MAT 305: Mathematical Computing

Lecture 9: 3d plots in Sage

John Perry

University of Southern Mississippi

Fall 2009

Outline

You should be in worksheet mode to repeat the examples.

Outline

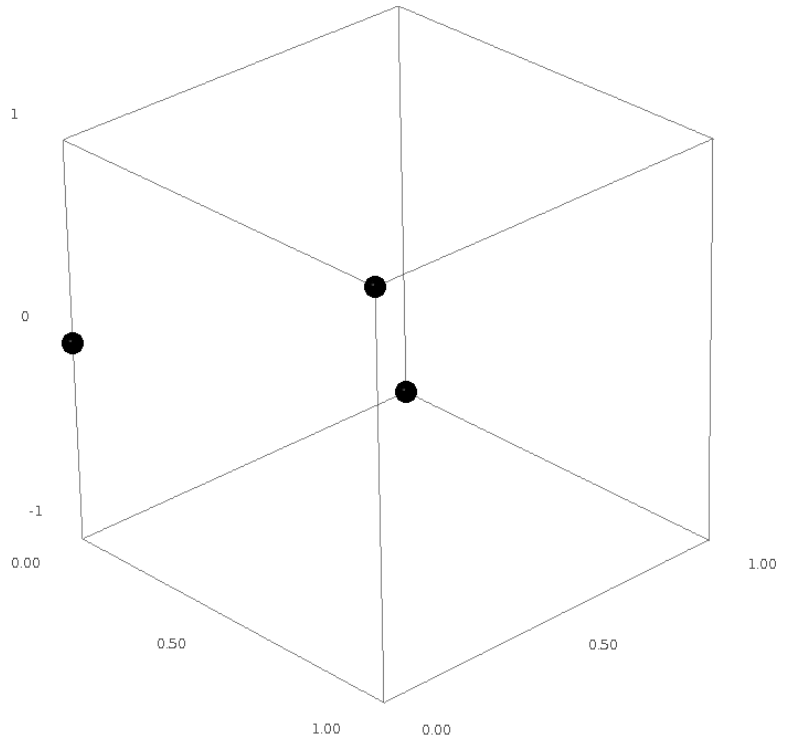
The `point3d()` command

`point3d(`(x,y,z)`,` *options*`)` where

- (x,y,z) is a 3-tuple (the location in $\mathbb{R}^3$ of this point)
  - can send list of points $[(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots]$
- *options* include
  - `rgbcolor`
  - `size` (*not* `pointsize`; default is 5)
  - `opacity` (more on this later)

Example

```
sage:  point3d([(0,0,0),(0,1,-1),(1,0,1)],
                rgbcolor=(0,0,0), size=20)
```



notice that the pointer can grab the image and rotate it for a better view!

