Temaøvelse 2

Algebraens fundamentalsætning siger, at ethvert polynomium $p(Z) \in \mathbb{C}[Z]$ af grad mindst én har en rod $\lambda \in \mathbb{C}$ (se Sætning 4.6.1 i lærebogen). For praktiske anvendelser er det vigtigt at kunne finde en sådan rod eller i det mindste at kunne finde en god numerisk tilnærmelse, også kaldet en approksimation. Målet med denne temaøvelse er at opnå algoritmisk indsigt i, hvordan numeriske approksimationer af rødderne i polynomier med reelle koefficienter kan findes. Vi vil gennemgå to metoder: bisektionsmetoden og Newton-Raphson-metoden.

Del I: Bisektionsmetoden

Begynd ved åbne en kommandolinjeversion af Python.

1. For at kunne arbejde med rødderne i polynomier $p(X) \in \mathbb{R}[X]$ definerer vi funktionen, som et sådant polynomium giver anledning til: $p : \mathbb{R} \to \mathbb{R}$, hvor $x \mapsto p(x)$. Det første skridt er at finde ud af, hvordan man definerer en sådan funktion i Python.

   (a) Overvej som et eksempel polynomiet $p(X) = X^3 - X - 6$. Den tilsvarende funktion $p : \mathbb{R} \to \mathbb{R}$ kan defineres i Python som følger:

   ```
   def p(x):
    return x**3 - x - 6
   ```

   Skriv denne kode i kommandolinje-Python (husk at indrykke anden linje). Efter at have indtastet den første linje vil prompten ændre sig fra `>>>` til `...`. Efter at have indtastet den anden linje vil prompten stadig være `...`, men efter at have trykket på Enter-tasten vil prompten ændre sig tilbage til `>>>`. Når den gør det, har du færdiggjort indtastningen af funktionen i Python.

   (b) Nu kan du beregne værdierne af funktionen p ved hjælp af Python. Skriv for eksempel $p(10)$ og kontroller i hånden, at Python returnerer den korrekte værdi.

   (c) Brug Python til at beregne $p(-1)$, $p(0)$, $p(1)$, $p(2)$ og $p(3)$. Kan du nu angive en rod i polynomiet $p(X)$?

Generelt vil man sjældent være heldig at finde en rod i et polynomium $f(X)$ blot ved at beregne nogle enkelte af dens værdier. Derfor vil vi nu studere et kriterium, der af og til kan anvendes til i det mindste at bestemme, hvor en rod cirka er placeret. Vi vil starte med at formulere en sætning for kontinuerte funktioner. Vi vil ikke definere, hvad en kontinuert funktion præcist er, men meget løst sagt er en funktion kontinuert, hvis dens graf ikke har nogen "huller". Du må frit benytte i denne temaøvelse, at enhver funktion $f : \mathbb{R} \to \mathbb{R}$, hvor $x \mapsto f(x)$, og hvor $f(X) \in \mathbb{R}[X]$ er et polynomium, er kontinuert. Kontinuerte funktioner har nogle rare egenskaber, blandt andet den følgende sætning, kendt som *mellemværdisætningen*.

Sætning 1 *Lad a, b være reelle tal, der opfylder $a < b$. Lad yderligere $f : [a, b] \to \mathbb{R}$ være en kontinuert funktion.*

- *Hvis $f(a) < f(b)$, og y opfylder $f(a) < y < f(b)$, så er $y = f(x)$ for nogle x i intervallet $[a, b]$.*
- *Hvis $f(a) > f(b)$, og y opfylder $f(b) < y < f(a)$, så er $y = f(x)$ for nogle x i intervallet $[a, b]$.*

Vi vil ikke bevise denne sætning, men du må bruge den frit. Hvis $f : \mathbb{R} \to \mathbb{R}$ er en funktion, og $\lambda \in \mathbb{R}$ opfylder $f(\lambda) = 0$, kalder vi λ for et nulpunkt i f. Hvis f er defineret ud fra et polynomium $f(X)$, er et nulpunkt i f simpelthen en rod i polynomiet $f(X)$. Mellemværdisætningen leder til en nyttig konsekvens vedrørende nulpunkter i kontinuerte funktioner:

Korollar 2 *Lad a, b være reelle tal, der opfylder $a < b$. Lad yderligere $f : [a, b] \to \mathbb{R}$ være en kontinuert funktion, der opfylder $f(a) \cdot f(b) < 0$. Da eksisterer der $x \in [a, b]$, således at $f(x) = 0$. Med andre ord: funktionen f har et nulpunkt i intervallet $[a, b]$.*

2. Målet med dette spørgsmål er at forstå, hvorfor Korollar 2 er en logisk konsekvens af Sætning 1.

   (a) Vis først og fremmest, at $f(a) \cdot f(b) < 0$ medfører, at

   $$f(a) < 0 \ \wedge \ f(b) > 0 \quad \text{eller} \quad f(a) > 0 \ \wedge \ f(b) < 0.$$

   (b) Konkludér, at hvis $f(a) \cdot f(b) < 0$, så er 0 en værdi mellem $f(a)$ og $f(b)$. Med andre ord: konkludér, at vi har enten $f(a) < 0 < f(b)$ eller $f(b) < 0 < f(a)$.

