An Introduction to Matlab

Fall 2018

Instructor: Dr. Bo-Wen Shen*

RA for Symbolic Computing: Mr. Jialin Cui

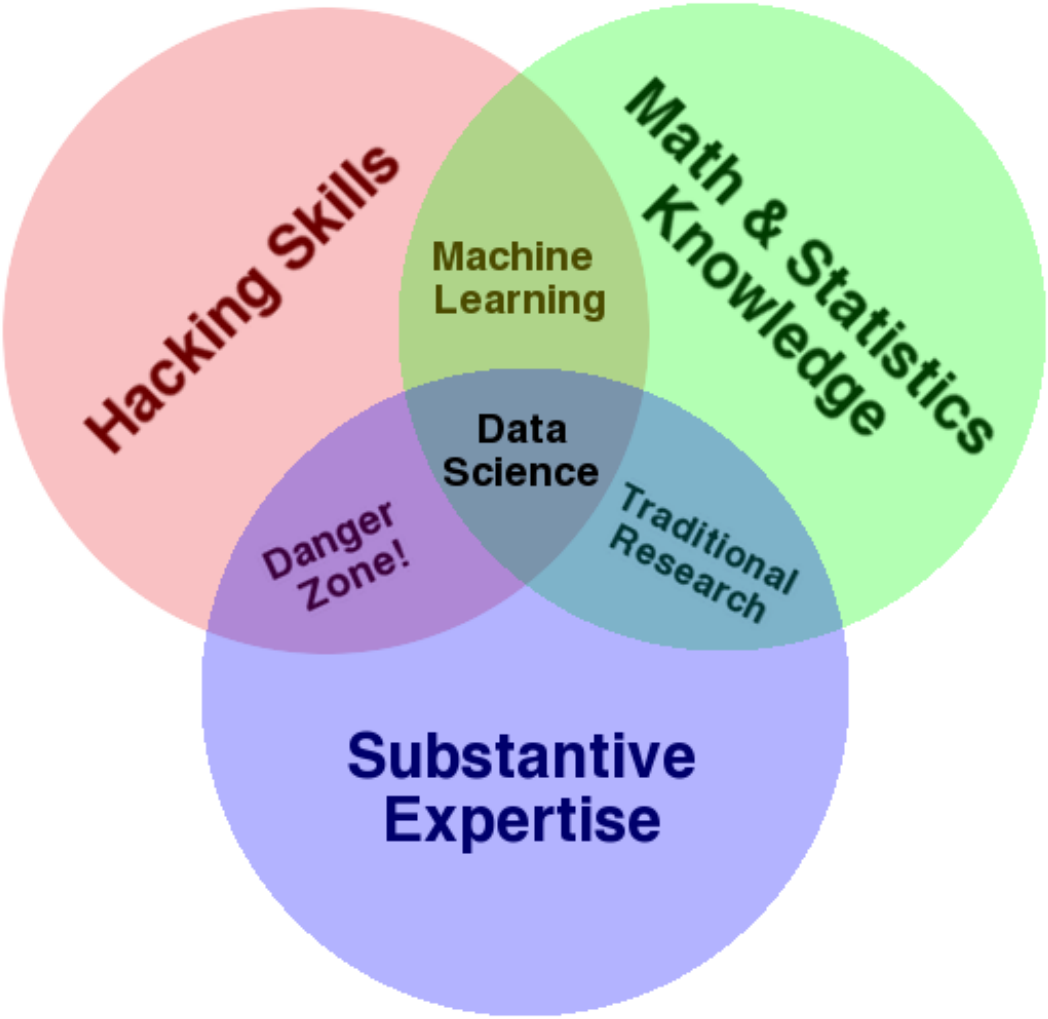
* sdsu.math252.shen@gmail.com

cjl742952519@gmail.com

Department of Mathematics and Statistics

San Diego State University

THE DATA SCIENCE VENN DIAGRAM



Variety
Velocity
Volume
Value

http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

A 10 minutes Introduction to Matlab

- MatLab is a powerful software created by MathWorks, which is used extensively in mathematics, engineering, and the sciences.
- MatLab stands for Matrix Laboratory.

Three key capabilities are

1. Computing
2. Visualizations (or Graphics)
3. Symbolic computation (or computer algebra)

➔ Symbolic plotting (to get plots for symbolic functions)

References:

http://www-rohan.sdsu.edu/~jmahaffy/courses/f15/math337/Lectures.html
http://www-rohan.sdsu.edu/~jmahaffy/courses/f15/math337/beamer/matlab/basic_matlab.pdf
http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
http://www.maths.dundee.ac.uk/ftp/na-reports/MatlabNotes.pdf
http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

Matlab and Mathematical Software

- Many forms of general-purpose numerical software are available commercially and in the public domain. Most of the early software was written for mainframe computers. Now that personal computers are sufficiently powerful, standard numerical software is available for them. Most of this numerical software is written in FORTRAN, although some packages are written in C, C++, and FORTRAN90.
- ALGOL procedures were presented for matrix computations in 1971 in [WR]. A package of FORTRAN subroutines based mainly on the ALGOL procedures was then developed into the EISPACK routines.
- LINPACK is a package of FORTRAN subroutines for analyzing and solving systems of linear equations and solving linear least squares problems.
- MATLAB was originally written to provide easy access to matrix software developed in the LINPACK and EISPACK projects. The first version was written in the late 1970s for use in courses in matrix theory, linear algebra, and numerical analysis. There are currently more than 500,000 users of MATLAB in more than 100 countries.
- In addition to Matlab, Maple (developed in 1980) and Mathematica (released in 1988) are equally popular.

References:

Burden, R., D. Faires and A. Burden, 2015: Numerical Analysis. Cengage Learning. 912pp. 10th ed. Jan. 2015.

Symbolic Computation: Differential and Integral Calculus

```
syms x                    %In Symbolic Math Toolbox™, symbolic variables are complex variables by default.
y=x^5
diff (y, x)
➔5*x^4
=====================================
syms x n real             #assumption
y=x^n
diff (y, x)
➔ ans =
    n*x^(n - 1)
=====================================
assume (n>0)
int(y,x)
➔ ans =
    x^(n + 1)/(n + 1)
```

Symbolic Computation: the Inner Product (Section 12.3)

```
syms x y z                  #In Symbolic Math Toolbox™, symbolic variables are complex variables by default.
r=[x, y, z]
syms x0 y0 z0
r0=[x0, y0, z0]
syms a b c real             #assumption
n=[a, b, c]
dot (n, r-r0)
```

➔ a*(x - x0) + b*(y - y0) + c*(z - z0)

Symbolic Computation: the Cross Product (Section 12.4)

```
syms p1 p2 p3 real
P=[p1, p2, p3]
syms q1 q2 q3 real
Q =[q1, q2, q3]
syms r1 r2 r3 real
R =[r1, r2, r3]

n=cross(Q-P, R-P)      %PQ x PR
dot (n, r-r0)
```

%In Symbolic Math Toolbox™, symbolic variables are complex variables by default.

➔((p2 - q2)*(p3 - r3) - (p3 - q3)*(p2 - r2))*(x - x0) - ((p1 - q1)*(p3 - r3) - (p3 - q3)*(p1 - r1))*(y - y0) + ((p1 - q1)*(p2 - r2) - (p2 - q2)*(p1 - r1))*(z - z0)

```
syms x y z real
X=[x, y, z]
planefunction = dot (n, X-P)      %PX*n
```

Section 12.6: Cylinders and Quadric Surface using ezsurf of MatLab

- A cylinder is a surface that consists of all lines (called rulings) that are parallel to a given line and pass through a given plane curve.
- A quadric surface is the graph of a second-degree equation in three variables (x, y, z).

$$Ax^2 + Bx^2 + Cx^2 + Dxy + Eyz + Fxz + Gx + Hy + Iz + J = 0$$

Easy-to-use 3-D colored surface plotter

- ezsurf(fun)
- ezsurf(funx,funy,funz)
- ezsurf(...,'circ')

EXAMPLE 1 Sketch the graph of the surface $z = x^2$.

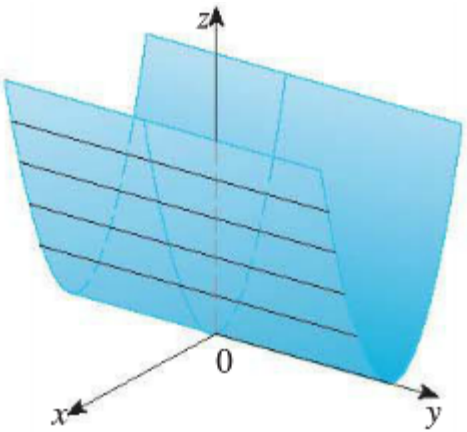
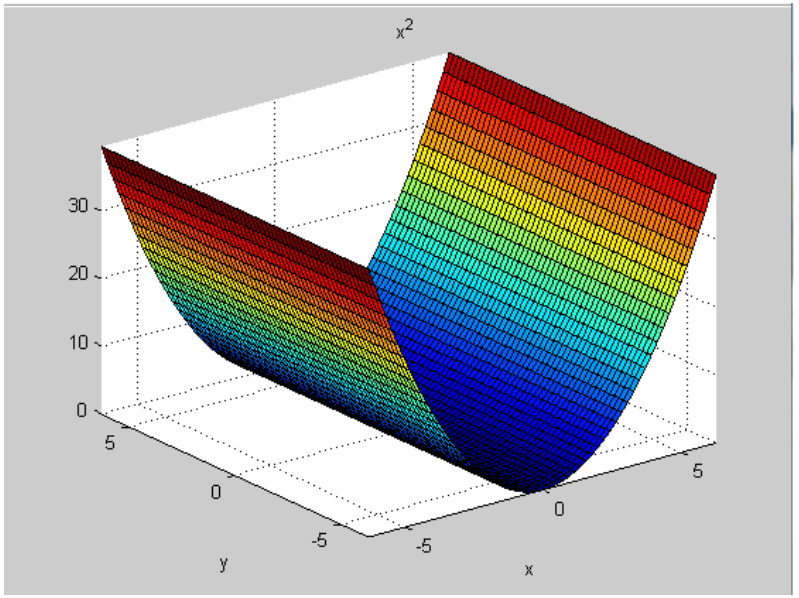


FIGURE 1
The surface $z = x^2$ is a parabolic cylinder.

The rulings of the cylinder are parallel to the y-axis.



Section 12.6: Cylinders and Quadric Surface using ezsurf of MatLab

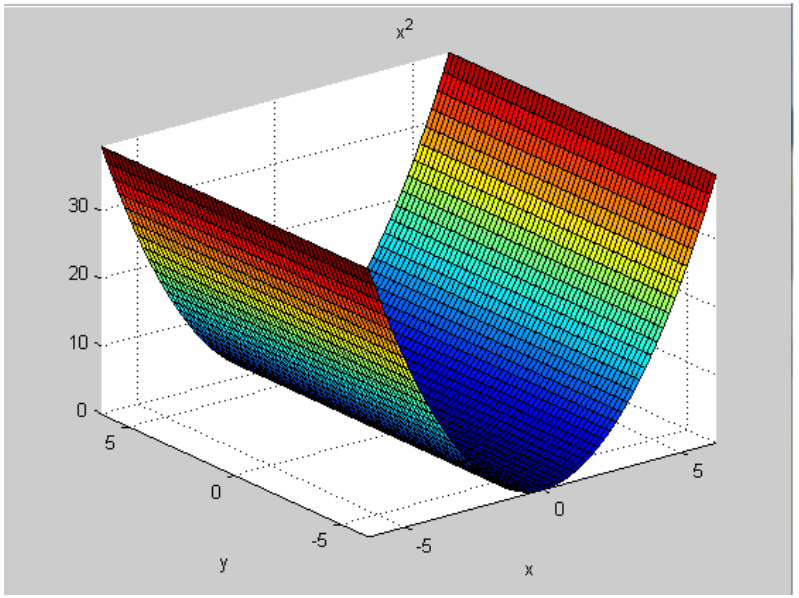
Easy-to-use 3-D colored surface plotter

- ezsurf(fun)
- ezsurf(funx,funy,funz)
- ezsurf(...,'circ')

```
syms x y z
ezsurf(x^2)
```

EXAMPLE 1 Sketch the graph of the surface $z = x^2$.

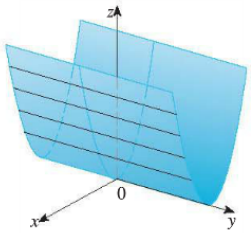
- The rulings of the cylinder are parallel to the y-axis.



