2014-04-21.sagews

April 21, 2014

Contents

1 Math 480b Sage Course **1**
1.1 Drawing Plots using Sage, Matplotlib, R, etc. 1
1.2 April 21, 2014 1
1.3 2d Sage Graphics (like Mathematica, but better) 1
1.4 2d Graphics using Matplotlib (like Matlab, but better) 5
1.5 3d Sage Graphics (like Mathematica) 7
1.6 Other You can also draw plots using R 8
1.7 Or even plot with octave 9

1 Math 480b Sage Course

1.1 Drawing Plots using Sage, Matplotlib, R, etc.

1.2 April 21, 2014

Screencast: REMEMBER!!!!!

Plan

- Questions?
- Homework 4; grading of homework 3 available for you to do; peer grading of homework 2 returned.
- 2D (and 3D) graphics in Sage

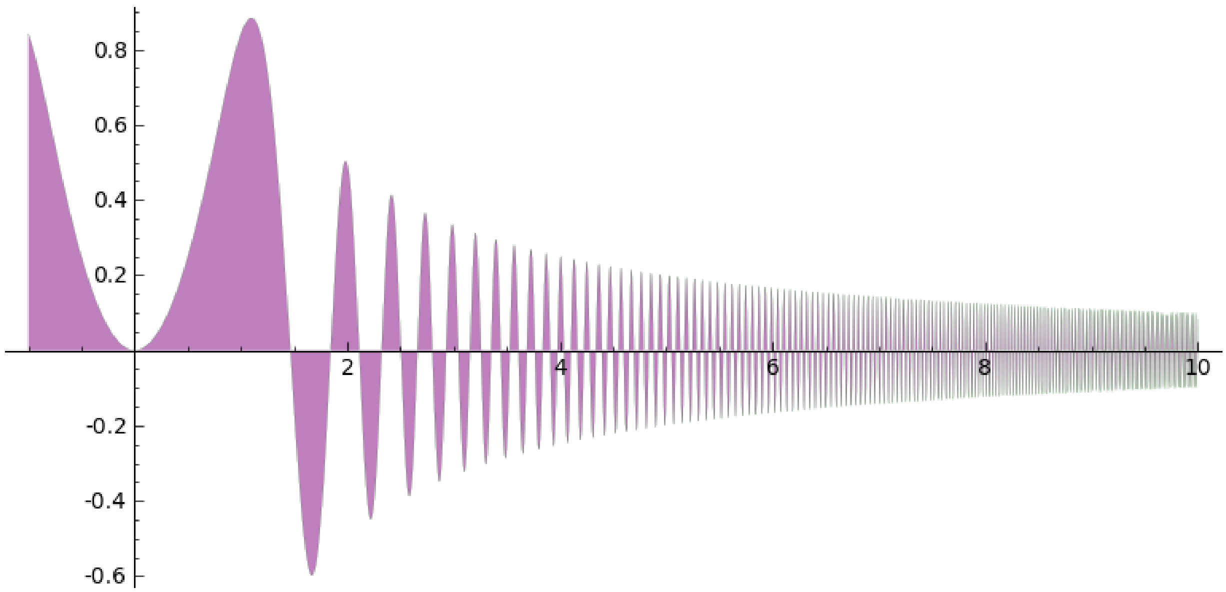
1.3 2d Sage Graphics (like Mathematica, but better)

- sequence of line segments
- a function
- interlude: combining plot objects using +
- points
- polygons, ellipses, etc.
- arrows
- contour plot

