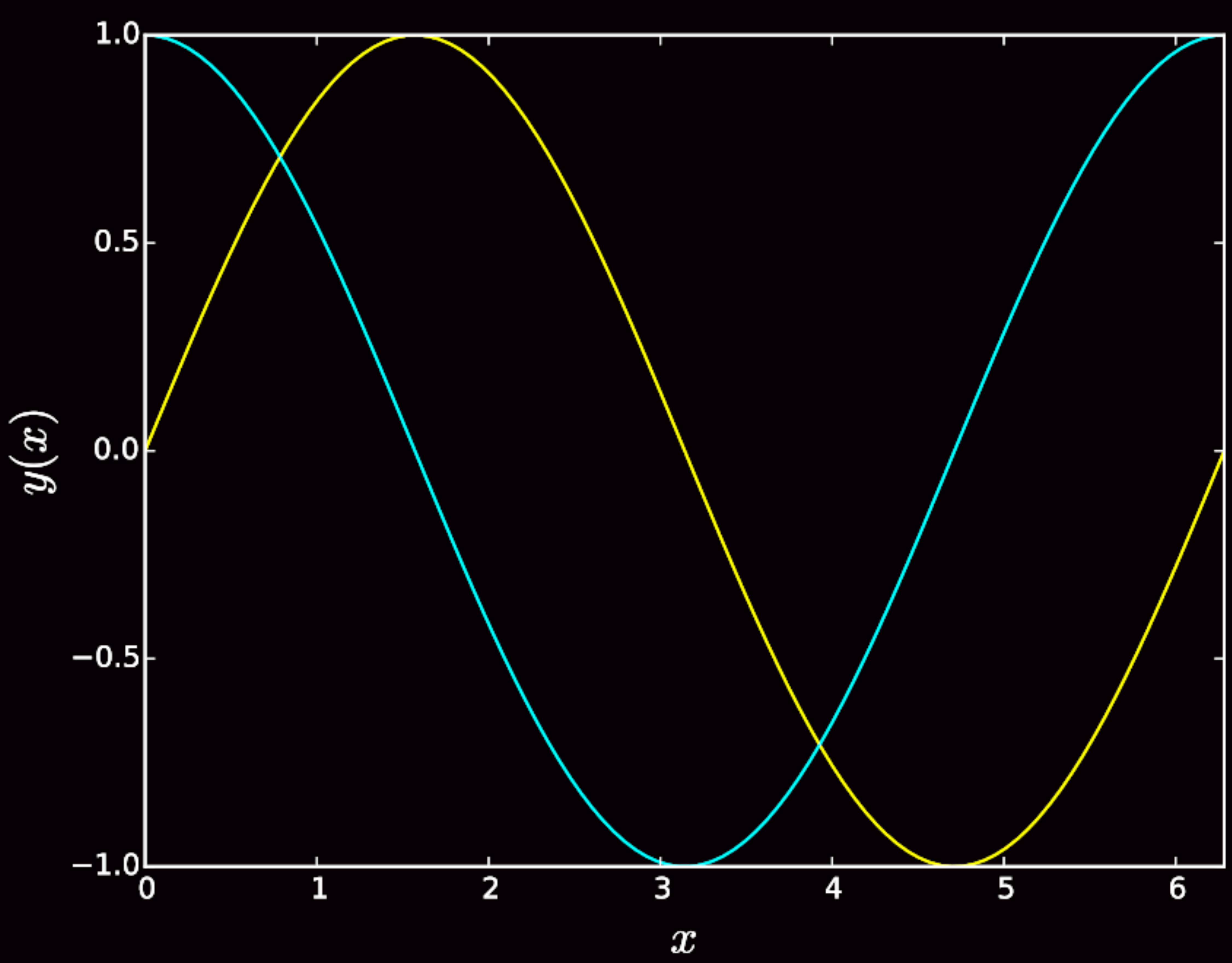
Plotting using Python

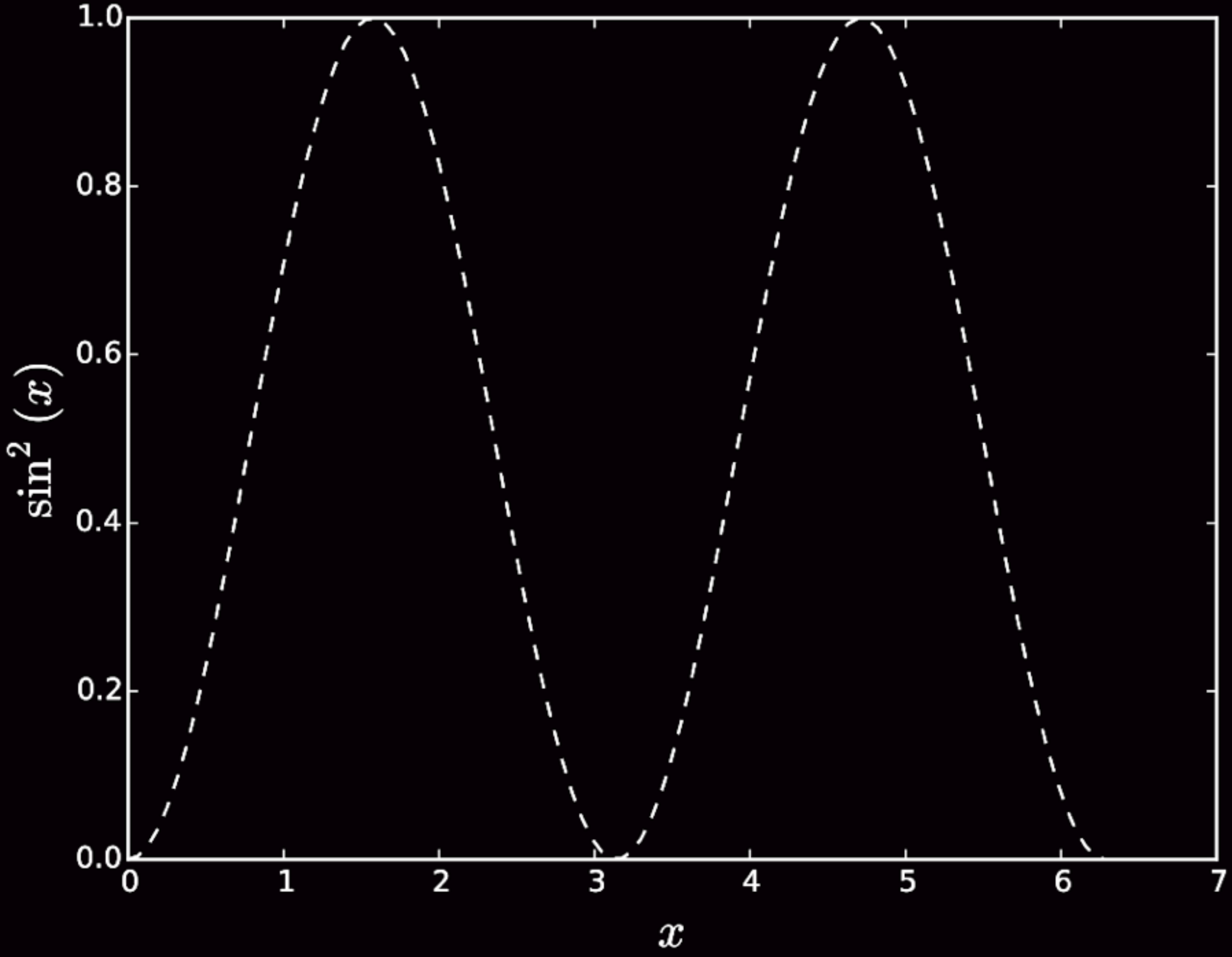
I-Introduction

In pylab

```
In [1]: x=linspace(0,2*pi,100)

In [2]: y=sin(x)

In [3]: plot(x,y)
Out[3]: [<matplotlib.lines.Line2D at 0x1087214d0>]

In [4]: xlabel('$x$',size=20)
Out[4]: <matplotlib.text.Text at 0x100635190>

In [5]: ylabel('$y(x)$',size=20)
Out[5]: <matplotlib.text.Text at 0x1086c5790>

In [6]: plot(x,cos(x),'r-')
Out[6]: [<matplotlib.lines.Line2D at 0x10a3e69d0>]

