Python for Science and Engg: Solving Equations & ODEs

FOSSEE

Department of Aerospace Engineering
IIT Bombay

26 September, 2010
Day 2, Session 1

Outline

1. Solving linear equations
   - Exercises
2. Finding Roots
3. ODEs

Solution of equations

Consider,

$$3x + 2y - z = 1$$
$$2x - 2y + 4z = -2$$
$$-x + \frac{1}{2}y - z = 0$$

Solution:

$$x = 1$$
$$y = -2$$
$$z = -2$$

Solving using Matrices

Let us now look at how to solve this using **matrices**

```
In []: A = array([[3,2,-1],
                 [2,-2,4],
                 [-1, 0.5, -1]])
In []: b = array([1, -2, 0])
In []: x = solve(A, b)
```

Solution:

```
In []: x
Out[]: array([ 1., -2., -2.])
```

Let's check!

```
In []: Ax = dot(A, x)
In []: Ax
Out[]: array([  1.00000000e+00,  -2.00000000e+00,
-1.11022302e-16])
```

The last term in the matrix is actually 0!
We can use `allclose()` to check.

```
In []: allclose(Ax, b)
Out[]: True
```

15 m

Outline

1. Solving linear equations
   - Exercises
2. Finding Roots
3. ODEs

Problem

Solve the set of equations:

$$\begin{aligned} x + y + 2z - w &= 3 \\ 2x + 5y - z - 9w &= -3 \\ 2x + y - z + 3w &= -11 \\ x - 3y + 2z + 7w &= -5 \end{aligned}$$

25 m

Solution

Use `solve()`

$$x = -5$$
$$y = 2$$
$$z = 3$$
$$w = 0$$

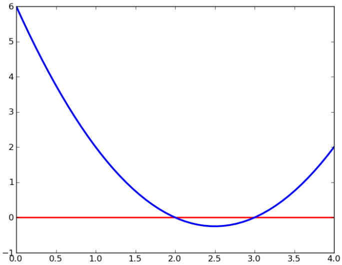
Outline

1. Solving linear equations
   - Exercises
2. Finding Roots
3. ODEs

Scipy Methods - `roots`

- Calculates the roots of polynomials
- To calculate the roots of $x^2 - 5x + 6$

```
In []: coeffs = [1, -5, 6]
In []: roots(coeffs)
Out[]: array([3., 2.])
```



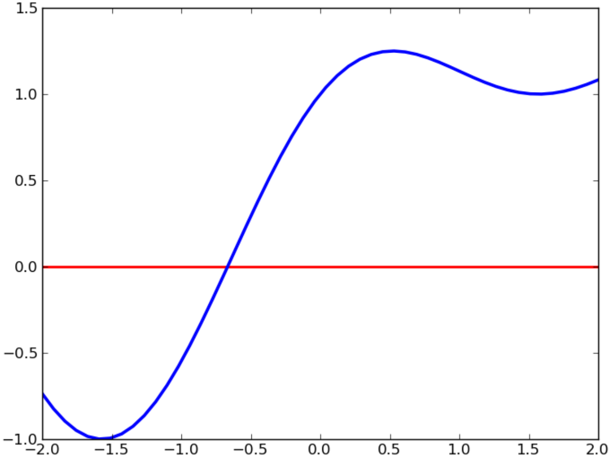
Scipy Methods - `fsolve`

```
In []: from scipy.optimize import fsolve
```

- Finds the roots of a system of non-linear equations
- Input arguments - Function and initial estimate
- Returns the solution

fsolve

Find the root of $sin(z) + cos^2(z)$ nearest to 0



fsolve

Root of $sin(z) + cos^2(z)$ nearest to 0

```
In []: fsolve(sin(z)+cos(z)*cos(z), 0)
NameError: name 'z' is not defined
```

fsolve

```
In []: z = linspace(-pi, pi)
In []: fsolve(sin(z)+cos(z)*cos(z), 0)
```

```
TypeError:
'numpy.ndarray'object is not callable
```

Functions - Definition

We have been using them all along. Now let's see how to define them.

```
In []: def g(z):
 ....:     return sin(z)+cos(z)*cos(z)
```

- `def`
- name
- arguments
- `return`

Functions - Calling them

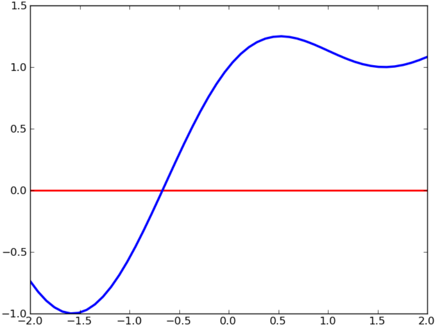
```
In []: g()
---------------------------------------
TypeError:g() takes exactly 1 argument
(0 given)
In []: g(0)
Out[]: 1.0
In []: g(1)
Out[]: 1.1333975665343254
```

More on Functions later . . .