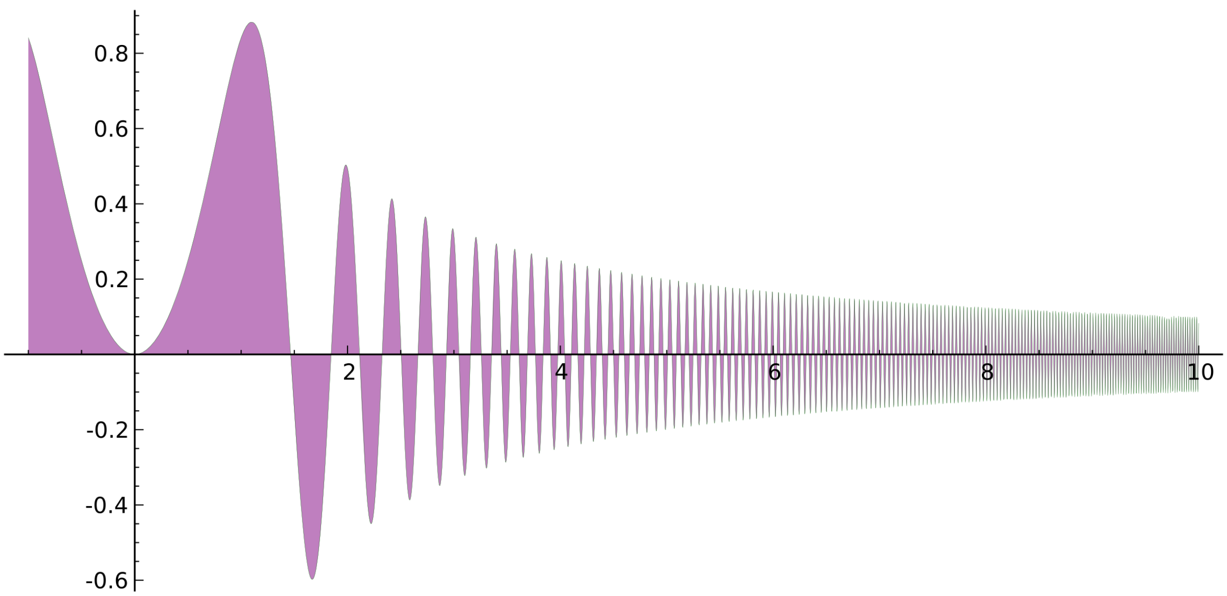
- implicit plot
- saving plots as pdf (e.g,. to include in a LaTeX document)

```
@interact
def f(c=Color('green')):
    show(plot(sin, (-1, 10), color=c))
```

```
g = plot(sin(x^3)/x, (-1, 10), color='darkgreen', thickness=.1, fillcolor\
   ='purple', fill=True)
g
```

