Nonlinear Programming

Introduction

Nonlinear problems appear everywhere in engineering and science. They often involve equations or optimization problems that cannot be expressed as simple linear forms. Solving nonlinear equations is essential for modeling, simulations, and design. This requires numerical methods, since closed-form solutions rarely exist.

Key Concepts

- Nonlinear Equations
- Random Bracketing
- Fixed Point Iteration
- Newtons Method
- MATLAB fsolve

Definitions

- **Nonlinear Equation:** An equation where the unknown variable(s) appear with powers, products, exponentials, logs, or other nonlinear operations. Example: $x^3 - 2x + e^{-x} = 0$. Note that you can determine if an equation is non-linear using the following rule:

$$f(ax + by) \neq af(x) + bf(y)$$

- **Root:** The value of x where $f(x) = 0$

Bracketing Methods

Bracketing methods are based on enclosing the root between two guesses a and b where $f(a)$ and $f(b)$ have opposite signs. You want to pick a point on either side of your root, and slowing iterate and move those points closer and closer. When the points are so close, you know you found your root.

You take the following steps:

- **Guess** Choose some guesses for a and b
- **Find your point** If your gap comes within a tolerance, terminate the iteration
- **Determine** If $\text{sign}(f(a_i)) = \text{sign}(f(x_i))$, then the root must be on the other side, so you want to reflect that:

$$a_{i+1} = x_i \; and \; b_{i+1} = b_i$$

But if that isn't true, your root is between a_i and x_i"

$$a_{i+1} = a_i \; and \; b_{i+1} = x_i$$

Bracketing methods are *slow but guaranteed* to converge if a root exists.

Fixed-Point Iteration

The idea is to rearrange the nonlinear equation $e^{-x} - x = 0$ into the simpler form:

$$x = e^{-x}$$

And so we can say:

$$g(x) = e^{-x}$$

Then, we apply:

$$x_{k+1} = g(x_k) = e^{-x_k}$$

Which will look like (assume initial guess $x_0 = 0.5$):

$$x_0 = 0.5$$
$$x_1 = e^{-.5} \approx 0.6065$$
$$x_2 = e^{-.6065} \approx 0.5462$$
$$\vdots$$
$$x_7 = e^{-.5649} \approx 0.5684$$

Convergence depends on the derivative of $g(x)$ (You want to walk a smooth path when iterating). The program may or may not converge, but if it does converge it will converge linearly.

Newton Method

The Newtons method is one of the fastest root-finding methods (quadratic convergence near the root). It uses the tangent line at the current guess to approximate the next guess.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

This method requires:

- An initial guess x_0,
- The function $f(x)$,
- Its derivative $f'(x)$.

Say we have a system of equations. Since systems can essentially be written as matrices, we need to introduce a system of our derivatives, called the Jacobian:

$$f(x) = \begin{pmatrix} f_1(x_1, x_2) & f_1(x_1, x_2) \\ f_2(x_1, x_2) & f_2(x_1, x_2) \end{pmatrix}$$
$$J(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix}$$

MATLAB: `fsolve`

MATLAB provides `fsolve` to numerically solve nonlinear equations or systems:

```
x = fsolve(@fun, x0)
```

where:

- `fun` is a function handle for $f(x)$,
- `x0` is the initial guess,
- `x` is the computed root.

`fsolve` internally uses Newton-like methods with numerical Jacobians. It is powerful for multi-variable nonlinear systems but depends heavily on good initial guesses.

Recap

- Nonlinear equations rarely have closed-form solutions.
- Bracketing methods are slow but reliable.
- Fixed-point iteration is simple but conditional on convergence.
- Newton–Raphson is fast but risky without good initial guesses.
- MATLAB's `fsolve` is the go-to tool for real-world engineering nonlinear systems.