   (c) Anvend nu Sætning 1 til at konkludere, at f har et nulpunkt i intervallet $[a, b]$.

3. Polynomiet $p(X) = X^3 - X - 6$ blev med vilje valgt på en sådan måde, at det havde den "pæne" rod 2. Vi vælger nu et andet polynomium uden sådanne pæne rødder.

   (a) Betragt polynomiet $f(X) = X^3 - X - 5$, og definér i Python den tilsvarende funktion $f : \mathbb{R} \to \mathbb{R}$, det vil sige funktionen, der opfylder $x \mapsto x^3 - x - 5$.

   (b) Benyt Python til at udregne $f(x)$ for $x \in \{0, 1, 2, 3\}$. Konkludér ved hjælp af Korollar 2 og det faktum, at $f(1) < 0$, og $f(2) > 0$, at polynomiet $f(X) = X^3 - X - 5$ har en rod i intervallet $[1, 2]$. Som nævnt tidligere må du frit benytte, at en polynomiefunktion er kontinuert.

   (c) For et givet interval $[a, b]$ kalder man værdien $(a + b)/2$ for *midtpunktet* af dette interval og værdien $b - a$ for *bredden* af intervallet. For eksempel har intervallet $[1, 2]$ midtpunktet $(1 + 2)/2 = 1.5$ og bredden $2 - 1 = 1$. Udregn nu $f(x)$ i midtpunktet af intervallet $[1, 2]$. Har polynomiet $X^3 - X - 5$ en rod i intervallet $[1, 1.5]$ eller i intervallet $[1.5, 2]$? Bemærk, at disse intervaller har bredden $1/2$, det vil sige halvdelen af bredden af intervallet $[1, 2]$.

   (d) Bestem nu, baseret på værdien af $f(x)$ i midtpunktet af det interval, du lige har fundet, et interval med bredden $1/4$ indeholdende en rod i polynomiet $X^3 - X - 5$.

   (e) Man kan i princippet fortsætte proceduren fra 3 (d) adskillige gange, hvor man hver gang finder et interval, der indeholder en rod i $X^3 - X - 5$, med en bredde, der er halvt så stor som det forrige intervals. Udfør yderligere to trin, og bestem et interval med bredden $1/16$, der indeholder en rod i polynomiet $X^3 - X - 5$.

Idéerne, der blev præsenteret herover, giver anledning til en algoritme, der kan finde approksimationer af rødderne i et polynomium med reelle koefficienter. Mere præcist: givet $p(X) \in \mathbb{R}[x]$ og $a, b \in \mathbb{R}$ sådan at $a < b$, og $p(a) \cdot p(b) < 0$, så kan vi rekursivt definere en sekvens af intervaller $[a_0, b_0], [a_1, b_1], [a_2, b_2], \ldots$ som følger:

$$[a_n, b_n] = \begin{cases} [a, b] & \text{hvis } n = 0 \\ \left[a_{n-1}, \dfrac{a_{n-1} + b_{n-1}}{2}\right] & \text{hvis } n \geq 1 \text{ og } p(a_{n-1}) \cdot p\left(\dfrac{a_{n-1} + b_{n-1}}{2}\right) \leq 0 \\ \left[\dfrac{a_{n-1} + b_{n-1}}{2}, b_{n-1}\right] & \text{hvis } n \geq 1 \text{ og } p(a_{n-1}) \cdot p\left(\dfrac{a_{n-1} + b_{n-1}}{2}\right) > 0. \end{cases}$$

At benytte midtpunkterne i alle de fundne intervaller giver derefter bedre og bedre approksimationer af en rod i polynomiet $p(X)$. Med andre ord: sekvensen af reelle tal $r_0, r_1, r_2, \ldots$ defineret som

$$r_n = \frac{a_n + b_n}{2}$$

er approksimationer af en rod i $p(X)$. Jo større værdien af n er, desto bedre bliver approksimationen. Denne metode til tilnærmelse af rødder kaldes bisektionsmetoden.

