, particularly in 2-D graphics.

We have also used “Drawing” tools which are available on a separate ribbon by clicking on the plot output. The tools are very elementary but it allowed us to use framing the text as well as arrow pointers as seen on the figure 2.

3-D Example

In general 3-D graphics are more difficult to make but Maple has made that task one of the flagship achievements of the program and this is probably the very best software for this particular use. Thus even though our 3-D graphics seem to be more elaborate they are actually simpler than the previous 2-D example. One of the things to be avoided with 3-D graphics is extensive use of transparency. This is because rendered 3-D graphics are actually 2-dimensional illusions of 3-dimensional space and adding transparency to the elements of plot output creates diffusion of these elements as where they are not expected. Sometimes this leaves graphical artifacts as well.

Our example is about display of geometrical interpretation of topic of Partial Derivatives in Calculus.

The initial worksheet is loading basic libraries and defining global coordinate boundaries with the following program instructions.

Libraries and bounds

```
>with(plots):
>with(linalg):
>with(LinearAlgebra):
>with(plottools):
>k1:=-5:k2:=5:
>l1:=-5:l2:=5:
>m1:=-5:m2:=5:
```

In this example we are not using any custom grid because it would introduce unnecessary clutter in the final plot output. In the final (third) worksheet we selected “boxed” option for the coordinate system which is usually more desirable in case of presenting curved surfaces as it is the case in our example.

The second worksheet is about textual elements for two separate plots and the following are the program instructions from the worksheet.

Text for 3d graph

```
>text1tangent:=textplot3d([[0,3,15,"Line:",'font'=["times","roman",12],color=red],[0,3,10
,r(x)=[-1+x,1,1+4*x]]],`align`=`right`, color=red):
```

```
>text1spacecurve:=textplot3d([[0,3,25,"Space
curve:",'font'=["times","roman",12],color=red],[0,3,20,s(x)=[x,1,3-
2*x^2]]],`align`=`right`, color=red):
>text1surface:=textplot3d([[3,0,40,"Surface:",'font'=["times","roman",12],color=black],[3
,0,35,z(x,y)=4-2*x^2-y^2]],`align`=`right`, color=black):

>text2tangent:=textplot3d([[0,3,15,"Line:",'font'=["times","roman",12]],[0,3,10,r(y)=[-
1,1+y,1-2*y]]],`align`=`right`, color=blue):
>text2spacecurve:=textplot3d([[0,3,25,"Space
curve:",'font'=["times","roman",12]],[0,3,20,s(y)=[-1,1+y,2-y^2]]],`align`=`right`,
color=blue):
>text2surface:=textplot3d([[3,0,40,"Surface:",'font'=["times","roman",12]],[3,0,35,z(x,y)
=4-2*x^2-y^2]],`align`=`right`, color=black):
```

Positioning of the text has been experimented with during compilation as it is hard to predict exact positions of all of the 3-dimensional plot elements beforehand. One can also manipulate 3-dimensional plot output with axial rotation and select the best view of the plot. This manipulation is also available upon clicking on the plot output.

The final worksheet is about creating two graphics shown in figure 3 and 4 respectively. The reason why we have both graphics on the same worksheet is because we wanted to achieve the same size of both images. Unfortunately a selection of precise image size is not available within the Maple program. The following are program instructions of the final worksheet.

Graphic elements

```
>trace1:=spacecurve([t,1,3-2*t^2],t=k1..k2, thickness=2,color=red):
>trace2:=spacecurve([-1,t,2-t^2],t=k1..k2, thickness=2,color=blue):
>line1:=spacecurve([-1+t,1,1+4*t],t=l1..l2, color=red):
>line2:=spacecurve([-1,1+t,1-2*t],t=l1..l2, color=blue):
>graph0:=plot3d(4-2*x^2-y^2, x=k1..k2, y=l1..l2, axes=boxed, numpoints=25000,
style=polygonOutline, contours=100):
```

First plot

```
>plots[display](graph0, line1, trace1, text1tangent,text1spacecurve, text1surface,
axes=boxed, labels=[`x`,`y`,``]);
```

Second plot

```
>plots[display](graph0, line2, trace2, text2tangent,text2spacecurve, text2surface,
axes=boxed, labels=[`x`,`y`,``]);
```

Text elements can also be inserted from tab of “Drawing” tools but if chosen as such they would change with every plot manipulation which makes them less useful than strict positioning through instructions.