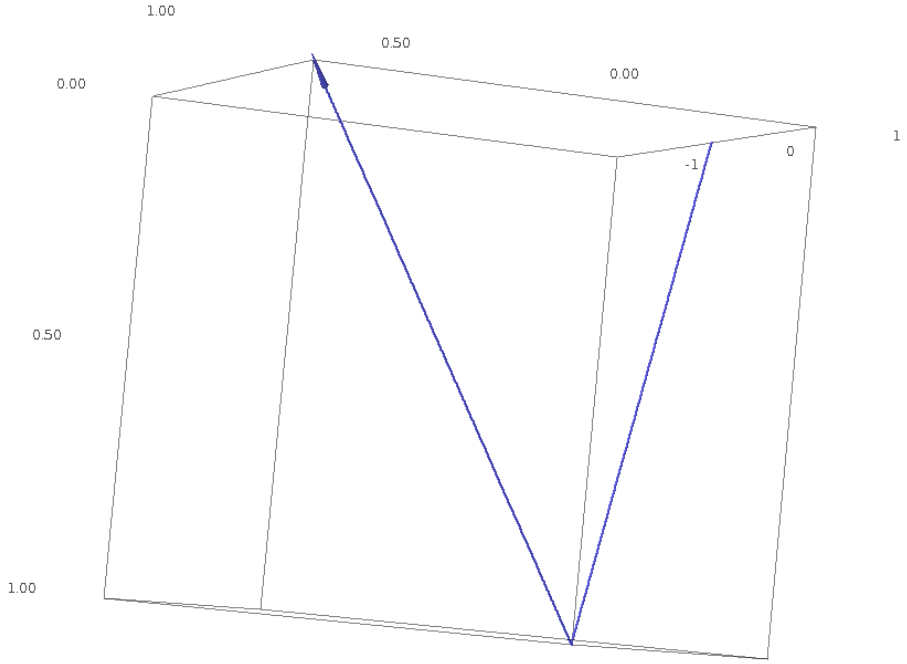
The `line3d()` command

`line3d(`$[(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots]$, *options*`)` where

- $[(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots]$ is a list of *at least two* points
  - more than two points? consecutive lines
- *options* include
  - `rgbcolor`
  - `thickness`
  - `arrow_head=True` for arrow on final point
  - `opacity` (more on this later)

Example

```
sage:  line3d([(0,0,0),(0,1,-1),(1,0,1)],
                thickness=2, arrow_head=True)
```



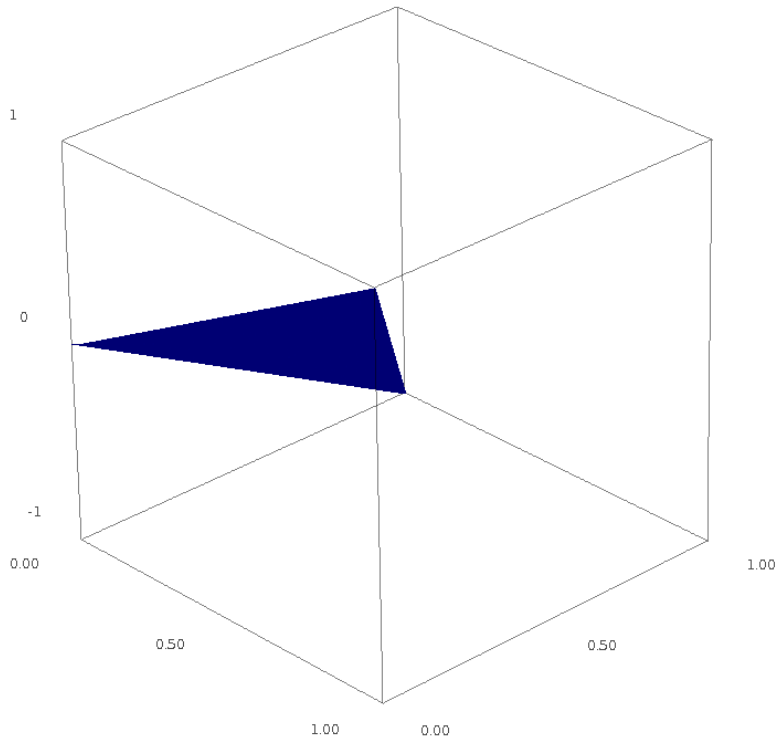
The `polygon3d()` command

`polygon3d(`$[(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots]$, *options*`)` where

- $[(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots]$ is a list of *at least two* points
  - fewer than 2? nothing drawn
- *options* include
  - `rgbcolor`
  - `opacity` (more on this later)

