高知大学教育学部の情報数学のテキスト

文責　：　高知大学名誉教授　中村　治

参考： sage によるプログラミング

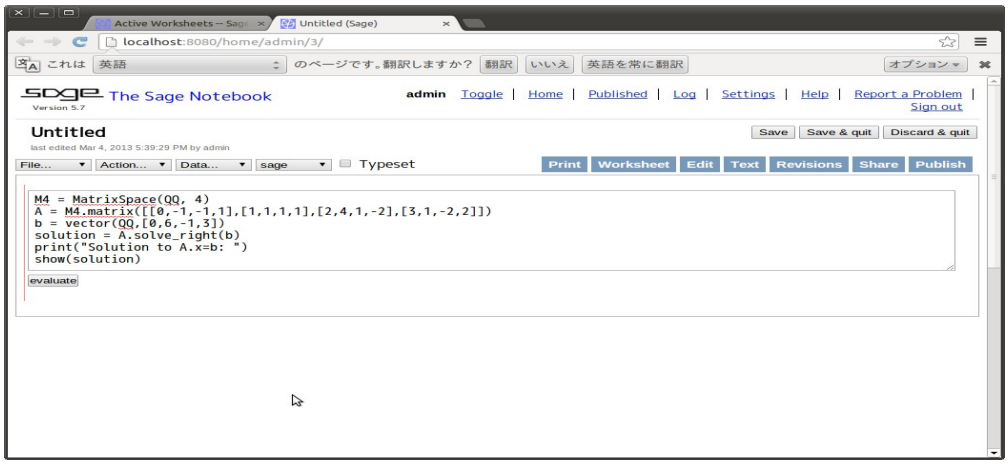
sage という数式処理ソフトを使うとこのテキストで述べた計算が簡単にできます。

$$\begin{cases} -x_2 - x_3 + x_4 = 0 \\ x_1 + x_2 + x_3 + x_4 = 6 \\ 2x_1 + 4x_2 + x_3 - 2x_4 = -1 \\ 3x_1 + x_2 - 2x_3 + 2x_4 = 3 \end{cases}$$

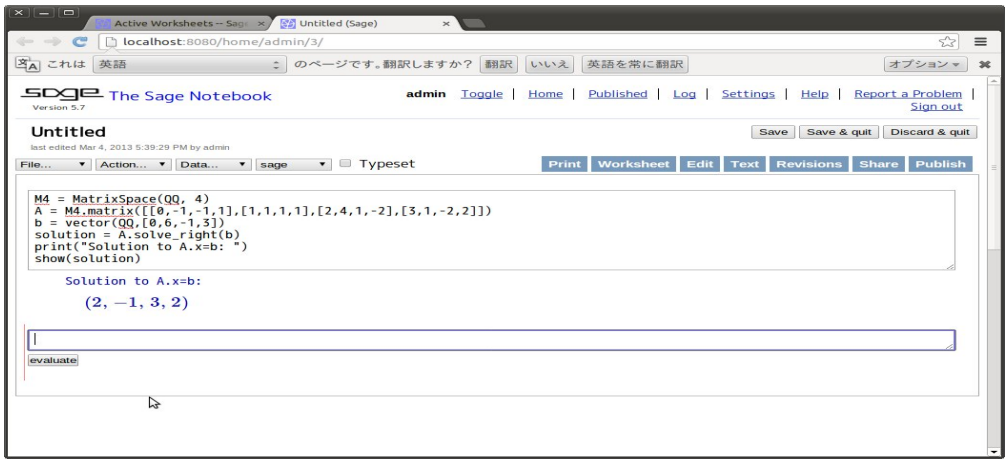
という連立方程式を解くには、sage の notebook で

```
A  =  Matrix(QQ, [[0, -1, -1, 1],[1, 1, 1, 1],[2, 4, 1, -2][3, 1, -2, 2]])
B  =  vector([0, 6, -1, 3])
solution = A.solve_right(B)
show(solution)
```

と入力し、



evaluate のボタンをクリックすれば良いです。



$$A = \begin{pmatrix} 2 & 5 & 4 \\ 3 & 1 & 2 \\ 5 & 4 & 6 \end{pmatrix} \tag{1}$$