4. (a) Benyt Python på polynomiet $f(X) = X^3 - X - 5$ med $[a, b] = [1, 2]$ til at bestemme r_4. Du kan genbruge de intervaller, du allerede har bestemt i den tidligere opgave. Benyt også Python til at udregne $f(r_4)$. Svar: $r_4 = 1.90625$, og $f(r_4) = 0.020660400390625$.

   (b) Antag fortsat, at $f(X) = X^3 - X - 5$, og $[a, b] = [1, 2]$, og vis ved induktion på n, at bredden af intervallet $[a_n, b_n]$ er lig med $1/2^n$ for alle $n \in \mathbb{Z}_{\geq 0}$. En konsekvens af dette er, at afstanden fra r_n til en rod i $f(X)$ højst er $1/2^{n+1}$.

Bemærkning 3 *Bemærk som en lille advarsel, at man i en realistisk anvendelse måske kun kender koefficienterne af $p(X)$ til en vis numerisk præcision. Hvis $(a_{n-1} + b_{n-1})/2$ allerede er tæt på en rod i $p(X)$, kan det derfor være umuligt pålideligt at bestemme, om $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2) > 0$ eller $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2) \leq 0$, og man kan ikke være sikker på andet end, at $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2)$ er meget tæt på nul. Sådanne overvejelser er emner for numerisk analyse og er vigtige, når man udvikler såkaldt numeriske algoritmer.*

Del II: Newton-Raphsons metode til bestemmelse af rødder i polynomier med reelle koefficienter

Hvis alt gik godt, fandt du i det forrige, at polynomiet X^3-X-5 har en rod, der er indeholdt i intervallet $[1.875, 1.9375]$. Midtpunktet i dette interval er derfor en rimelig approksimation for en rod i $X^3 - X - 5$. Som allerede udregnet i den forrige opgave er dette midtpunkt lig med $(1.875+1.9375)/2 = 1.90625$. Vi vil nu behandle en anden måde, hvorpå man kan finde numeriske tilnærmelser af rødderne i et polynomium $p(X) \in \mathbb{R}[X]$.

5. Lad $p(X) \in \mathbb{R}[X]$ være et polynomium af grad mindst én. Desuden antages det, at $\lambda \in \mathbb{R}$ er en rod i $p(X)$. Som før definerer vi den funktion, som dette polynomium giver anledning til, ved $p : \mathbb{R} \to \mathbb{R}$, hvor $x \mapsto p(x)$.

   (a) Antag, at λ er en rod i $p(X)$ med multiplicitet m, hvor m er mindst to. Benyt Definition 4.6.1 i lærebogen til at konkludere, at der i dette tilfælde eksisterer et polynomium $g(X)$, således at $f(X) = (X-\lambda)^m \cdot g(X)$. Benyt nu antagelsen om, at m er mindst to, til at udlede, at der findes et polynomium $h(X)$, således at $f(X) = (X-\lambda)^2 \cdot h(X)$.

   (b) For et givet polynomium

   $$f(X) = a_0 + a_1X + a_2X^2 + \cdots + a_nX^n$$

   defineres den afledte af $f(X)$, betegnet ved $f'(X)$ eller sommetider også $f(X)'$, på sædvanlig måde som

   $$f'(X) = a_1 + 2a_2X + \cdots + na_nX^{n-1}.$$

   Vis, at hvis λ er en rod i $f(X)$ med multiplicitet mindst to, så er λ en rod i $f'(X)$. Hint: udnyt, at $f(X) = (X-\lambda)^2 \cdot h(X)$ for et polynomium $h(X)$, og benyt produktreglen for differentiation.

   (c) Antag, at λ er en rod i $f(X)$ med multiplicitet én. Vis, at der i dette tilfælde eksisterer et polynomium $g(X)$, således at $f(X) = (X-\lambda) \cdot g(X)$, og $g(\lambda) \neq 0$.

   (d) Vis, at hvis λ er en rod i $f(X)$ med multiplicitet én, så er $f'(\lambda) \neq 0$. Hint: benyt igen produktreglen til at finde et udtryk for $f'(x)$, denne gang på produktet $f(X) = (X-\lambda) \cdot g(X)$. Udnyt derefter det faktum, at $g(\lambda) \neq 0$.

En rod i et polynomium $f(X)$ med multiplicitet én kaldes også en *simpel* rod. I en mere generel sammenhæng har vi, at hvis $f : \mathbb{R} \to \mathbb{R}$ er en differentiabel funktion, så kaldes et element $\lambda \in \mathbb{R}$ et *simpelt nulpunkt* i f, hvis $f(\lambda) = 0$, og $f'(\lambda) \neq 0$.