Example

```
sage: polygon3d([(0,0,0),(0,1,-1),(1,0,1)])
```



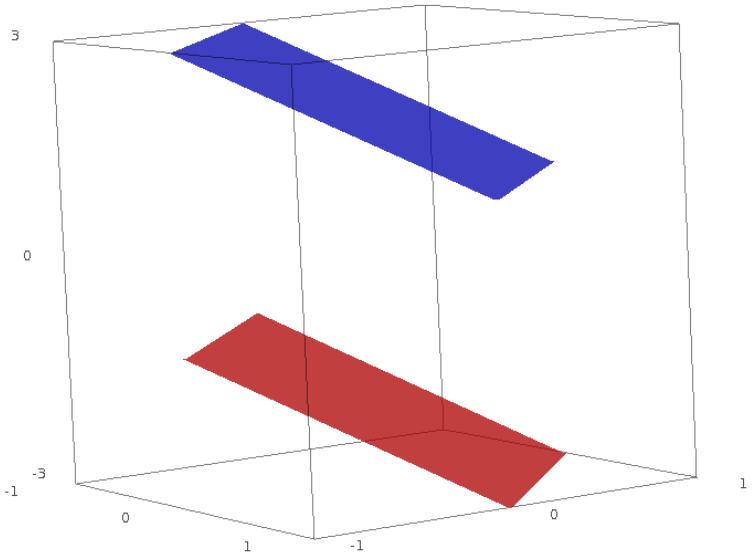
Opacity

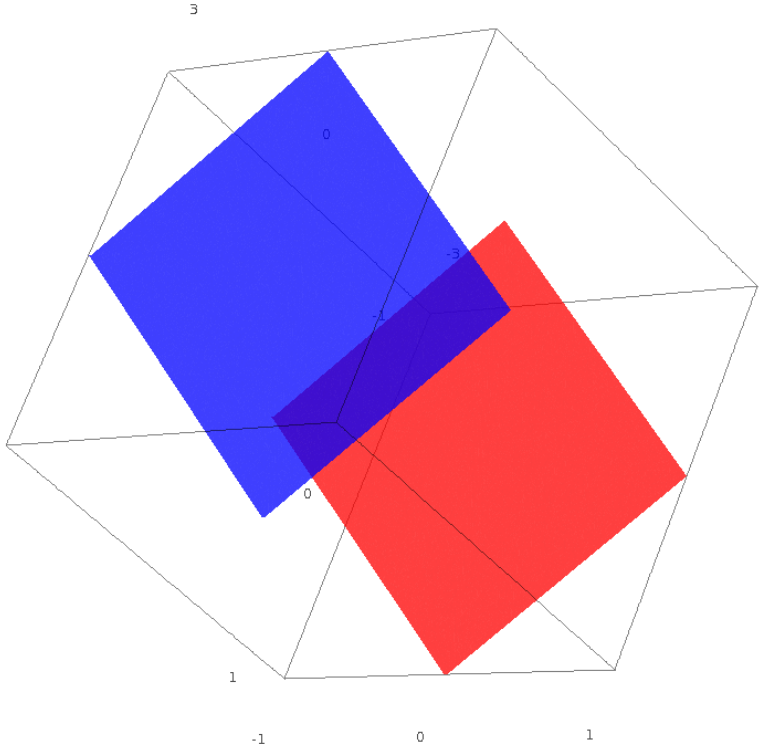
controls whether you can "see through" the object

- ranges from 0 to 1
- 0: completely translucent (invisible)
- 1: completely opaque
- useful when combining many objects

Example: two parallel planes

```
sage:  p1 = polygon3d([(1,0,1),(0,1,1),(-1,0,3),(0,-1,3)],
                      opacity=0.75)
sage:  p2 = polygon3d([(1,0,-3),(0,1,-3),(-1,0,-1),(0,-1,-1)],
                      rgbcolor=(1,0,0), opacity=0.75)
sage:  p1 + p2
```





Outline

Variables

- "standard" 2d: y depends on x
  - can define otherwise if necessary
  - Q as functions of t
- "standard" 3d: z depends on x, y
  - can define otherwise
  - x given (unless changed); must define at least y