In [7]: xlim([0,2*pi])
Out[7]: (0, 6.283185307179586)
```


1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6
$y(x)$
x

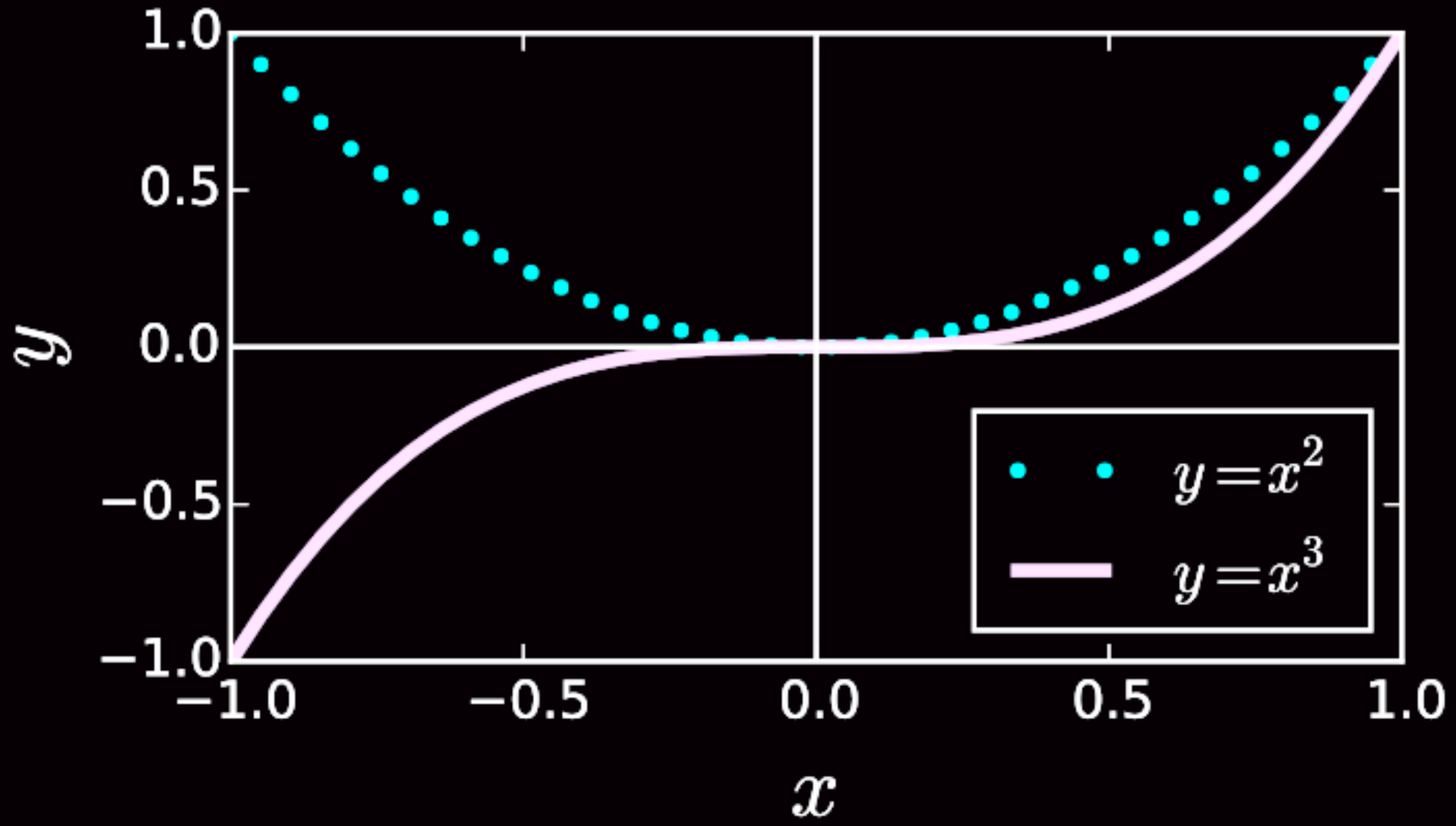
```
In [17]: figure()
Out[17]: <matplotlib.figure.Figure at 0x109e736d0>

In [18]: plot(x,sin(x)**2,'k--')
Out[18]: [<matplotlib.lines.Line2D at 0x11083dcd0>]

In [19]: xlabel('$x$',size=20)
Out[19]: <matplotlib.text.Text at 0x110809190>

In [21]: ylabel('$\sin^2(x)$',size=20)
Out[21]: <matplotlib.text.Text at 0x10aeca910>
```