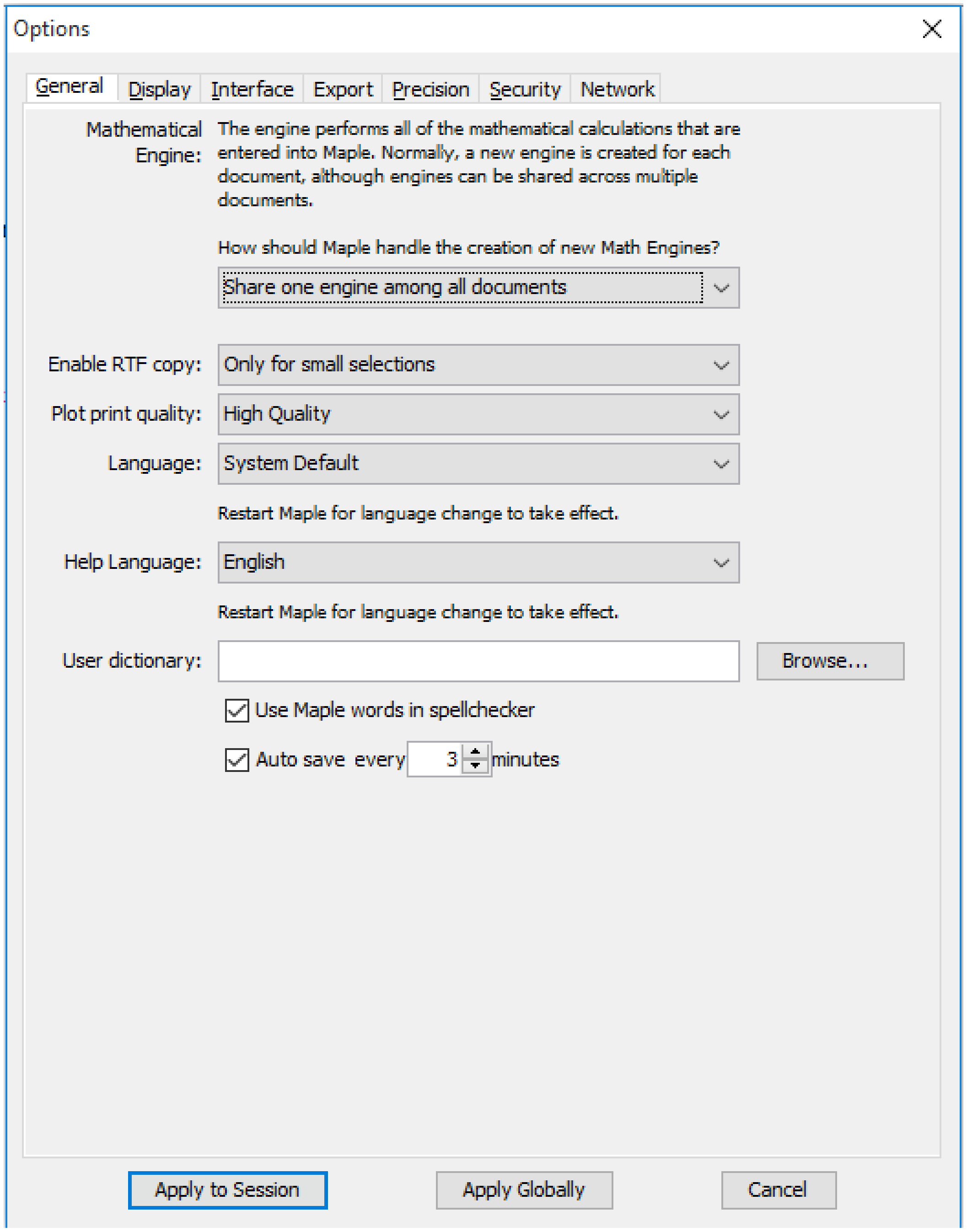
Deficiency and Possible Improvements

Even though the software abilities are greatly improved since its inception we can see a number of deficiencies that should be avoided in the future releases of the program. The current version of software has no option of rotating “textplot” elements. This is necessary to be able to align text elements with graphical elements which is currently not possible. Image processing options are very elementary. Since they are only available through instructions or through the so-called “palettes” they are not applicable directly on the plot output which would be more desirable. Drawing tools outputs are not precise enough and with any change of size of a plot they end up misaligned with other graphical elements. As we noted earlier the ability to save images from the plot output does not have an option to make a precise image size in pixels which is one of the most important image processing choices. To change size of the saved image outside the program by using separate image processing inevitably reduces the sharpness of the plot content. These are the main problems that can be worked out in the future releases of Maple. We would also like to see Maple help files with more examples of plotting options.

References

1. https://www.maplesoft.com/support/help/maple/view.aspx?path=reference
2. https://www.maplesoft.com/documentation_center/maple2017/UserManual.pdf
3. Ian Thompson (2016). Understanding Maple. Cambridge University Press.

Appendix: Figures



Options

General | Display | Interface | Export | Precision | Security | Network

Mathematical Engine: The engine performs all of the mathematical calculations that are entered into Maple. Normally, a new engine is created for each document, although engines can be shared across multiple documents.

How should Maple handle the creation of new Math Engines?

Share one engine among all documents

Enable RTF copy: Only for small selections

Plot print quality: High Quality

Language: System Default

Restart Maple for language change to take effect.

Help Language: English

Restart Maple for language change to take effect.

User dictionary: Browse...