の逆行列は

```
A = matrix(QQ, [[2, 5, 4],[3, 1, 2],[5, 4, 6]])
show(A.inverse())
```

と入力し、evaluate のボタンをクリックすれば良いです。

C や C++ で不定積分を計算するのは大変難しいですが、sage では簡単で

```
var('x')
f(x) = e^x * cos(x)
f_int(x) = integrate(f, x)
show(f_int)
```

と入力すればよく、$\sqrt{(1-x^2)}$ の 0 から 1 の定積分は

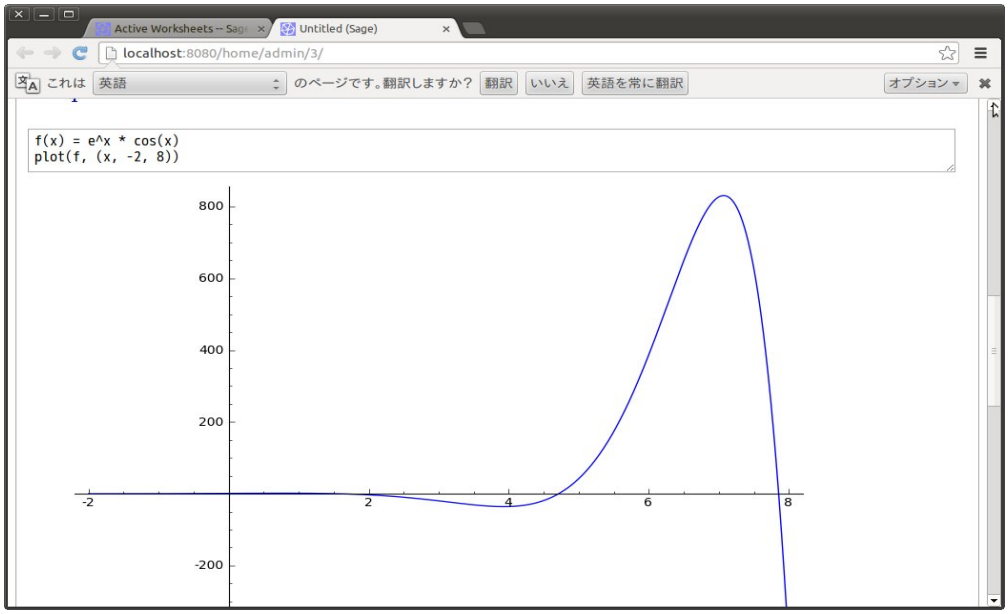
```
f(x) = sqrt(1 - x^2)
f_integral = integrate(f, (x, 0, 1))
show(f_integral)
```

と入力すればいいです。

関数のグラフを描くのも簡単で

```
f(x) = e^x * cos(x)
plot(f, (x, -2, 8))
```

と入力すれば



を描きます。三次元空間内の曲面のグラフも簡単に描けます。

sage は Windows でも使えるとインターネットには書いていますが、その方法をいくつかやってみましたが、私はいずれも成功しませんでした。Windows にインストールするのは難しいです。

４万円の怪しげなノートパソコンをパソコン工房で購入し、それに無料の ubuntu という linux の OS をインストールし、それに sage をインストールして使っています。sage は１００種類ぐらいのフリーのソフトを Python でまとめ上げたソフトだそうです。

linux の世界には無数のフリーソフトがあります。例えば、めちゃくちゃ強い無料の将棋ソフトもあります。使えるプリンタが限られているとか、ソフトをインストールするのに分からなくて、インターネットで情報を探し回ったりしなければなりませんが、十年前と比べると ubuntu 12.04

は隔世の感があるくらい使いやすくなっています。昔は linux では、ソフトをインストールし、それが動くことで満足していましたが、現在は各種フリーソフトを駆使し、実際に仕事をする時代になりました。光ファイバにインターネットがつながっていれば、すごく快適です。Windows と随分勝手が違い、私のように教えてくれる友人がいなければ、時間が無駄に過ぎていき、苦労しますが、得られるものは多いです。linux の世界は宝の山です。