Section 12.6: Surface plots

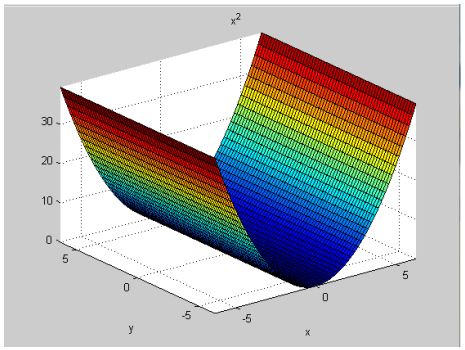
```
syms x y z
ezsurf(x^2)
```

EXAMPLE 1 Sketch the graph of the surface $z = x^2$.



parabolic cylinder

rulings: parallel to the y-axis




```
syms t
x=cos(t)
y=sin(t)
ezsurf(x,y,z)
```

EXAMPLE 2 Identify and sketch

(a) $x^2 + y^2 = 1$

axis: the z-axis
rulings: vertical lines
given curve: the circle

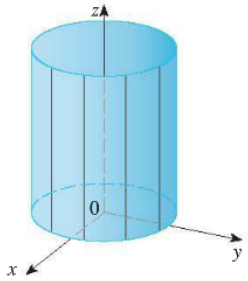
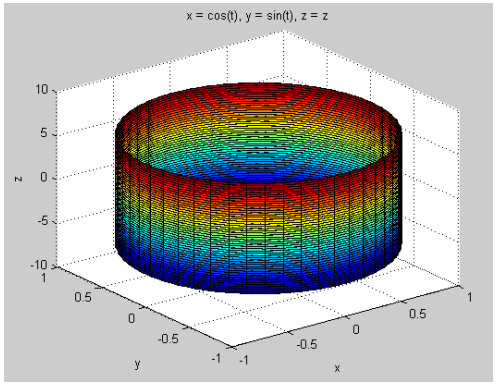


FIGURE 2 $x^2 + y^2 = 1$




```
syms t x y z
y=cos(t)
z=sin(t)
ezsurf(x,y,z)
```

(b) $y^2 + z^2 = 1$

axis: the x-axis
rulings: parallel to the axis

circular cylinder

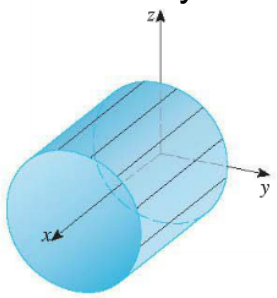
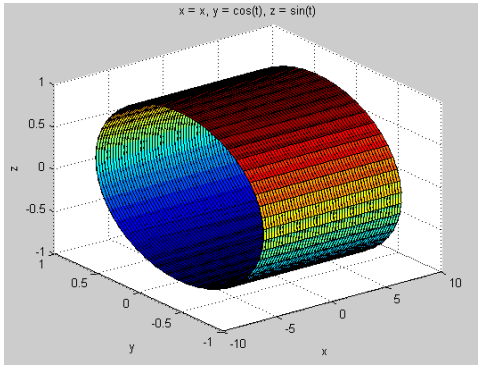
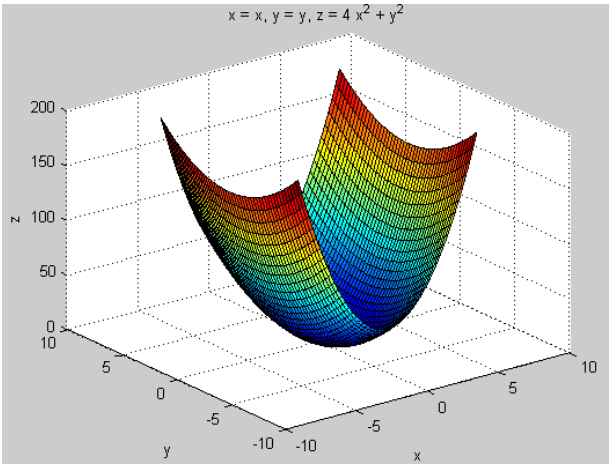


FIGURE 3 $y^2 + z^2 = 1$



Section 12.6: Surface plots

EXAMPLE 4 Use traces to sketch the surface $z = 4x^2 + y^2$.




```
syms x y z
z=4*x^2 + y^2
ezsurf(x, y, z)
```

elliptic paraboloid

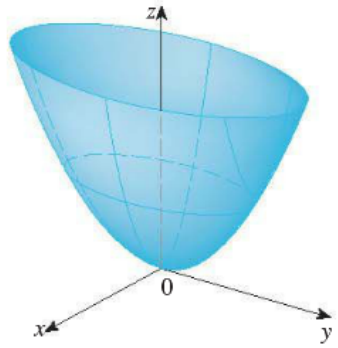
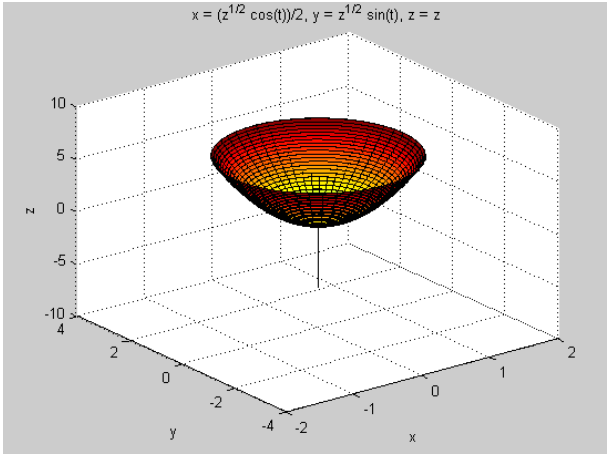
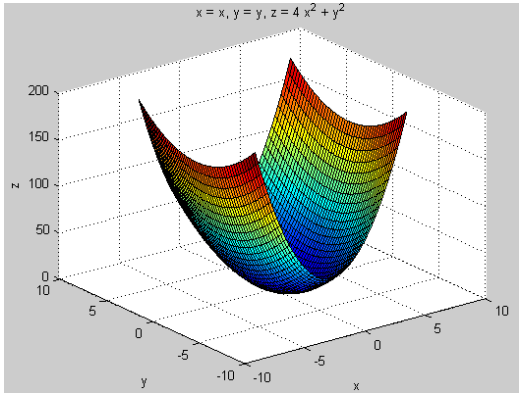


FIGURE 5
The surface $z = 4x^2 + y^2$ is an elliptic paraboloid. Horizontal traces are ellipses; vertical traces are parabolas.

```
syms x y z t
x=sqrt(z)/2*cos(t)
y=sqrt(z)*sin(t)
ezsurf(x,y,z)
```



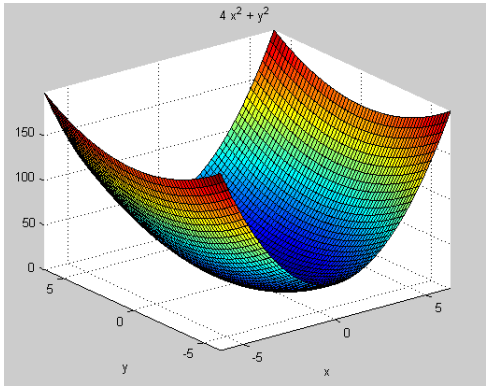
Section 12.6: Surface plots



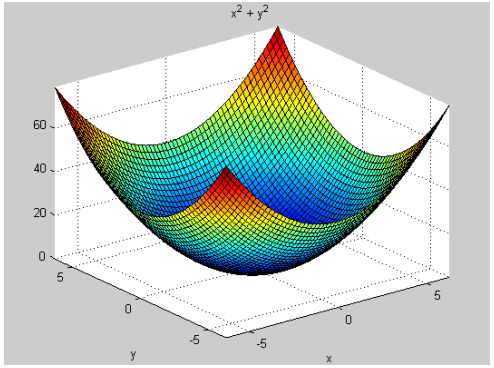
$$z = 4x^2 + y^2.$$

(left, top)
syms x y z
z=4*x^2 + y^2
ezsurf(x, y, z)

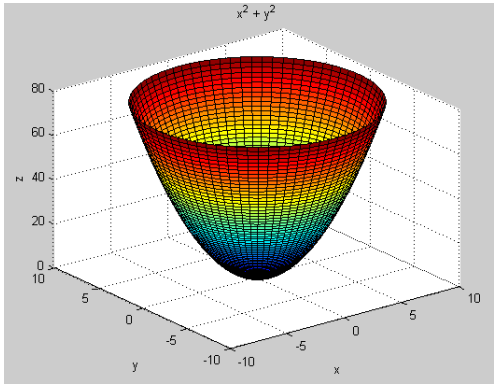
syms x y z
ezsurf(4*x^2 + y^2)



$$x^2 + y^2 = 1$$



(left, bottom)
syms x y z
ezsurf(x^2 + y^2)



(right, bottom)
syms x y z
ezsurf(x^2+y^2, 'circ')

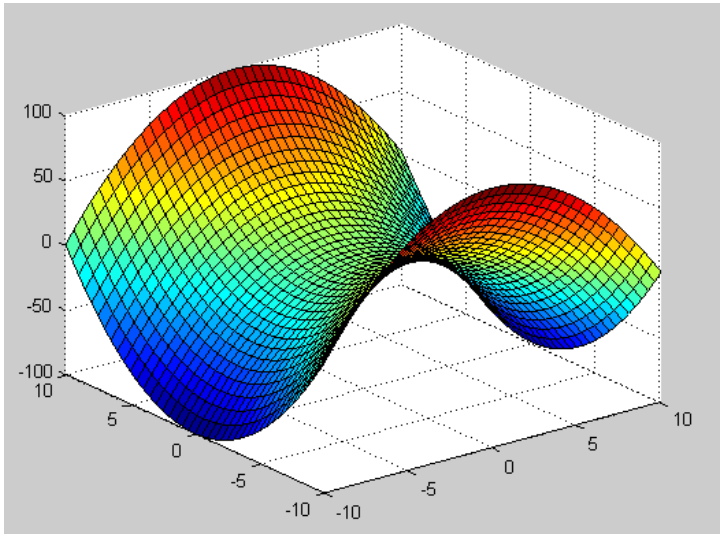
%ezsurf(...,'circ') plots fun ("function") over a disk centered on the domain.

Section 12.6: Surface plots

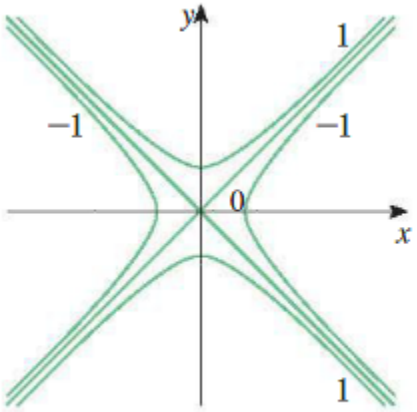
EXAMPLE 5 Sketch the surface $z = y^2 - x^2$.

```
syms x y z
ezsurf(y^2-x^2)
```

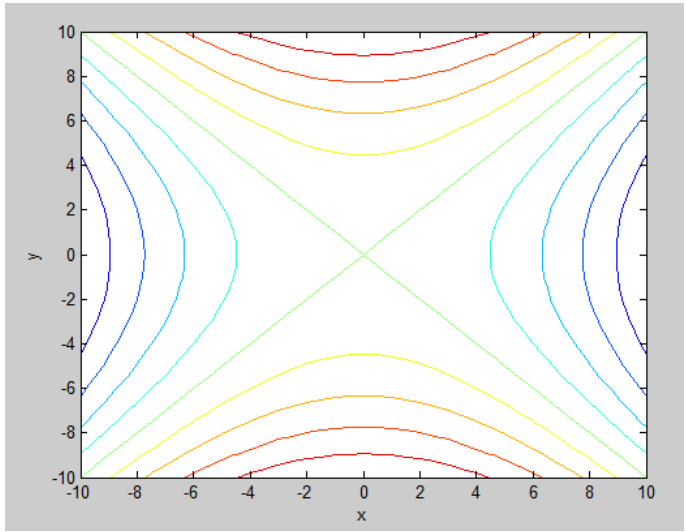
hyperbolic paraboloid




```
%Hyperbolic paraboloid
x = [-10:.5:10]; y=[-10:.5:10];
[X, Y] = meshgrid(x,y);
Z = Y.^2-X.^2;
surf(X,Y,Z)
contour(X,Y,Z)
xlabel x
ylabel y
zlabel z
```



Traces in $z = k$ are $y^2 - x^2 = k$

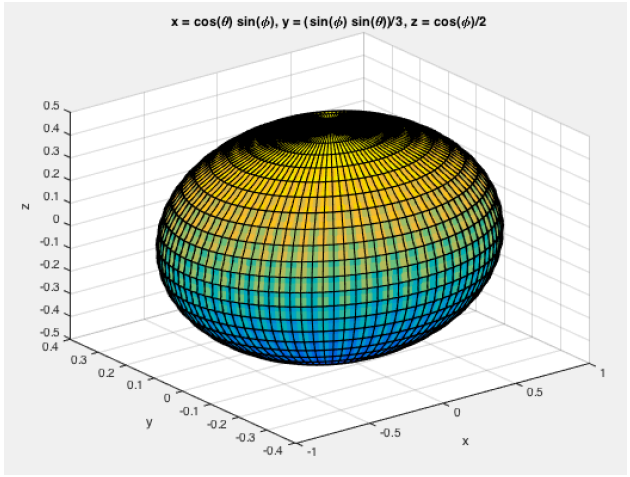


Section 12.6: Surface plots

EXAMPLE 3 Use traces to sketch the quadric surface with equation

$$x^2 + \frac{y^2}{9} + \frac{z^2}{4} = 1$$

ellipsoid




```
syms x y z phi theta real

x=sin(phi)*cos(theta)
y=1/3*sin(phi)*sin(theta)
z=1/2*cos(phi)
ezsurf(x, y, z)
```

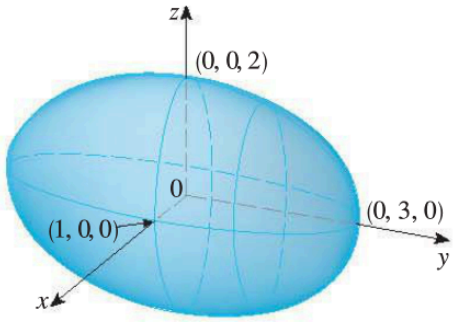
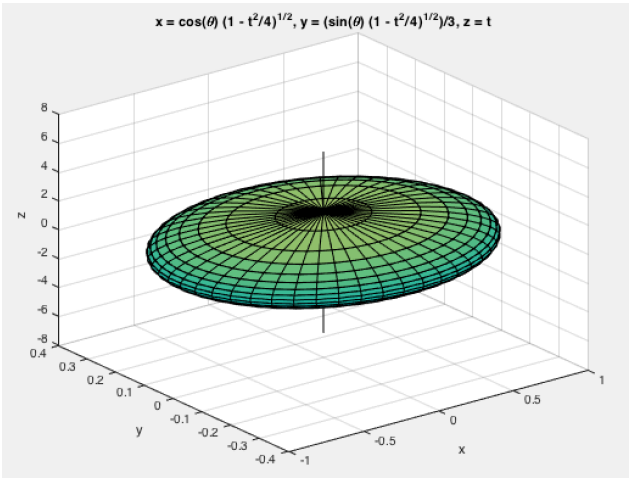


FIGURE 4

The ellipsoid $x^2 + \frac{y^2}{9} + \frac{z^2}{4} = 1$

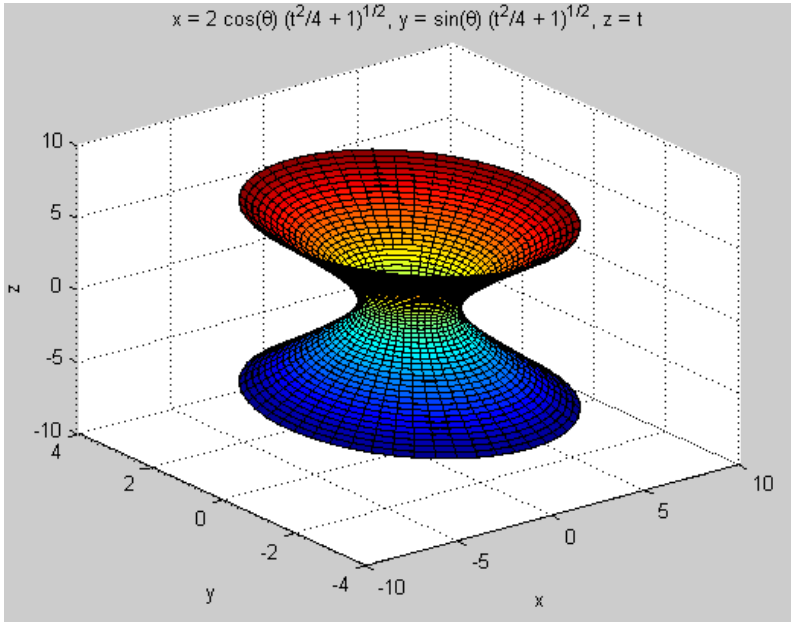



```
syms x y z t theta
z=t
x=sqrt(1-t^2/4)*cos(theta)
y=1/3*sqrt(1-t^2/4)*sin(theta)
ezsurf(x, y, z)
```

