KU LEUVEN kulak

Bachelor in de fysica
Bachelor in de informatica
Bachelor in de wiskunde
Ingenieurswetenschappen

Examenoefeningen MATLAB

Academiejaar 2018 – 2019

Numerieke Wiskunde

Oefening 1

Beschouw volgend MATLAB-script

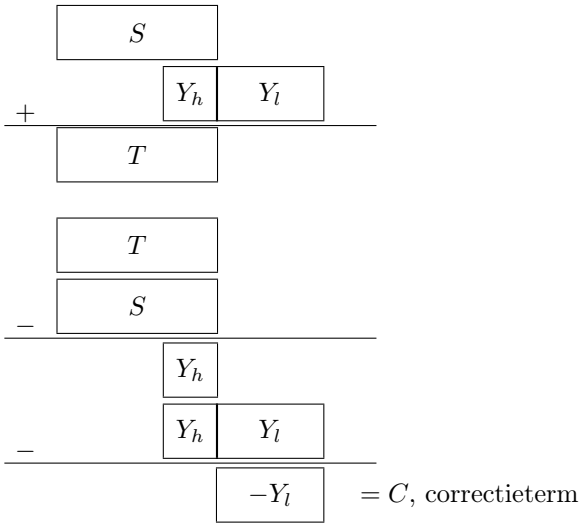
```
x = 0.1;
y = a*x;
if (y == b)
  disp('y is gelijk aan b')
else
  disp('y is niet gelijk aan b')
end
```

1. Implementeer deze code in het bestand `nauwkeurigheid.m`. Zorg hierbij dat je eenvoudig de parameters a en b kan instellen.
2. Voer de code uit met $(a, b) = (1, 0.1)$, $(a, b) = (2, 0.2)$, $(a, b) = (3, 0.3)$, $(a, b) = (4, 0.4)$ en $(a, b) = (5, 0.5)$.
3. Bepaal welke van alle bovenstaande getallen exact kunnen voorgesteld worden in Matlab en welke niet.
4. Maak een grondige foutenanalyse voor dit probleem.
5. Verklaar de output die je verkreeg in het tweede puntje voor elk van de koppels (a, b). (Hint: gebruik de antwoorden van punt 3 en 4.)
6. Pas de derde regel van het programma op een zinvolle manier aan, zodat je steeds het verwachte antwoord krijgt voor alle mogelijke koppels (a, b). Verantwoord je voorstel!

Oefening 2

Wanneer we de som van n getallen willen berekenen, kunnen we dit op verschillende manieren doen. De ene manier is al beter dan de andere...

1. Leg uit wat het probleem is met de klassieke manier.
2. Probeer aan de hand van de schematische voorstelling in Figuur 1 te achterhalen wat de achterliggende idee is van het sommatiealgoritme van Kahan.
3. Implementeer het algoritme van Kahan (in het bestand `kahan.m`) op basis van de gegeven pseudocode (Algoritme 1).
4. Toon aan de hand van een goed gekozen voorbeeld aan wat het voordeel is van de methode van Kahan ten opzichte van de klassieke methode.



Figuur 1: Schematische voorstelling van het sommatiealgoritme van Kahan.

Algoritme 1 Sommatiealgoritme van Kahan.

$S \leftarrow a_1$
$C \leftarrow 0$
for $j = 2$ to n **do**
  $Y \leftarrow a_j - C$
  $T \leftarrow S + Y$
  $C \leftarrow (T - S) - Y$
  $S \leftarrow T$
end for

Oefening 3

Beschouw de functie

$$f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x) = \frac{(x+1)(x-1)(x-2)}{x^2-4}.$$

1. Implementeer zelf een algoritme om deze functie te evalueren in het bestand `mijnf.m`. Denk hierbij aan de stabiliteit...
2. Maak een plot van deze functie voor $x \in [-5, 5]$ en $y \in [-10, 10]$. (Hint: `ylim`)
3. Bespreek op basis van deze plot de conditie van deze functie.
4. Bespreek de stabiliteit van het geïmplementeerde algoritme zo nauwkeurig mogelijk.
5. Bespreek de conditie van de nulpunten van deze functie. Plot hiervoor een rode lijn op de x-as op dezelfde figuur als de functie.

Oefening 4

1. Implementeer de methode van Newton-Raphson voor het vinden van een nulpunt van een functie f.
   Deze functie volgt de signatuur `ws = newtonraphson(f, df, start, tol)` met

   - Input `f` is een function handle met het voorschrift van de functie waarvan een nulpunt gezocht wordt.
   - Input `df` is een function handle met het voorschrift van de afgeleide van de functie waarvan een nulpunt gezocht wordt. (Je berekent deze functie dus analytisch, definieert een function handle `df` en geeft die mee als input.)
   - Input `start` is een startwaarde.
   - Input `tol` is een gewenste tolerantie. Indien de gebruiker geen tolerantie doorgeeft aan de functie, gebruik je de standaardtolerantie $\varepsilon = 10^{-6}$.
   - Output `ws` bevat de benaderingen *uit elke stap* voor de gevonden wortel.