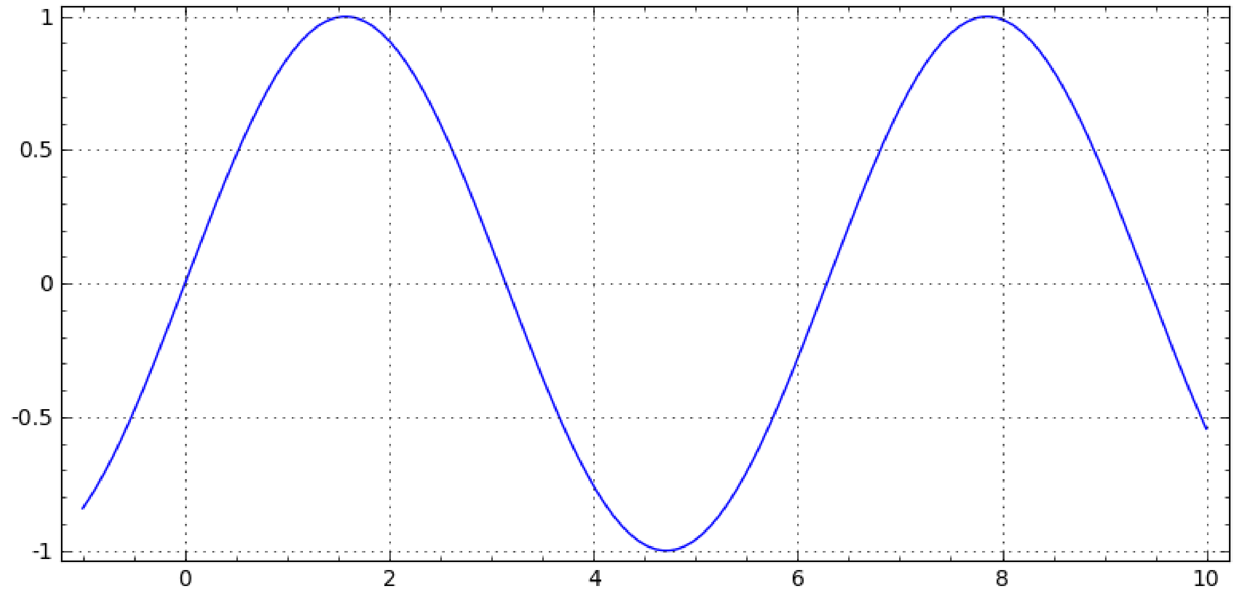



```
show(g, svg=True)
```



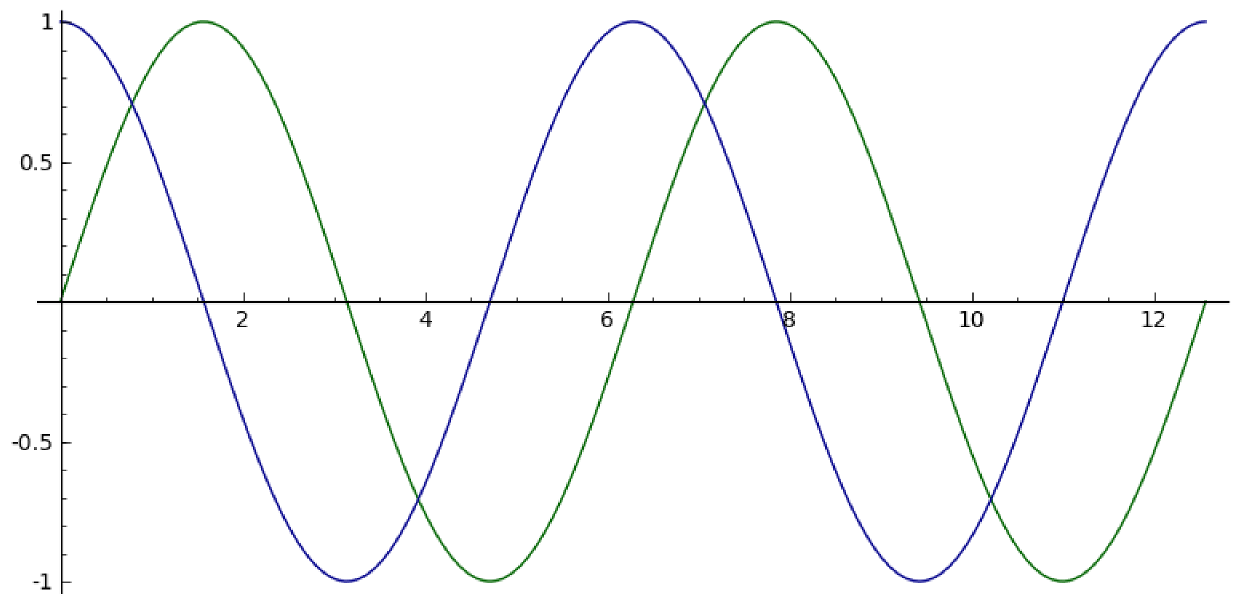
```
g.save('a.pdf')
```

```
plot(sin, (-1, 10), frame=True, axes=False, gridlines=True)
```




```
g = plot(sin(x), (x, 0, 4*pi), color='darkgreen')
h = plot(cos(x), (x, 0, 4*pi), color='darkblue')
```

```
g + h  # salvus.file(...) behind the scenes.
```