Section 12.6: Surface plots

EXAMPLE 6 Sketch the surface $\frac{x^2}{4} + y^2 - \frac{z^2}{4} = 1$.

```
syms x y z t theta
z=t
x=2*sqrt(1+t^2/4)*cos(theta)
y=sqrt(1+t^2/4)*sin(theta)
ezsurf(x, y, z)
```



Hyperboloid of One Sheet

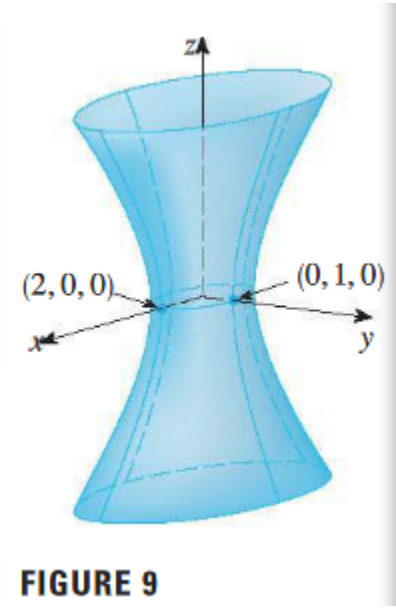
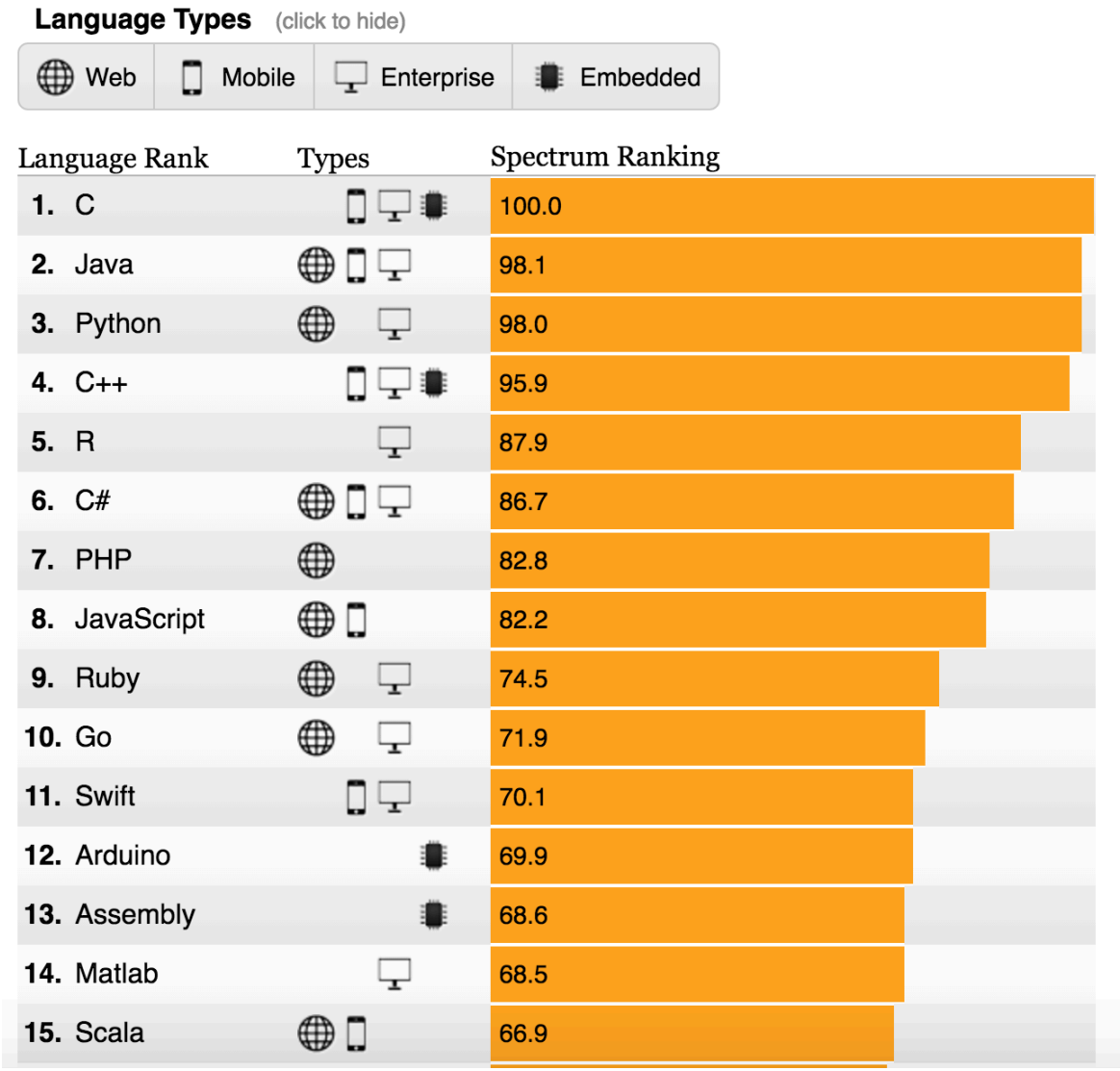


FIGURE 9

The Top Programming Languages 2016

IEEE Spectrum

Enterprise

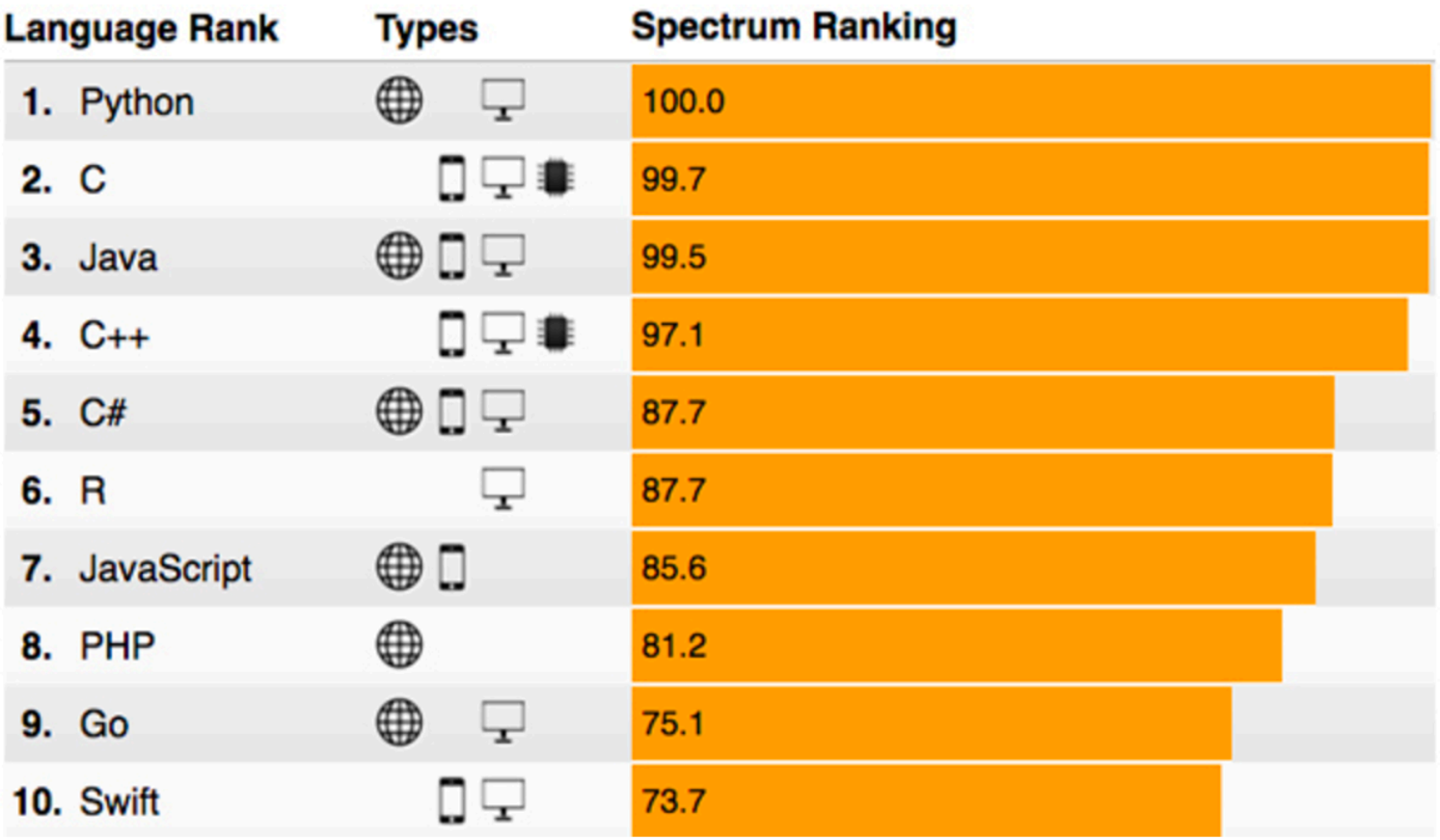


Python Facts

- Python is an open-source, high-level, OOP scripting languages;
- Python Is a widely used, general-purpose, interpreted language;
- Python is similar to MATLAB;
- Python is recommended by the Raspberry Pi Foundation

The Top Programming Languages 2017

So what are the Top Ten Languages for the typical *Spectrum* reader?



Python Facts

Python is often used for

- symbolic computation,
- scientific computing and plotting,
- multi-tasking (i.e., fork),
- multi-threading,
- network programming (TCP/IP, IPC),
- MPI parallel computing,
- web development, and
- game programming.

Symbolic Computation in Python

```
from sympy import *
x,y,z=symbols("x y z")
y=x**6
#y=x^5, error
ans1=diff(y,x)
print(ans1)
```

→ 6*x**5

```
#
#
ans2=integrate(y,x)
print(ans2)
```

→ x**7/7

```
#
simplify(ans2)
#
```

```
from sympy import *
x,y,z, n=symbols("x y z n")
y=x**n
#y=x^5, error
ans1=diff(y,x)
print(ans1)
```

→ n*x**n/x

```
#
#
ans2=integrate(y,x)
print(ans2)
```

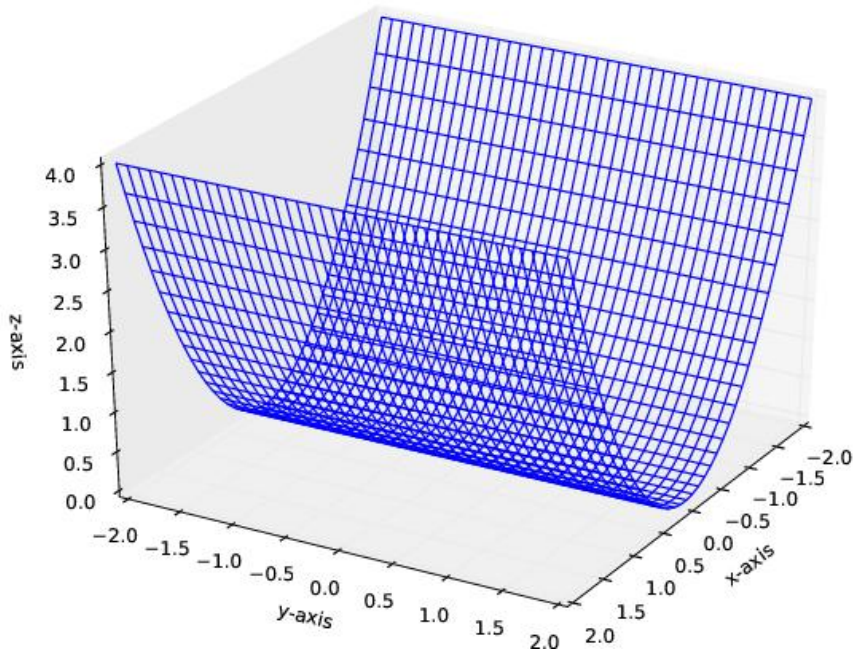
→ Piecewise((log(x), n == -1), (x**(n + 1)/(n + 1), True))

```
#
simplify(ans2)
#
```

Ex1 in Python

EXAMPLE 1 Sketch the graph of the surface $z = x^2$.

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig=plt.figure()
ax=axes3d.Axes3D(fig,azim=30,elev=30)

x=np.linspace(-2,2,40)
y=np.linspace(-2,2,40)

X, Y=np.meshgrid(x,y)
Z=X**2

ax.plot_wireframe(X,Y,Z)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

plt.savefig('ex1.ps')
plt.show()
```

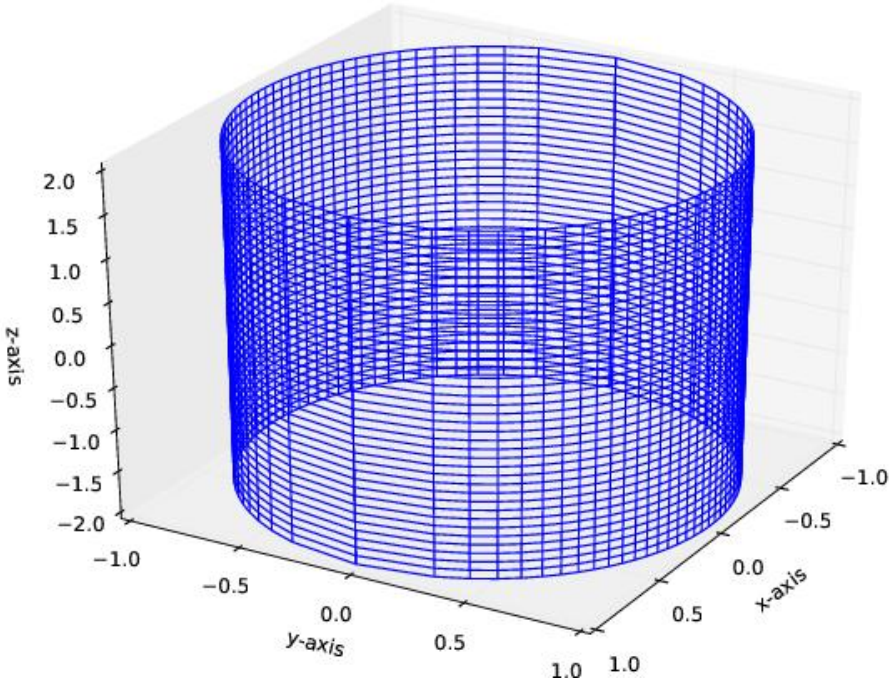


Ex2 in Python

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig=plt.figure()
ax=axes3d.Axes3D(fig,azim=30,elev=30)

x=np.linspace(-1,1,40)
z=np.linspace(-2,2,40)

X, Z=np.meshgrid(x,z)
Y=np.sqrt(1-X**2)

ax.plot_wireframe(X,Y,Z)
ax.plot_wireframe(X,-Y,Z)

ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

plt.savefig('ex2.ps')
plt.show()
```

EXAMPLE 2 Identify and sketch the surfaces.