The `plot3d()` command

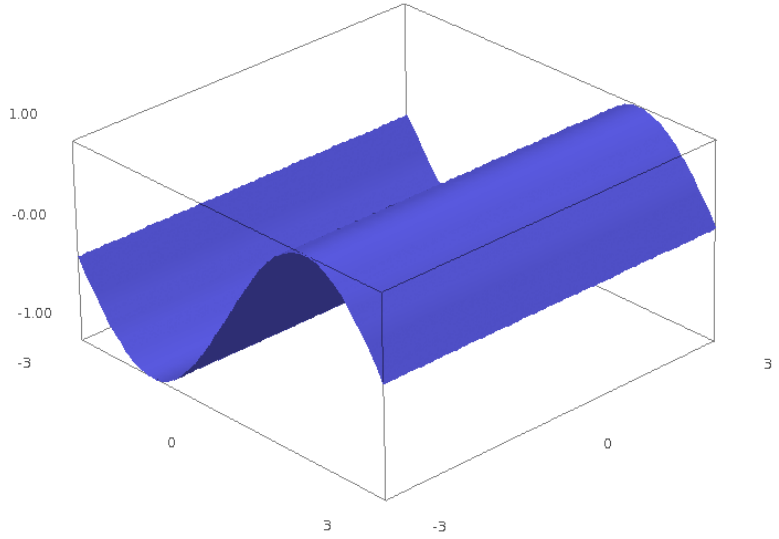
`plot3d(`$f(x,y)$`, (`x`, xmin, xmax), (`y`, ymin, ymax),`
options`)`

where

- $f(x,y)$ is a function of x and y
- *options* include
  - `adaptive=True` for a better-looking graph (but slower)
  - `mesh=True` for mesh grid lines
  - `dots=True` to show dots at grid points
  - `color`
  - `opacity`

Example 1

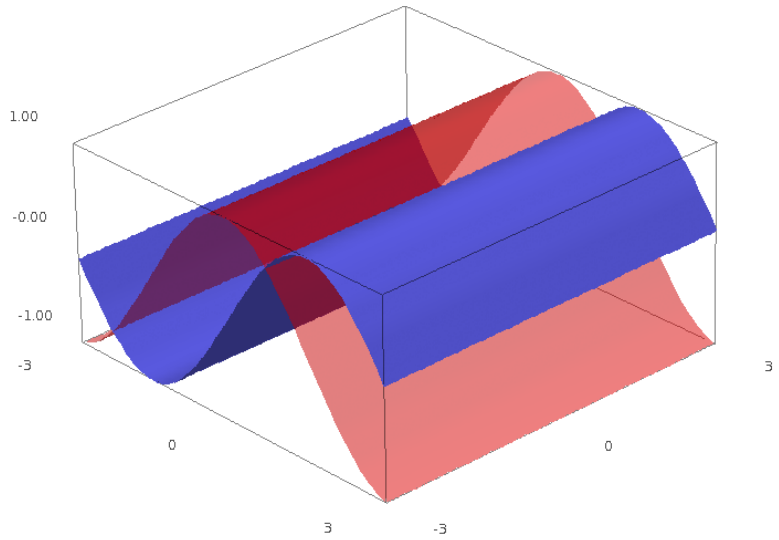
```
sage: plot3d(sin(x), (x,-3,3), (y,-3,3))
```



notice that the pointer can grab the image and rotate it for a better view!

Example 2: color and opacity

```
sage:  p1 = plot3d(sin(x), (x,-3,3), (y,-3,3))
sage:  p2 = plot3d(cos(x), (x,-3,3), (y,-3,3),
                   rgbcolor=(1,0,0), opacity=0.5)
sage:  p1+p2
```



Let's do something marginally useful

Plot $z = \sin x \cos y$ and the tangent plane at $(x, y, z) = \left(\frac{\pi}{6}, \frac{\pi}{3}, \frac{1}{4}\right)$.
(Tangent plane is $z = f_x(x_0, y_0) \cdot (x - x_0) + f_y(x_0, y_0) \cdot (y - y_0) + z_0$.)
Make the plane red and translucent.