$\sin^2(x)$
x
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7

Plotting using files
2D plot

```python
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 5, 3
# figure of the size 5in x 3in

x = np.linspace(-1,1,10)
y = x**2
y1 = x**3

plt.plot(x,y, 'r.', label = r'$y=x^2$')
plt.plot(x,y1, lw = 3, color = 'g', label = r'$y=x^3$')

plt.axhline(0, color='k') # draw hor axis
plt.axvline(0, color='k') # draw vertical axis

plt.xlim(-1,1)
plt.ylim(-1,1)

plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)
plt.legend(loc=4)

plt.savefig('plot2d.pdf')
plt.show()
```



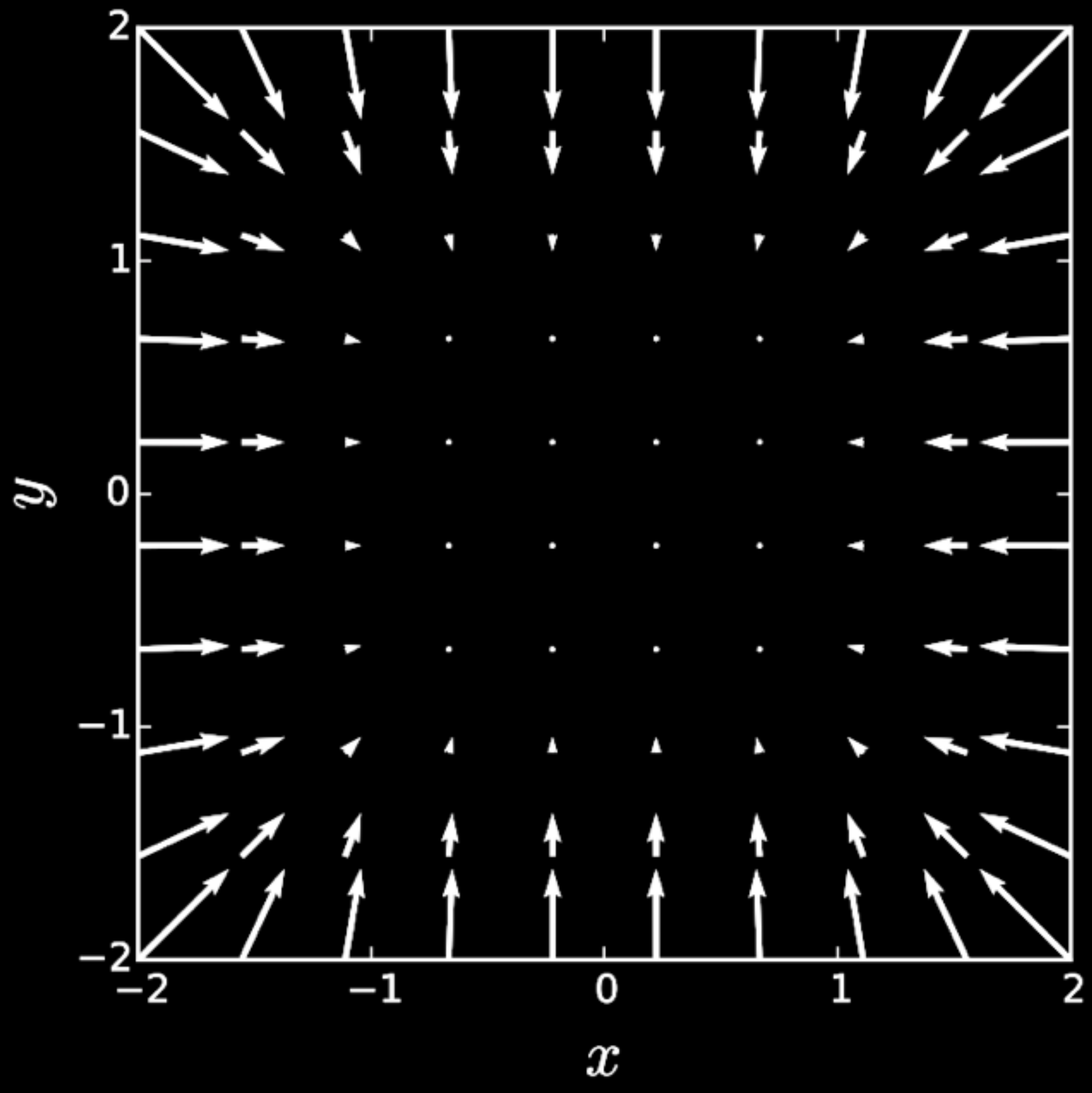
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x
$y = x^2$
$y = x^3$