(a) $x^2 + y^2 = 1$ (b) $y^2 + z^2 = 1$

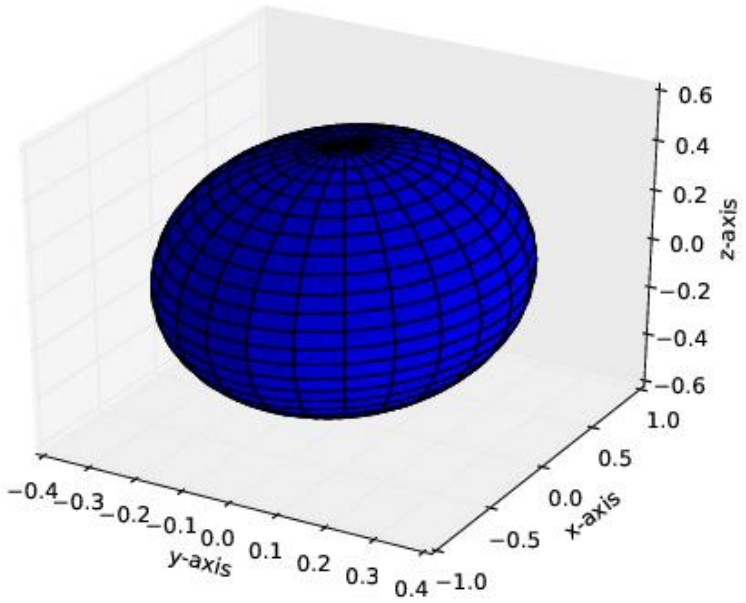


Ex3 in Python

EXAMPLE 3 Use traces to sketch the quadric surface with equation

$$x^2 + \frac{y^2}{9} + \frac{z^2}{4} = 1$$

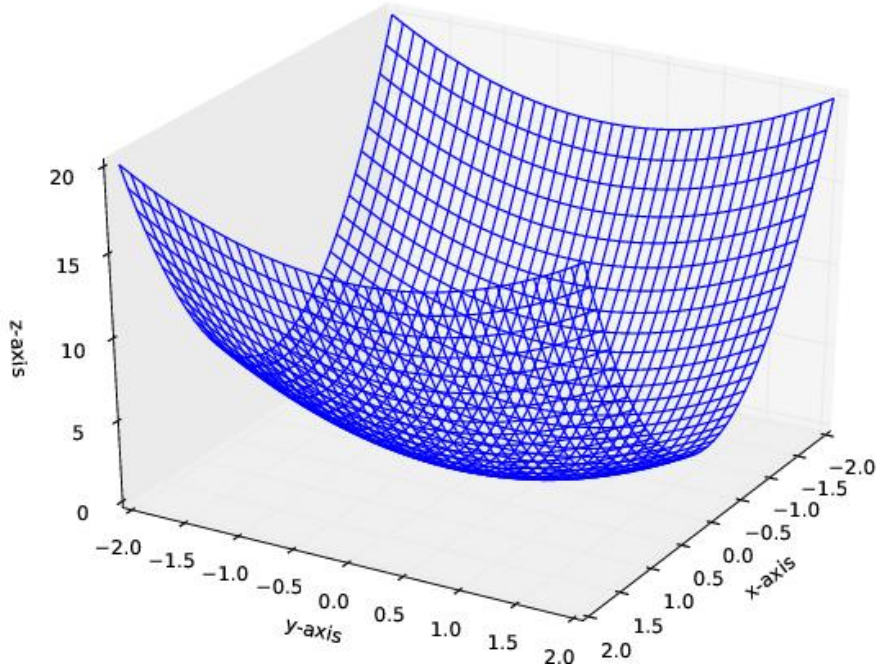
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
u = np.linspace(0, 2 * np.pi, 100)        #theta
v = np.linspace(0, np.pi, 100)            #phi
x = 1. * np.outer(np.cos(u), np.sin(v))   #outer:
y = 1./3. * np.outer(np.sin(u), np.sin(v))
z = 1./2. * np.outer(np.ones(np.size(u)), np.cos(v))
#ax.plot_surface(x, y, z,  rstride=4, cstride=4, color='b')
ax.plot_surface(y, x, z,  rstride=4, cstride=4, color='b')

ax.set_xlabel('y-axis')
ax.set_ylabel('x-axis')
ax.set_zlabel('z-axis')
plt.savefig('ex3.ps')
plt.show()
```



Ex4 in Python

EXAMPLE 4 Use traces to sketch the surface $z = 4x^2 + y^2$.

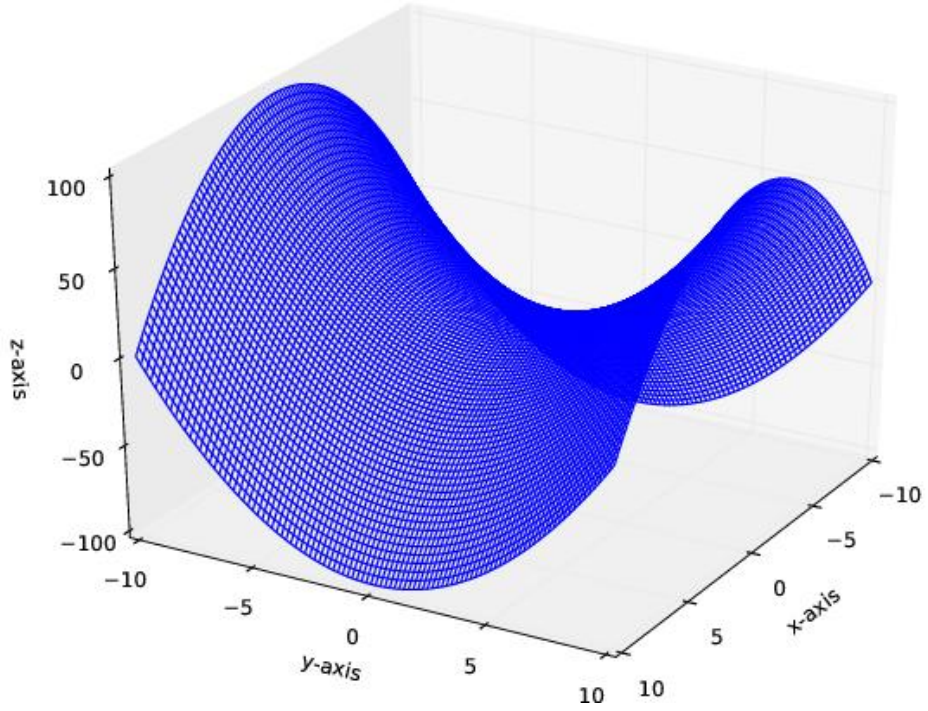
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig=plt.figure()
ax=axes3d.Axes3D(fig,azim=30,elev=30)

x=np.linspace(-2,2,40)
y=np.linspace(-2,2,40)

X, Y=np.meshgrid(x,y)
Z=4*X**2 + Y**2

ax.plot_wireframe(X,Y,Z)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

plt.savefig('ex4.ps')
plt.show()
```



Ex5 in Python

EXAMPLE 5 Sketch the surface $z = y^2 - x^2$.

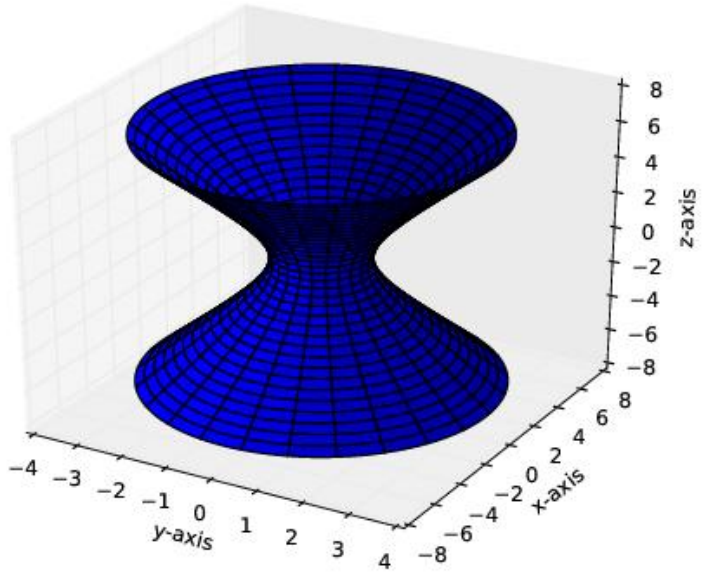
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig=plt.figure()
ax=axes3d.Axes3D(fig,azim=30,elev=30)

x=np.linspace(-10,10,100)
y=np.linspace(-10,10,100)

X, Y=np.meshgrid(x,y)
Z= Y**2 - X**2

ax.plot_wireframe(X,Y,Z)
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

plt.savefig('ex5.ps')
plt.show()
```



Ex6 in Python

EXAMPLE 6 Sketch the surface $\frac{x^2}{4} + y^2 - \frac{z^2}{4} = 1$.

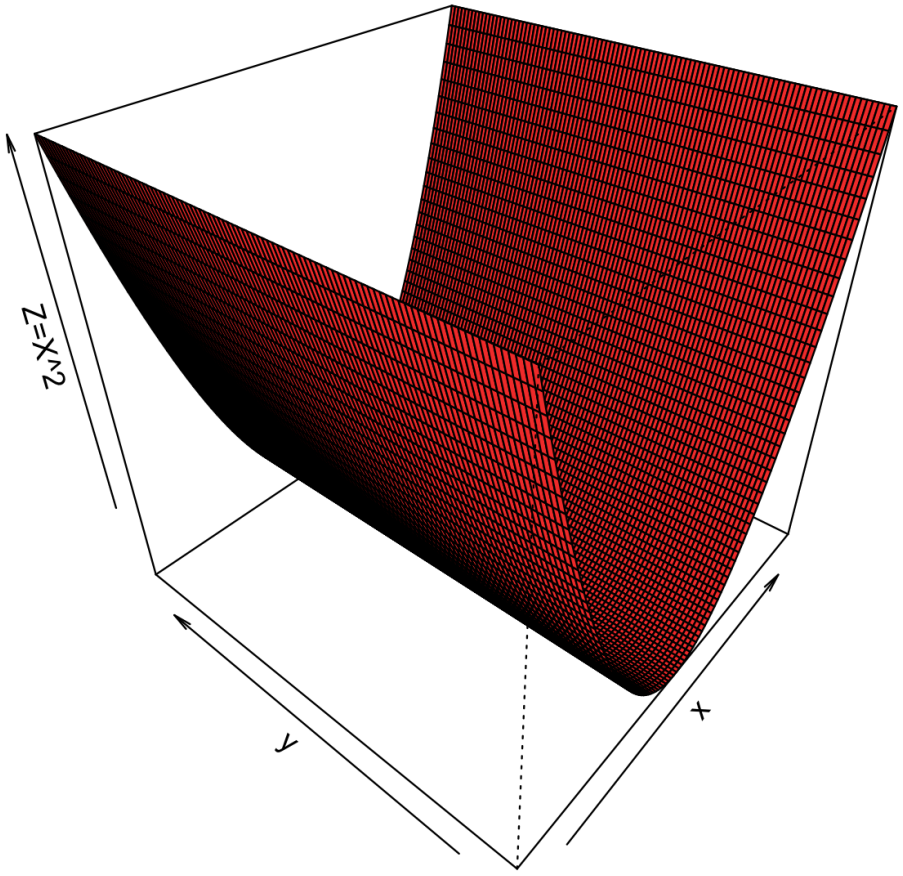
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
u = np.linspace(0, 2 * np.pi, 100)        #theta
t = np.linspace(-7, 7, 140)               #t
x = 2. * np.outer(np.cos(u), np.sqrt(1+t**2/4))
y = 1. * np.outer(np.sin(u), np.sqrt(1+t**2/4))
z = 1. * np.outer(np.ones(np.size(u)), t)
ax.plot_surface(y, x, z,  rstride=4, cstride=4, color='b')
#ax.plot_surface(x, y, z,  rstride=4, cstride=4, color='b')

ax.set_xlabel('y-axis')
ax.set_ylabel('x-axis')
ax.set_zlabel('z-axis')

plt.savefig('ex6.ps')
plt.show()
```



Ex1 in R

EXAMPLE 1 Sketch the graph of the surface $z = x^2$.

```
library(package = "TDA")
x=seq(-5, 5, length=100)
y=seq(-5, 5, length=100)
z=outer(x,y, function(x,y) {x^2})
persp(x,y,z, xlab='x', ylab='y', zlab="Z=X^2
theta=-50, phi=35, col='firebrick2')
dev.copy2pdf(file="ex1.pdf")
png("ex1.png")
```

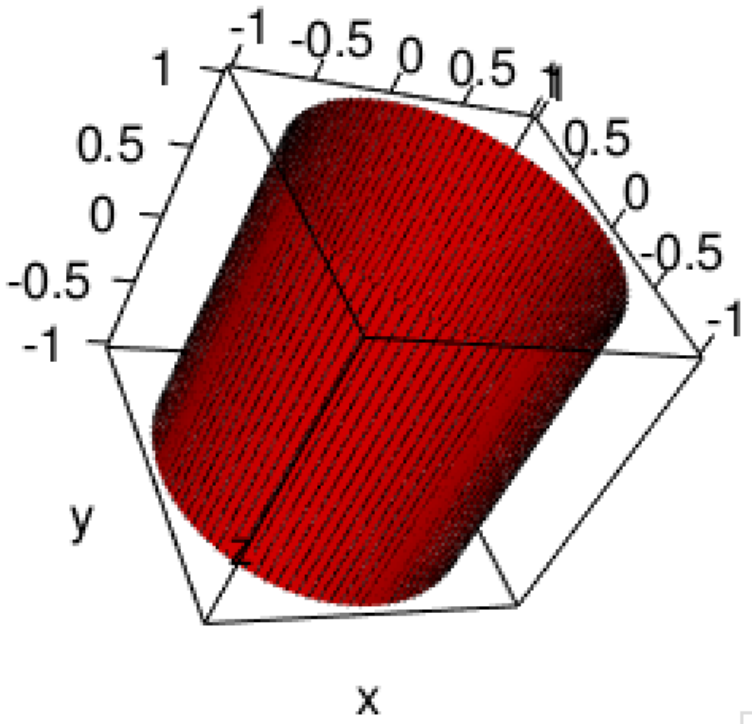


Ex2 in R

EXAMPLE 2 Identify and sketch the surfaces.