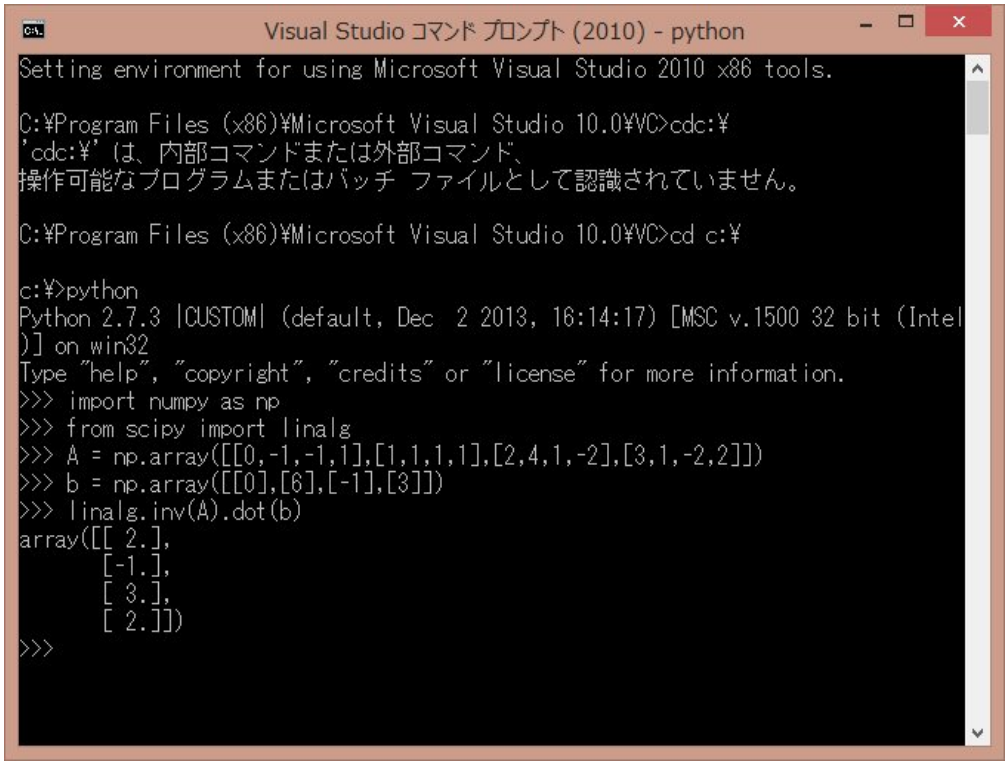
近似値で良ければ、Python のライブラリ (NumPy, SciPy, matplotlib) を使っても同じことが出来ます。これなら Windows でも実行できます。

$$\begin{cases} -x_2 - x_3 + x_4 = 0 \\ x_1 + x_2 + x_3 + x_4 = 6 \\ 2x_1 + 4x_2 + x_3 - 2x_4 = -1 \\ 3x_1 + x_2 - 2x_3 + 2x_4 = 3 \end{cases}$$

という連立方程式を解くには、sage の notebook で

```
import numpy as np
from scipy import linalg
A = np.array([[0,-1,-1,1],[1,1,1,1],[2,4,1,-2],[3,1,-2,2]])
b = np.array([[0],[6],[-1],[3]])
linalg.inv(A).dot(b)
```