Vector plot

$\mathbf{v} = -\mathbf{r}$

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm
from pylab import rcParams
rcParams['figure.figsize'] = 5, 3

# Vector plot
x = np.linspace(-2,2,10)
y = np.linspace(-2,2,10)

xv, yv = np.meshgrid(x,y)
# meshgrid is a 2D grid: (xv,yv) provides the coordinate at points
a mesh of Nx*Ny grid.

plt.figure()
plt.quiver(xv, yv, -xv**3, -yv**3)
# quiver = array: At point (xv,yv), make vector of (-xv**3,-yv**3).
```

```python
plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)
plt.xticks(np.linspace(-2, 2, 5, endpoint=True))
plt.yticks(np.linspace(-2, 2, 5, endpoint=True))
plt.axes().set_aspect('equal')
plt.tight_layout()

plt.savefig("vector.pdf")
plt.show()
```

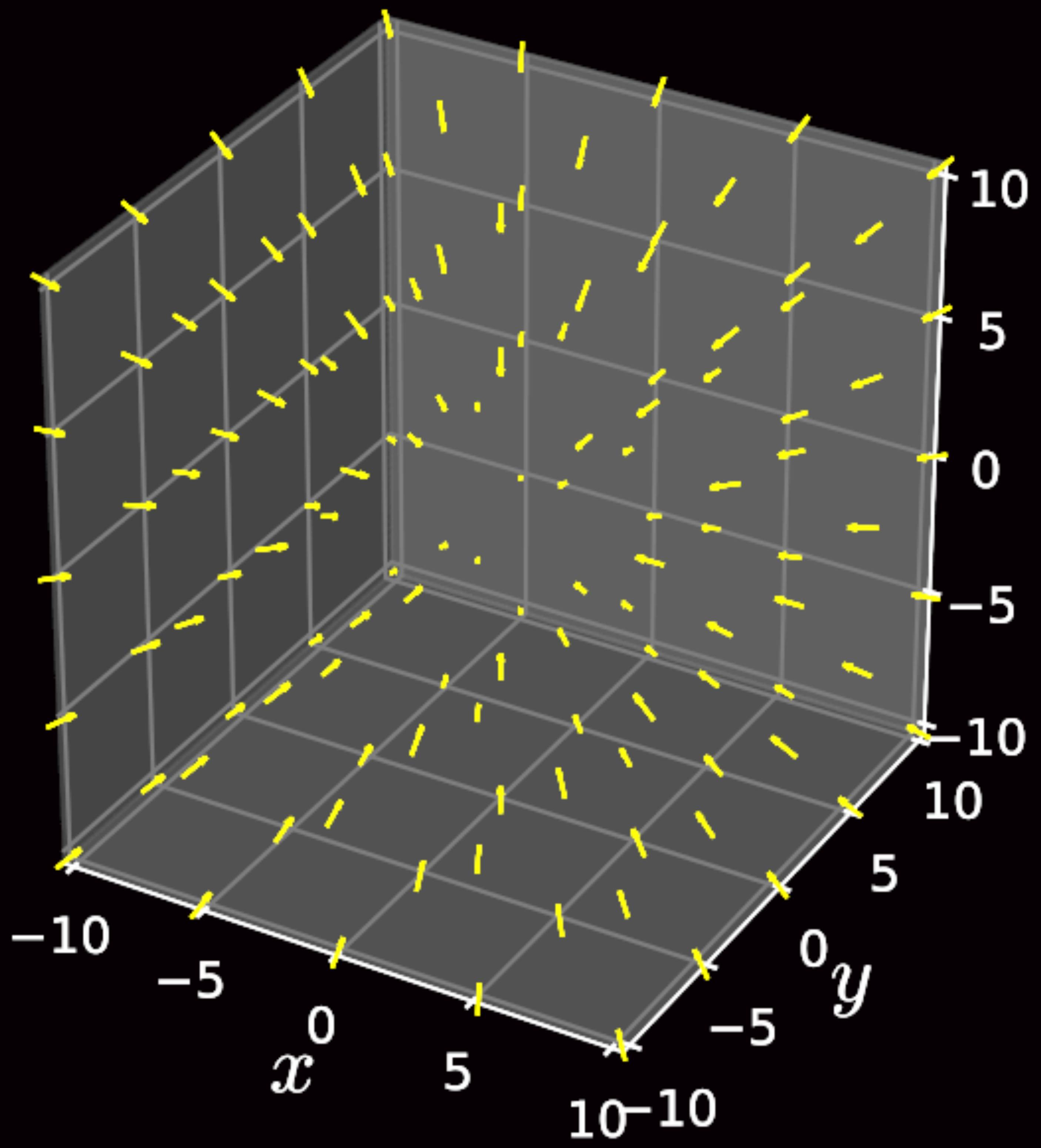


2
1
0
−1
−2
y
−2
−1
0
1
2
x

Vector3d plot

v = -r

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm


plt.gca(projection='3d') #gca = get current axis

# Vector plot
x = np.linspace(-10,10,5)
y = np.linspace(-10,10,5)
z = np.linspace(-10,10,5)

xv, yv,zv = np.meshgrid(x,y,z)
# meshgrid is a 3D grid: (xv,yv,zv)=coordinate

plt.quiver(xv, yv, zv, -xv, -yv, -zv)
plt.savefig("vector3d.pdf")
plt.show()
```



