

C: Solving equations

Very frequent task in numerical analysis:

Given a function $f(x)$

Find an x^* such that $f(x^*) = 0$

Some examples:

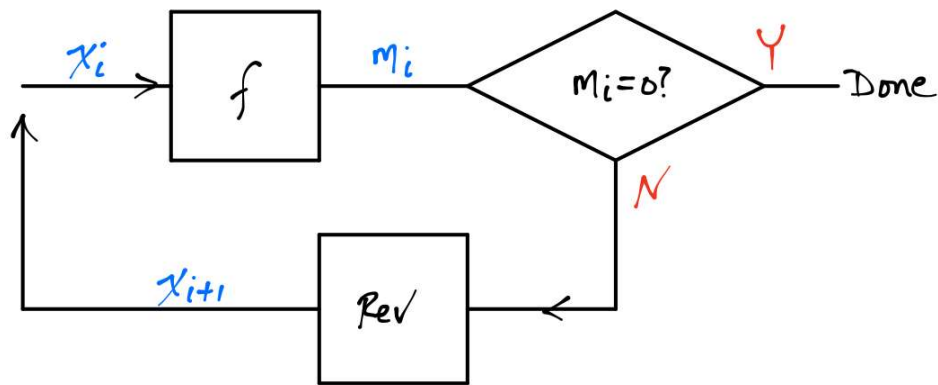
- IRR calculation:
Find an r^* such that $NPV(r^*) = 0$
- Market equilibrium:
Find a price p^* such that $Q^D(p^*) - Q^S(p^*) = 0$
- Carbon tax t^* to reduce emissions to a target level
- Clean energy credit c^* that causes renewable electricity to hit 50%
- Low income tax credit z^* to eliminate regressivity of a tax
- ... many, many, many more

Core technique is Newton's Method:

Iterative process:

1. At trial i guess x_i
2. Evaluate $f(x_i)$ at the guess to find the *miss distance*: $f(x_i) = m_i$
3. If m_i isn't 0, revise the guess to x_{i+1} and try again

Schematically:

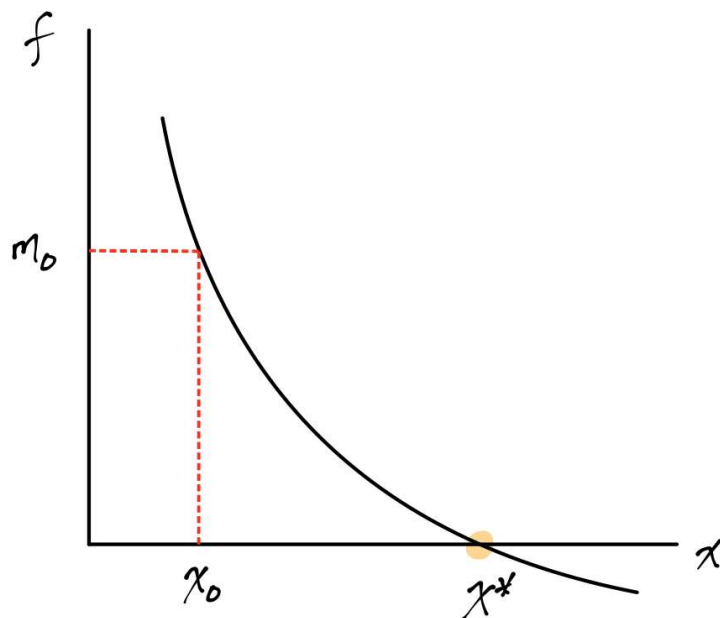


The heart of Newton's Method is the revision procedure:

