PA081: Programování numerických výpočtů

2. Nelineární rovnice o jedné neznámé

Aleš Křenek

jaro 2019

Obecná formulace problému

- hledáme řešení rovnice

$$F(x) = G(x)$$

  kde alespoň jedna z funkcí F, G není lineární
- víceméně všechny numerické metody jsou iterační
  - začínáme s odhadem řešení x_0
  - v každém kroku odhad postupně zpřesňujeme
- výpočet končí dosažením kritéria zastavení
  - dosáhli jsme dostatečně přesné aproximace řešení

Obecná formulace problému

- hledáme řešení rovnice

$$F(x) = G(x)$$

  kde alespoň jedna z funkcí F, G není lineární
- víceméně všechny numerické metody jsou iterační
  - začínáme s odhadem řešení x_0
  - v každém kroku odhad postupně zpřesňujeme
- výpočet končí dosažením kritéria zastavení
  - dosáhli jsme dostatečně přesné aproximace řešení
- anebo
  - rovnice nemá řešení
  - použitá metoda pro danou rovnici nefunguje
  - špatně jsme odhadli x_0
  - dosáhli jsme falešného řešení
- žádná metoda není dokonale univerzální
- bez jisté analýzy vlastností rovnice se neobejdeme

Železniční příklad

Kolejnice délky $2d$ je ohnutá do oblouku tak, že její konce jsou ve vzálenosti $2a$. Jaká je vzdálenost středu kolejnice od spojnice krajních bodů?

- označíme R poloměr oblouku, α úhel poloviny úseče, r hledanou vzdálenost
- platí rovnice

$$d = R\alpha \qquad R\sin\alpha = a \qquad r = R(1 - \cos\alpha)$$

- dosazením a jednoduchou úpravou

$$\frac{d}{a}\sin\alpha = \alpha$$

- hledáme pevný bod funkce
- kandidát na metodu prosté iterace

Metoda prosté iterace

- rovnice ve tvaru $x = f(x)$
- řešením je pevný bod funkce f
- počítáme opakovaně $x_{i+1} = f(x_i)$
  - až k dosažení kritéria konvergence

Metoda prosté iterace

- rovnice ve tvaru $x = f(x)$
- řešením je pevný bod funkce f
- počítáme opakovaně $x_{i+1} = f(x_i)$
  - až k dosažení kritéria konvergence
- kdy a proč to funguje? – věta o pevném bodě

  Je-li pro $K \subseteq \mathbb{R}$ funkce $f\colon K \to K$ kontrakce, tj. existuje $q \in (0,1)$ tak, že $\forall x, y \in K\colon |f(x) - f(y)| \leq q|x - y|$, potom existuje $x^* \in K$ takové, že pro libovolné $x_0 \in K$ je x^* limitou posloupnosti $x_{i+1} = f(x_i)$.
- idea důkazu

$$|x_{i+1} - x^*| = |f(x_i) - f(x^*)| < |x_i - x^*|$$

Metoda prosté iterace

Pro železniční příklad

- je zobrazení $\alpha \mapsto \frac{d}{a} \sin \alpha$ kontrakce?
- postačující podmínka

$$\forall x \in K \colon f'(x) < 1 \quad \text{tedy} \quad \cos \alpha < \frac{a}{d}$$

- $\frac{d}{a} \sin \alpha = \alpha$ bude mít řešení v $(\arccos \frac{a}{d}, \frac{\pi}{2})$

Metoda prosté iterace

Pro železniční příklad

- je zobrazení $\alpha \mapsto \frac{d}{a} \sin \alpha$ kontrakce?
- postačující podmínka

$$\forall x \in K\colon f'(x) < 1 \quad \text{tedy} \quad \cos \alpha < \frac{a}{d}$$

- $\frac{d}{a} \sin \alpha = \alpha$ bude mít řešení v $(\arccos \frac{a}{d}, \frac{\pi}{2})$
- implementačně velmi jednoduchá metoda
- zdánlivě velmi speciální případ
  - řešený problém lze často transformovat, aby splňoval podmínku kontrakce
- rychlost konvergence záleží na vlastnostech $f(x)$

Pohyb planet

- poloha planety na eliptické dráze o poloosách a, b

$$x = a\cos E - e \qquad y = a\sqrt{1-e^2}\sin E$$

kde

$$e = \sqrt{1 - \frac{b^2}{a^2}} \qquad E = \omega t + e\sin E$$

- nelze snadno řešit analyticky
- jeden z nejznámějších motivačních příkladů pro numerické řešení rovnic
- viz např. „Kepler's equation“ na Wikipedii