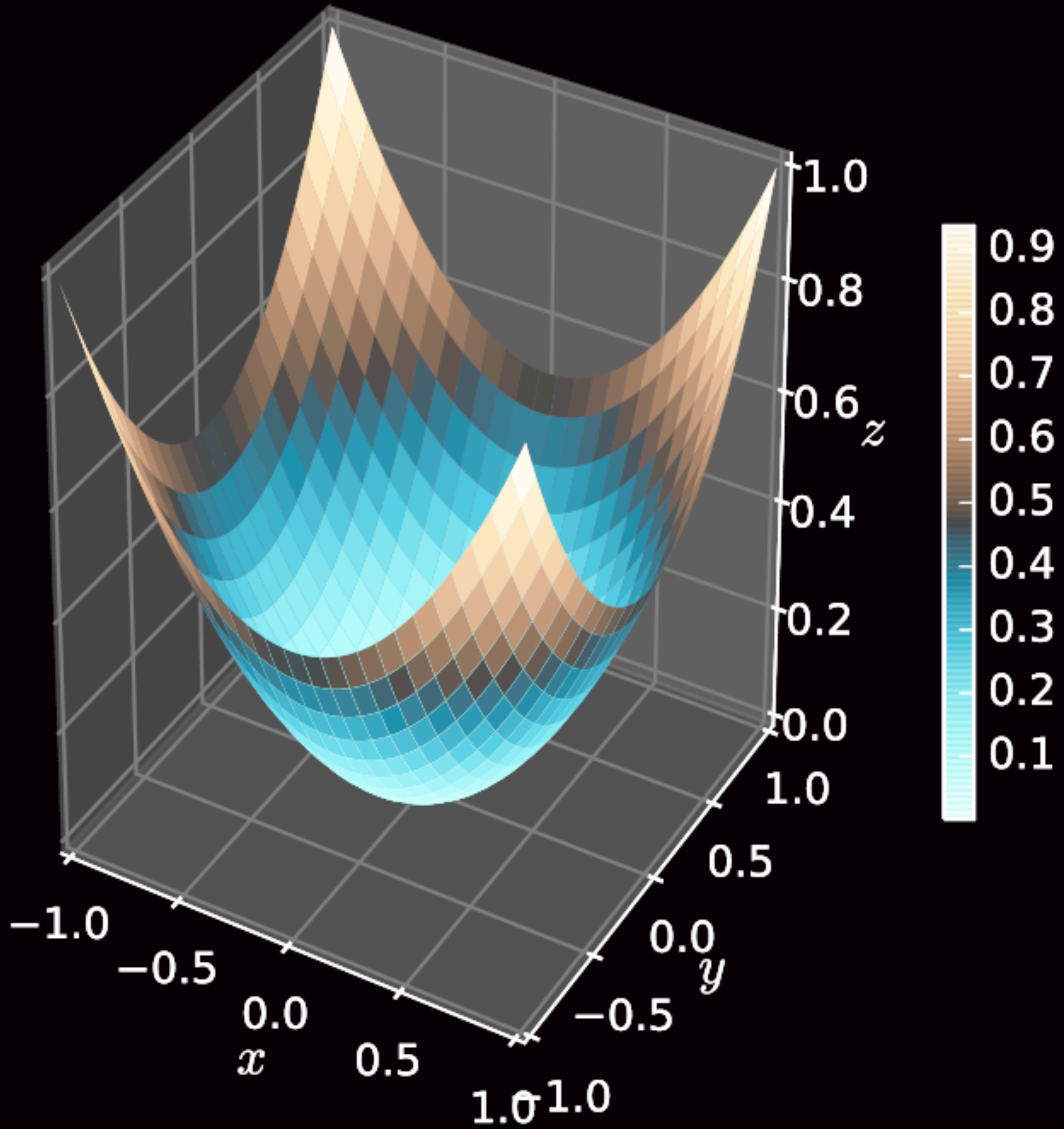
−10
−5
0
5
10
x
−10
−5
0
5
10
y
−10
−5
0
5
10

Surface plot

$$f(x,y) = x^2 + y^2$$

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from pylab import rcParams
rcParams['figure.figsize'] = 5, 5