   Bouw zeker een veiligheid in om een oneindige lus te voorkomen.

2. Gebruik vervolgens de door jou geïmplementeerde code op de functie

   $$f(x) = x \sin x$$

   met volgende startwaardes: $x_0 = 2$ en $x_0 = 7$.

3. Plot de relatieve fout voor beide startwaardes in functie van de index van de iteratiestap.

4. Bespreek de bekomen plot.

Oefening 5

Evalueer een veelterm $p_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ in een complex getal $\alpha + \beta i$.

1. Pas de pseudocode voor het dubbelrijig Hornerschema (Algoritme 2) aan zodat er enkel strikt positieve indices gebruikt worden aangezien Matlab bij 1 begint te indexeren.

2. Implementeer het dubbelrijig Hornerschema in het bestand `dubbelhorner.m`.

3. Voorzie in deze code ook twee variabelen die het aantal optellingen en vermenigvuldigingen exact bijhouden. (De implementatie hiervan is niet beschreven in de pseudo code van Algoritme 2.)

4. Pas de methode toe om de veelterm $p(x) = x^7 + 3x^4 - x^3 + 4x - 3$ te evalueren in $x = 2 - i$. Leg uit hoe je de correctheid van je code test.

5. Vergelijk het experimenteel aantal optellingen en vermenigvuldigen (= de waarde van de geïmplementeerde variabelen) met het theoretisch verwachte aantal optellingen en bewerkingen.

Algoritme 2 Dubbelrijig Hornerschema.

$b_{-1} \leftarrow 0$
$b_0 \leftarrow a_0$
$\rho \leftarrow 2\alpha$
$\mu \leftarrow -\left(\alpha^2 + \beta^2\right)$
for $k = 1$ to n **do**
  $b_k \leftarrow a_k + \rho b_{k-1} + \mu b_{k-2}$
end for
$A \leftarrow b_n - \alpha b_{n-1}$
$B \leftarrow \beta b_{n-1}$
output $\leftarrow A + Bi$

Oefening 6

In deze cursus hebben we zowel directe als iteratieve methodes gezien om een stelsel op te lossen. De iteratieve methodes die besproken werden, zijn de methode van Jacobi en de methode van Gauss-Seidel. Beide methodes werden geïmplementeerd en toegepast op een bepaald stelsel in `linstelsel.m`.

Bij deze oefening geldt dat het aantal bewerkingen gelijk is aan de som van het aantal optellingen en het aantal vermenigvuldigingen.

Het is in deze oefening de bedoeling dat je het bestand `linstelsel.m` aanpast om op onderstaande vragen te kunnen antwoorden. In het geval er door een bepaalde aanpassing iets niet meer zou werken, kan je de oorspronkelijke code terugvinden in `linstelsel_template.m`.

1. Voer het script linstelsel uit en zorg dat je begrijpt wat er gebeurt.

2. Toon met behulp van een voorwaarde aan dat er zeker convergentie zal optreden bij beide methodes voor het stelsel dat beschouwd wordt in de code.

3. Bepaal een benadering voor de complexiteit van het iteratievoorschrift van beide methodes. (Je kijkt hiervoor enkel naar de regel waarnaast in commentaar *iteratievoorschrift* staat. Je moet geen rekening houden met de lus die rond deze regel staat.)

4. Maak een plot van de nauwkeurigheid in functie van de stap k voor beide methodes op dezelfde figuur.

5. Bespreek de bekomen plot.

Oefening 7

In deze opgave beschouwen we de methode van Crout om de LU-factorisatie van een n-bij-m matrix te berekenen met optimale pivotering (Algoritme 3).

1. Implementeer de de pseudo-code van Algoritme 3.
   Hanteer hierbij de signatuur `[C, ind] = my_crout(C)`, waarbij `ind` en vector is met informatie over de rijwisselingen.
   Hints:
   - Implementeer eerst letterlijk de pseudo-code, voeg pas daarna de `ind` uitvoerparameter toe.
   - $\sum_{t=1}^{i-1} c_{jt}c_{ti}$ en $\sum_{l=1}^{i-1} c_{il}c_{lj}$ zijn vector-vector producten.
   - Je kan een lus afbreken (stoppen) door gebruik te maken van het commando `return`.
2. Schrijf een tweede functie met signatuur `[L, U, P] = croutfactors(C, ind)` die de matrices `L`, `U` en `P` bepaalt op basis van uit de uitvoerparameters van de functie `my_crout` zodat $LU = PC$. (Hints: `triu`, `tril`, `eye`). (Merk op dat je deze tweede functie kan implementeren voor een willekeurige matrix `C`, ook indien `my_crout` niet werkt.)
3. Test je code. (Of, indien je code niet werkt, leg uit hoe je je code kan testen.)
4. Definieer een stelsel met 5 vergelijkingen en 5 onbekenden (Hint: `rand`), dat je vervolgens oplost door gebruik te maken van de door jou geïmplementeerde code.