とすれば



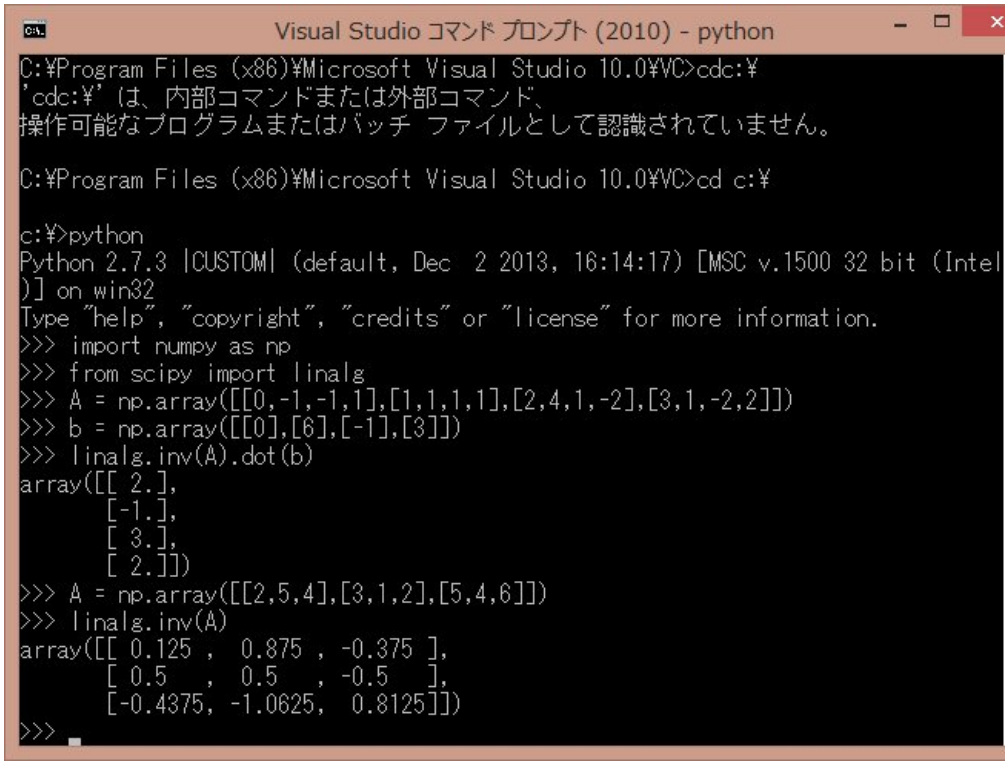
を得ます。

$$A = \begin{pmatrix} 2 & 5 & 4 \\ 3 & 1 & 2 \\ 5 & 4 & 6 \end{pmatrix} \tag{2}$$

の逆行列は、続けて

```
A = np.array([[2,5,4],[3,1,2],[5,4,6]])
linalg.inv(A)
```

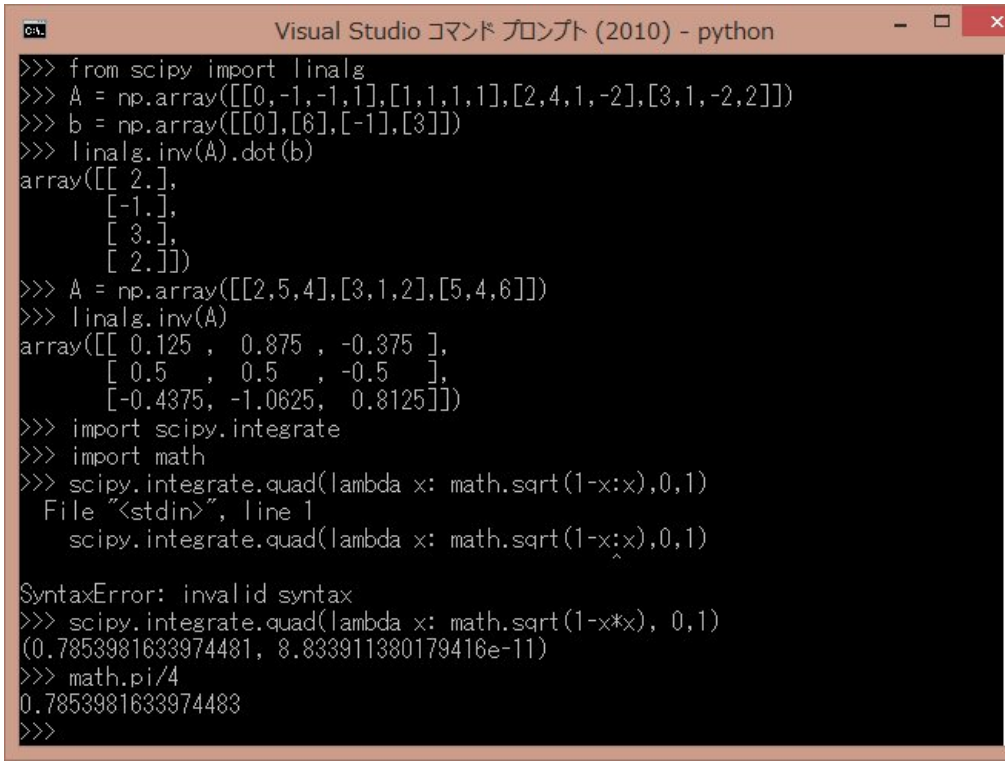
と入力すれば良いです。



$\sqrt{(1-x^2)}$ の 0 から 1 の定積分は

```
import scipy.integrate
import math
scipy.integrate.quad(lambda x: math.sqrt(1-x*x), 0, 1)
math.pi/4
```