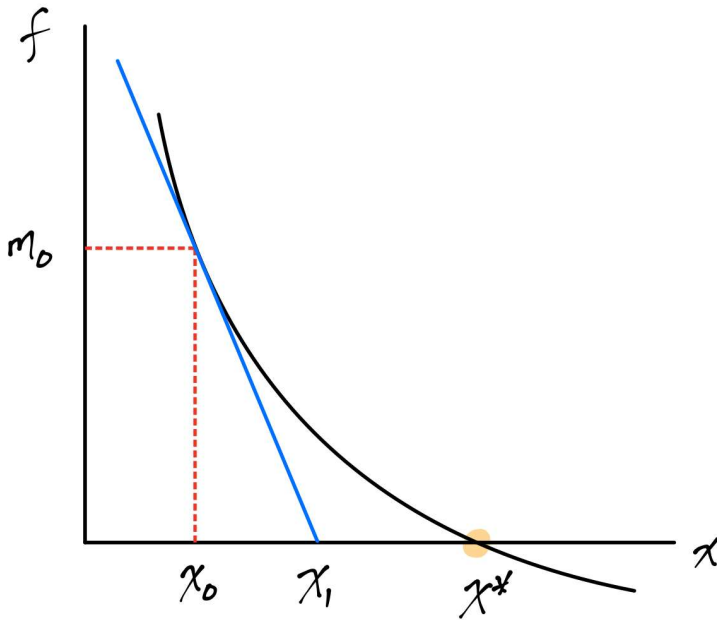
- Uses the slope of the function at x_i
- Very fast in a wide range of cases

Graphically:

Initial guess and miss distance:



Using the slope of the function to find a new guess:



Why try this?

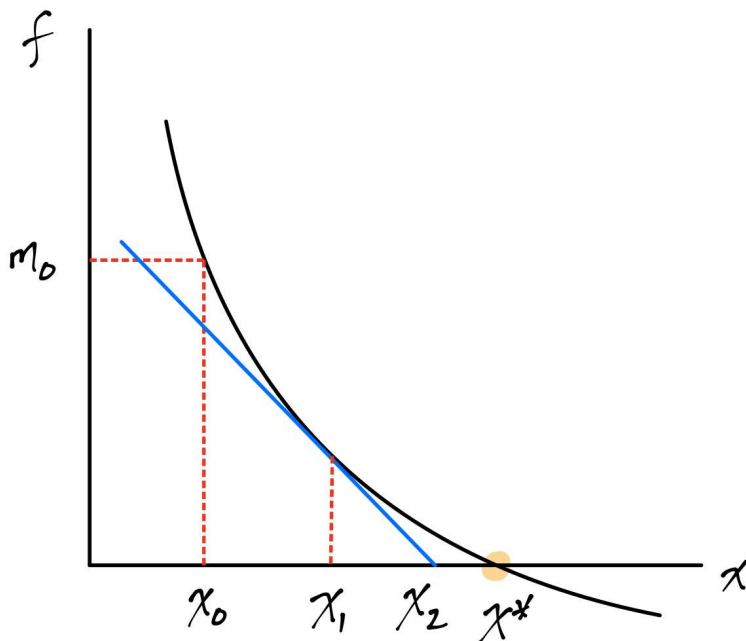
Hits x^* on the first try if f is linear

Still makes a big improvement for many other functions.

Algebraically, if the slope at x_0 is s_0 , can show:

$$x_1 = x_0 - \frac{m_0}{s_0}$$

Updating x , m and s and generating a new guess:



Next guess:

$$x_2 = x_1 - \frac{m_1}{s_1}$$

Repeat until $f(x_i) = 0$

Outline of invoking Newton in Python:

1. Import the **scipy** module
2. Define a **miss distance** function
3. Create a **starting guess**
4. **Call the** Newton's Method function in scipy; will return x^*

Sketching the code:

```
1 import scipy.optimize as opt
```

```
2 def miss_func( guess:float, ... ) -> float:  
    ...  
    return miss
```

```
3 start_guess = ...
```

```
4 sol = opt.newton( miss_func, start_guess, ... )
```

Variable sol will be the solution x^*

Minor **complication**: passing **additional variables** to the miss function

- The **miss** function usually takes *several arguments*:
Example: npv(r, cashflow)

- `opt.newton()` iterates over guesses of the **first argument**

Example: `r`

- To pass **additional arguments** give them as a **LIST** via optional parameter **args**:

`opt.newton(..., args=[cashflow])`

- Additional arguments are not altered by `opt.newton()`

g07 demo.py