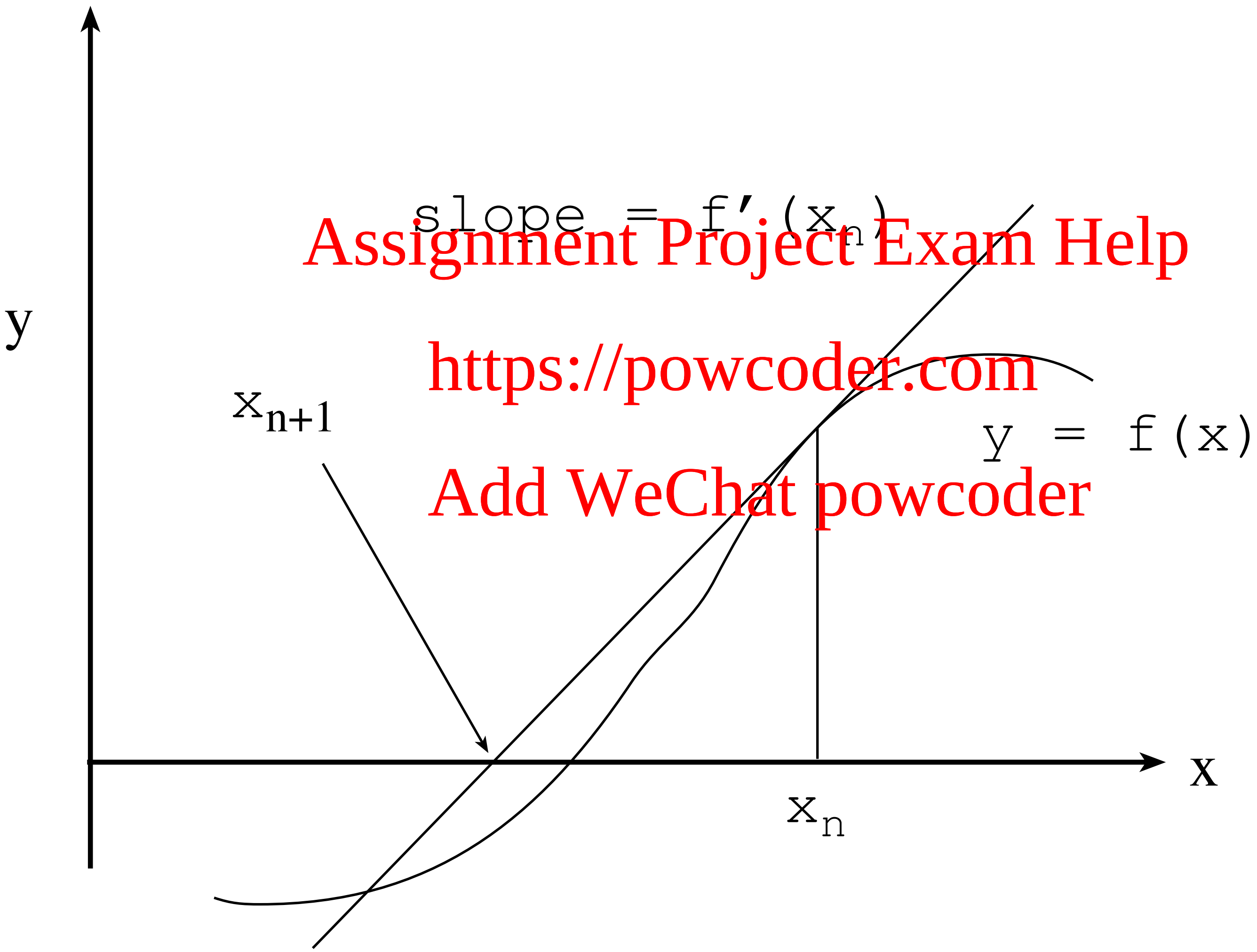
Newton's method



based on **slope** of function

Derivation

Taylor series around x_n:

$$f(x_{n+1}) \approx f(x_n) + f'(x_n)(x_{n+1} - x_n) = 0$$

which gives

Newton-Raphson iteration:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Again, a first order recurrence relation.
$\implies$ we expect problems if $f'(x_n) = 0$

Simple roots

Definition

f has a **simple root** at x^* if $f(x^*) = 0, f'(x^*) \neq 0$.

Definition

f has a **double root** at x^* if $f(x^*) = 0, f'(x^*) = 0, f''(x^*) \neq 0$.

