Lecture 4
Error Plotting, Recursion, Newton-Raphson Iteration

Summary of Last Lecture

Bisection

- Evaluate function only.
- Need to find a and b such that f is continuous and has unique zero on $[a, b]$.
- Upper bound for error: $|x^{(k)} \leq \epsilon^{(k)} = |b^{(k)} - a^{(k)}|$.
- Decreases as $\epsilon^{(k+1)} = \frac{\epsilon^{(k)}}{2}$...
- ...so that $\epsilon^{(k)} = \frac{\epsilon^{(0)}}{2^k}$, aka *linear* convergence.
- Straight line on semilog plot.
- Works only for one unknown. Adobe

Newton's Method/Secant Method

- Uses function and derivative/two initial points.
- Function must be continuously differentiable/continuous around x^*.
- Need one/two initial guesses.
- Error estimate: $\epsilon^{(k+1)} \approx |\delta x^{(k)}|$.
- Decreases as $\epsilon^{(k+1)} \approx (\epsilon^{(k)})^{2^k}$...
- ... so that $\epsilon^{(k)} \approx (\epsilon^{(0)})^{2^k}$, aka *quadratic* convergence.
- Steeper than linear on semilog plot.

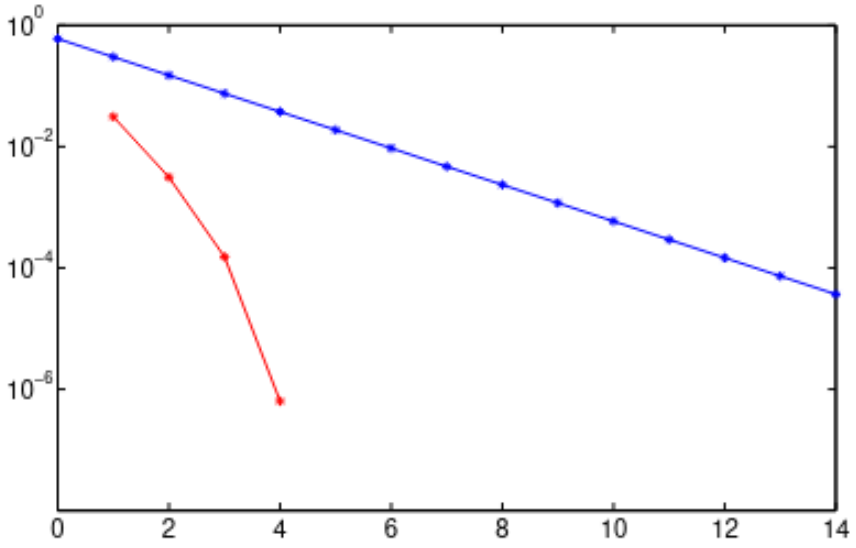

Figure 1:

Error Plotting of Newton's vs. Secant Method

Recursion

Simple example of recursive programming:

```
def bisection(a, b, k_max, eps_x, eps_f):
        mid = (a + b) / 2
        x = mid
        err = abs(a - b) / 2\]
        fm = func(mid)
        res = abs(fmid)
        if k_max > 0:
                if err > eps_x or res > eps_f:
                        fa = func(a)
                        if fm * fa > 0:
                                a = mid
                        else:
                                b = mid
                        x, err, res = bisection(a, b, k_max - 1, eps_x, eps_f)
        else:
                print("Warning: no convergence...")
        return x, err, res
```

Pros:

- clean, concise, elegant code;
- conceptually clear.

Cons:

- code harder to follow;
- uses more memory;
- small mistake can give bad crash!

Newton-Raphson Iteration

Remark 1 Newton iteration can be generalized to n equations with n unknowns.

Alternative derivation in 1D:

$$f(x+\delta x) \approx f(x) + f'(x)\delta x = 0 \implies \delta x = -\frac{f(x)}{f'(x)}$$

Now in 2D. We want to find x_1 and x_2 such that:

$$f_1(x_1, x_2) = 0$$

$$f_2(x_1, x_2) = 0$$

Note that in general, we need **the same number of equations and unknowns** to find (isolated) solutions.

$$f_1(x_1+\delta x_1, x_2+\delta x_2) \approx f_1(x_1, x_2) + \frac{\partial f_1}{\partial x_1}(x_1, x_2)\delta x_1 + \frac{\partial f_1}{\partial x_2}(x_1, x_2)\delta x_2$$

$$f_2(x_1+\delta x_1, x_2+\delta x_2) \approx f_1(x_1, x_2) + \frac{\partial f_2}{\partial x_1}(x_1, x_2)\delta x_1 + \frac{\partial f_2}{\partial x_2}(x_1, x_2)\delta x_2$$

which gives:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x_1, x_2) & \frac{\partial f_1}{\partial x_2}(x_1, x_2) \\ \frac{\partial f_2}{\partial x_1}(x_1, x_2) & \frac{\partial f_2}{\partial x_2}(x_1, x_2) \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} = - \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}$$