Hledání kořenů

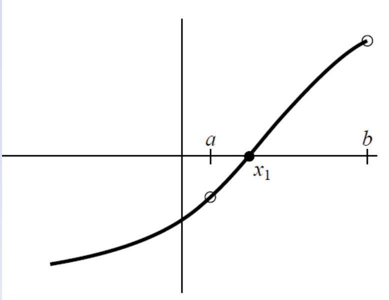
- pro rovnici $F(x) = G(x)$ uvažujeme $f(x) = F(x) - G(x)$
- provedeme separaci kořenů $f(x)$
  - definiční obor $f(x)$ rozdělíme na intervaly $[a_i, b_i]$
  - v každém $[a_i, b_i]$ má funkce právě jeden kořen
  - $f(a_i)f(b_i) < 0$
  - f je na $[a_i, b_i]$ spojitá
- nepovede-li se separace dokonale, musíme být připraveni na následky
- vybereme vhodnou metodu hledání kořene na $[a_i, b_i]$
- stanovíme konkrétní podmínku ukončení

Separace kořenů

Spojitá funkce

- máme štěstí a f je spojitá, platí varianta věty o střední hodnotě

  Je-li f na $[a, b]$ spojitá a $f(a)f(b) < 0$, existuje $x \in [a, b]$ tak, že $f(x) = 0$



Separace kořenů

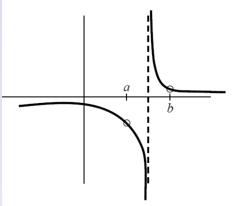
Nespojitá funkce

- není-li f spojitá, ale je ohraničená
  - „kříží“ osu x v bodě nespojitosti
  - numericky nerozlišitelné od kořene

Separace kořenů

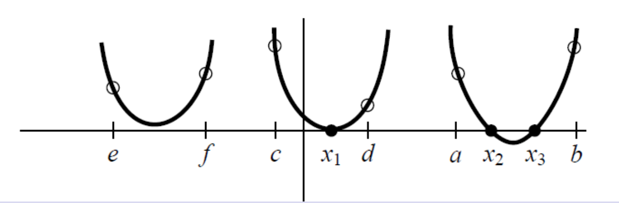
Nespojitá funkce

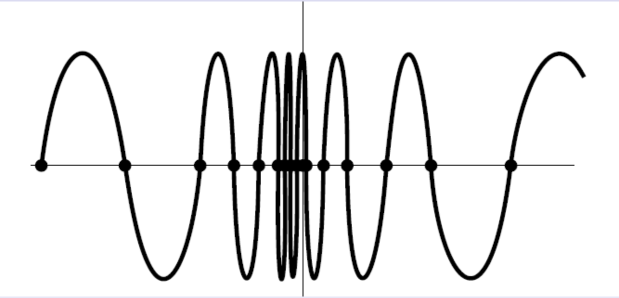
- není-li f spojitá, ale je ohraničená
  - „kříží“ osu x v bodě nespojitosti
  - numericky nerozlišitelné od kořene
- není-li ani ohraničená
  - např. $\frac{1}{x-c}$
  - některé metody najdou „kořen“ v c
  - snadno identifikovatelný problém



Separace kořenů

Patologické případy





Separace kořenů

Praktický postup

- základní analýza vlastností f je nezastupitelná
  - netušíme-li vůbec, zdali má kořeny, kde jsou, kolik jich je atd., je problém zpravidla někde jinde

Separace kořenů

Praktický postup

- základní analýza vlastností f je nezastupitelná
  - netušíme-li vůbec, zdali má kořeny, kde jsou, kolik jich je atd., je problém zpravidla někde jinde
- algoritmus „look inward"
  - alespoň nahrubo známe interval, kde by kořen měl být
  - rozdělíme na n menších a postupně prohledáváme
  - v případě neúspěchu můžeme opakovat s větším n

Separace kořenů

Praktický postup

- základní analýza vlastností f je nezastupitelná
  - netušíme-li vůbec, zdali má kořeny, kde jsou, kolik jich je atd., je problém zpravidla někde jinde
- algoritmus „look inward“
  - alespoň nahrubo známe interval, kde by kořen měl být
  - rozdělíme na n menších a postupně prohledáváme
  - v případě neúspěchu můžeme opakovat s větším n
- algoritmus „look outward“
  - poslední zoufalý pokus
  - počáteční interval expandujeme exponenciálně na tu stranu, kde se funkce blíží k ose x
  - zpravidla funguje pro funkce, které mají pro $x \to \pm\infty$ opačné zneménko

Metoda půlení intervalů

Princip

- začínáme se separovaným kořenem
  - pro daný interval $[a, b]$ platí $f(a)f(b) < 0$
- není-li $\frac{b+a}{2}$ přesně kořen, platí právě jedna nerovnost

$$f\left(\frac{b+a}{2}\right) f(a) < 0 \qquad f\left(\frac{b+a}{2}\right) f(b) < 0$$

- interval rozpůlíme a pokračujeme rekurzivně
- metoda vždy dokonverguje ke kořeni nebo k singularitě
- je-li jich více, odhalí jen jeden

