Oefeningen Numerieke Wiskunde

Oefenzitting 4 (PC): Bewegende kommavoorstelling en benaderings- en afrondingsfouten

In deze oefenzitting maak je gebruik van Matlab om kennis te maken met de bewegende kommavoorstelling en het verschil tussen benaderingsfouten en afrondingsfouten. De m-bestanden die je nodig hebt, kan je vinden op Toledo.

1 Matlab

Bij het werken met MATLAB krijg je op het scherm telkens een prompt. Dit is: `>>` . Hierachter kan je de gewenste bevelen intikken. Als je op de enter-toets drukt, wordt het bevel uitgevoerd. Merk op dat je enkel achtereenvolgende commando's kan ingeven. Het is niet mogelijk om bijvoorbeeld met de muis op een eerder uitgevoerd commando te klikken om dit te wijzigen. Er bestaat wel de mogelijkheid om met de pijltjes toetsen te navigeren doorheen de commando's die je eerder ingegeven hebt. Zo kan je eenvoudig een commando opnieuw onder de cursor halen en eventueel aanpassen.

MATLAB bevat een uitgebreide helpfunctie. Door het intypen van

```
>> doc
```

verschijnt het helpvenster, dat ook een ingebouwde zoekfunctie heeft. Je kan met dit commando ook de documentatie bekijken van een bepaalde functie, bvb.

```
>> doc exp
```

Een alternatief is het commando

```
>> help exp
```

De documentatie verschijnt hierdoor in de Command Window waarin je aan het werken bent, wat vaak handiger is.

2 Bewegende kommavoorstelling

Probleem 1. (Berekening van ϵ_{mach}) Gebruik `bepaalb.m` om de basis te berekenen van het talstelsel waarmee MATLAB werkt. Met `bepaalp.m` kan je het aantal cijfers in

de mantisse berekenen. Bekijk deze bestanden. Zet de `fprintf` regels uit commentaar (verwijder het `%` teken) en roep beide functies opnieuw aan.

Vergelijk de berekende waarde ϵ_{mach} (formule handboek) met de voorgedefinieerde variabele `eps` in MATLAB. Opgelet: `eps` stelt niet de machineprecisie voor, maar wel het verschil tussen 1 en het kleinste getal groter dan 1, voorstelbaar in de floating point-voorstelling. Controleer dit door 1 af te trekken van $(1+\mathbf{eps})$ en daarna 1 af te trekken van $(1+\frac{\mathbf{eps}}{3})$. Wat gebeurt er als je 1 aftrekt van $(1+0.70*\mathbf{eps})$ en hoe verklaar je dit?

Probleem 2. (Floating Point) De functie `dumpfp(x)` geeft de binaire floating point voorstelling van een decimaal getal x terug zoals op een i386 architectuur. Het geeft dus terug hoe je computer getallen bitsgewijs bijhoudt. Gebruik `dumpfp` voor de getallen:

0.125, 0.25, 0.5, 1, 2, 4, 8 en

0.001, 0.01, 0.1, 1, 10, 100, 1000.

Welke getallen worden correct bijgehouden? Welke niet? Waarom? Hoe groot is de fout ongeveer op de voorstelling van 0.1? Maak gebruik van de binaire voorstelling.[1] En als je zou werken met een geheugencel waarvoor de mantisse slechts 3 bits telt?

3 Numerieke differentiatie: benaderingsfouten en afrondingsfouten

Probleem 3. Een differentiequotiënt is een eenvoudige benadering voor de afgeleide $f'(x)$ van een functie $f(x)$. In deze oefening maken we een Matlab script waarin we de afgeleide van de functie $f(x) = 5e^x$ in $x = 0$ benaderen met volgende differentieformules

$$y_1 = \frac{f(x+h) - f(x)}{h}, \tag{1}$$

$$y_2 = \frac{f(x+h) - f(x-h)}{2h}. \tag{2}$$

Open een nieuw script en sla het op als `numdiff.m`. Je voert de gebruikte commando's best eerst eens uit in de Command Window, zodat je weet wat ze precies doen, waarna je ze in het script kan kopiëren. Als je in het script de uitvoer wilt onderdrukken, plaats je een `;` na het commando.

Benadering 1 We evalueren formule (1) in steeds kleiner wordende waarden van h

```
>> for k=1:15, h(k) = 10^(-k); end
```

We kunnen nu een vector opstellen van opeenvolgende benaderingen met formule (1)

```
>> x = 0
>> y1 = (5*exp(x+h)-5*exp(x))./h
```

[1] Beschouw enkel de eerste verwaarloosde bit.

Merk op dat we hier twee vectoren elementsgewijs door mekaar delen.