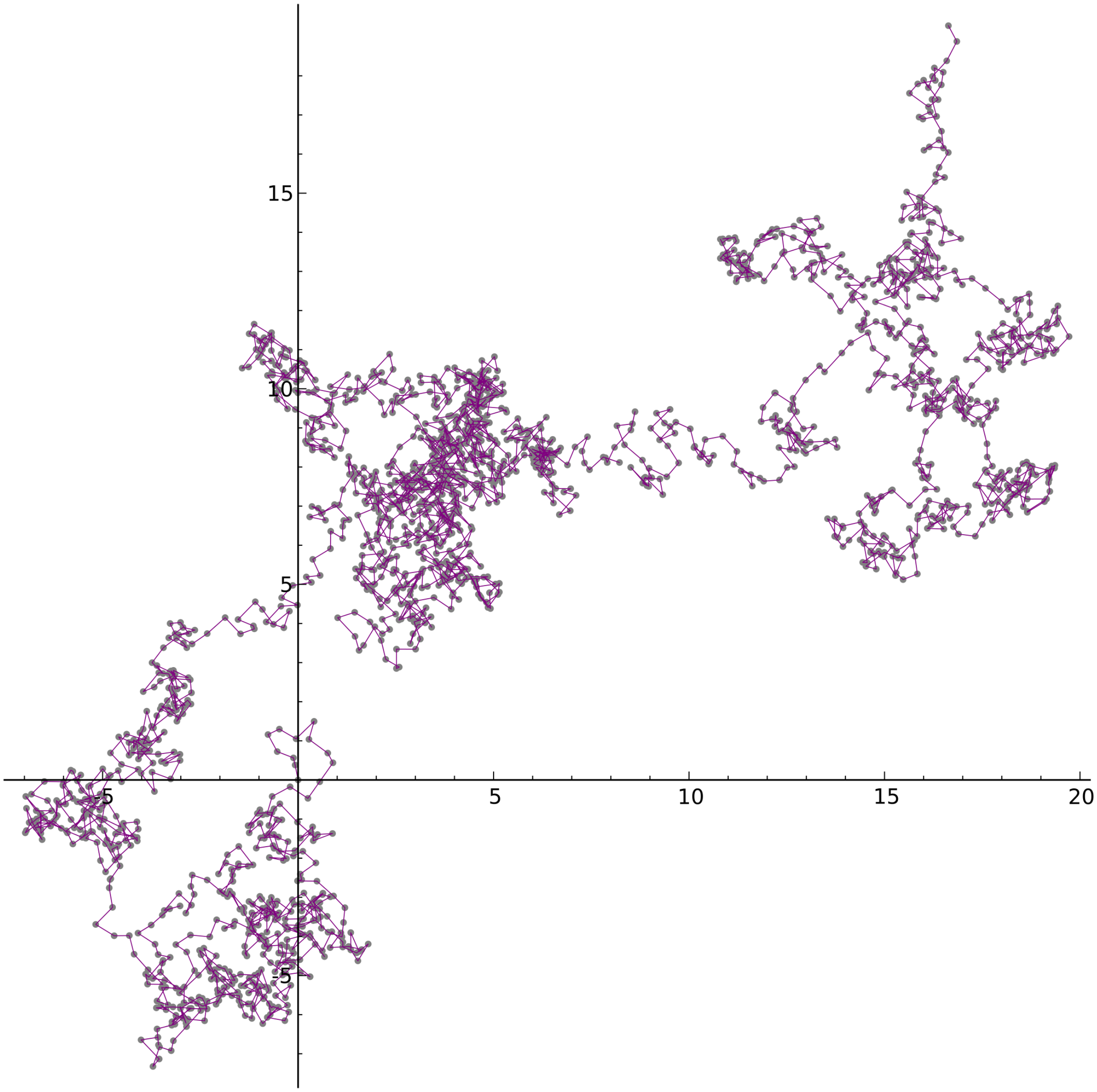



```
random()
```

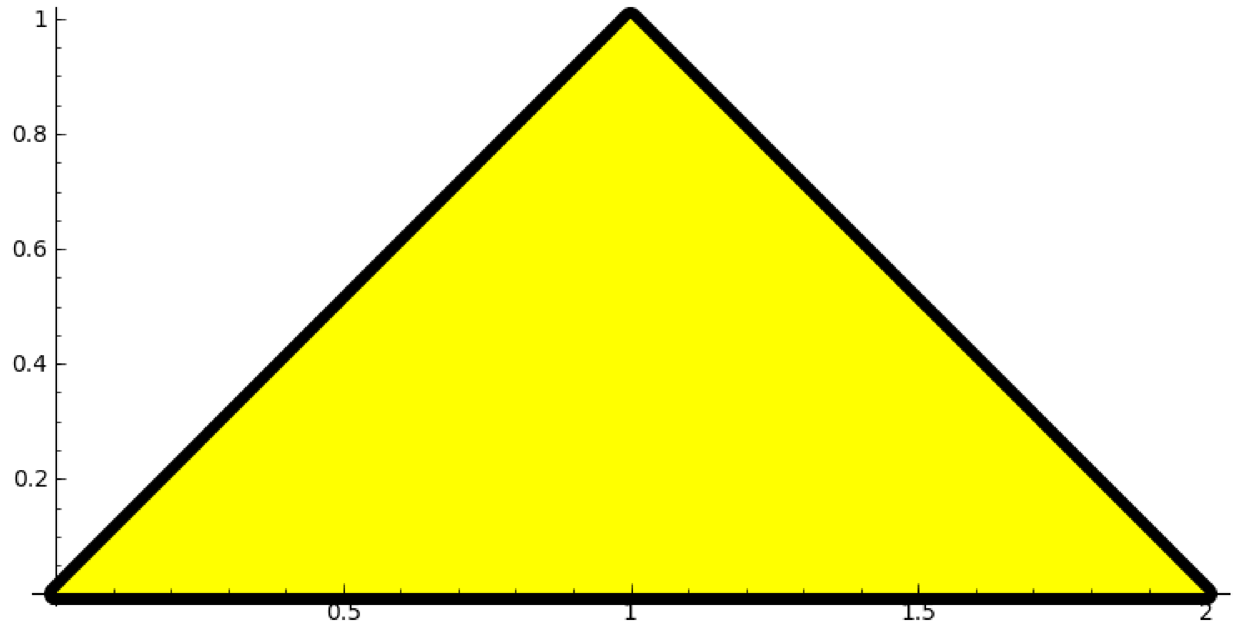
0.8181731957981786

```
v = [(0,0)]
```

```
for i in range(2000):
    last = v[-1]
    v.append((last[0]+(random()-.5), last[1]+(random()-.5)))

g = line(v, thickness=.4, color='purple')
g += points(v, pointsize=10, color='grey')
g.show(aspect_ratio=1, svg=True, figsize=10)
```




```
a = polygon2d([(0,0), (1,1), (2,0)], fill=False, color='black', thickness\
   =10)
b = polygon2d([(0,0), (1,1), (2,0)], color='yellow')