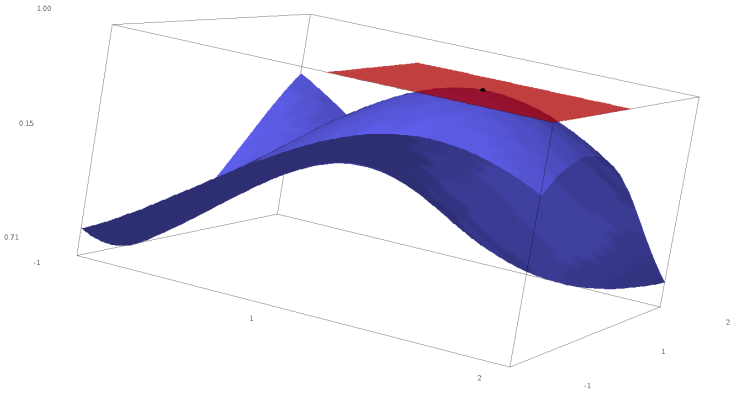
Let's do something marginally useful

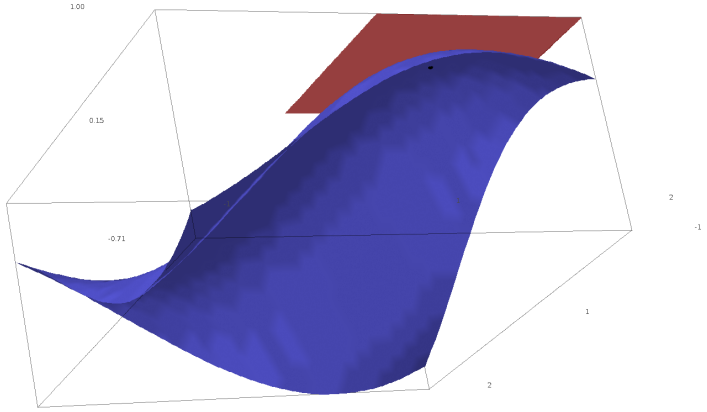
Plot $z = \sin x \cos y$ and the tangent plane at $(x, y, z) = \left(\frac{\pi}{6}, \frac{\pi}{3}, \frac{1}{4}\right)$.
(Tangent plane is $z = f_x(x_0, y_0) \cdot (x - x_0) + f_y(x_0, y_0) \cdot (y - y_0) + z_0$.)
Make the plane red and translucent.

```
sage:  f = sin(x)*cos(y)
sage:  p1 = plot3d(f, (x,-pi/4,3*pi/4), (y,-pi/4,3*pi/4))
sage:  dfx = diff(f,x)  (Need partial derivatives)
sage:  dfy = diff(f,y)
sage:  a=pi/6; b = pi/3
sage:  tanplane = dfx(x=a,y=b)*(x-a) + dfy(x=a,y=b)*(y-b)
                  + f(x=a,y=b)
sage:  p2 = plot3d(tanplane, (x,pi/4,3*pi/4), (y,-pi/4,pi/4),
                   rgbcolor=(1,0,0), opacity=0.75)
sage:  p3 = point3d((a,b,f(x=a,y=b)),rgbcolor=(0,0,0),size=10)
sage:  p1+p2+p3
```

...and you get...

The graduating seagull!





