Newton Raphson method

Method to find the root of f(x), i.e. x s.t. f(x)=0.

Method works if:

f(x) and derivative f '(x) can be easily calculated.

A good initial guess x_0 for the root is available.

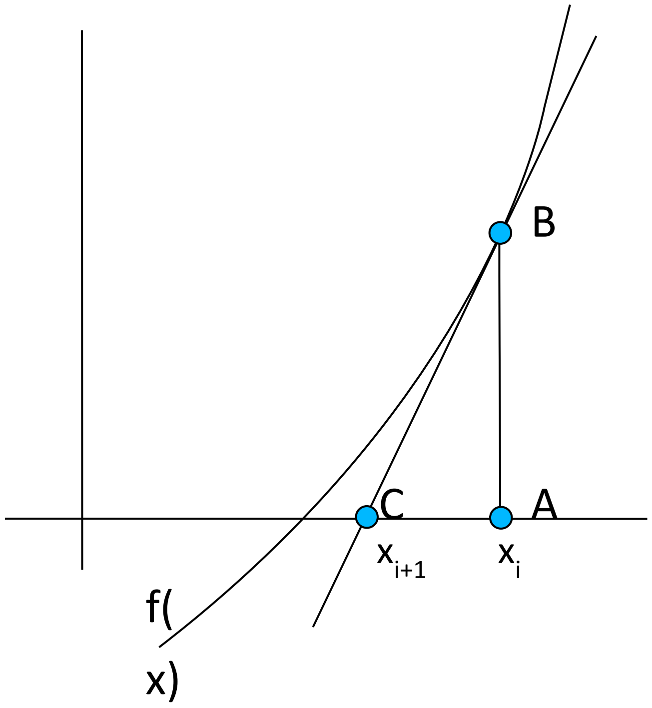
Example: To find square root of y.

use $f(x) = x^2 - y$. $f'(x) = 2x$.

f(x), f ' (x) can be calculated easily. 2,3 arithmetic ops.

Initial guess $x_0 = 1$ is good enough!

How to get better x_{i+1} given x_i



Point A =$(x_i,0)$ known.

Calculate $f(x_i)$.

Point B=$(x_i,f(x_i))$

Approximate f by tangent
C= intercept on x axis
C=$(x_{i+1},0)$

$$x_{i+1} = x_i - AC = x_i - AB/(AB/AC) = x_i - f(x_i) / f'(x_i)$$

Square root of y

$x_{i+1} = x_i - f(x_i) / f'(x_i)$

$f(x) = x^2 - y, \quad f'(x) = 2x$

$x_{i+1} = x_i - (x_i^2 - y)/(2x_i)$

$= (x_i + y/x_i)/2$

Starting with $x_0=1$, we compute x_1, then x_2, …

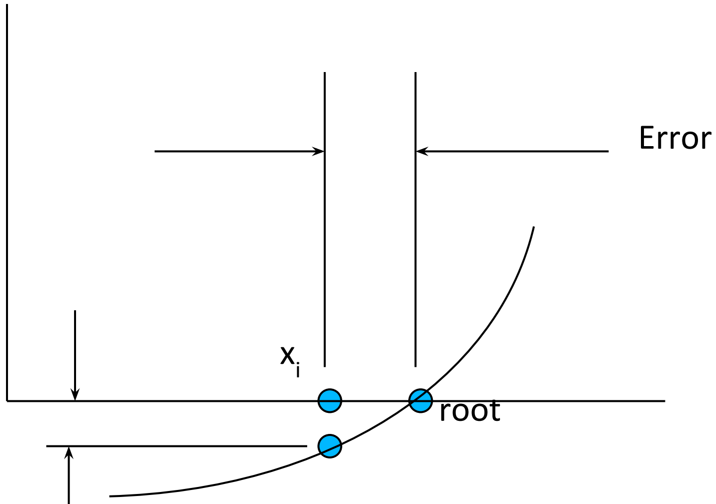
We can get as close to sqrt(y) as required.

Proof not part of the course.

Code

```
double y;  cin >> y;
double xi=1;      // Guess. Known to work.
repeat(10){
   xi = (xi + y/xi)/2;
}
cout << xi << endl;
```

How to iterate until error is small



Error Estimate = $|f(x_i)| = |x_i * x_i - y|$

Make $|x_i*x_i - y|$ small

```
double y; cin >> y;
double xi=1;
while(abs(xi*xi – y) > 0.001){
   xi = (xi + y/xi)/2 ;
}
cout << xi << endl;
```

Error Analysis

- Number of correct bits double with each iteration!
- Proof not in course.

Remarks

- Babylonians used this method to find square roots 3500 years ago!
- Newton's method is very commonly used.
- Also useful in multiple dimensions. Given functions f, g, h, … Find x, y, z, w, ... such that
  - f(x, y, z, w, …) = 0
  - g(x, y, z, w, …) = 0
  - h(x, y, z, w, …) = 0.
  - . . .

–But it is trickier too (Chapter 29).

Concluding Remarks

- If you want to find f(x), then
  - use Taylor series for f, if f and its derivatives can be evaluated at some point x_0 close to x.
  - Express f as an integral of some easily evaluable function g, and use numerical integration
  - Express f as the root of some easily evaluable function g, and use bisection or Newton-Raphson.
- All the methods are iterative, i.e. the accuracy of the answer improves with each iteration.
- The ideas of this chapter are very fundamental, and will appear later also, e.g. Chapter 19, Chapter 29.

♞♞