g = a+b; g
```




```
g.rotate  # project idea; also g.copy()
```

```
v = [(0,0)]
for i in range(5):
    last = v[-1]
    v.append((last[0]+(random()-.5), last[1]+(random()-.5)))

plots = []
for i in range(1,5):
    g = line(v[:i], thickness=.4, color='purple')
    g += points(v[:i], pointsize=10, color='grey', aspect_ratio=1)
    plots.append(g)
```

```
len(plots)
```

4

```
animate(plots)
```

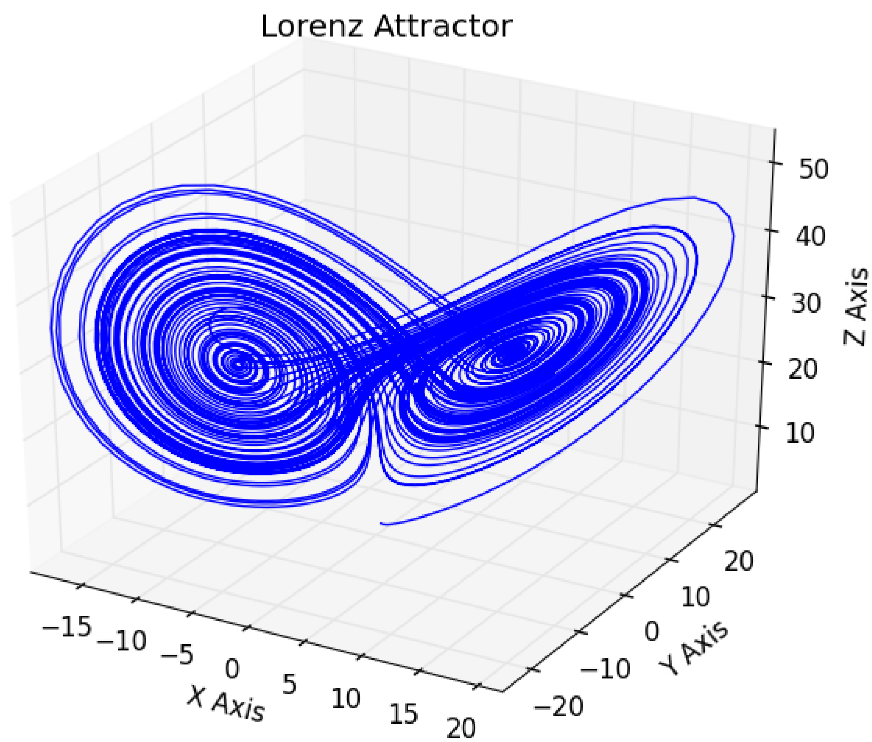
https://cloud.sagemath.com/blobs/tmp_4wrqiA.gif?uuid=a8fbe218-a5d9-45b7-9ce2-89113c19293e