(a) $x^2 + y^2 = 1$ (b) $y^2 + z^2 = 1$

```
library(package = "TDA")
library("rgl")

u = seq(0, 2 * pi, length=100)        #theta
ones = array(1, dim=length(u))
t = seq(-1, 1, length=100)      #phi
ones_t = array(1, dim=length(v))
x = 1. * outer(cos(u), ones_t)   #outer:
y = 1. * outer(sin(u), ones_t)
z = 1. * outer(ones, t)

plot3d(x, y, z,
xlab="x",ylab="y",zlab="z", type="s", xlim=c(-1,1),
ylim=c(-1,1), size=0.25, aspect=FALSE)
surface3d(x,y,z, color='red')
dev.copy2pdf(file="ex2.pdf")
png("ex2.png")
```

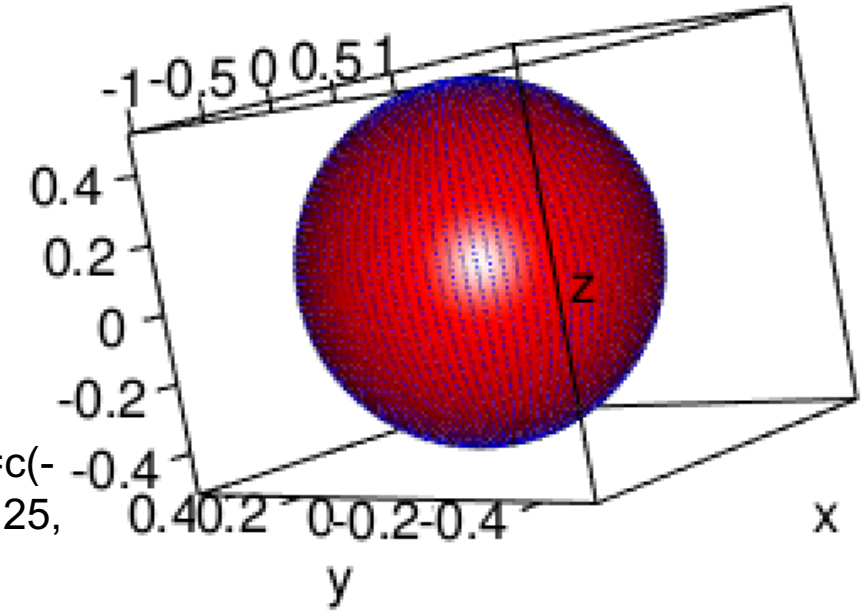


Ex3 in R

EXAMPLE 3 Use traces to sketch the quadric surface with equation

$$x^2 + \frac{y^2}{9} + \frac{z^2}{4} = 1$$

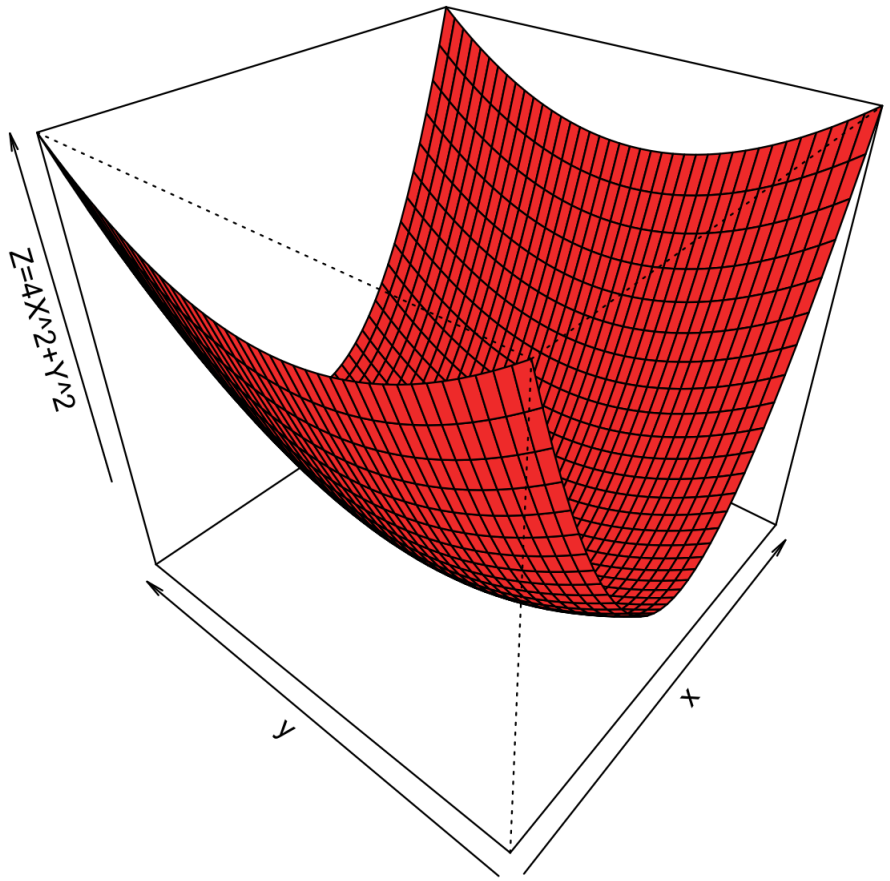
```
library(package = "TDA")
library("rgl")

u = seq(0, 2 * pi, length=100)       #theta
ones = array(1, dim=length(u))
v = seq(0, pi, length=100)           #phi
x = 1. * outer(cos(u), sin(v))   #outer:
y = 1./3. * outer(sin(u), sin(v))
z = 1./2. * outer(ones, cos(v))

plot3d(x, y, z,
xlab="x",ylab="y",zlab="z", type="s", xlim=c(-
1,1), ylim=c(-0.5,0.5), col = "blue", size=0.25,
aspect=FALSE)
surface3d(x,y,z, color='red')
dev.copy2pdf(file="ex3.pdf")
png("ex3.png")
```



Ex4 in R

EXAMPLE 4 Use traces to sketch the surface $z = 4x^2 + y^2$.

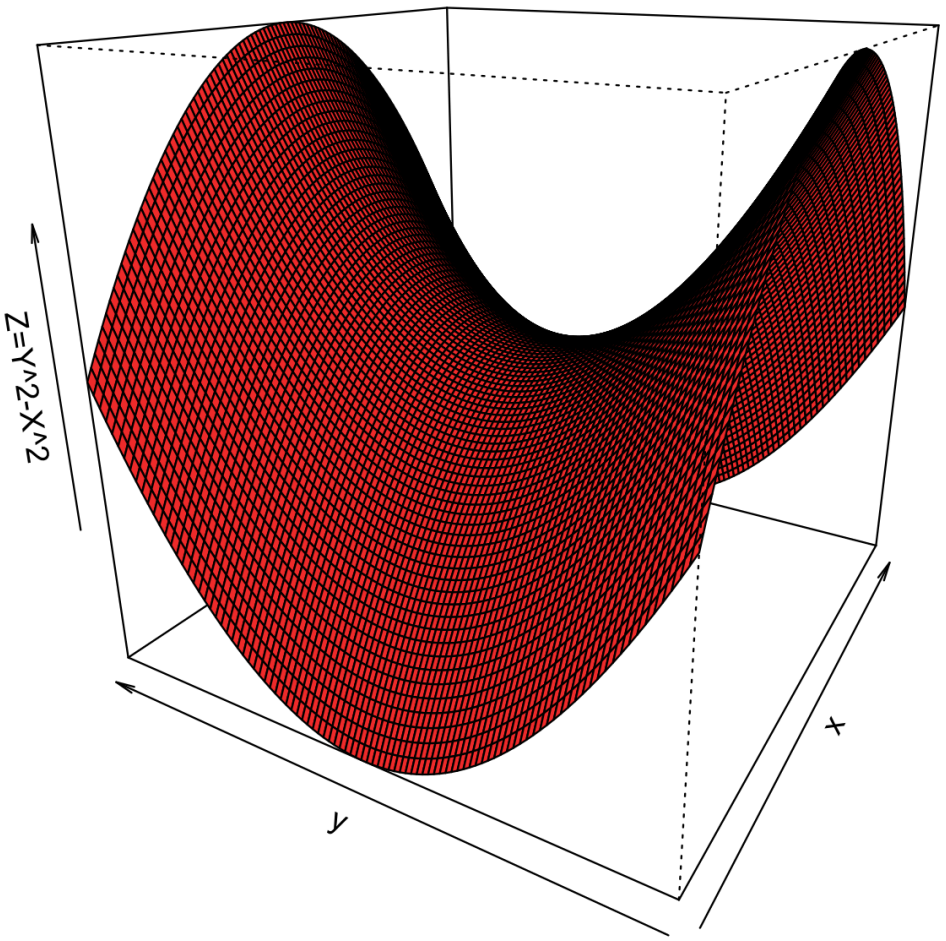
```
library(package = "TDA")
x=seq(-2, 2, length=40)
y=seq(-2, 2, length=40)
z=outer(x,y, function(x,y) {4*x^2+y^2})
persp(x,y,z, xlab='x', ylab='y',
zlab="Z=4X^2+Y^2", theta=-50, phi=35,
col='firebrick2')
dev.copy2pdf(file="ex4.pdf")
png("ex4.png")
```



Ex5 in R

EXAMPLE 5 Sketch the surface $z = y^2 - x^2$.

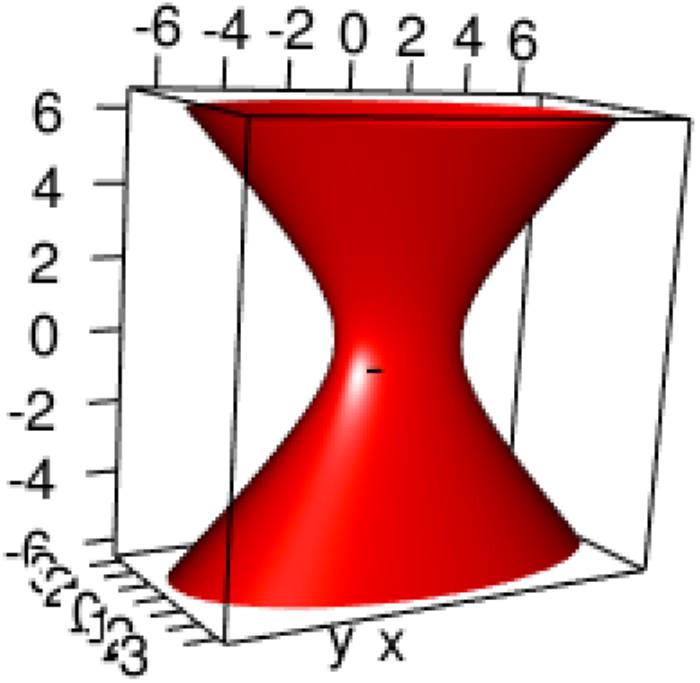
```
library(package = "TDA")
x=seq(-10, 10, length=100)
y=seq(-10, 10, length=100)
z=outer(x,y, function(x,y) {y^2-x^2})
persp(x,y,z, xlab='x', ylab='y',
zlab="Z=Y^2-X^2", theta=-60, phi=20,
col='firebrick2')
dev.copy2pdf(file="ex5.pdf")
png("ex5.png")
```



Ex6 in R

EXAMPLE 6 Sketch the surface $\frac{x^2}{4} + y^2 - \frac{z^2}{4} = 1$.

