Session 3
Sep. 9, 2019

other types of annuities:

- annuity due : pays C at beginning of year

$$FV = \sum_{i=1}^{n} C(1+r)^i = C(1+r)\sum_{i=0}^{n-1}(1+r)^i = C(1+r)\left(\frac{(1+r)^n - 1}{r}\right)$$

- general annuity: m payments per year

↳ ordinary: $FV = \sum_{i=0}^{nm-1} C\left(1+\frac{r}{m}\right)^i = C\left(\frac{\left(1+\frac{r}{m}\right)^{nm} - 1}{\frac{r}{m}}\right)$

What is PV?

annuitydue: $PV = \sum_{i=1}^{m\cdot n} C\left(1+\frac{r}{m}\right)^{-i} = C\sum_{i=1}^{m\cdot n}\left(\frac{1}{1+\frac{r}{m}}\right)^i$

$$= C\left(\frac{1}{1+\frac{r}{m}}\right)\left(\frac{\left(1+\frac{r}{m}\right)^{-nm} - 1}{\left(\frac{1}{1+\frac{r}{m}}\right) - 1}\right)$$

$$= C\left(\frac{1-\left(1+\frac{r}{m}\right)^{-nm}}{\frac{r}{m}}\right)$$

- perpetual annuity: $n \to \infty$

$$\Rightarrow PV = \lim_{n\to\infty} C\left(\frac{1-\overbrace{\left(1+\frac{r}{m}\right)^{-n\cdot m}}^{\to 0}}{\frac{r}{m}}\right) = C\frac{m}{r}$$

Amortization:

→ repay loan with regular payments

↳ payments for principal (repay) + interest

traditional mortgage = equal regular payments

$$C = PV\left(\frac{\frac{r}{m}}{1-(1+\frac{r}{m})^{-n\cdot m}}\right)$$

remaining principal after k payments: $\sum_{i=1}^{m\cdot n-k} C(1+\frac{r}{m})^{-i}$

↳ HW: create an amortization schedule

Internal Rate of Return (IRR):

given n, C_i, P, the r that solves $PV(r) = \sum_{i=1}^{n} \frac{C_i}{(1+r)^i} = P$ is called IRR.

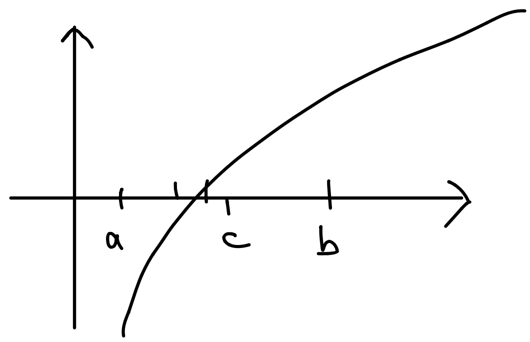
price of financial instrument

sometimes one defines the net-present value $NPV(r) = PV(r) - P$

$\Rightarrow$ IRR = zero of NPV

Root Finding Algorithms:

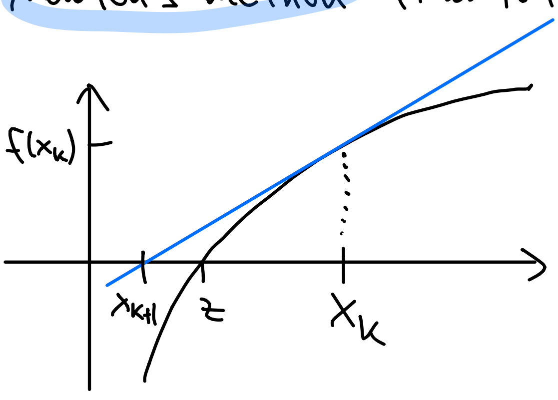
- **Bisection:**



- choose $a<b$, s.t. $f(a)\cdot f(b)<0$
  (if $f(a)f(b)=0 \Rightarrow$ done)
- set $c=\frac{a+b}{2}$
  - $\rightarrow$ if $f(c)=0 \Rightarrow$ done
  - $\rightarrow$ if $f(a)\cdot f(c)<0 \Rightarrow$ root is in $[a,c]$
  - $\rightarrow$ if $f(b)\cdot f(c)<0 \Rightarrow$ root is in $[c,b]$