Metoda půlení intervalů

Konvergence

- je-li požadovaná přesnost ϵ, je třeba
  $$n = \log_2 \frac{b-a}{\epsilon}$$
  iteračních kroků
- lineární rychlost konvergence
  - v každém kroku přibývá konstatně platných číslic přesnosti
- kritéria ukončení
  - jsou-li a, b řádově srovnatelná, nemá smysl více iterací než než počet bitů mantisy
  - v opačném případě opět musíme vědět, co chceme
  - standardně se používá zastavení při velikosti intervalu
    $$\epsilon \frac{|a| + |b|}{2}$$
    kde ϵ je přesnost daného datového typu
- metoda je robustní ale relativně pomalá

Metoda půlení intervalů

```
float rtbis(float (*func)(float), float x1, float x2,
     float xacc,int *iter)
{
    int j;
    float dx,f,fmid,xmid,rtb;
    f=(*func)(x1);
    fmid=(*func)(x2);
    rtb = f < 0.0 ? (dx=x2-x1,x1) : (dx=x1-x2,x2);
    for (j=1;j<=JMAX;j++) {
        fmid=(*func)(xmid=rtb+(dx *= 0.5));
        if (fmid <= 0.0) rtb=xmid;
        if (fabs(dx) < xacc || fmid == 0.0) {
            *iter = j;
            return rtb;
        }
    }
}
```

Newtonova metoda

- také Newton-Raphsonova metoda
- odvozena z Taylorova rozvoje $x^* = x_i + \delta_i$

$$f(x_i + \delta_i) = f(x_i) + f'(x_i)\delta_i + \frac{f''(x_i)\delta_i^2}{2!} + \dots$$

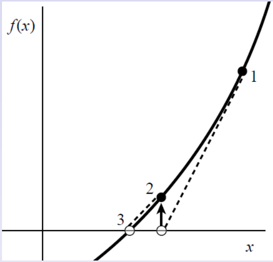
- zanedbáme vyšší derivace

$$\delta_i \doteq -\frac{f(x_i)}{f'(x_i)}$$

- opakujeme iterační krok

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

- nutno spočítat i derivaci



Newtonova metoda

Konvergence

- označíme x^* kořen, a ϵ_i odchylky $x_i - x^*$

$$x^* + \epsilon_{i+1} = x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x^* + \epsilon_i - \frac{f(x_i)}{f'(x_i)}$$

  tedy $\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)}$
- Taylorův rozvoj

$$0 = f(x^*) = f(x_i) - \epsilon_i f'(x_i) + \frac{\epsilon_i^2 f''(\xi_i)}{2!}$$

  kde $\xi_i \in [0, \epsilon_i]$
- podělením $f'(x_i)$ a dosazením dostaneme

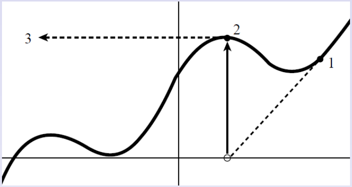
$$\epsilon_{i+1} = -\epsilon_i^2 \frac{f''(\xi_i)}{2 f'(x_i)}$$

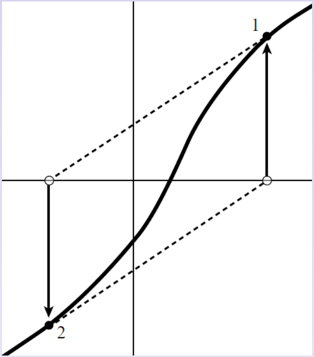
- kvadratická konvergence
  - v každém kroku se zdvojnásobí počet platných číslic

Newtonova metoda

Nešvary

- špatná globální konvergence
  - velká citlivost na lokální vlastnosti f mimo kořen





- prakticky nepoužitelná sama o sobě
  - nutno kombinovat s jinými metodami
  - vhodná k „vyleštění" nahrubo nalezeného kořene

Newtonova metoda

```
float rtnewt(void (*funcd)(float, float *, float *),
             float x1, float x2, float xacc)
{
   int j;
   float df,dx,f,rtn;
   rtn=0.5*(x1+x2);
   for (j=1;j<=JMAX;j++) {
      (*funcd)(rtn,&f,&df);
      dx=f/df;
      rtn -= dx;
      if ((x1-rtn)*(rtn-x2) < 0.0) {
        /* chyba, utekli jsme */
         return 0;
      }
      if (fabs(dx) < xacc) return rtn;
   }
   /* Příliš mnoho iterací */
   return 0;
}
```

Halleyova metoda

- máme k dispozici i 2. derivaci
- zanedbáme až kubický člen Taylorova rozvoje

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)\left(1 - \frac{f(x_i)f''(x_i)}{2f'(x_i)^2}\right)}$$

Halleyova metoda

- máme k dispozici i 2. derivaci
- zanedbáme až kubický člen Taylorova rozvoje

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)\left(1 - \frac{f(x_i)f''(x_i)}{2f'(x_i)^2}\right)}$$