Algoritme 3 Factorisatie volgens Crout met optimale pivotering.

for $i = 1$ to n **do**
  $k \leftarrow i$
  $s \leftarrow 0$
  for $j = i$ to n **do**
    $c_{ji} \leftarrow c_{ji} - \sum_{t=1}^{i-1} c_{jt}c_{ti}$
    if $|c_{ji}| > s$ **then**
      $s \leftarrow |c_{ji}|$
      $k \leftarrow j$
    end if
  end for
  if $s = 0$ **then**
    stop
  end if
  if $k \neq i$ **then**
    verwissel rij i en k
  end if
  for $j = i + 1$ to m **do**
    $c_{ij} \leftarrow \left(c_{ij} - \sum_{l=1}^{i-1} c_{il}c_{lj}\right) / c_{ii}$
  end for
end for

Oefening 8

Als een computerarchitectuur het delen van twee getallen niet ondersteunt, kan men toch de inverse van een getal berekenen op een iteratieve manier (uiteraard mag de iteratieformule dan ook geen deling bevatten!). Om $1/a$ te berekenen kunnen we bijvoorbeeld het nulpunt van $f(x) = a - 1/x$ zoeken, aangezien

$$f(x) = a - \frac{1}{x} = 0 \quad \Leftrightarrow \quad x = \frac{1}{a}.$$

Implementeer alle code voor deze oefening in het bestand `mijndeling.m`.

1. Welke formule levert Newton-Raphson voor deze vergelijking?
2. Maak een plot van de gevonden substitutiemethode voor $a = 2$. Plot op dezelfde figuur ook de eerste bissectrice.
3. Is de methode consistent? Is ze reciprook consistent?
4. Voor welke startwaarden is er convergentie naar de correcte waarde $1/a$?
5. Wat gebeurt er als $a = 0$?
6. Implementeer de gevonden methode om een deling iteratief te berekenen tot de oplossing 14 juiste cijfers heeft. Houd hierbij rekening met het geval $a = 0$.
7. Maak een plot van de fout in functie van de index van de iteratiestap voor $a = 7$ met startwaarde $x_0 = 0.1$. Plot op dezelfde figuur ook de theoretisch verwachte convergentie.

Oefening 9

We wensen alle nulpunten van een veelterm te vinden met de methode van Newton-Raphson. Om dit te doen willen we gebruik maken van deflatie. Je kan hierbij gebruik maken van de functie `newtonraphsonpoly.m`, die één nulpunt van een veelterm kan vinden.

Hint: Je kan inspiratie vinden in `newtonraphsonpoly.m` voor de verdere implementatie.

1. Neem de functie `newtonraphsonpoly.m` door, zodat je weet hoe je deze moet gebruiken.
2. Implementeer een functie `newtonraphsondeflate` met als output alle wortels van de opgegeven polynoom. De inputparameters mag je zelf kiezen.
3. Gebruik de door jou geïmplementeerde functie om alle wortels van de veelterm $x^6 + 1$ te berekenen. Gebruik als startwaarde $1 + i$.
4. Plot de gevonden wortels in het complexe vlak.
5. We bekomen hier complexe getallen als oplossing. Bepaal een maat om de nauwkeurigheid van een complex getal te bepalen. Met andere woorden, definieer de fout op een complex getal. Verantwoord de gedefinieerde maat, en verduidelijk met een plot.

Hint bij het implementeren: vergeet niet dat Matlab bij 1 begint te indexeren.

Oefening 10

Omdat het niet eenvoudig is een gepaste integratieregel te vinden voor een gevraagde nauwkeurigheid, maakt men in de praktijk vaak gebruik van een automatische integratiemethode. In dat geval wordt een samengestelde regel gebruikt dat op een steeds fijner gediscretiseerd interval wordt ingezet.

1. Implementeer automatische integratie met de regel van Simpson (Algoritme 4) in het bestand `autosimp.m`. Verantwoord de gekozen inputparameters.
2. Kies hierbij een eigen stopcriterium, en bespreek je keuze.
3. Pas je code toe op de functie $f(x) = x\sin(x)$ en het interval $[0, 5]$.
4. Plot de relatieve fout bij dit voorbeeld in functie van de gebruikte stapgrootte h en bespreek.

Algoritme 4 Automatische integratie met regel van Simpson.

$h \leftarrow \frac{b-a}{2}$
$S_0 \leftarrow f\left(\frac{b+a}{2}\right)$
$I \leftarrow \frac{h}{3}\left[f(a) + f(b) + 4S_0\right]$
$k \leftarrow 0$
while $k < K_{\max}$ en eigen stopcriterium niet voldaan **do**
  $h \leftarrow \frac{h}{2}$
  $S \leftarrow S_0$
  $S_0 \leftarrow 0$
  for $j = 1$ to 2^{k+1} **do**
    $x \leftarrow a + (2 * j - 1)h$
    $S_0 \leftarrow S_0 + f(x)$
  end for
  $I \leftarrow \frac{1}{2}I + \frac{h}{3}\left(4S_0 - 2S\right)$
  $k \leftarrow k + 1$
end while