1.4 2d Graphics using Matplotlib (like Matlab, but better)

- matplotlib is an easy-to-install standard Python plotting library (which Sage uses extensively).
- examples/docs at the gallery: `http://matplotlib.org/gallery.html`
- how to get them to appear in SageMathCloud worksheets

```
# from http://matplotlib.org/examples/mplot3d/lorenz_attractor.html

import numpy as np
```

```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def lorenz(x, y, z, s=10, r=28, b=2.667) :
    x_dot = s*(y - x)
    y_dot = r*x - y - x*z
    z_dot = x*y - b*z
    return x_dot, y_dot, z_dot


dt = 0.01
stepCnt = 10000

# Need one more for the initial values
xs = np.empty((stepCnt + 1,))
ys = np.empty((stepCnt + 1,))
zs = np.empty((stepCnt + 1,))

# Setting initial values
xs[0], ys[0], zs[0] = (0., 1., 1.05)

# Stepping through "time".
for i in range(stepCnt) :
    # Derivatives of the X, Y, Z state
    x_dot, y_dot, z_dot = lorenz(xs[i], ys[i], zs[i])
    xs[i + 1] = xs[i] + (x_dot * dt)
    ys[i + 1] = ys[i] + (y_dot * dt)
    zs[i + 1] = zs[i] + (z_dot * dt)

fig = plt.figure()
ax = fig.gca(projection='3d')

_ = ax.plot(xs, ys, zs)
_ = ax.set_xlabel("X Axis"), ax.set_ylabel("Y Axis"), ax.set_zlabel("Z \
   Axis"), ax.set_title("Lorenz Attractor")

plt.show()
```



1.5 3d Sage Graphics (like Mathematica)