$\implies$ Newton's method has trouble at a double root

Example

Convergence rates

At a simple root:

$$e_{n+1} \sim Ce_n^2$$

quadratic convergence

At a double root:

$$e_{n+1} \sim Ce_n$$

linear convergence

Definition

Order of convergence If $e_{n+1} \sim Ce_n^p \implies$ order of convergence $=$ p

Explanation by Taylor series ...

Can derive this more rigorously.

Linear convergence

Write $e_n = 10^{-b_n}$

b_n = number of correct decimal digits
Linear Convergence $\Longrightarrow$

$$e_{n+1} \approx C e_n \quad C < 1$$

$$\Longrightarrow \; b_{n+1} \approx b_n - \log_{10} C$$

$\rightarrow$ number of correct digits increases by a constant

Quadratic Convergence

$$\implies \quad e_{n+1} \approx C e_n^2$$

$$\implies b_{n+1} \approx 2b_n - \log_{10} C$$

$\rightarrow$ number of correct digits doubles each iteration!!

$\implies$ Quadratic Convergence **very desirable**

Problems with Newton's method

- $f'(x)$ may be hard to calculate
- it may not converge :-(

Theorem

If $f''(x)$ is C^1 then $\exists \delta > 0$ such that $\forall x_0 \in [x^* - \delta, x^* + \delta]$, Newton's method converges to the root x^*

$\Longrightarrow$ converges provided we start close enough

BUT

- don't know x^*
- don't know δ

$\Longrightarrow$ get close with a globally convergent method, then switch to a fast method