## 3D surface_plot
fig = plt.figure()
axes = fig.gca(projection='3d') #gca = get current axis

x = np.linspace(-1,1,100)
y = np.linspace(-1,1,100)

xv, yv = np.meshgrid(x,y)
z = (xv**2 + yv**2)/2
```

```
# surface_plot with color grading and color bar
p = axes.plot_surface(xv,yv,z, rstride=4, cstride=4, cmap=cm.RdBu,
    linewidth=0, antialiased=False)
fig.colorbar(p, shrink=0.5)

axes.set_xlabel('$x$',fontsize=15)
axes.set_ylabel('$y$',fontsize=15)
axes.set_zlabel('$z$',fontsize=15)
plt.tight_layout()
fig.savefig(“surface.pdf”)
plt.show()
```

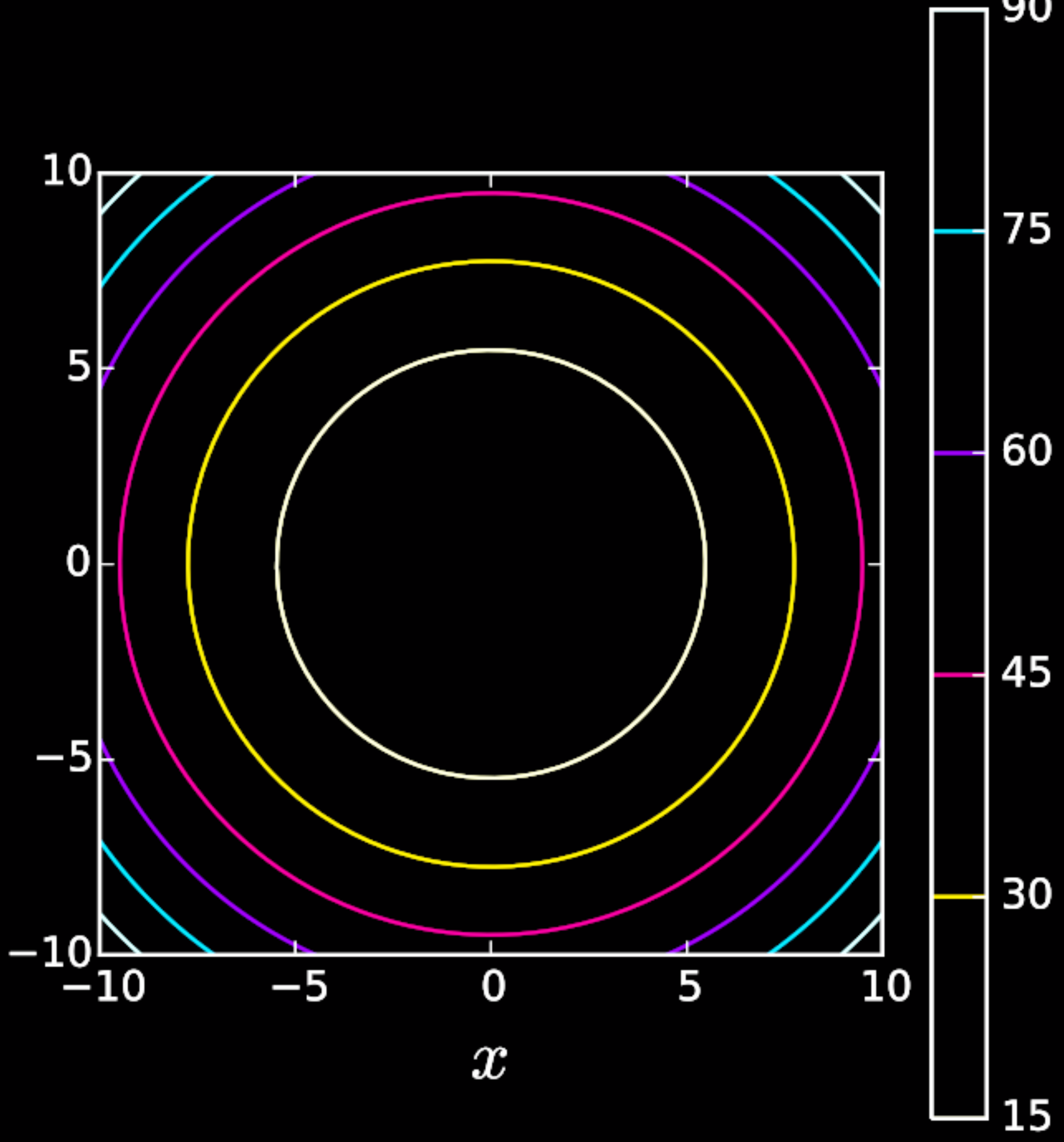


x
y
z
−1.0
−0.5
0.0
0.5
1.0
−1.0
−0.5
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Contour plot

$$f(x,y) = -\log(\sqrt{x^2 + y^2})$$

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 5, 5

### contour plot
#
plt.figure()

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

xv, yv = np.meshgrid(x,y)
#z = (xv**2 + yv**2)/2
z = -np.log(np.sqrt((xv**2 + yv**2)))

curves = plt.contour(xv,yv,z,6)
plt.colorbar()
plt.xlabel('$x$',size=20)
plt.ylabel('$y$',size=20)
plt.axes().set_aspect('equal')
plt.tight_layout()
plt.savefig("contour.pdf")
plt.show()
```