- repeat with either $[a,c]$ or $[c,b]$
- Advantage: • robust, <u>only continuity necessary</u>
  (except if $f(x)\geq 0 \ \forall x$)
- Disadvantage: • slow, linear convergence (error reduces by $\frac{1}{2}$ in each step)

- **Newton's method** (Newton-Raphson):



- we have: $f'(x_k)=\dfrac{f(x_k)}{x_k-x_{k+1}}$

$$\Rightarrow x_k-x_{k+1}=\frac{f(x_k)}{f'(x_k)}$$

$$\Rightarrow x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \longrightarrow \text{iterate}$$

- Convergence?

use Taylor expansion around x_k

$$f(z) = f(x_k) + f'(x_k)(z - x_k) + \frac{f''(x_k)}{2}(z - x_k)^2 + \underbrace{O((z-x_k)^3)}_{=R}$$

Let z be the root, i.e., $f(z) = 0$

$$\Rightarrow 0 = f(x_k) + f'(x_k)(z - x_k) + \frac{f''(x_k)}{2}(z - x_k)^2 + R$$

$$x_k = x_{k+1} + \frac{f(x_k)}{f'(x_k)}$$

$$\Rightarrow 0 = f(x_k) + f'(x_k)\left(z - x_{k+1} - \frac{f(x_k)}{f'(x_k)}\right) + \frac{f''(x_k)}{2}(z - x_k)^2 + R$$

$$\Rightarrow z - x_{k+1} = \frac{-f''(x_k)}{2f'(x_k)}(z - x_k)^2 + \underbrace{O((z-x_k)^3)}_{\text{neglect}}$$

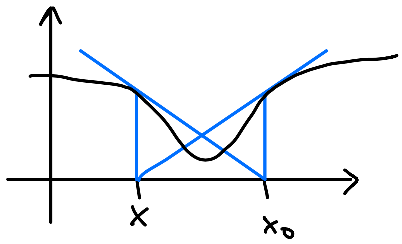
error in k-th step $\varepsilon_k = |z - x_k|$

$$\Rightarrow \varepsilon_{k+1} \leq \underbrace{\left|\frac{f''(x_k)}{2f'(x_k)}\right|}_{\text{suppose } \leq C} \varepsilon_k^{2}$$

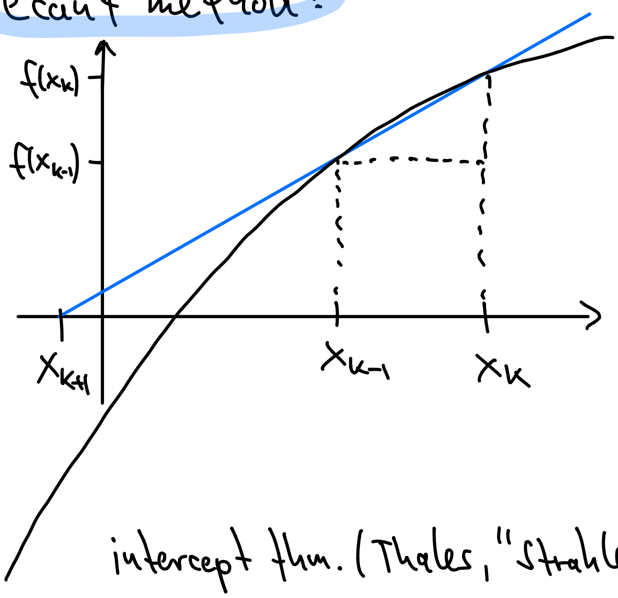
$\rightarrow$ order of convergence

- Advantage: · fast (quadratic convergence)
- Disadvantages: · need differentiability
  · need derivative explicitly

- need more conditions for convergence
  ↳ possible problems: – $f'(x_k)=0$ for some x_k
  – f'' not continuous
  – x_0 too far away from root
  – cyclic behavior



- **Secant method:**



– take secants instead of tangents

intercept thm. (Thales, "Strahlensatz"):

$$\frac{f(x_k)}{x_k - x_{k+1}} = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \implies x_k - x_{k+1} = \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

$$\implies \text{iteration } x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

– Advantages: • still fast, order of convergence = 1.62 (Golden Ratio!)
(under some conditions similar to Newton's method)

• derivative not needed

otherwise similar to Newton

- Python's brentq fct.:
  - combines advantages of several methods (especially bisection and secant)
  - always converges for cont. fct.s

  => robust and relatively fast