☑ Use Maple words in spellchecker

☑ Auto save every 3 minutes

Apply to Session | Apply Globally | Cancel

Fig 1. Initial Program Options from “Tools” Tab

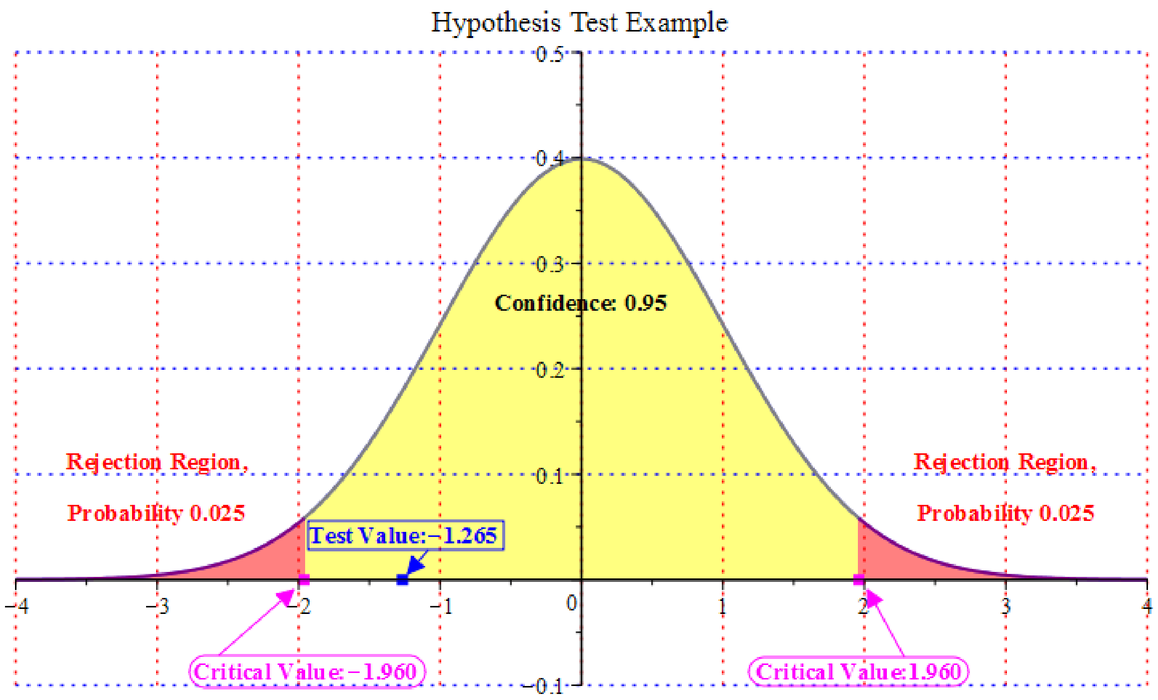


Fig 2. 2-D Example from Statistics

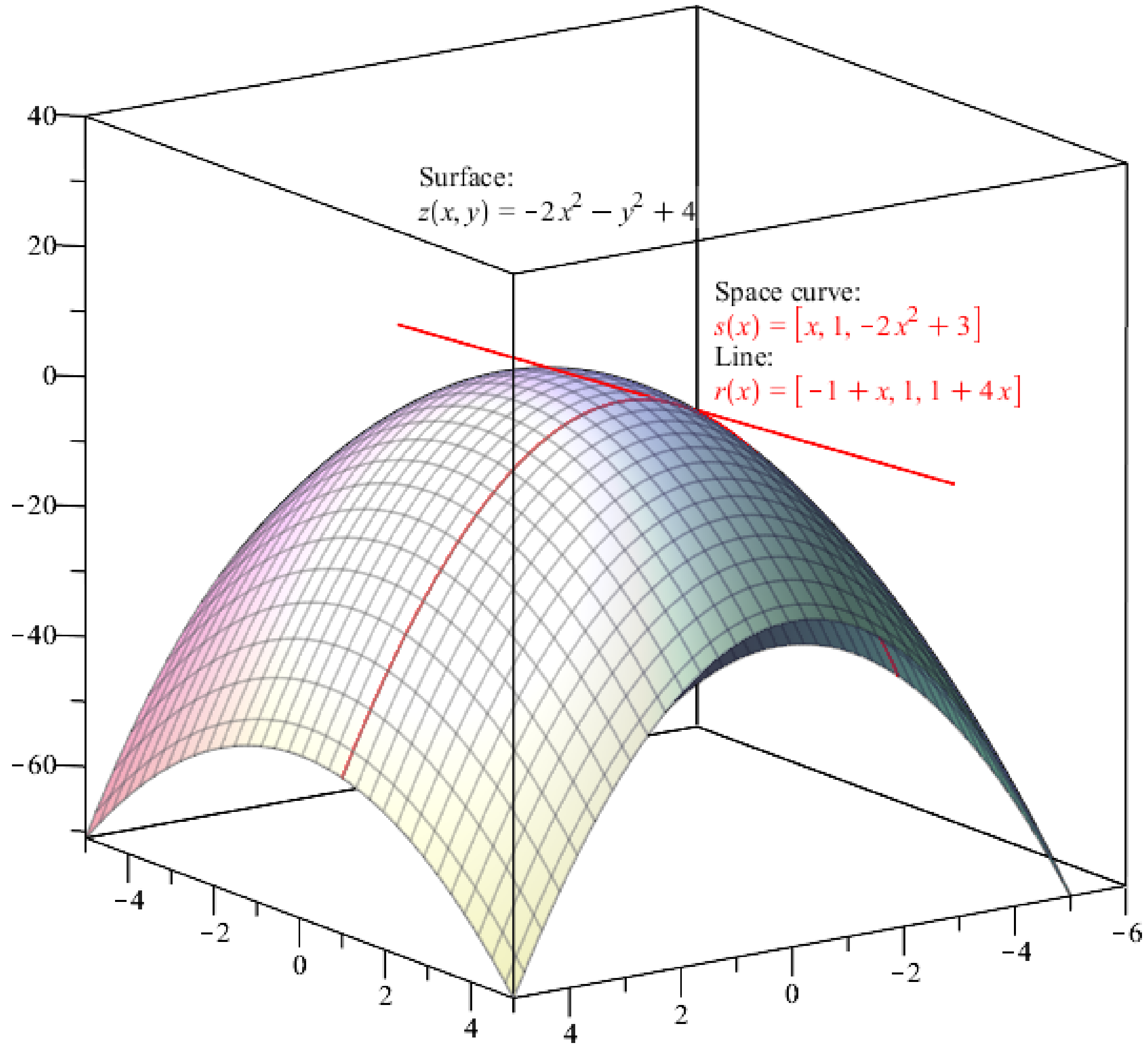