- rychlost konvergence 3
  - za cenu výpočtu f''i
  - proč tedy nespočítat místo toho další krok N. metody (rychlost konvergence 4)?
  - v mnoha případech lze recyklovat části výpočtu, f'' je téměř zadarmo

Halleyova metoda

- máme k dispozici i 2. derivaci
- zanedbáme až kubický člen Taylorova rozvoje

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)\left(1 - \frac{f(x_i)f''(x_i)}{2f'(x_i)^2}\right)}$$

- rychlost konvergence 3
  - za cenu výpočtu f''i
  - proč tedy nespočítat místo toho další krok N. metody (rychlost konvergence 4)?
  - v mnoha případech lze recyklovat části výpočtu, f'' je téměř zadarmo
- může i škodit
  - pragmaticky ořízneme člen $1 - \frac{f(x_i)f''(x_i)}{2f'(x_i)^2}$ do mezí $[0.8, 1.2]$

Seminewtonovské metody

- výpočet derivace nemusí být možný nebo žádoucí
- aproximace

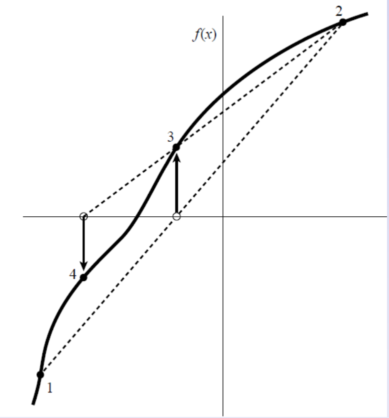
$$f'(x) = \frac{f(x+\delta) - f(x)}{\delta}$$

  není vhodná
  - potřebuji 2× vyhodnocení f, tj. rychost konvergence jen $\sqrt{2}$
  - malé δ – numerická nestabilita
  - velké δ – nepřesnost
- seminewtonovské metody – aproximace derivace „uvnitř“
  - metoda sečen
  - metoda regula falsi

Seminewtonovské metody

Metoda sečen

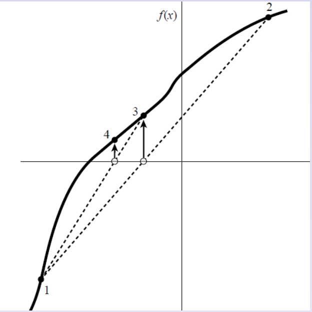
- derivace je aproximována směrnicí spojnice dvou odhadů
- vždy počítá s posledními dvěma odhady
- konverguje obecně rychleji
  - zlatý řez (1.618...)
- může porušit separaci kořene



Seminewtonovské metody

Regula falsi

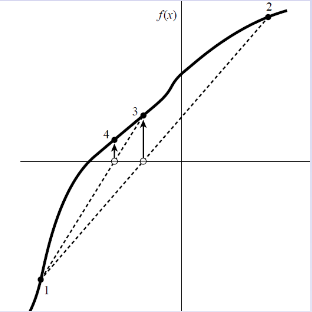
- důsledně zachovává separaci kořene
- v případě potřeby použije starší odhad

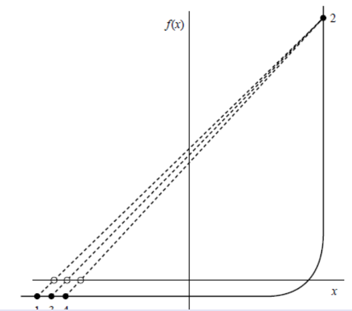


Seminewtonovské metody

Regula falsi

- důsledně zachovává separaci kořene
- v případě potřeby použije starší odhad





- v patologických případech pomalá konvergence

Riddersova metoda

- patologické chování předchozích metod
  - jedním z důvodů je proložení přímkou
- idea Riddersovy metody – proložit exponenciálou

$$p(x) = a + be^{cx}$$

- potřebné 3 body x_0, x_1, x_2, omezené na $x_1 - x_0 = x_2 - x_1 = d$
- $p(x) = 0$ v bodě

$$x_4 = x_0 + d\frac{\ln b}{\ln a} \qquad \text{kde} \qquad a = \frac{f(x_0) - f(x_1)}{f(x_1) - f(x_2)}, \quad b = \frac{f(x_0) - f(x_1)}{f(x_0) - af(x_1)}$$

Riddersova metoda

- vyžaduje výpočet dvou logaritmů, příliš náročné
- nahrazeno aproximací

$$x_3 = x_0 + d\frac{u(3+u^2)}{v(3+v^2)} \qquad \text{kde} \qquad u = \frac{b-1}{b+1} \quad v = \frac{a-1}{a+1}$$