$\rightarrow$ **hybrid method**

Convex functions

Note: for some functions, Newton's method is guaranteed to converge

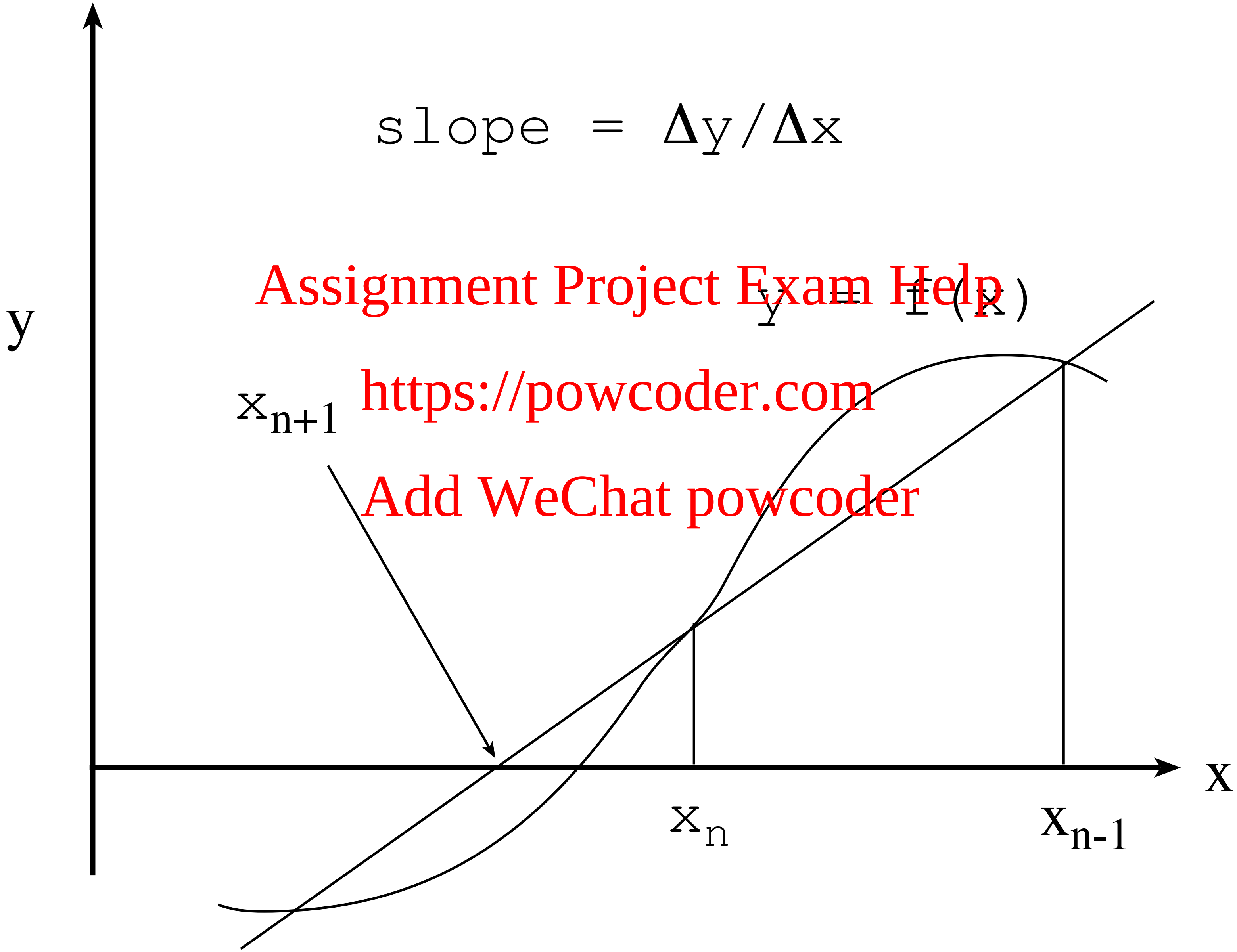
Example

Secant method

What if the derivative is really nasty? $\implies$ try the secant method instead, which **doesn't need the derivative function**.

The secant method.

For this method, use **two current "guesses"** at a root of $f(x) = 0$.
Want to replace the pair x_0, x_1 by the pair x_1, x_2 where x_2 is a new guess.
Use the straight line L from $(x_0, f(x_0))$ to $(x_1, f(x_1))$ (a **secant** of the curve $y = f(x)$) to give the slope.


slope = Δy/Δx
y = f(x)
y
x_{n+1}
x
x_n
x_{n-1}

Secant method algorithm

Just replace $f'(x_n)$ from NR method by

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

which gives

Secant method:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}.$$

Since each new guess depends on the previous two guesses $\implies$ a second order recurrence relation.

Properties

The secant method:

- quite efficient: converges quickly towards a root :-)
- must start with two guesses :-(
- don't need derivative :-)

Order of convergence

Using Taylor series, $\rightarrow$ (at simple root)

$$e_{n+1} \sim C e_n e_{n-1}$$

$$\implies e_{n+1} \sim C e_n^p \text{ where } p = \frac{1+\sqrt{5}}{2} \approx 1.618$$

$\implies$ secant is **superlinearly convergent but not quadratically convergent**
(but only need 1 function evaluation/iteration; Newton needs 2)

Summary

fixed point	$e_{n+1} \sim Ce_n$	may diverge
bisection	$e_{n+1} \sim \frac{1}{2}e_n$	globally convergent
Newton	$e_{n+1} \sim Ce_n^2$	simple root
Newton	$e_{n+1} \sim \frac{1}{2}e_n$	double root
secant	$e_{n+1} \sim Ce_n^{1.618}$	simple root

Hybrid methods

Attractive to combine **global convergence** of bisection with **speed** of another method e.g. Newton or secant
→ hybrid methods that switch between basic methods such that interval is always bracketing

Example

MATLAB's `fzero`

- switches between bisection, secant and Inverse Quadratic Interpolation (faster than secant)
- needs no derivative

What about systems of nonlinear equations?

Try Newton's Method for system of 2 equations in 2 variables.

$$f(x,y) = 0 \quad g(x,y) = 0$$

Expand in Taylor series about current iterate to get tangent plane approximation:

$$\begin{aligned}
f(x_{n+1}, y_{n+1}) \approx f(x_n, y_n) + \frac{\partial f}{\partial x}\,|_n\,(x_{n+1} - x_n) + \frac{\partial f}{\partial y}\,|_n\,(y_{n+1} - y_n) &= 0 \\
g(x_{n+1}, y_{n+1}) \approx g(x_n, y_n) + \frac{\partial g}{\partial x}\,|_n\,(x_{n+1} - x_n) + \frac{\partial g}{\partial y}\,|_n\,(y_{n+1} - y_n) &= 0
\end{aligned}$$

or in matrix notation

$$\begin{bmatrix} f(x_n, y_n) \\ g(x_n, y_n) \end{bmatrix} + \begin{bmatrix} \frac{\partial f}{\partial x}\Big|_n & \frac{\partial f}{\partial y}\Big|_n \\ \frac{\partial g}{\partial x}\Big|_n & \frac{\partial g}{\partial y}\Big|_n \end{bmatrix} \begin{bmatrix} x_{n+1} - x_n \\ y_{n+1} - y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where partial derivs are evaluated at (x_n, y_n)
$\implies$ at each iteration, must solve a **linear system** of the form

$$\mathrm{J}\,\big|_n\,(\mathrm{x}_{n+1} - \mathrm{x}_n) = -\mathrm{f}\,\big|_n$$

$\implies$ must learn how to solve linear systems first!

End of Lecture 10