Lecture 15 – Scipy Root Finding

* Quiz / Prayer

I. Scipy Root Finding Tools

II. Solving Engineering Problems

I. Scipy RootFinding

Scipy – Scientific Python

↳ optimize

```
import scipy
import scipy.optimize as opt

x_soln = opt.root(f, x_guess).x
```

- `x_soln` ↑ solution
- `opt.root` ↑ function call to scipy
- `f` ↑ function I will define
- `x_guess` ↑ guess
- `.x` ↑ return

root → dictionary

x: [~]

or: [~]

⋮

: [ ]

- comments:
  - The function f needs to be defined using def.

Good:

```
def f(x):
    return x**2
```

Bad:

```
f = x**2
```

  - The function f can be a vector function
    ↳ multiple equations / unknowns

Good:

```
def f(x):
    z = x[0]
    y = x[1]
    eq0 = z*y
    eq1 = y**z
    return np.array([eq0, eq1])
```

Bad

```
def f(x,y):
    return x**y, y+x
```

Activity

II. Solving Engineering NLEs

* Solve NLE in Engineering we have additional issues to work with.

(0) $\frac{\Delta P}{\rho g} = \frac{-v^2}{2g}\left(\frac{4Lf}{D}\right)$ ← pressure loss

(1) $\frac{1}{\sqrt{f}} = 4\log_{10}\left(Re\sqrt{f}\right) - 0.4$ ← Reynold's #

(2) $Re = \frac{\rho v D}{\mu}$

knowns: $\rho, g, \mu, v, L, \Delta P$ ← know speed, pressure loss

unknowns: Re, D, f solve for D

(1) Units

- Be consistent → Eng. / SI
- dimensionless

(2) Converting to vector notation

(residual form $\underline{f}(\underline{x}) = 0$)

$$f_0 = 0 = \frac{\Delta P}{\rho g} + \frac{v^2}{2g}\left(\frac{4x_1}{x_0}\right) \qquad x_0 = D/L, \quad x_1 = f$$

$$f_1 = 0 = \frac{1}{\sqrt{x_1}} - 4\log_{10}\left(x_2\sqrt{x_1}\right) + 0.4 \qquad x_2 = Re$$

$$f_2 = 0 = x_2 - \frac{\rho v D}{\mu}\cdot\frac{L}{L}$$

$$= x_2 - \frac{\rho v L}{\mu}\cdot x_0$$

$$\underline{f}(\underline{x}) = \begin{bmatrix} \frac{\Delta P}{\rho g} + \frac{v^2}{2g}\left(\frac{4x_1}{x_0}\right) \\ \frac{1}{\sqrt{x_1}} - 4\log_{10}\left(x_2\sqrt{x_1}\right) + 0.4 \\ x_2 - \frac{\rho v L}{\mu} x_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

det $\underline{f}(\underline{x})$:

$x_0 = x$ [0]

⋮

(3) Getting root to converge

- Good guess → plot
  - ↳ ✱ Physical Intuition
    - D: a few inches or less (1 in., 0.25 in)
    - f: $\frac{16}{Re}$ / 10^{-3}
- even w/ a good guess → doesn't always converge.

- Try substituting → reduce # Eq's & unknowns
- Get rid of exp, log ← Thermo
- Can provide an analytical Jacobian

(root uses numerical Jacobian)

```
x = opt.root(f, x_guess, jac=J).x

def J(x):
    ⋮
    return np.array([[ ∂f0/∂x0  ...  ∂f0/∂x_{n-1} ],
                     [      ⋮                     ],
                     [ ∂f_{n-1}/∂x0  ...  ∂f_{n-1}/∂x_{n-1} ]])
```