- Ridders, C. *A new algorithm for computing a single root of a real continuous function*. IEEE Transactions on Circuits and Systems 26: 979-980, 1979.
- x_4 vždy spadne do $[x_0, x_2]$
- vybereme nejbližší z x_0, x_1, x_2 a dopočítáme třetí do stejné vzdálenosti
- rychlost konvergence $\sqrt{2}$
  - po krocích kvadraticky, ale vyžaduje dvojí vyhodnocení

Brentova metoda

- půlení intervalu jako bezpečný základ
- proložení inverzní kvadratickou funkcí pro urychlení konvergence
  - x jako kvadratická funkce y
- opět tři aktuální body odhadu x_1, x_2, x_3, výpočet vede na

$$x_3 = x_1 + \frac{P}{Q}$$

  kde P, Q jsou vyjádřeny z x_1, x_2, x_3 a $f(x_1), f(x_2), f(x_3)$ elementárními aritmetickými operacemi
- algoritmus hlídá zejména $|Q| \gg 0$, jinak se vrací k půlení intervalů
- prakticky zřejmě nejuniverzálnější metoda
  - nejsou-li k dispozici derivace
  - neřešíme-li speciální případ, kde jiná metoda funguje lépe a/nebo rychleji

Multiplikativní kalkulus

- *geometrická derivace*

$$f^*(x) = \lim_{h \to 0} \left(\frac{f(x+h)}{f(x)} \right)^{\frac{1}{h}}$$

- pro „rozumné“ funkce

$$f^*(x) = e^{\frac{f'(x)}{f(x)}}$$

- platí řada analogií „normálního“ kalkulu, zejména Taylorova věta

$$f(x+h) = f(x) f^*(x)^h f^{**}(x)^{\frac{h^2}{2!}} \dots f^{*(n)}(x)^{\frac{h^n}{n!}} f^{*(n+1)}(x+\theta)^{\frac{h^n}{n!}}$$

Multiplikativní kalkulus

- Newtonova metoda: pro $f(x) = 1, f^*(x) \neq 1$ bude posloupnost
  $$x_{i+1} = x_i - \frac{\ln f(x_i)}{\ln f^*(x_i)}$$
  konvergovat k x
- speciálně pro funkce tvaru $e^{g(x)}$ odstraní exp. složku
- v jistých případech rychlejší praktická konvergence
  - kombinace exponenciálních a goniometrických funkcí
- viz `http://dx.doi.org/10.1155/2016/8174610`

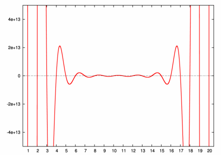
Kořeny polynomů

- speciální případ nelineární rovnice
  - lze aplikovat zjednodušená (rychlejší, přesnější) řešení
  - silnější sklony ke špatnému chování

Kořeny polynomů

- speciální případ nelineární rovnice
  - lze aplikovat zjednodušená (rychlejší, přesnější) řešení
  - silnější sklony ke špatnému chování
- špatná podmíněnost, např. Wilkinsonův polynom

$$w_n(x) = \prod_{i=1}^{n} (x - i)$$



- reálný polynom stupně n má n kořenů

Kořeny polynomů

Potenciální problémy

- násobné kořeny
  - derivace jsou nulové, selhávají Newtonova a seminewtonovské metody
  - sudě násobné kořeny nelze separovat
- kořeny mohou být komplexní, co s nimi?

Kořeny polynomů

Faktorizace kořenů

- pro reálný kořen

$$P_n(x) = (x - x_n)Q_{n-1}(x)$$

- pro dvojici komplexních kořenů

$$\begin{aligned} P_n(x) &= (x - (a + ib))(x - (a - ib))Q_{n-2}(x) \\ &= (x^2 - 2ax + a^2 + b^2)Q_{n-2}(x) \end{aligned}$$

Kořeny polynomů

Faktorizace kořenů

- pro reálný kořen

$$P_n(x) = (x - x_n)Q_{n-1}(x)$$

- pro dvojici komplexních kořenů

$$\begin{aligned} P_n(x) = (x - (a + ib))(x - (a - ib))Q_{n-2}(x) \\ = (x^2 - 2ax + a^2 + b^2)Q_{n-2}(x) \end{aligned}$$

- známe-li x_n, dokážeme spočítat koeficienty $Q(x)$

$$\begin{aligned} (q_0 + q_1 x + \dots)(x - x_n) \\ = -x_n q_0 + (q_0 - x_n q_1)x + (q_1 - x_n q_2) + \dots \end{aligned}$$

  a tedy

$$q_0 = -\frac{p_0}{x_n} \qquad q_i = \frac{p_i - q_{i-1}}{x_n}$$

- analogicky opačným směrem, počínaje q_n

Kořeny polynomů

Faktorizace kořenů

- dvojitá rekurentní formule, hrozí numerická nestabilita
  - vypočteme nepřesné koeficienty Q_{n-1}, použijeme k výpočtu $Q_{n-2}, \ldots$
- stabilní chování
  - kořen s největší absolutní hodnotou, počítáme od q_0
  - kořen s nejmenší absolutní hodnotou, počítáme od q_n
- „leštění kořenů“
  - postupně nalezené kořeny chápeme jako aproximaci
  - použijeme v původním $P(x)$ např. do Newtonovy metody
  - příliš velká chyba může svést leštění k jinému kořenu (lze ohlídat)