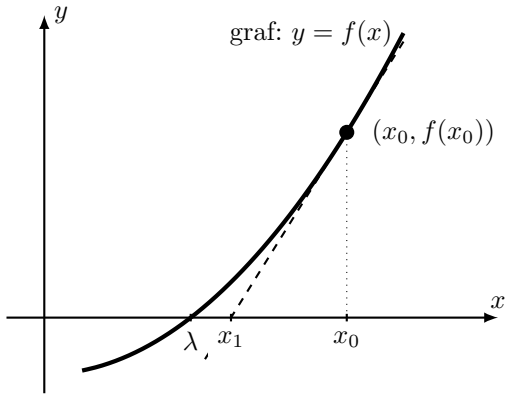
6. Lad $f : \mathbb{R} \to \mathbb{R}$ være en differentiabel funktion, og lad $x_0 \in \mathbb{R}$ være givet. Du må gerne antage, at $f'(x_0) \neq 0$.

   (a) Tjek, at tangentlinjen til grafen af f i punktet $(x_0, f(x_0))$ er givet ved ligningen $y = f'(x_0) \cdot (x - x_0) + f(x_0)$.

   (b) Vis, at skæringspunktet mellem denne tangentlinje og x-aksen er givet ved punktet $(x_1, 0)$, hvor

   $$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Hvis det indledende "gæt" x_0 er tæt på λ, så vil x_1 typisk være tættere på λ end x_0. Se den følgende figur for en illustration.



Idéen med Newton-Raphson-metoden (ofte kaldet Newton-Raphson-algoritmen) er nu at udføre iteration med denne procedure: når $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$ er beregnet, kan man definere $x_2 = x_1 - \dfrac{f(x_1)}{f'(x_1)}$. Hvis startpunktet x_1 er tættere på λ end x_0, vil man forvente, at x_2 er endnu tættere! Søger vi en endnu bedre approksimation af λ, kan vi ganske enkelt gentage proceduren nogle gange. Mere formelt finder vi en sekvens af reelle tal $x_0, x_1, x_2, \ldots$ der er rekursivt defineret som følger:

$$x_n = \begin{cases} x_0 & \text{if } n = 0 \\ x_{n-1} - \dfrac{f(x_{n-1})}{f'(x_{n-1})} & \text{if } n \geq 1 \end{cases}$$

Det viser sig, at følgende gælder:

Sætning 4 *Lad $f : \mathbb{R} \to \mathbb{R}$ være en differentiabel funktion, og antag, at λ er et simpelt nulpunkt i f. Så findes der et reelt tal $\epsilon > 0$, således at de reelle tal x_n, der er defineret ovenfor, for ethvert $x_0 \in]\lambda - \epsilon, \lambda + \epsilon[$ vil konvergere mod λ for voksende n.*

Formelt siger man, at sekvensen $x_0, x_1, x_2, \ldots$ *konvergerer* mod λ, eller med andre ord, at $\lim_{n\to\infty} x_n = \lambda$. Konvergens af sekvenser er et emne for andre kurser, og vi vil kun anvende denne terminologi på en uformel måde. Lad os afprøve Newton-Raphson-algoritmen i et specifikt eksempel.

7. Lad $f : \mathbb{R} \to \mathbb{R}$ være defineret ved $f(x) = x^3 - x - 5$, og lad $x_0 = 2$.

   (a) Udregn x_1 og x_2 ved hjælp af Python direkte fra formlerne $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$ og $x_2 = x_1 - \dfrac{f(x_1)}{f'(x_1)}$.

   (b) For udregning af yderligere værdier af x_n er det mere bekvemt at benytte sig af en rekursivt defineret funktion $F : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, hvor $F(n) = x_n$. For at definere den skal vi i Python først definere funktionen $f : \mathbb{R} \to \mathbb{R}$ med $x \mapsto x^3 - x - 5$ samt funktionen

$fprime : \mathbb{R} \to \mathbb{R}$ med $x \mapsto 3x^2 - 1$. Bemærk, at $fprime(x) = f'(x)$. Disse funktioner kan defineres i Python på følgende måde:

```
def f(x):
 return x**3 - x - 5
def fprime(x):
 return 3*x**2 - 1
```

Vi sætter nu x_0 lig med 2 og definerer funktionen $F : \mathbb{Z}_{\geq 0} \to \mathbb{R}$, hvor $F(n) = x_n$, på en rekursiv måde som følger i Python:

```
x0=2
def F(n):
 if n==0:
  return x0
 else:
  return (F(n-1)-f(F(n-1))/fprime(F(n-1)))
```

Indtast nu koden givet ovenfor i Python for at sætte x_0 lig med 2 og for at definere funktionerne f, $fprime$ og F. Verificér ved hjælp af Python, at $F(0)$, $F(1)$ og $F(2)$ giver de rigtige output: $F(0) = 2$, $F(1) = x_1$ og $F(2) = x_2$. Du kan sammenligne med værdierne af x_1 og x_2, som du allerede har udregnet tidligere af denne opgave.