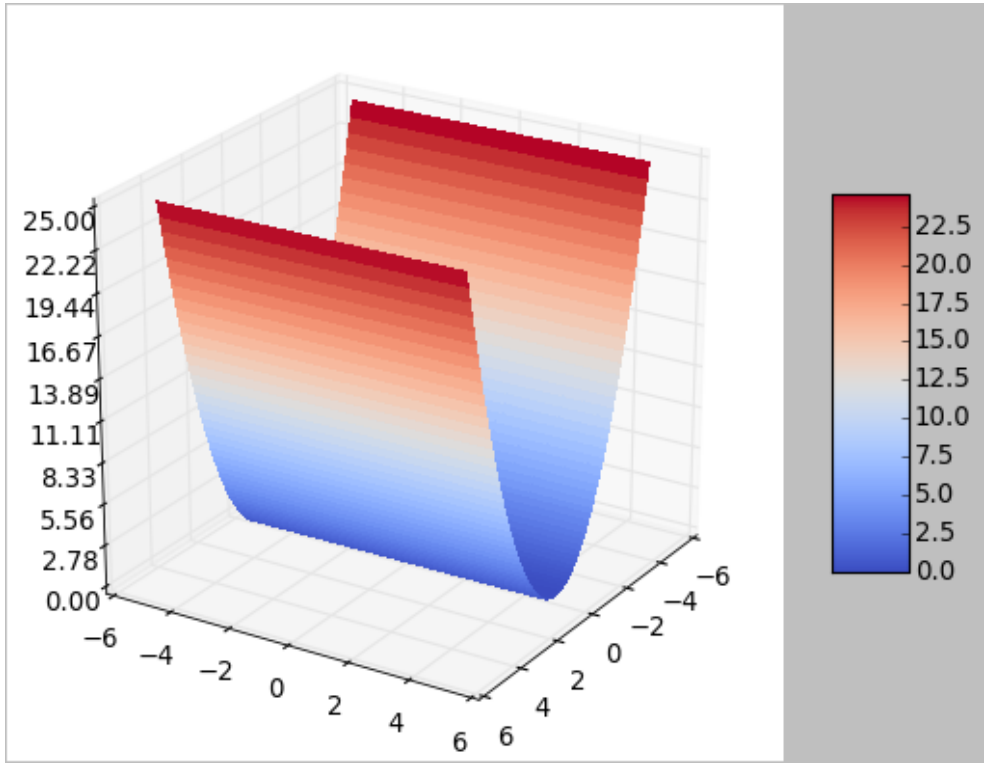
```
library(package = "TDA")
library("rgl")

u = seq(0, 2 * pi, length=100)       #theta
ones = array(1, dim=length(u))
t = seq(-2*pi, 2*pi, length=100)          #phi
x = 2. * outer(cos(u), sqrt(1+t^2/4))   #outer:
y = 1. * outer(sin(u), sqrt(1+t^2/4))
z = 1. * outer(ones, t)

plot3d(x, y, z,
xlab="x",ylab="y",zlab="z", type="s", xlim=c(-
1,1), ylim=c(-0.5,0.5), col = "blue", size=0.25,
aspect=FALSE)
surface3d(x,y,z, color='red')
dev.copy2pdf(file="ex6.pdf")
png("ex6.png")
```



Ex1 in Julia

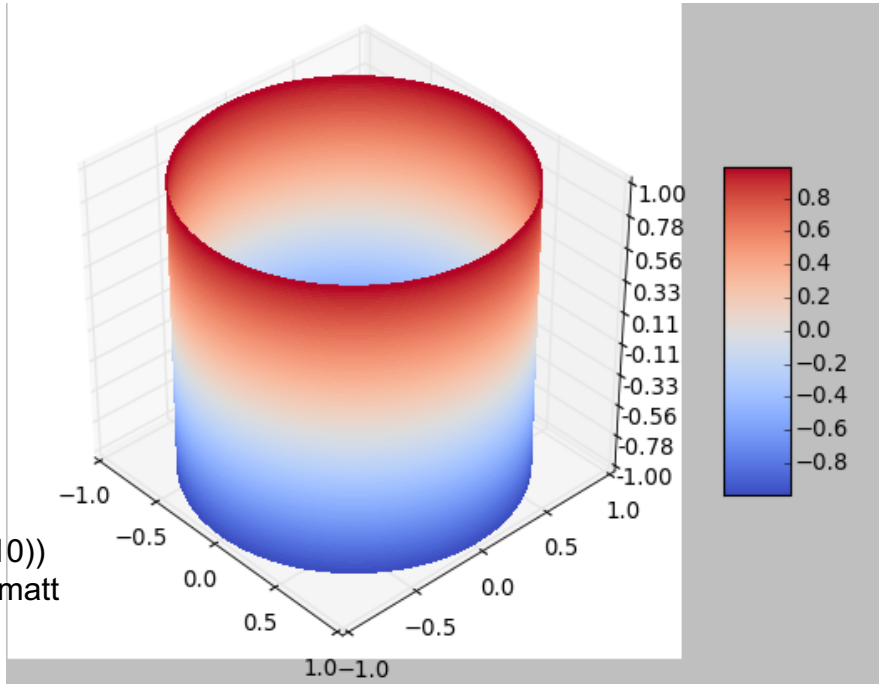
EXAMPLE 1 Sketch the graph of the surface $z = x^2$.

```
using PyPlot
using QuantEcon

fig = figure()
x = linspace(-5, 5, 100)
y = linspace(-5, 5, 100)
X, Y = meshgrid(collect(x), collect(y))
Z = X.^2
#surf = plot_surface(X, Y, Z, rstride=1, cstride=1,
linewidth=0, antialiased=false,
cmap="coolwarm")
surf = plot_surface(x, y, Z, rstride=1, cstride=1,
linewidth=0, antialiased=false,
cmap="coolwarm")
#zlim(-1.01,1.01)
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:
LinearLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker]
[:FormatStrFormatter]("%.02f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex1.pdf")
```



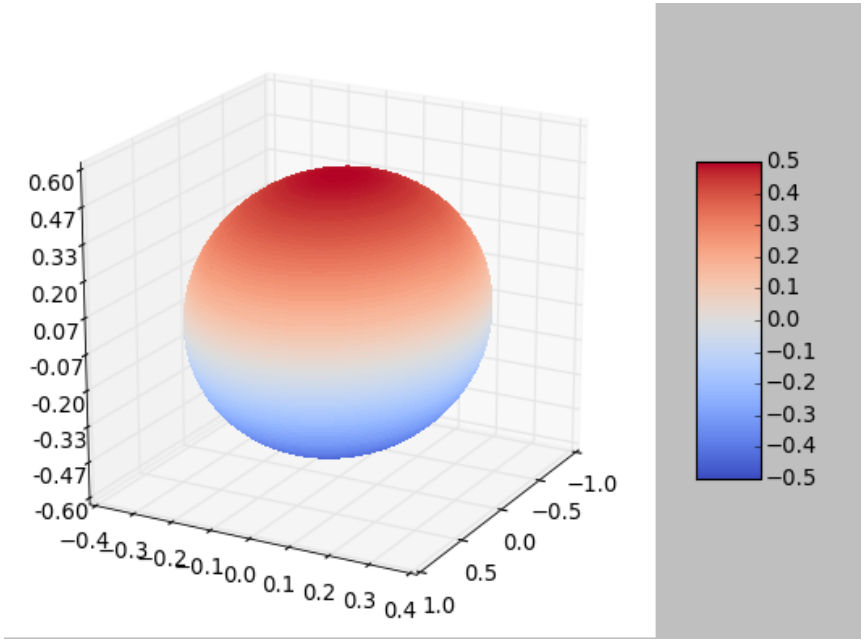
Ex2 in Julia

```
using PyPlot
using QuantEcon
fig = figure()
n=100
u = linspace(0, 2pi, n)       #theta
v = linspace(-1, 1, n)

x = Array(Float64, n, n)
y = Array(Float64, n, n)
z = Array(Float64, n, n)
xf(xx, yy) = cos(xx)
yf(xx, yy) = sin(xx)
zf(xx, yy) = yy
for i in 1:n
    for j in 1:n
        x[j, i] = xf(u[i], v[j])
        y[j, i] = yf(u[i], v[j])
        z[j,i]  = zf(u[i], v[j])
    end
end
surf = plot_surface(x, y, z, rstride=1, cstride=1, linewidth=0,
antialiased=false, cmap="coolwarm")
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:LinearLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker][:FormatStrFormatt
er]("%.02f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex2.pdf")
```



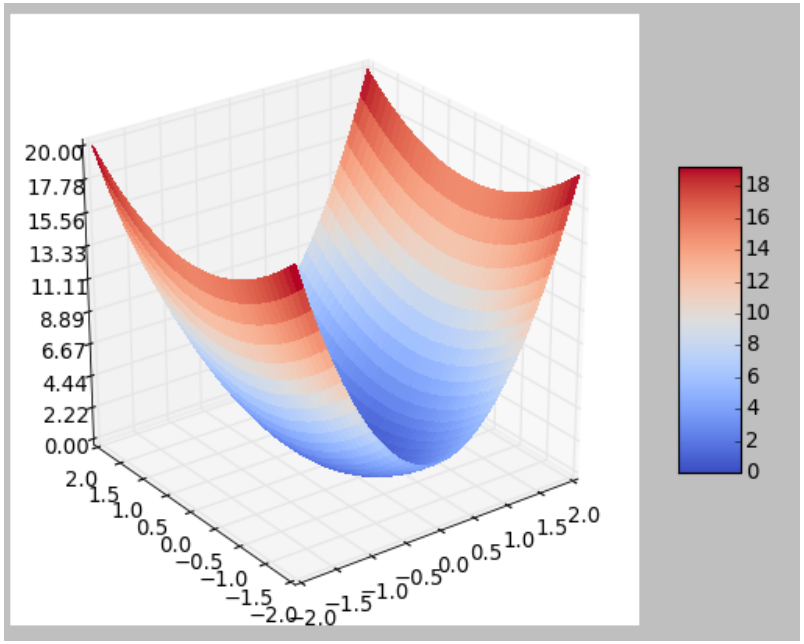
Ex3 in Julia

```
using PyPlot
using QuantEcon

fig = figure()
n=100
u = linspace(0, 2pi, n)       #theta
v = linspace(0, pi, n)          #phi

x = Array(Float64, n, n)
y = Array(Float64, n, n)
z = Array(Float64, n, n)
xf(xx, yy) = cos(xx)*sin(yy)
yf(xx, yy) = 1.0/3.0*sin(xx)*sin(yy)
zf(xx, yy) = 1.0/2.0*cos(yy)
for i in 1:n
    for j in 1:n
        x[j, i] = xf(u[i], v[j])
        y[j, i] = yf(u[i], v[j])
        z[j,i]  = zf(u[i], v[j])
    end
end

surf = plot_surface(x, y, z, rstride=1, cstride=1, linewidth=0,
antialiased=false, cmap="coolwarm")
#zlim(-1.01,1.01)
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:LinearLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker][:FormatStrFormatter]("%.0
2f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex3.pdf")
```



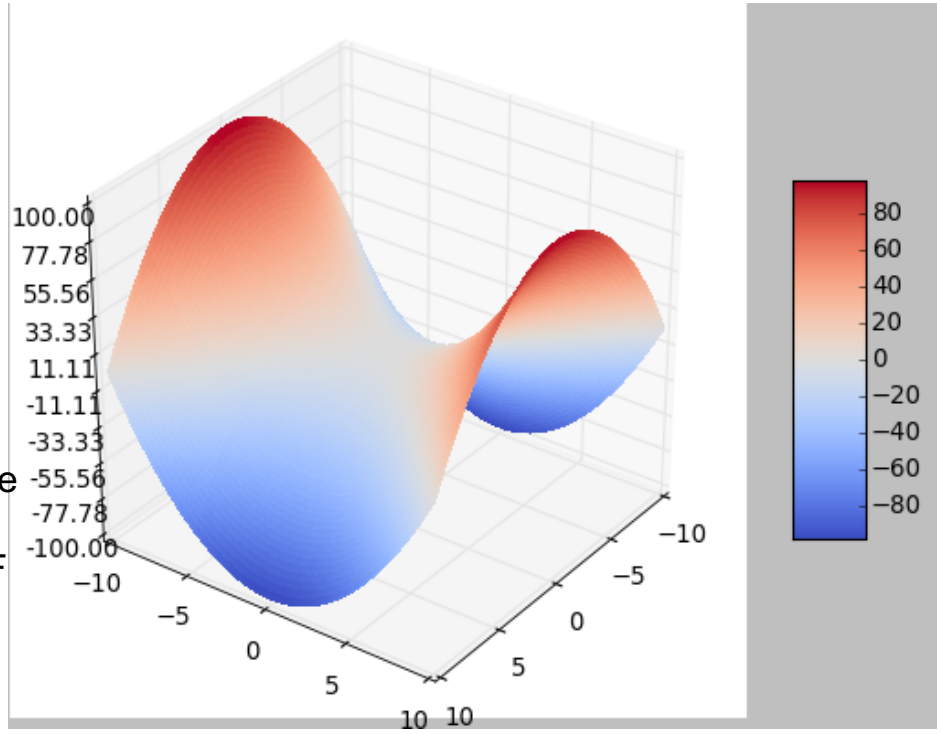
Ex4 in Julia

```
using PyPlot
#using QuantEcon: meshgrid
using QuantEcon

fig = figure()
#x = linspace(-5, 5, 100)'
n = 40
x = linspace(-2, 2, n)
y = linspace(-2, 2, n)
X, Y = meshgrid(collect(x), collect(y))
Z = 4 * X.^2 + Y.^2
surf = plot_surface(x, y, Z, rstride=1, cstride=1,
linewidth=0, antialiased=false, cmap="coolwarm")
#zlim(-1.01,1.01)
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:Line
arLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker][:F
ormatStrFormatter]("%.02f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex4.pdf")
```



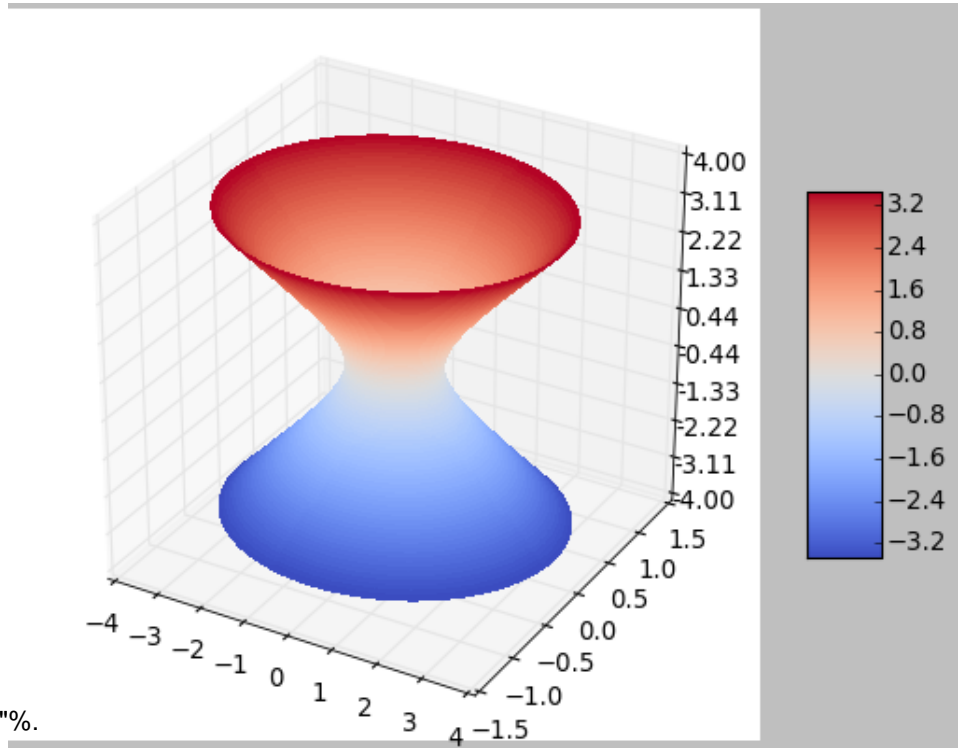
Ex5 in Julia

```
using PyPlot
#using QuantEcon: meshgrid
using QuantEcon

fig = figure()
#x = linspace(-5, 5, 100)'
n = 100
x = linspace(-10, 10, n)
y = linspace(-10, 10, n)
X, Y = meshgrid(collect(x), collect(y))
Z = Y.^2 - X.^2
surf = plot_surface(x, y, Z, rstride=1, cstride=1,
linewidth=0, antialiased=false, cmap="coolwarm")
#zlim(-1.01,1.01)
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:Line
arLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker][:F
ormatStrFormatter]("%.02f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex5.pdf")
```