Kořeny polynomů

Přímočará metoda

- odchytíme reálný kořen dříve popsanými metodami
  - separace metodou pokusu a omylu
  - řešení zpravidla Newtonovou metodou
    - výpočet derivace polynomu je triviální
- provedeme faktorizaci, opakujeme pro další reálný kořen
  - zpravidla včetně „leštění"
- následně faktorizace kvadratických polynomů
  - Mullerova metoda – zobecnění metody sečen, aproximace parabolou
  - Bairstowova metoda – vede na Newtonovu metodu ve dvou dimenzích

Kořeny polynomů

Další metody

- Laguerre
  - iterační hledání jednoho kořene včetně komplexních
  - triková manipulace s $\ln |P(x)|$ a jeho derivacemi
  - funguje i pro komplexní koeficienty
  - faktorizace a opakované použití
  - iterační krok lze použít k leštění
- Jenkins-Traub
  - propracovaná metoda, základ obecných knihoven
  - odvození vyžaduje 4 kapitoly v knize ...
- Lehmer-Shur
  - generalizace separace kořenů na kruhy v komplexní rovině

Polynomy – vyhodnocení

- výpočetní náročnost

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$$

  - n násobení $x^i x$
  - n násobení $p_i x^i$
  - n sčítání

Polynomy – vyhodnocení

- výpočetní náročnost

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$$

  - n násobení $x^i x$
  - n násobení $p_i x^i$
  - n sčítání
- Hornerův algoritmus

```
p[n]=a[n]
for (i=n-1;i>=0;i-)
    p[i]=xp[i+1]+a[i]
return p[0]
```

  - pouze $2n$ operací, výhradně `madd`
  - lze ukázat numerickou stabilitu
  - lze přímočaře rozšířit na výpočet derivací

Kořeny polynomů

Vlastní hodnoty matic

- vlastní hodnoty matice A jsou kořeny charakteristického polynomu
$$P(x) = \det |A - xI|$$
- lze zkonstuovat matici
$$A = \begin{pmatrix} -\frac{p_{m-1}}{p_m} & -\frac{p_{m-2}}{p_m} & \dots & -\frac{p_0}{p_m} \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$
jejíž charakteristický polynom je právě $P(x) = \sum p_i x^i$
- lze aplikovat metody hledání vlastních hodnot
  - zpravidla pomalejší ale celkově robustnější

Cirkusové číslo

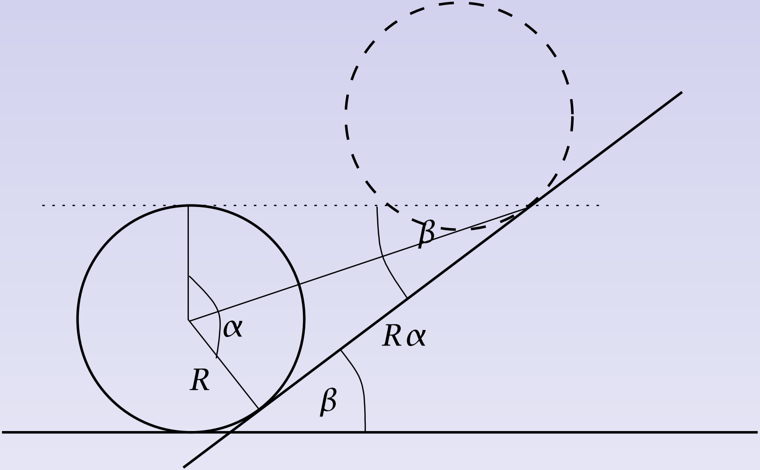
Artisté připravují nové vystoupení.

Na začátku se jeden z nich zachytí rukama i nohama v kruhové konstrukci o průměru 180 cm a další konstrukci vykoulí na plošinu.

Je třeba, aby se na konci tohoto pohybu původně nejvyšší bod kruhové konstrukce dotýkal plošiny a byl ve stejné výšce jako na začátku.

Jaká je potřebná délka a sklon plošiny?