`fsolve` ...

Find the root of $sin(z) + cos^2(z)$ nearest to 0

```
In []: fsolve(g, 0)
Out[]: -0.66623943249251527
```

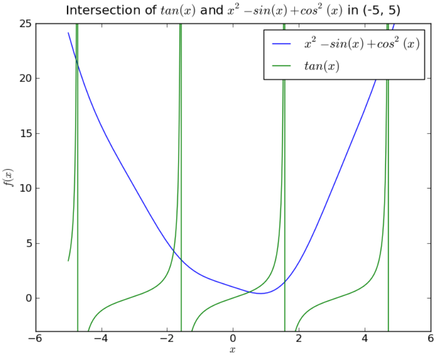


Exercise Problem

Find the root of the equation
$x^2 - sin(x) + cos^2(x) = tan(x)$ nearest to 0

Solution

```
def g(x):
    return x**2 - sin(x) + cos(x)*cos(x) - tan(x)
fsolve(g, 0)
```



Outline

1. Solving linear equations
   - Exercises
2. Finding Roots
3. ODEs

Solving ODEs using SciPy

- Let's consider the spread of an epidemic in a population
- $\frac{dy}{dt} = ky(L-y)$ gives the spread of the disease
- L is the total population.
- Use L = 250000, k = 0.00003, y(0) = 250
- Define a function as below

```
In []: from scipy.integrate import odeint
In []: def epid(y, t):
  ....     k = 0.00003
  ....     L = 250000
  ....     return k*y*(L-y)
  ....
```

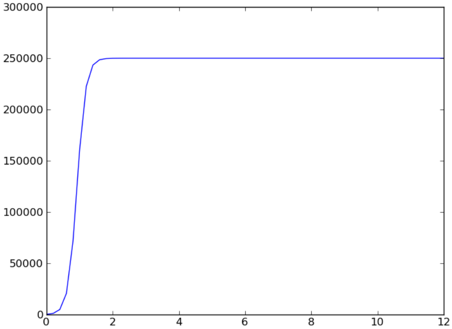
Solving ODEs using SciPy . . .

```
In []: t = linspace(0, 12, 61)

In []: y = odeint(epid, 250, t)

In []: plot(t, y)
```

Result



ODEs - Simple Pendulum

We shall use the simple ODE of a simple pendulum.

$$\ddot{\theta} = -\frac{g}{L} sin(\theta)$$

- This equation can be written as a system of two first order ODEs

$$\dot{\theta} = \omega \tag{1}$$

$$\dot{\omega} = -\frac{g}{L} sin(\theta) \tag{2}$$

$$\text{At } t = 0 :$$

$$\theta = \theta_0(10^o) \quad \& \quad \omega = 0 \ (\textit{Initial values})$$

ODEs - Simple Pendulum . . .

- Use `odeint` to do the integration

```
In []: def pend_int(initial, t):
  ....         theta = initial[0]
  ....         omega = initial[1]
  ....         g = 9.81
  ....         L = 0.2
  ....         F=[omega, -(g/L)*sin(theta)]
  ....         return F
  ....
```

ODEs - Simple Pendulum . . .

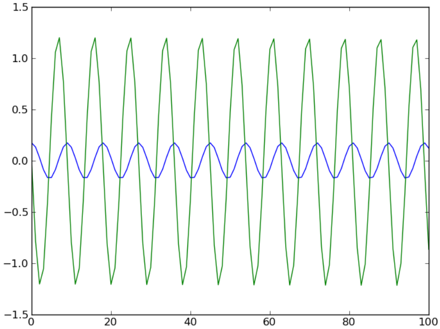
- `t` is the time variable
- `initial` has the initial values

```
In []: t = linspace(0, 20, 101)
In []: initial = [10*2*pi/360, 0]
```

ODEs - Simple Pendulum . . .

```
In []: from scipy.integrate import odeint
In []: pend_sol = odeint(pend_int,
                         initial,t)
```

Result



Things we have learned

- Solving Linear Equations
- Defining Functions
- Finding Roots
- Solving ODEs