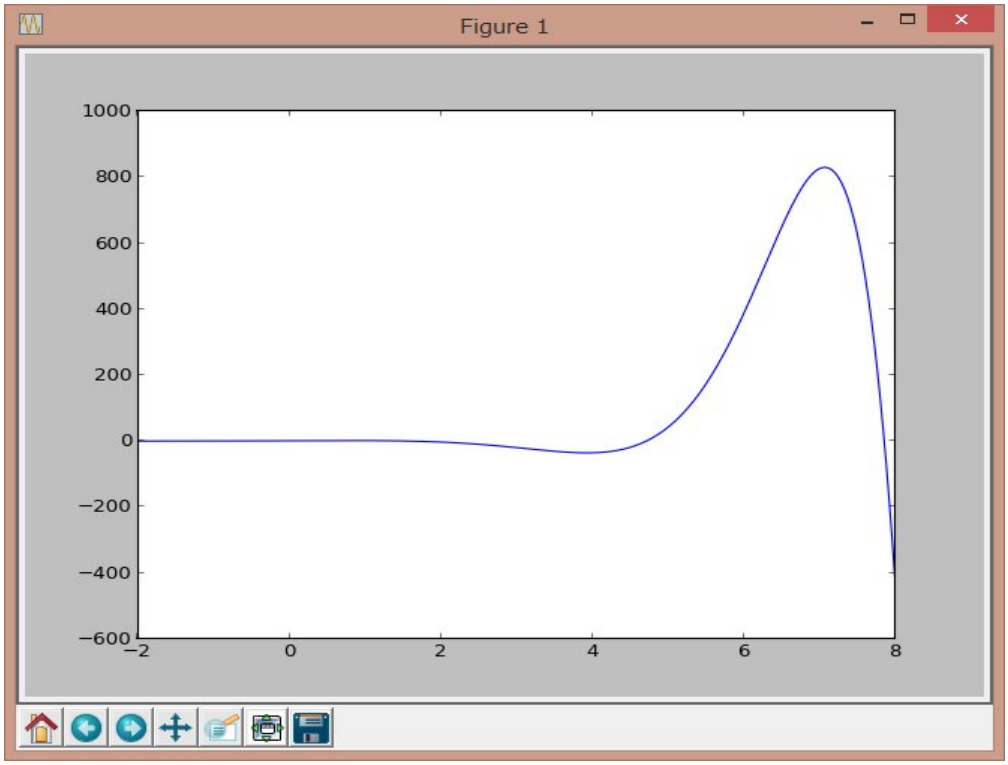
と入力すればいいです。



関数のグラフを描くのも簡単で、Python 2.7.6 Shell を使って

```
import pylab as pl
import numpy as np
X = np.linspace(-2, 8, 256, endpoint=True)
F = np.exp(X)*np.cos(X)
pl.plot(X, F)
pl.show()
```