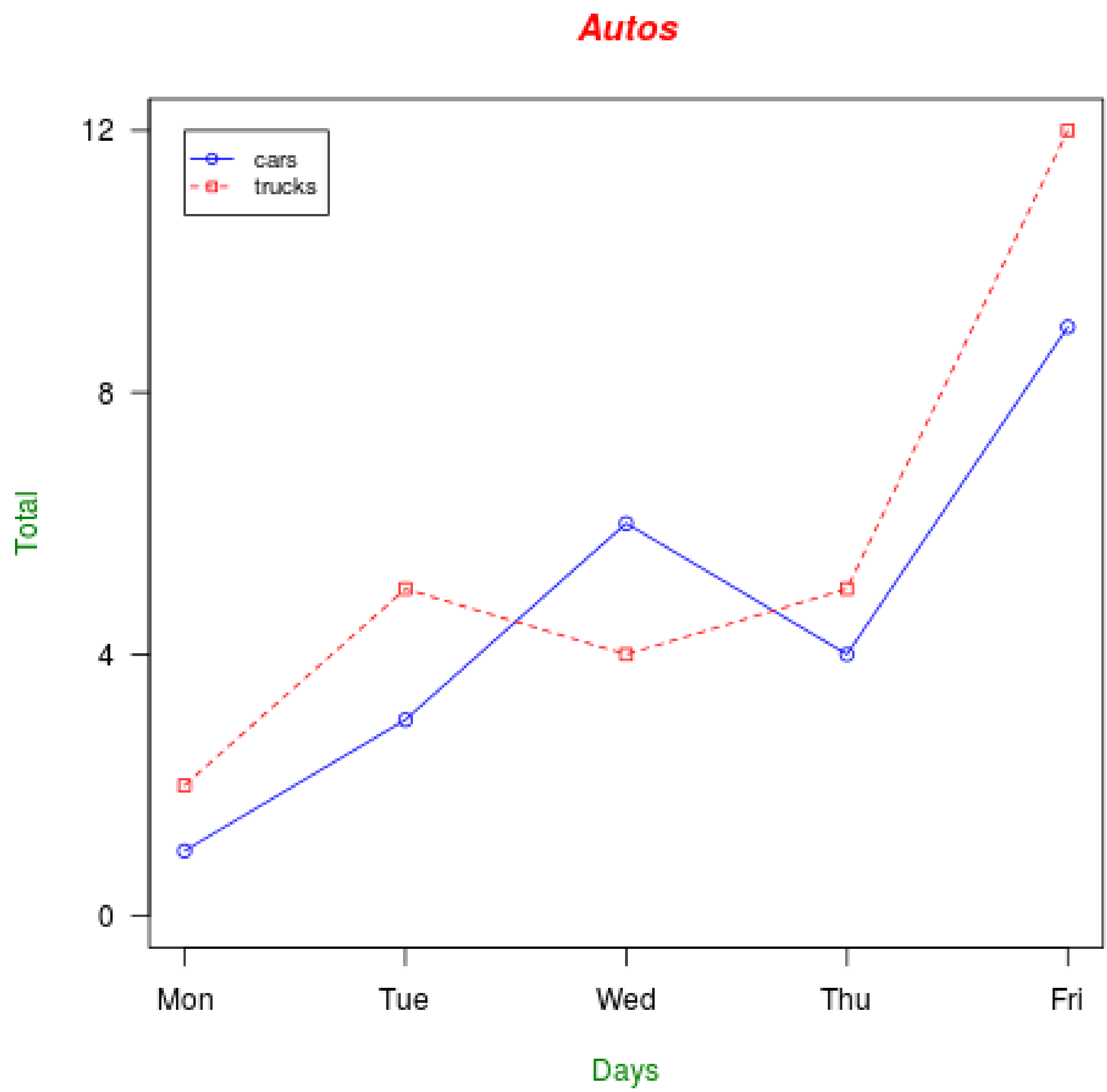
- function of two variables (surface)
- regular polyhedra
- sphere
- implicit plot of surface
- text

```
T = RDF(golden_ratio)
%var x,y,z
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(\
    z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=30)
```

1.6 Other You can also draw plots using R

To draw a plot (or do anything) using R, put

```
%r

# Define 2 vectors
cars <- c(1, 3, 6, 4, 9)
trucks <- c(2, 5, 4, 5, 12)

# Calculate range from 0 to max value of cars and trucks
g_range <- range(0, cars, trucks)

# Graph autos using y axis that ranges from 0 to max
# value in cars or trucks vector.  Turn off axes and
# annotations (axis labels) so we can specify them ourself
plot(cars, type="o", col="blue", ylim=g_range,
   axes=FALSE, ann=FALSE)

# Make x axis using Mon-Fri labels
axis(1, at=1:5, lab=c("Mon","Tue","Wed","Thu","Fri"))

# Make y axis with horizontal labels that display ticks at
# every 4 marks. 4*0:g_range[2] is equivalent to c(0,4,8,12).
axis(2, las=1, at=4*0:g_range[2])

# Create box around plot
box()

# Graph trucks with red dashed line and square points
lines(trucks, type="o", pch=22, lty=2, col="red")

# Create a title with a red, bold/italic font
title(main="Autos", col.main="red", font.main=4)

# Label the x and y axes with dark green text
title(xlab="Days", col.lab=rgb(0,0.5,0))
title(ylab="Total", col.lab=rgb(0,0.5,0))

# Create a legend at (1, g_range[2]) that is slightly smaller
# (cex) and uses the same line colors and points used by
# the actual plots
legend(1, g_range[2], c("cars","trucks"), cex=0.8,
   col=c("blue","red"), pch=21:22, lty=1:2);
```



1.7 Or even plot with octave

```
%octave
cd('/tmp')
M = rand(10);
h = figure('visible', 'off');
plot(M);
saveas(h,"figure2.png");
```

```
salvus.file('/tmp/figure2.png')
```