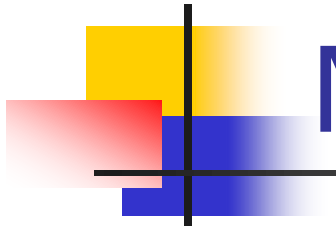
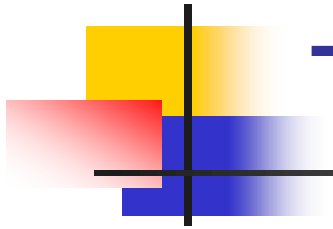
Computer Methods (MAE 3403)

Root finding



Motivation

- Find the solutions to f(x)=0 (i.e., roots of f(x) =0), where the function f is given.
- For example, quadratic functions have a formula
- Without a formula, e.g., f(x) = cos(x) – x, we use iterative procedures to find roots:
  - Start with an initial guess (important)
  - Iteratively refine the guess



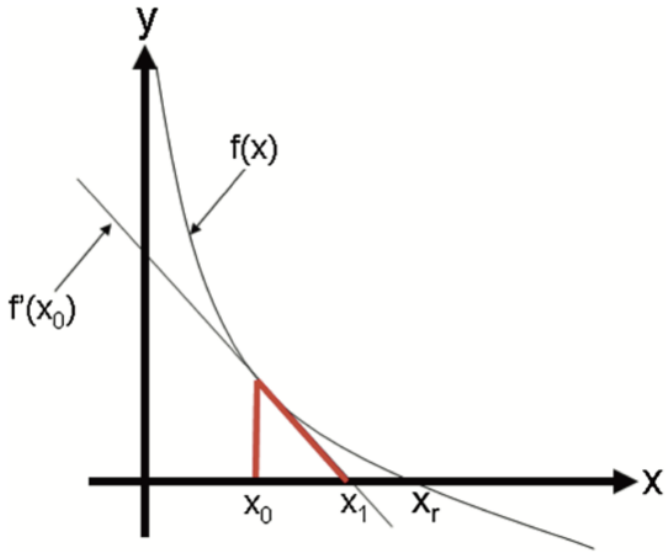
Tolerance

- Tolerance: a level of error acceptable for an engineering application
- Iterative computation of roots: numerical convergence implies a certain level of tolerance of errors
- Possible choices: $|f(x)|$ is small enough, $|x_{i+1} - x_i|$ is small enough
  - Small enough: less than a small threshold e.

Newton-Raphson Method

- Best known method for finding roots: simple, fast
- Makes use of the derivative f'(x) and f(x)
- Iterative update of the estimate of a root as

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}.$$

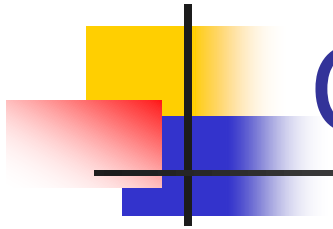


Algorithm

Let x be an estimate of the root of $f(x) = 0$.
Do until $|\Delta x| < \varepsilon$:
    Compute $\Delta x = -f(x)/f'(x)$.
    Let $x \leftarrow x + \Delta x$.

- One iteration:

```
import numpy as np
f = lambda x: x**2 - 2
f_prime = lambda x: 2*x
x0 = 1.4
newton_raphson = x0 - (f(x0))/(f_prime(x0))
print("newton_raphson =", newton_raphson)
print("sqrt(2) =", np.sqrt(2))
```



Coding

Can you write a function myNewton(f, df, x0, tol) where the output is an estimate of the root of f, f is a function object, df is a function object to f', x0 is an initial guess and tol is the error tolerance on |f(x)| (i.e., when |f(x)|<tol, convergence is achieved) ?

You can also add an argument "maxiter"

Secant method

- Replace the derivative f'(x) with an approximation
- Step 1: initialization of x_0 and x_1
- Step 2: for n = 1, 2, 3, …
- $x_{n+1} = x_n - f(x_n) * (x_n - x_{n-1})/(f(x_n) - f(x_{n-1}))$
- Repeat until $|x_{n+1} - x_n|$ is small

Try these examples with Newton and Secant method

- Try $f(x) = x^3 + 3x^2 - 2x - 5$ with an initial guess $x = 0.29$
  - derivative is close to zero
- Try $f(x) = x^3 - 100x^2 - x + 100$ with an initial guess $x = 0$
  - What's the obtained root?

Alternative in Python

- Existing function that performs root-finding
- fsolve from scipy.optimize

```
from scipy.optimize import fsolve
f = lambda x: x**3-100*x**2-x+100
xsol = fsolve(f, [2, 80])
```

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html

scipy.optimize.

fsolve

fsolve(*func*, *x0*, *args=()*, *fprime=None*, *full_output=0*, *col_deriv=0*, *xtol=1.49012e-08*, *maxfev=0*, *band=None*, *epsfcn=None*, *factor=100*, *diag=None*)

[source]

Find the roots of a function.

Return the roots of the (non-linear) equations defined by `func(x) = 0` given a starting estimate.

Parameters:

func : *callable* `f(x, *args)`

A function that takes at least one (possibly vector) argument, and returns a value of the same length.

x0 : *ndarray*

The starting estimate for the roots of `func(x) = 0`.

args : *tuple, optional*

Any extra arguments to *func*.

fprime : *callable* `f(x, *args)`, *optional*

A function to compute the Jacobian of *func* with derivatives across the rows. By default, the Jacobian will be estimated.

- See examples