Intro to 1D rootfinding

Thursday, August 27, 2020 11:57 PM

Chapter 2 covers 1D $\underbrace{\text{root-finding}}_{\text{aka zeros}}$ and fixed-point equations,

i.e., solving a single scalar equation $f(x)=0$ ← why 0? just a convention: to solve $f(x)=13$, define $g(x)=f(x)-13$ and solve $g(x)=0$

Ex

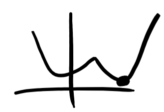
Our country produces $\left(\cos\left(\pi\cdot\frac{t}{12}\right)+1\right)\cdot e^{.01\cdot t}$ amount of CO_2 per year (t is in years)

How long before we produce 500 amount of CO_2?

Solve $(\cos(\ldots)+1)\,e^{.01t}-500=0$

Ex

Find the minimum of $g(x)=2x^4-3x^3+x^2-x+5$

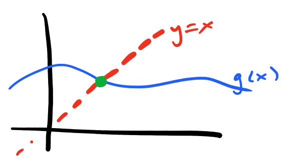


set $\underbrace{g'(x)}_{\text{call this } f(x)}=0$

* solving polynomial root-finding has special methods that are better than the generic methods in this chapter

Fixed-pt. equations

$g(x)=x$



A root-finding problem in disguise!

$\underbrace{g(x)-x}_{f(x)}=0$

(...and vice versa: $f(x)=0$
iff $-c\cdot f(x)=0$ ($c\neq 0$) — choose c wisely!
iff $\underbrace{x-c\cdot f(x)}_{g(x)}=x$ ... a fixed-pt. problem)

Big Picture

1D rootfinding is easy, not many new developments in past 100 years.

TL;DR ... in Matlab, use fzero (combination of bisection, secant, inverse quad. interpolation)

in Python, use scipy.optimize.root_scalar (you can choose Brent's, bisection, Newton, secant, etc.)

(fsolve is deprecated)

Solving Equations

	linear	non-linear
scalar	Trivial $3x=2$	Easy. Ch. 2 $(\tan^{-1}(x))^2=x^{1/3}$
multi-dimension	OK*. Ch. 6 $3x+4y=2$, $2x-5y=5$	Challenging, active research (many special cases)

* laptop can solve 10^4 equations in a few seconds, but 10^5 is tricky

Solving linear equations (multi-dimensional) is one of the most common scientific computing operations. We're good at doing it, so we try to exploit it when possible, i.e., linearizing nonlinear systems

Justification that 1D root finding is a "static" (old-school) topic

First edition of Burden & Faires is about 40 yrs ago.

i.e. Newton, Raphson, Gregory } all 1600's

Let's say computers are now 1,000 times faster (not exactly, but 1000 is a nice number)

Method	1980	2020
• Find a root on an interval $[a,b]$, w/ accuracy ε, using bisection	$[0,2]$ accuracy ε	either accuracy ε on interval $[0, \underbrace{2^{1000}}_{\simeq 10^{300}}]$ or interval $[0,2]$ and accuracy $\varepsilon/2^{1000}$ Nice!
• Find a root via Newton's method e.g., error $e_n = (0.99999)^{2^n}$	$n=10$ error is $e_{10} \simeq 3\cdot 10^{-5}$	$n = 10{,}000$ error is $< 10^{-300}$ (cannot compute error expression due to underflow! Even for $n=100$) Nice!
• Find a root by making a grid and checking each pt.	$[0,1]$ interval accuracy ε	either accuracy ε on interval $[0,1000]$ or interval $[0,1]$ w/ accuracy $\varepsilon/1000$. Not bad

= In comparison, linear or nonlinear problems in high dimensions are still active topics of research. Why? We need algorithms, not just computer power =

• Solve **n** linear equations with **n** variables	**n**	**10·n**	Not good (recall we have 1,000× speed)
• Find a max or min or root by making a grid in dimension **d**	**d = 3**, $[0,1]^3$	$[0,10]^3$	Not good
	d = 100, $[0,1]^{100}$	$[0,1.07]^{100}$	Terrible... barely noticeable difference

1D rootfinding research/innovation essentially stopped after Brent's 1973 book

Bottom line:

1D root-finding is easy

Why teach it? (you'll never need to implement it yourself)

- Basis for more complicated algorithms
- Classical
- Illustrates concepts like rate of convergence