(c) Benyt den rekursivt definerede funktion F til at beregne x_3 og x_4.

(d) Udregn x_5, og sammenlign med x_4. Hvilken sandhedsværdi har det logiske udtryk `F(4)==F(5)` ifølge Python? Har du kommentarer til Pythons svar?

(e) Husk fra tidligere, at man ved anvendelse af bisektionsmetoden efter fire iterationer opnåede approksimationen $r_4 = 1.90625$ for en rod i $f(X) = X^3 - X - 5$, som opfylder $f(r_4) = 0.020660400390625$. Udregn nu $f(x_4)$. Hvilken metode giver den bedste approksimation af en rod i $X^3 - X - 5$ efter fire iterationer: bisektionsmetoden eller Newton-Raphson-algoritmen?

Bemærkning 5 *Man kan vise, at (groft sagt) med hver iteration i bisektionsmetoden øges antallet af korrekte decimaler med den samme mængde. Omvendt, hvis dit valg af startværdi x_0 er rimeligt tæt på en rod (og den rod har multiplicitet én), så fordobles antallet af korrekte decimaler i hver iteration ved brug af Newton-Raphson-metoden! I praksis kombinerer man begge metoder: først anvendes bisektionsmetoden med kun få iterationer for at opnå en rimelig approksimation af en rod, hvorefter Newton-Raphson-metoden anvendes til hurtigt at finde en meget god approksimation af denne rod.*

Temaøvelse 2 er slut.

For dem, der ønsker mere, er her et par ekstra spørgsmål, men det er helt frivilligt at lave dem!

Vi vil nu inddrage komplekse tal, så vi starter med at indlæse samme pakke som i Temaøvelse 1.

```
import cmath
```

Blot som en påmindelse kan det komplekse tal $2 + 3i$ defineres som følger:

```
z = complex(2,3)
```

Det komplekse tal kan nu vises nemt ved at skrive:

```
z
```

Bemærk, at z vises af Python som $2 + 3j$. Altså foretrækker kommandolinje-Python at anvende 'j' i stedet for 'i'.

8. Vi har set et eksempel, hvor Newton-Raphsons algoritme fungerede ret godt. Dens svaghed er, at det indledende gæt x_0 skal være et godt gæt. Lad os betragte et eksempel, der demonstrerer dette. Vi vælger i denne øvelse $g : \mathbb{C} \to \mathbb{C}$ defineret ved $g(x) = x^3 - 2x^2$.

   (a) Vælg $x_0 = -1$, og benyt Python til at udregne x_n for nogle små værdier af n. Konkludér ligesom før, at sekvensen $x_0, x_1, x_2, \ldots$ ser ud til at nærme sig en rod i polynomiet $X^3 - 2X^2$.

   (b) Vælg nu $x_0 = 0$, og benyt Python til at (forsøge at) udregne x_1. Hvad går galt?

   (c) Lad os vælge $x_0 = 0.1$ for at undgå division med nul i det første trin. Udregn x_n for n op til 10. Ser sekvensen $x_0, x_1, x_2, \ldots$ ud til at konvergere?

9. Som et sidste eksempel viser vi, at Newton-Raphsons algoritme også kan finde approksimationer af komplekse (simple) rødder i polynomier. Vi vælger i denne opgave $h : \mathbb{C} \to \mathbb{C}$ defineret ved $h(z) = z^3 - 1$.

   (a) Rødderne i polynomiet $Z^3 - 1$ er de komplekse tal 1, $-\frac{1}{2} + \frac{1}{2}\sqrt{3} \cdot i$ og $-\frac{1}{2} - \frac{1}{2}\sqrt{3} \cdot i$. Verificér dette ved hjælp af teorien om binomiske ligninger.

   (b) Vælg nu $x_0 = i$, og benyt Python til at udregne x_n for nogle små værdier af n. Hvilken rod i $Z^3 - 1$ konvergerer sekvensen $x_0, x_1, x_2, \ldots$ imod?

   (c) Eksperimentér med værdien af x_0, og find en værdi, for hvilken den resulterende sekvens $x_0, x_1, x_2, \ldots$ konvergerer imod roden $-\frac{1}{2} - \frac{1}{2}\sqrt{3} \cdot i$.