Fig 3. 3-D Example from Calculus, First Plot

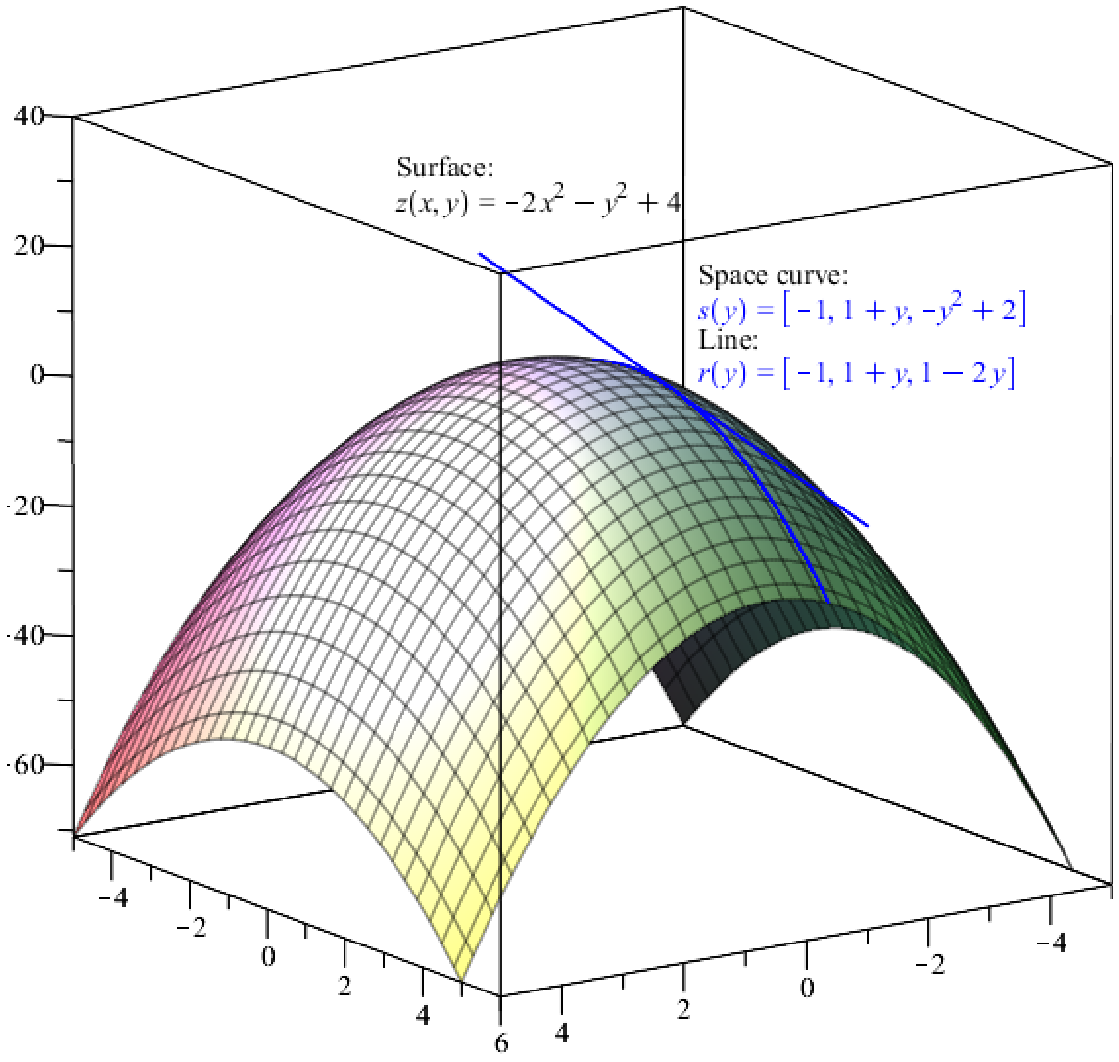


Fig 4. 3-D Example from Calculus, Second Plot