Cirkusové číslo



Cirkusové číslo

- označíme
  - R – poloměr kruhu
  - α – úhel otočení kruhu (v radiánech)
  - β – úhel naklonění plošiny
- délka pohybu kruhu po plošině je $R\alpha$
- z naznačeného čtyřúhelníku odečteme
  - $\alpha + \beta = \pi$ (další dva úhly jsou pravé)
  - $\tan \frac{\beta}{2} = \frac{R}{R\alpha}$
- z toho vyplývá řešená rovnice

$$\tan \frac{\beta}{2} = \frac{1}{\pi - \beta} \qquad \text{tedy} \qquad f(\beta) = (\pi - \beta) \tan \frac{\beta}{2} - 1 = 0$$

Cirkusové číslo

- separace kořene
  - β je ostrý úhel, tj. $\beta \in (0, \frac{\pi}{2})$
  - řešení je právě jedno
  - z geometrické podstaty problému
  - formulací rovnic jsme nepřidali falešný kořen
  - dáme si práci s exaktním důkazem nebo to rovnou zkusíme
- pro jistotu ověříme

$$f(0) = -1 \qquad f(\pi/2) = 0.5707963$$

- numericky nebezpečná by byla až oblast $\beta \to \pi$
  - vedla by na součin typu $0 \cdot \infty$
  - pohybujeme se v bezpečné vzdálenosti
- budeme předstírat, že neumíme spočítat derivaci
- ukážeme metodu sečen, Riddersovu, a Brentovu

Cirkusové číslo

Metoda puleni intervalu, požadovaná absolutní přesnost 10^{-4}

```
      n       beta            f(beta)
      1       +0.0000000      -1.0000000
      2       +1.5707963      +0.5707963
      3       +0.7853982      -0.0240323
      4       +1.1780972      +0.3119657
      5       +0.9817477      +0.1544612
      6       +0.8835729      +0.0679638
      7       +0.8344855      +0.0226702
      8       +0.8099419      -0.0005027
      9       +0.8222137      +0.0111281
...
     15       +0.8105171      +0.0000445
     16       +0.8104212      -0.0000467
```

- počínaje třetím krokem výpočet vždy ve středu intervalu
- pomalá konvergence

Cirkusové číslo

Riddersova metoda, požadovaná absolutní přesnost 10^{-4}

n	beta	f(beta)
1	+0.0000000	-1.0000000
2	+1.5707963	+0.5707963
3	+0.7853982	-0.0240323
4	+0.8103685	-0.0000968
5	+1.1905824	+0.3213144
6	+0.8104702	-0.0000001
7	+1.0005263	+0.1704016
(8)	+0.8104702	

4. první iterační výpočet
5. půlení intervalu mezi 2. a 4.
6. další iterace
7. půlení mezi 5. a 6.
8. výsledek, už je v toleranci

Cirkusové číslo

Brentova metoda, požadovaná absolutní přesnost 10^{-4}

n	beta	f(beta)
1	+0.0000000	-1.0000000
2	+1.5707963	+0.5707963
3	+1.0000000	+0.1699574
4	+0.7931377	-0.0165737
5	+0.8115179	+0.0009960
6	+0.8104759	+0.0000054
7	+0.8104259	-0.0000422
(8)	+0.8104759	

- počítá primárním způsobem, nedošlo na bezpečné půlení intervalů
- srovnatelně rychlá konvergence s Riddersem
- při větší požadované přesnosti se liší ± o jeden krok

Cirkusové číslo

- všechny metody se v rámci požadované tolerance shodují
- rychlost konvergence dle očekávání
  - zdvojnásobení požadované přesnosti zdvojnásobí počet kroků půlení intervalů
  - Riddersova metoda naroste ze 7 na 9
  - Brentova dokonce jen ze 7 na 8
  - lepší výsledky než teoretická rychlost konvergence $\sqrt{2}$

Cirkusové číslo

- délka pohybu po plošině je

$$R\alpha = R(\pi - \beta) = 2.098$$

- zbývá spočítat část plošiny od země k bodu dotyku

$$\tan\frac{\pi-\beta}{2} = \frac{R}{l} \qquad \text{tedy} \qquad l = \frac{R}{\tan((\pi-\beta)/2)} = 0.3861$$

- nehrozí významné numerické nepřesnosti
- celková délka plošiny je 2.484 m, sklon 46.44°

Obchodní případ

Americká garážová firma zahajuje montáž počítačů.

Kolik jich musí v prvním roce prodat, aby byla zisková?

Obchodní případ

Americká garážová firma zahajuje montáž počítačů.

Kolik jich musí v prvním roce prodat, aby byla zisková?