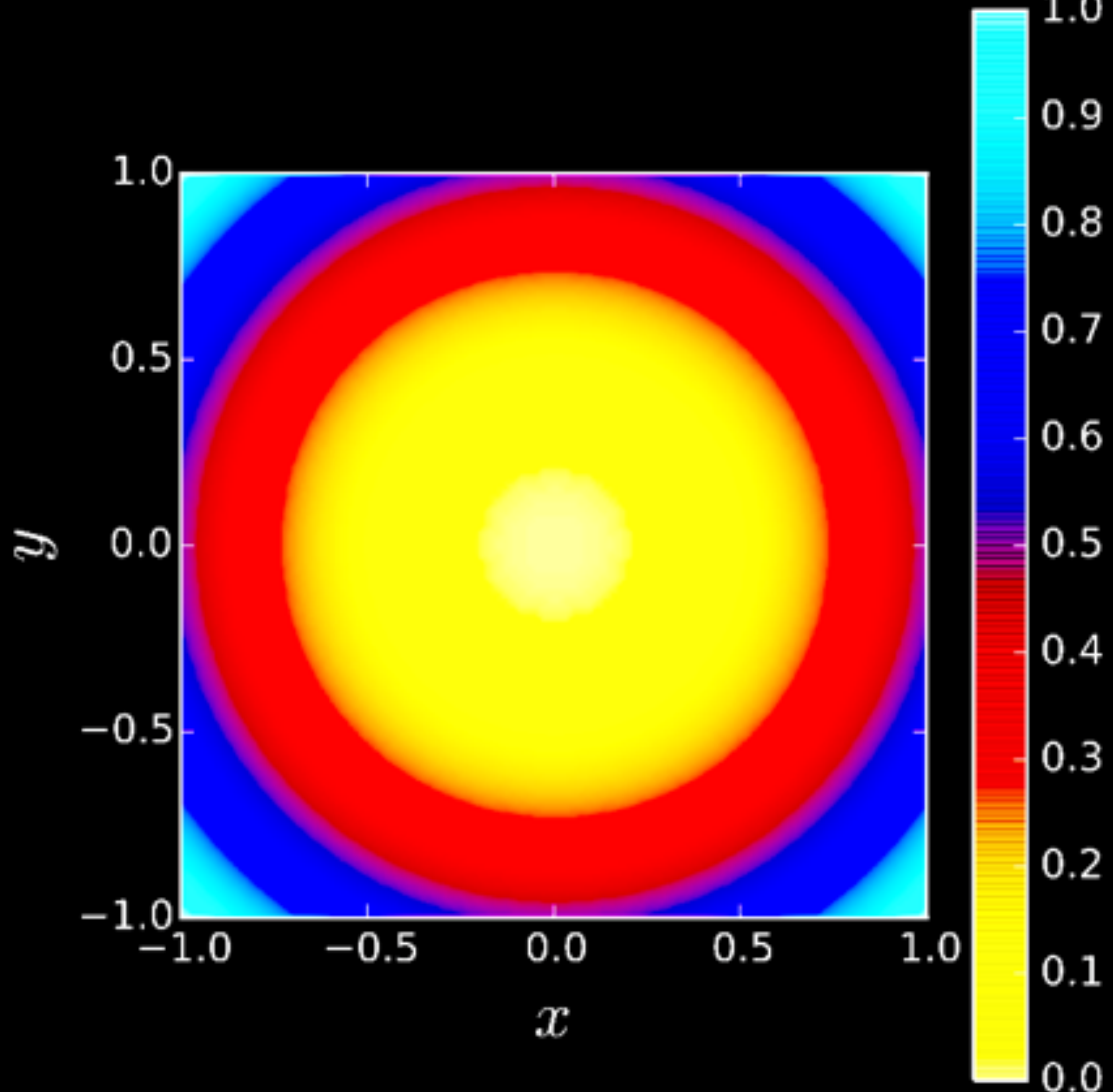


90
75
60
45
30
15
10
5
0
−5
−10
−10
−5
0
5
10
x
y

Density plot

$f(x,y) = x^2 + y^2$

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 5, 5

x = np.linspace(-1,1,100)
y = np.linspace(-1,1,100)

xv, yv = np.meshgrid(x,y)
z = (xv**2 + yv**2)/2

plt.imshow(z, vmin=abs(z).min(), vmax=abs(z).max(),
extent=[-1,1,-1,1])
plt.colorbar()
plt.xlabel('$x$',size=20)
plt.ylabel('$y$',size=20)
plt.axes().set_aspect('equal')
plt.tight_layout()
plt.savefig("density1.pdf")
plt.show()
plt.figure()
plt.pcolor(xv,yv, z)
plt.colorbar()
plt.xlabel('$x$',size=20)
plt.ylabel('$y$',size=20)
plt.axes().set_aspect('equal')
plt.tight_layout()
plt.savefig("density2.pdf")
plt.show()
```



Making Animation

$$f(x,t)=\frac{1}{2(\cosh(x-t)/2)^2}$$

from Stewart, Python for Scientists