Let's add a normal vector

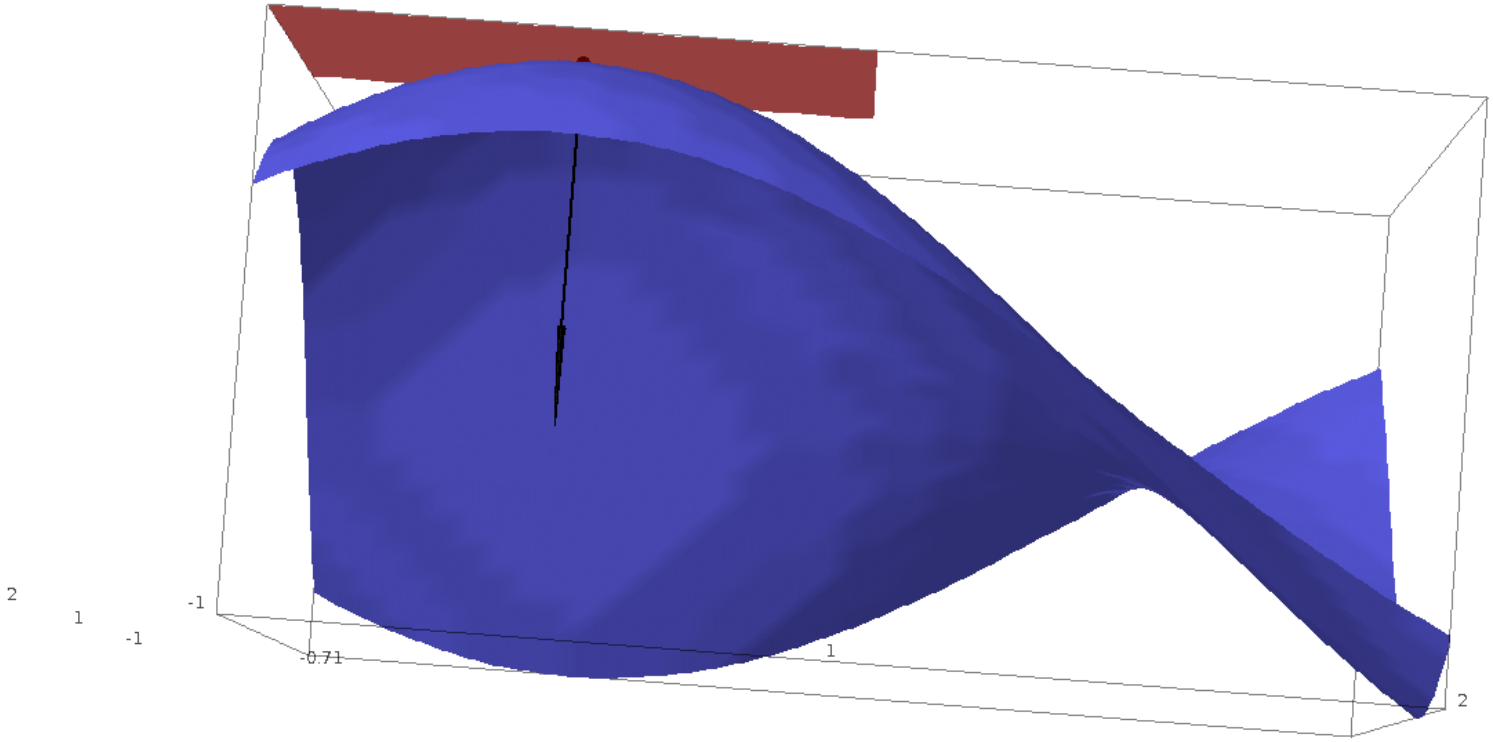
```
sage:  fgrad = (f - z).gradient()
```

(Think about why I had to do this)

```
sage:  fgrad
(cos(x)*cos(y), -sin(x)*sin(y), -1)
sage:  dx, dy, dz = fgrad(x=a,y=b)
sage:  c = f(x=a,y=b)
sage:  nvec = line3d([(a,b,c),(a+dx,b+dy,c+dz)],
                     rgbcolor=(0,0,0),thickness=2,
                     arrow_head=True)
sage:  p1 + p2 + p3 + nvec
```

...and you get...

Result



Outline

Summary

- Sage offers many ways to plot 3d objects and functions
  - plots can be rotated
  - images can be saved
- adjusting `opacity` allows one to see through an object