のプログラムを実行すれば



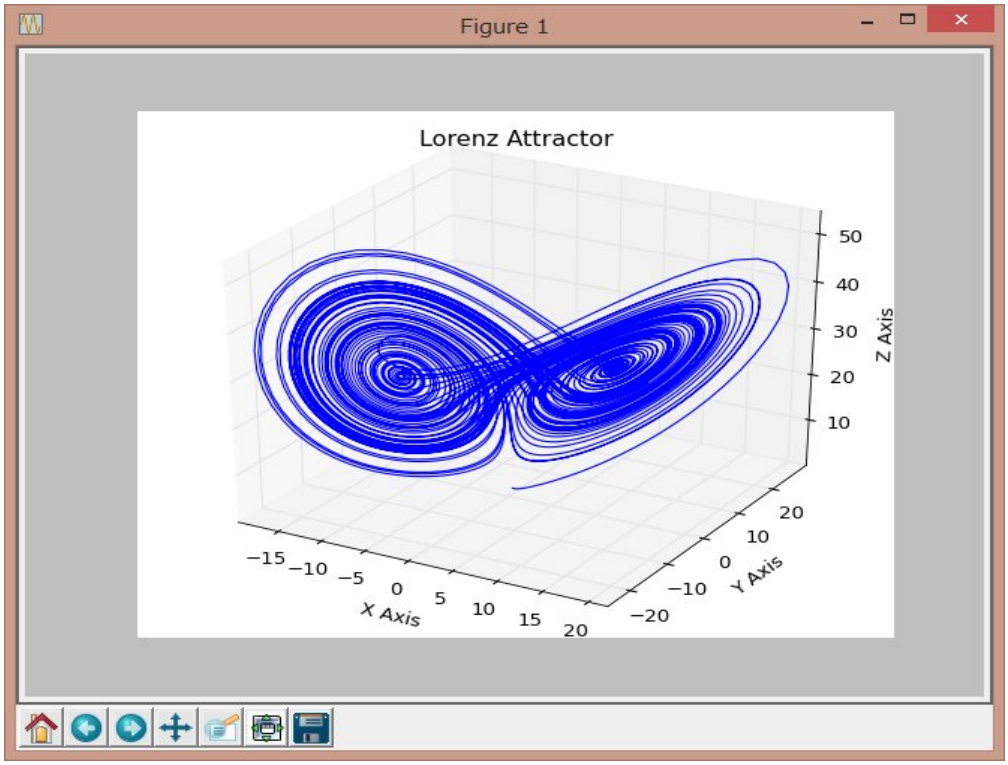
を描きます。

常微分方程式を数値的に解いて、解を三次元グラフで表示してみます。

```
import scipy.integrate
import numpy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

class LorenzAtractor(object):
    def __init__(self,p,r,b,t,x,y,z):
        self._p=p
        self._r=r
        self._b=b
        self._t=t
        self._x=x
        self._y=y
        self._z=z

    def Func(self,result,t0):
        F_dx=lambda result: -self._p*result[0]+self._p*result[1]
        F_dy=lambda result: -result[0]*result[2]+self._r*result[0]-result[1]
        F_dz=lambda result: result[0]*result[1]-self._b*result[2]
        return [F_dx(result),F_dy(result),F_dz(result)]
```

```
    def calc(self):
        initval=[self._x,self._y,self._z]
        result=scipy.integrate.odeint(self.Func,initval,self._t)
        self.show(result[:].T[0],result[:].T[1],result[:].T[2])

    def show(self,x,y,z):
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.plot(x, y, z)
        ax.set_xlabel("X Axis")
        ax.set_ylabel("Y Axis")
        ax.set_zlabel("Z Axis")
        ax.set_title("Lorenz Attractor")
        plt.show()
p=10.0
r=28.0
b=8/3.0
x=0.1
y=0.1
z=0.1
t=numpy.arange(0,100,0.01)
LorenzAtractor(p,r,b,t,x,y,z).calc()
```

のプログラムを実行すれば



を描きます。グラフはマウスで回転することが出来ます。これは、有名な常微分方程式

$$\frac{dx}{dt} = -10x + 10y$$

$$\frac{dy}{dt} = 28x - y - xz$$

$$\frac{dz}{dt} = -8/3 - z + xy$$

を解いたものです。Sage や Python はフリーソフトですから、書籍の情報は少ないですが、インターネットで調べれば、色々面白いことが出来ます。

また、Pygame や Pyopengl を導入すれば、アニメーションや所謂ゲームが作れます。興味があれば、インターネットで調べて下さい。

コンピュータの世界は、いかに楽をするかの戦いの歴史です。コンピュータのソフトも作り方を知らなくても使えますが、プログラミングの方法やアルゴリズムを知っていて、ソフトを自由自在に作れる人がいなくなれば、コンピュータが作り上げた文明も消滅します。子供達に聞かれれば、この授業で述べたくらいの簡単な仕組みを答えられるようになっていることが教師になる人達には必要なことだと思います。コンピュータの勉強は外国語の勉強と同じで、自分の頭で沢山のプログラムを組み立て、実際にコンピュータに打ち込んで実行してみて、発見したバグを繰り返し直し、あらゆる失敗を経験することが必要です。