```
# from Stewart, Python for Scientists
import numpy as np
import matplotlib.pyplot as plt
import time
def sol(t,x):
  return 0.5/np.cosh(0.5*(x-t))**2

x=np.linspace(0,60.0,1001)

plt.ion()
plt.xlabel('x')
plt.ylabel('y')
line,=plt.plot(x,sol(10,x))
# line(xdata, ydata)

for t in np.linspace(-10,70,161):
    line.set_ydata(sol(t,x))
    plt.draw()
    plt.title('Soliton wave at t = %5.1f' % t)
    time.sleep(0.1)
```

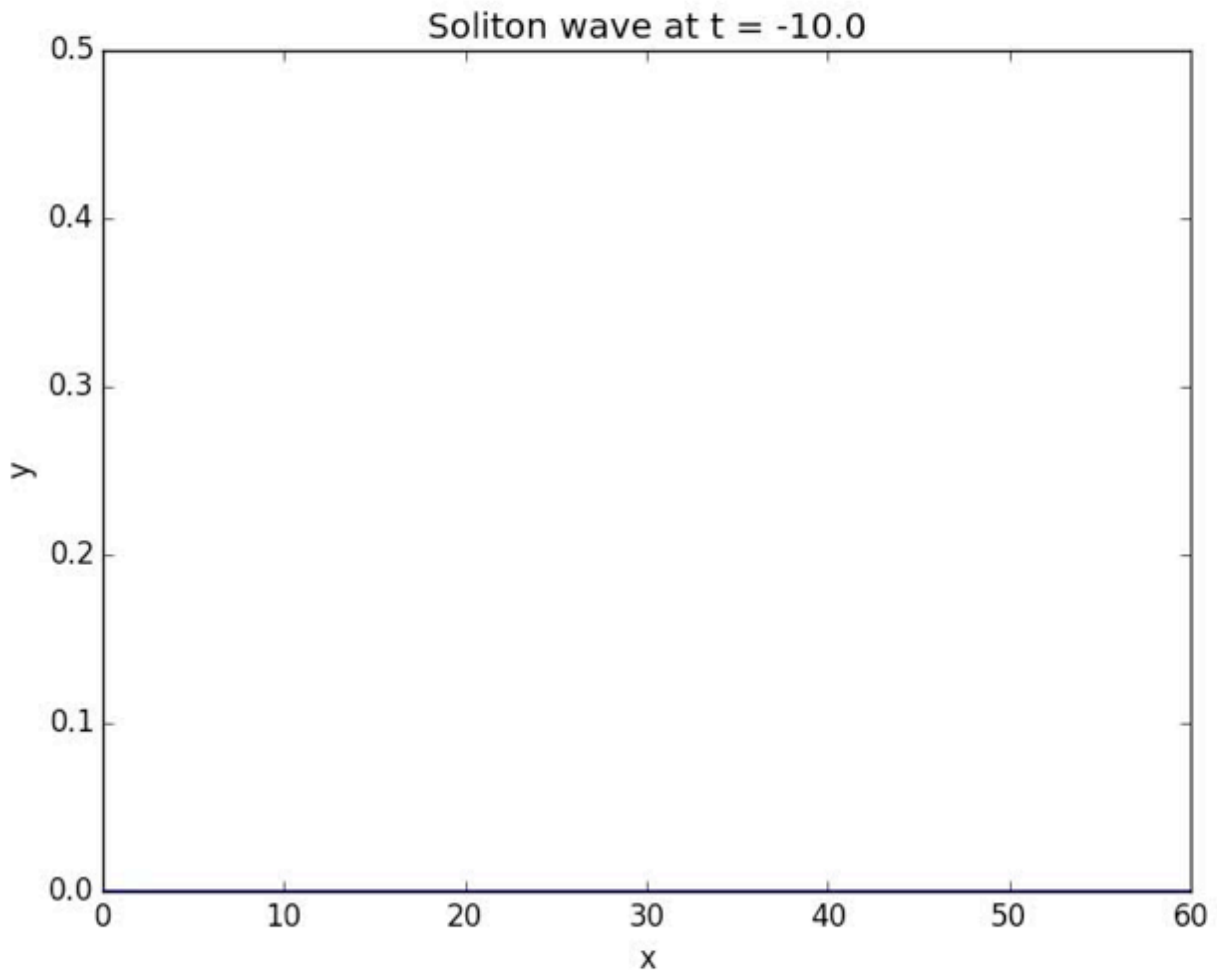
Movie

$$f(x,t) = \frac{1}{2(\cosh(x-t)/2)^2}$$

from Stewart, Python for Scientists

```
# from Stewart, Python for Scientists
import numpy as np
import matplotlib.pyplot as plt
def sol(t,x):
  return 0.5/np.cosh(0.5*(x-t))**2

def draw_frame(t,x):
  plt.plot(x,sol(t,x))
  plt.axis((0,60.0,0,0.5))
  plt.xlabel('x')
  plt.ylabel('y')
  plt.title('Soliton wave at t = %05.1f' % t)

x=np.linspace(0,60.0,1001)
t=np.linspace(-10,10,101)

for i in range(len(t)):
  file_name='_temp%05d.png' % i
    # file_name = _temp(i).png
  draw_frame(t[i],x)
  plt.savefig(file_name)
  plt.clf()  # clear current figure
```

```
import os
os.system("rm _movie.mpg")
os.system("/Users/mkv/local/bin/ffmpeg -r 25 " +
         " -i _temp%05d.png -b:v 1800 _movie.mpg")
# make a mpg file using the _temp files
os.system("rm _temp*.png")
```


Soliton wave at t = -10.0
y
x
0.0
0.1
0.2
0.3
0.4
0.5
0
10
20
30
40
50
60