- uvažované náklady
  - fixní jednorázové (pořízení vybavení): \$20000
  - fixní roční (nájem, web, …): \$15000
  - variabilní (komponenty, hodinová sazba práce, …): $\$625n$
  - semivariabilní (náročnější logistika atd.) $\$30n^{1.5}$
- předpokládané příjmy
  - prodej počítačů: $\$1500n$
  - slevy (množstevní, VIP zákazníci, …): $-\$10n^{1.5}$

Obchodní případ

- řešená rovnice

$$f(n) = TC(n) - TS(n) = 35000 - 875n + 40n^{1.5} = 0$$

- první dva členy – lineární funkce
  - prochází bodem (0,35000)
  - osu x protíná v $35000/875 = 40$
- člen $40n^{1.5}$ způsobí „prohnutí“ nahoru
  - posune kořen z bodu 40 dál
  - přidá druhý kořen pro vyšší n
- zisku dosahujeme v oblasti mezi těmito kořeny

Obchodní případ

- separace 1. kořene
  - podle předchozí úvahy $f(40) > 0$, skutečná hodnota 10119
  - zkusíme $f(80) = -6378$, vyhovuje
- separace 2. kořene
  - předchozí $f(80) = -6378$
  - hrubý odhad zanedbáním absolutního člene

$$-875n + 40n^{1.5} = 0 \qquad \text{tedy} \qquad n = \left(\frac{875}{40}\right)^2 = 478.51$$

  - zkusíme $f(500) = 44713$, vyhovuje
- numerická stabilita v dané separaci kořenů
  - součty/rozdíly čísel s minimálním řádovým rozdílem
  - reálně nás zajímá přesnost na jednotky
  - žádný potenciální problém

Obchodní případ

- vypočtené kořeny 62.7 a 384.0
- výsledky v požadované přesnosti shodné všemi metodami
- chování jednotlivých metod (počty iterací)

přesnost	10^{-1}		10^{-4}	
kořen	1	2	1	2
půlení	11	15	21	25
Ridders	5	9	7	11
Brent	5	9	7	10

- umělá přesnost 10^{-4} – zdvojnásobení počtu platných číslic
- dokládá očekávanou rychlost konvergence
  - lineární pro půlení intervalů
  - $\sqrt{2}$ pro ostatní metody

Shrnutí

- řešíme nelineární rovnice tvaru $F(x) = G(x)$ iteračními metodami
- redukce na hledání kořenů $f(x) = 0$
- separace kořenů je klíčová
  - nalezení intervalu, ve kterém leží právě jeden
- metoda půlení intervalů
  - robustní ale pomalá
- Newtonova metoda
  - rychlá konvergence
  - vyžaduje derivaci a dobrý počáteční odhad
- seminewtonovské metody
  - vnitřní aproximace derivace
  - metoda sečen, regula falsi
- pokročilé metody
  - aproximace funkce složitější křivkou
  - Ridders, Brent
- speciální metody pro polynomy

Domácí úkol

František poseděl s přáteli v restauraci a vrací se poněkud nejistým krokem domů. Cesta měří 2400 m a vede poblíž meandrujícího potoka. Spočítejte, zda a jak daleko od začátku cesty František do potoka spadne.

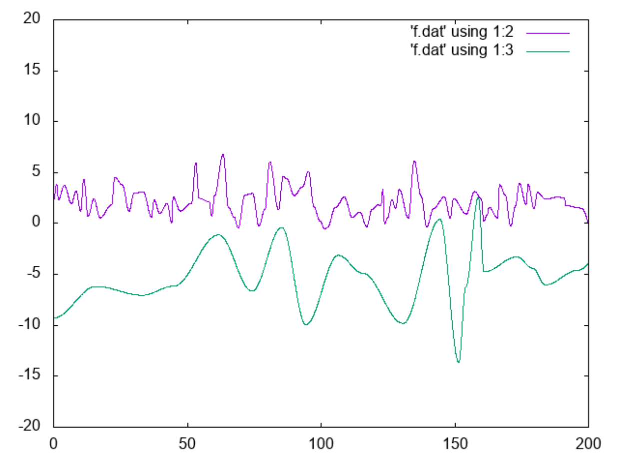
Domácí úkol

František poseděl s přáteli v restauraci a vrací se poněkud nejistým krokem domů. Cesta měří 2400 m a vede poblíž meandrujícího potoka. Spočítejte, zda a jak daleko od začátku cesty František do potoka spadne.

K dispozici máte:

- `double frantisek(double x, double *df)`
  - pro $0 \le x \le 2400$ na vrací Františkovu souřadnici y
  - není-li `df == NULL`, spočítá také derivaci dy/dx
  - při každém běhu programu počítá jinak!
- `double potok(double x, double *dp)`
  - totéž pro potok, počítá vždy stejně

Domácí úkol



Domácí úkol

- Nejsou přístupné zdrojové kódy
  - nevíte tedy přesně, jak se funkce chovají
- K separaci kořene využijte „přirozené" charakteristiky
  - meandry potoka jsou dlouhé v řádu desítek metrů
  - alkoholem ovlivněný krok se může vychýlit do strany i každý metr
  - obě funkce jsou spojité
  - experimentujte s hustotou vzorkování
- Počítejte s uměle vysokou přesností 10^{-12} v y
- Implementujte metody bez derivace i s derivací, srovnejte náročnost
  - funkce `long cena()` vrací počet vyhodnocení obou funkcí, derivace se počítá 2×
- Termín odevzdání 31.3., 1 bod ke zkoušce