Ex6 in Julia

```
using PyPlot
using QuantEcon

fig = figure()
n=100
u = linspace(0, 2pi, n)       #theta
#v = linspace(0, pi, n)          #phi
v = linspace(-7, 7, n)

x = Array(Float64, n, n)
y = Array(Float64, n, n)
z = Array(Float64, n, n)
xf(xx, yy) = cos(xx)*sqrt(1+yy^2/4)
yf(xx, yy) = 1.0/3.0*sin(xx)*sqrt(1+yy^2/4)
zf(xx, yy) = 1.0/2.0*yy
for i in 1:n
   for j in 1:n
      x[j, i] = xf(u[i], v[j])
      y[j, i] = yf(u[i], v[j])
      z[j,i]  = zf(u[i], v[j])
   end
end

surf = plot_surface(x, y, z, rstride=1, cstride=1, linewidth=0,
antialiased=false, cmap="coolwarm")
#zlim(-1.01,1.01)
ax = gca()
ax[:zaxis][:set_major_locator](matplotlib[:ticker][:LinearLocator](10))
ax[:zaxis][:set_major_formatter](matplotlib[:ticker][:FormatStrFormatter]("%.
02f"))
fig[:colorbar](surf, shrink=0.5, aspect=5)

savefig("ex6.pdf")
```



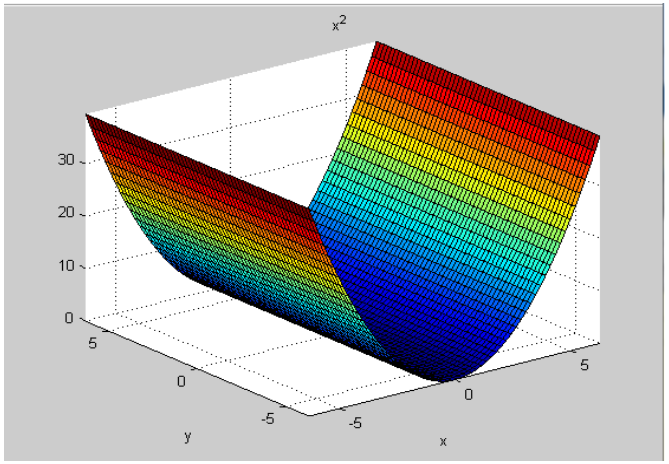
Performance

	Fortran gcc 5.1.1	Julia 0.4.0	Python 3.4.3	R 3.2.2	Matlab R2015b	Octave 4.0.0	Mathe-matica 10.2.0	JavaScript V8 3.28.71.19	Go go1.5	LuaJIT gsl-shell 2.3.1	Java 1.8.0_45
fib	0.70	2.11	77.76	533.52	26.89	9324.35	118.53	3.36	1.86	1.71	1.21
parse_int	5.05	1.45	17.02	45.73	802.52	9581.44	15.02	6.06	1.20	5.77	3.35
quicksort	1.31	1.15	32.89	264.54	4.92	1866.01	43.23	2.70	1.29	2.03	2.60
mandel	0.81	0.79	15.32	53.16	7.58	451.81	5.13	0.66	1.11	0.67	1.35
pi_sum	1.00	1.00	21.99	9.56	1.00	299.31	1.69	1.01	1.00	1.00	1.00
rand_mat_stat	1.45	1.66	17.93	14.56	14.52	30.93	5.95	2.30	2.96	3.27	3.92
rand_mat_mul	3.48	1.02	1.14	1.57	1.12	1.12	1.30	15.07	1.42	1.16	2.36

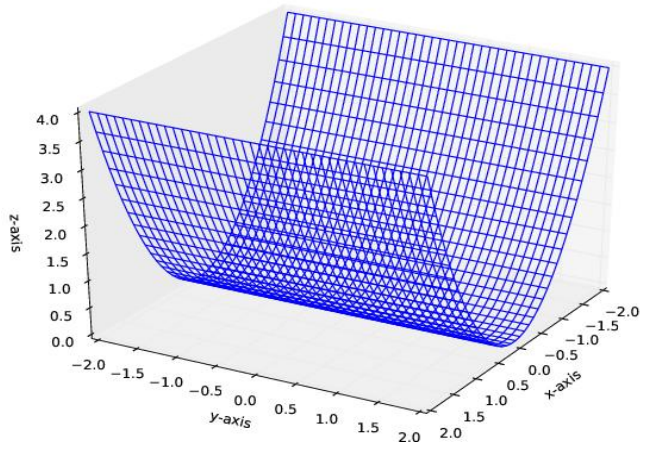
Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

EX-1
Section 12.6: Surface plots

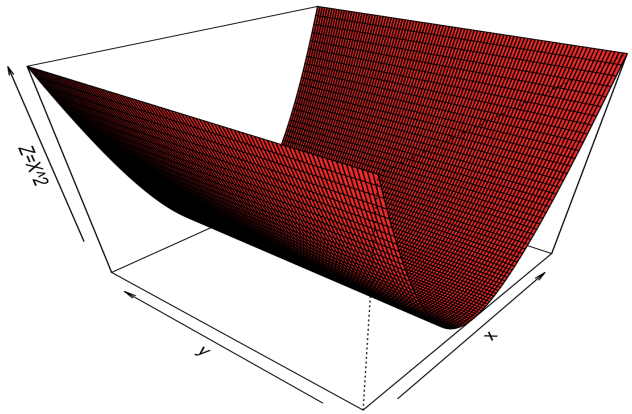
Matlab



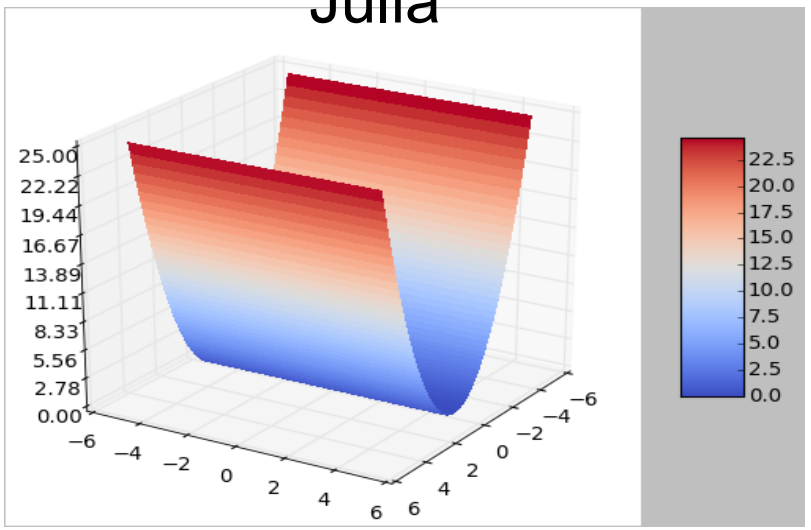
Python



R

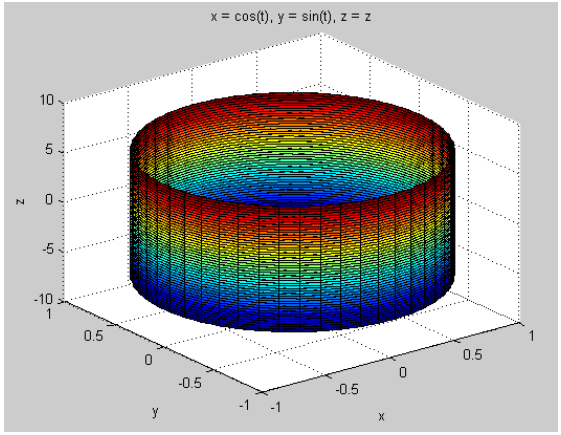


Julia

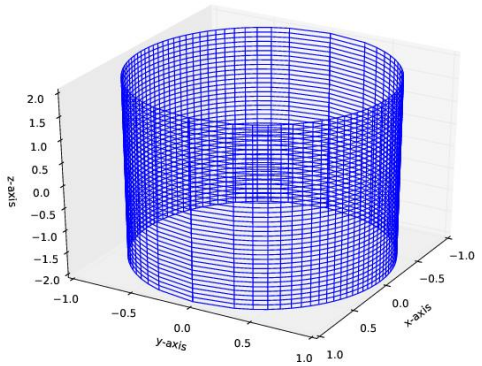


EX-2
Section 12.6: Surface plots

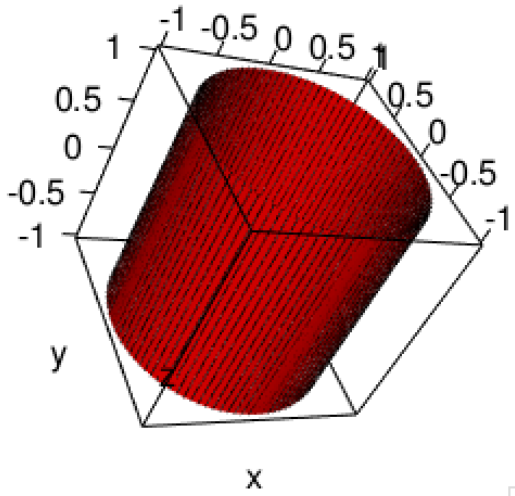
Matlab



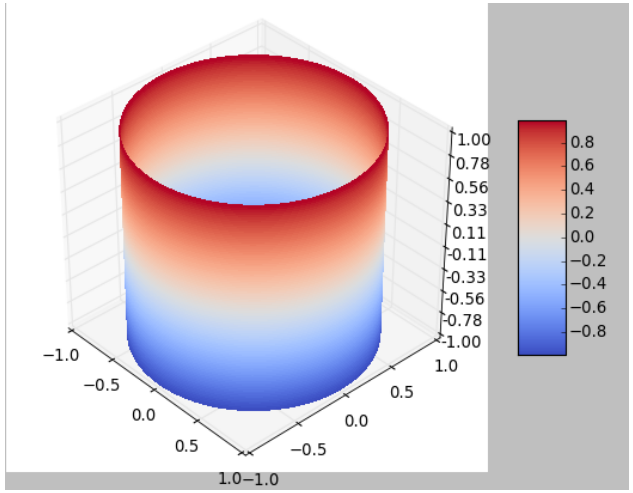
Python



R

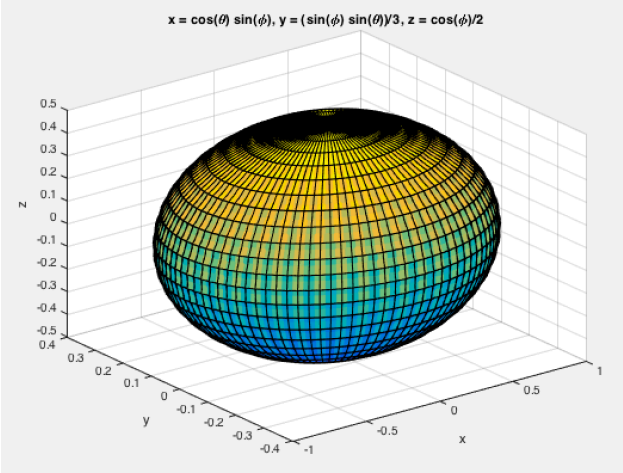


Julia

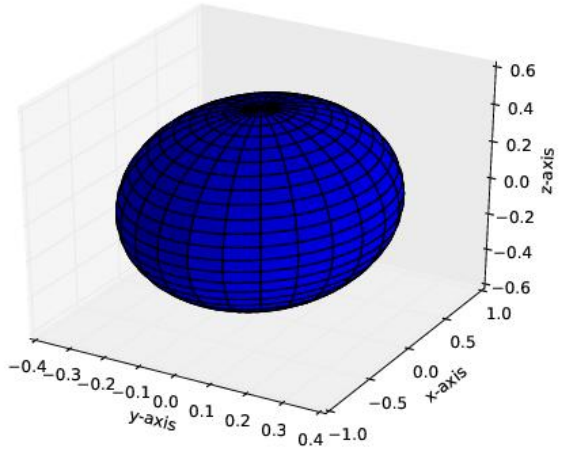


EX-3
Section 12.6: Surface plots

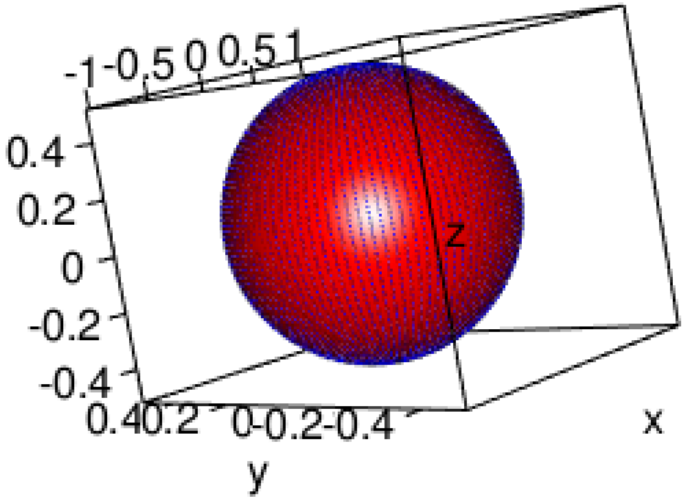
Matlab



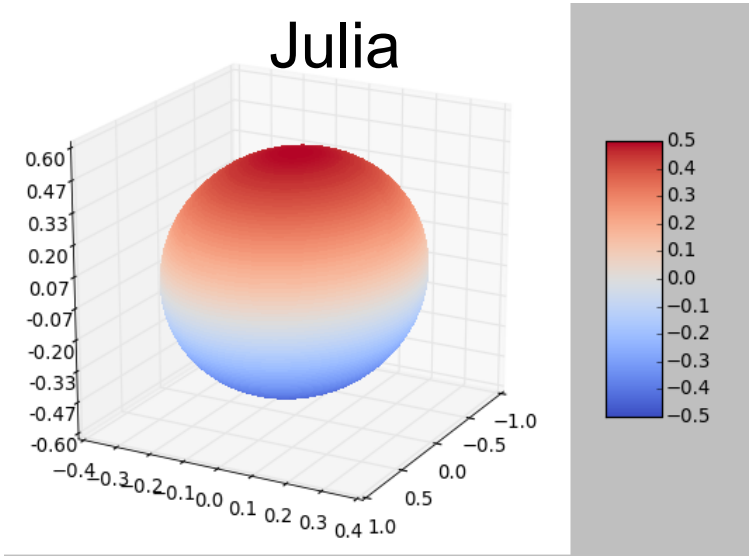
Python



R

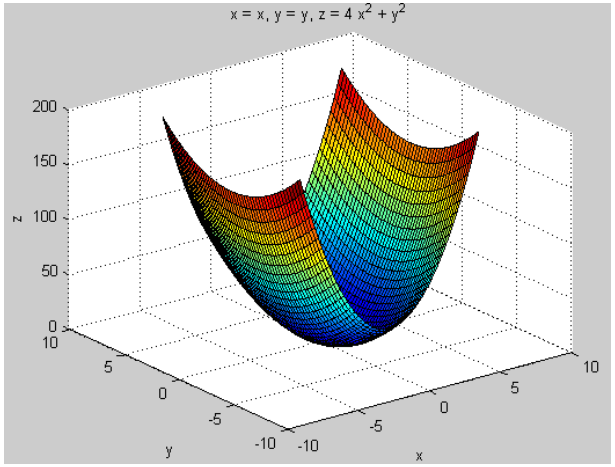


Julia

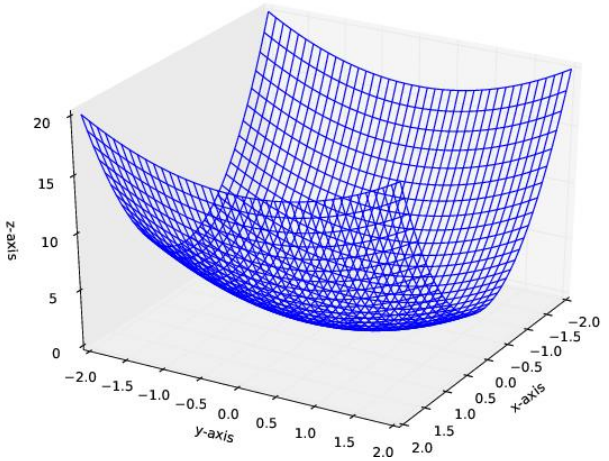


EX-4
Section 12.6: Surface plots

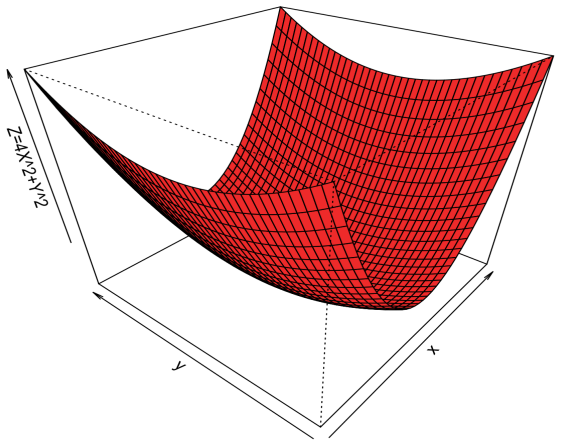
Matlab



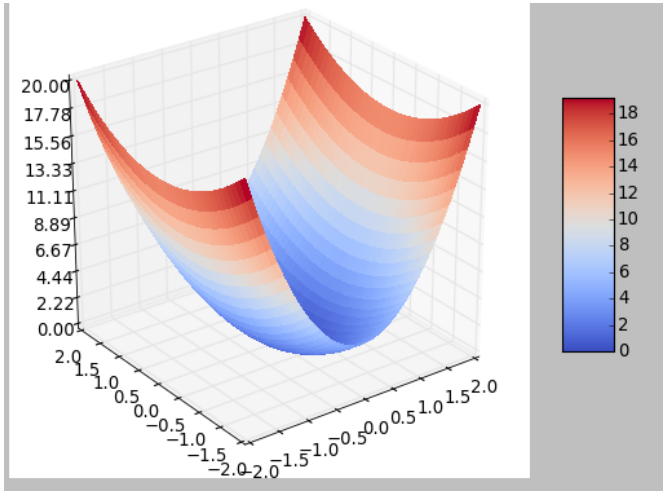
Python



R

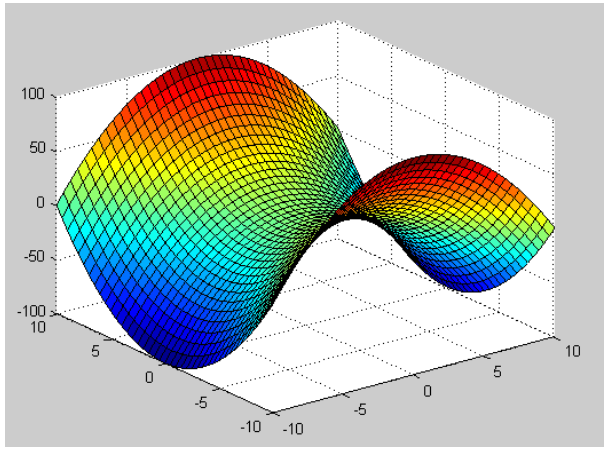


Julia

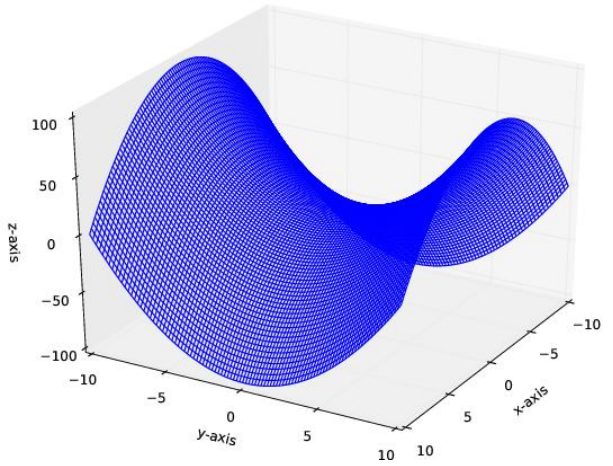


EX-5
Section 12.6: Surface plots

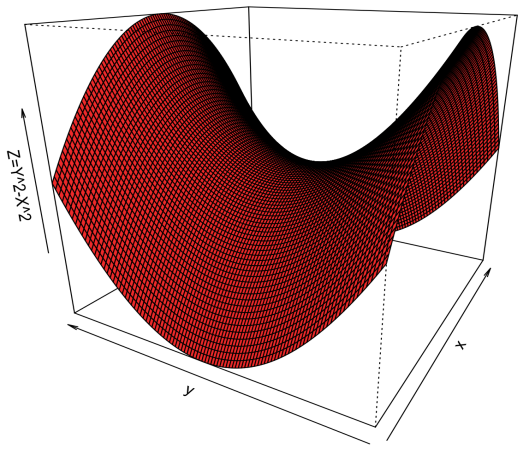
Matlab



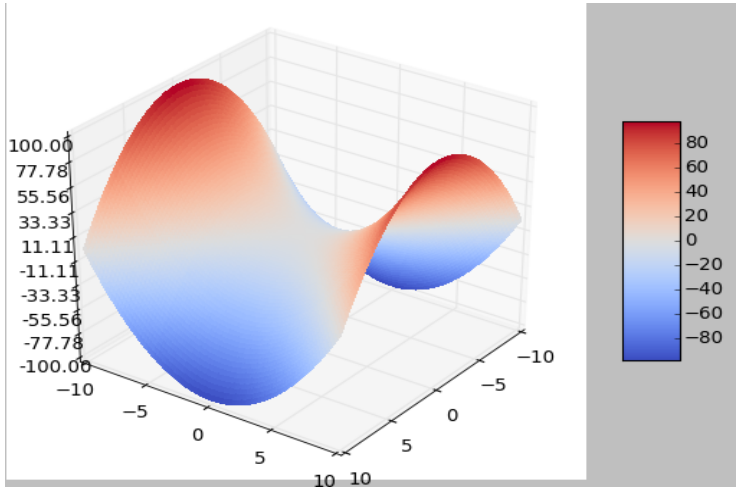
Python



R

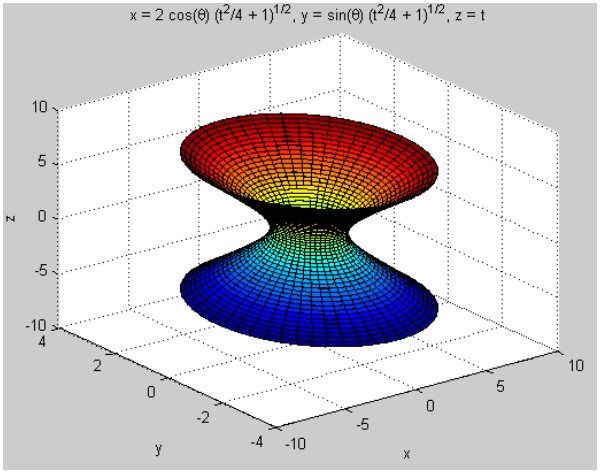


Julia

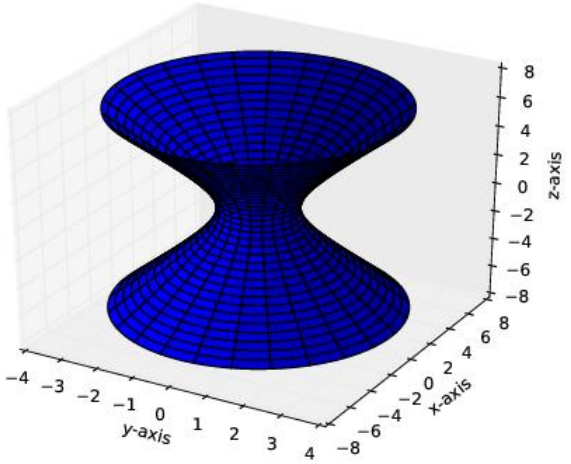


EX-6
Section 12.6: Surface plots

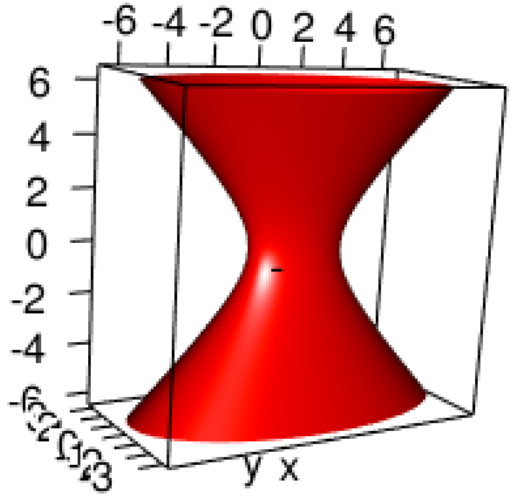
Matlab



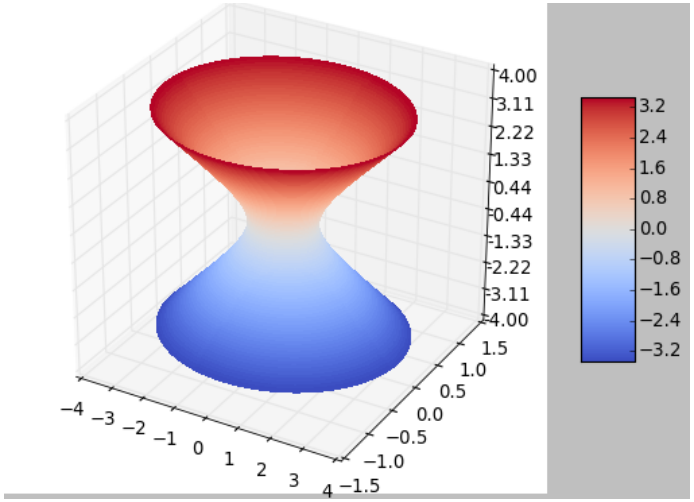
Python



R



Julia



Performance

	Fortran gcc 5.1.1	Julia 0.4.0	Python 3.4.3	R 3.2.2	Matlab R2015b	Octave 4.0.0	Mathe-matica 10.2.0	JavaScript V8 3.28.71.19	Go go1.5	LuaJIT gsl-shell 2.3.1	Java 1.8.0_45
fib	0.70	2.11	77.76	533.52	26.89	9324.35	118.53	3.36	1.86	1.71	1.21
parse_int	5.05	1.45	17.02	45.73	802.52	9581.44	15.02	6.06	1.20	5.77	3.35
quicksort	1.31	1.15	32.89	264.54	4.92	1866.01	43.23	2.70	1.29	2.03	2.60
mandel	0.81	0.79	15.32	53.16	7.58	451.81	5.13	0.66	1.11	0.67	1.35
pi_sum	1.00	1.00	21.99	9.56	1.00	299.31	1.69	1.01	1.00	1.00	1.00
rand_mat_stat	1.45	1.66	17.93	14.56	14.52	30.93	5.95	2.30	2.96	3.27	3.92
rand_mat_mul	3.48	1.02	1.14	1.57	1.12	1.12	1.30	15.07	1.42	1.16	2.36

Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

Midterm Q4

```
syms x y t b  r T  N  k dr dT real
assume (a>0)
assume (0<= t <= 2*pi)

x=cos(t)*a
y=sin(t)*a
% parameterization function is x=cos(t)*b y=sin(t)*b

r=[x, y]
dr=diff(r, t) % v(t)
d2r=diff(dr, t) % a(t)

dot1 = dot(r,dr) %value is 0, r and v(t) is orthogonal

dot2 = dot(r,d2r) %value is 0, r and a(t) is orthogonal

T=dr/norm(dr) % unit tangent vector

Tsim=simplify(T) % simplify the expression
dT=diff(Tsim,t) % dT/dt

k=norm(dT)/norm(dr) %curvature
pretty(simplify(k))
```

Problem 4

(a1) v(t):$\begin{pmatrix} -a\sin(t) & a\cos(t) \end{pmatrix}$

(a2) a(t):$\begin{pmatrix} -a\cos(t) & -a\sin(t) \end{pmatrix}$

(b) dot(r,v(t))=0 so r and v(t) are orthogonal

(c) dot(r,a(t))=0 so r and a(t) are orthogonal

(d) Unit tangent vector:$\begin{pmatrix} -\sin(t) & \cos(t) \end{pmatrix}$

(e) Curvature: $\frac{1}{a}$