Bereken met de hand de waarde van $f'(0)$ voor de gegeven functie en sla deze op in de variabele `y`. We bepalen dan de relatieve fout van de benaderingen `y1` als

```
>> error1 = abs(y1-y)/abs(y)
```

Bekijk de waarde van `y1` en de relatieve fout in de Command Window. Wat valt je op? Je gebruikt best

```
>> format long
>> y1'
>> error1'
```

om een goed idee te krijgen van het aantal juiste beduidende decimale cijfers. Wat is het maximum aantal juiste beduidende cijfers dat je bekomt?

Maak nu ook een plot van de fout. Open een nieuwe figuur

```
>> figure
```

en plot de fout in functie van de waarde van h

```
>> plot(h,error1)
```

Kan je iets afleiden uit de figuur? Het probleem is dat de waarden van h niet lineair gekozen zijn, maar logaritmisch, en dat de waarde van de fout sterk varieert. Plot de fout opnieuw met

```
>> loglog(h,error1)
```

Gebruik `help loglog` om te weten wat dit doet.

(a) Kan je nu iets afleiden uit de figuur?

(b) Identificeer de afrondingsfouten en de benaderingsfouten.

Benadering 2 Vul nu je script aan zodat je ook opeenvolgende benaderingen berekent met formule (2). Bereken de relatieve fout, analyseer het aantal juiste beduidende decimale cijfers en maak een plot van de fout. Je kan deze plot in het rood toevoegen aan dezelfde figuur met

```
>> hold on
>> loglog(h,error2,'r')
```

Orde van de benaderingsfout Een benadering $g(h;x)$ voor $f'(x)$ is van orde k als

$$g(h;x) = f'(x) + O(h^k), \quad \text{voor} \quad h \to 0. \tag{3}$$

Dit betekent dat de fout van de benadering afneemt zoals h^k wanneer $h \to 0$.

(a) Wat is de orde van de benaderingen y_1 en y_2? Ga dit experimenteel na door h^k in functie van h te plotten in dezelfde figuur voor verschillende k waarden. Dit kan met

```
loglog(h,h.^k,'--k')
```

(De `'--k'` betekent 'zwarte stippellijn'.)

(b) Waarom is de grafiek van h^k in functie van h een rechte?

(c) Bestudeer het volgende wiskundige bewijs voor de orde van benadering (1), gebruik makend van een Taylorreeksontwikkeling.

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f^{(3)}(x)}{3!}h^3 + \dots \\ &= f(x) + f'(x)h + O(h^2) \end{aligned}$$

en bijgevolg

$$y_1 = \frac{f(x+h) - f(x)}{h} = f'(x) + O(h), \qquad h \to 0.$$

Komt dit overeen met je observaties?

(d) Geef zelf het bewijs voor y_2. Gebruik hiervoor de Taylorreeksen van $f(x+h)$ en $f(x-h)$ rond x en vul deze in in formule (2).

Grootte-orde van de afrondingsfouten Men kan aantonen dat de afrondingsfouten voor beide formules zich gedragen als

$$\left| \frac{\bar{y} - y}{y} \right| = O(h^{-1})\epsilon, \qquad h \to 0,$$

met $|\epsilon| \leq \epsilon_{mach}$. Ga na dat dit klopt door de functie $h^{-1}\epsilon_{mach}$ in functie van h in dezelfde figuur te plotten met

```
loglog(h,(eps/2)*h.^(-1),'k--')
```

Conclusie Wat weet je nu over de differentieformules? Kan je deze gebruiken om de afgeleide van een functie te benaderen tot op machineprecisie? Welke formule zou je verkiezen?

4 Extra oefeningen: conditie van de wortel van een veelterm

Zij α een enkelvoudige wortel van een veelterm $p(x)$ en $\bar{\alpha} = \alpha + \Delta\alpha$ de naburige wortel van de geperturbeerde veelterm $\bar{p}(x) = p(x) + \Delta p(x)$. Dan is

$$\Delta\alpha \approx -\frac{\Delta p(\alpha)}{p'(\alpha)}. \tag{4}$$

In Matlab berekent het commando `roots` de wortels van een veelterm met gegeven coëfficiënten. Omgekeerd berekent het commando `poly` de coëfficiënten van de veelterm met gegeven wortels.

Opgaven

1. Beschouw de veelterm in x met parameter t

$$x^2 - 2x + t$$

   (a) Zij $t = -3$. Breng een fout $\Delta t = 10^{-6}$ aan op t en controleer de foutenschatting (4) voor de wortel $\alpha = 3$.

   (b) Zij $t = 1$. De veelterm heeft nu een dubbele wortel $\alpha = 1$. Observeer dat de wortel nu veel gevoeliger is voor een fout op t.

2. Construeer de veelterm met de 6-voudige wortel $\alpha = 2$ en bereken numeriek zijn wortels (in een vector r). Hoe nauwkeurig zijn ze? Maak een figuur met deze wortels in het complexe vlak met het matlab-bevel `plot(real(r),imag(r),'b*')`.

3. De **veelterm van Wilkinson** is gekend als

$$(x-1)(x-2)\cdots(x-20) = x^{20} - 210x^{19} + \cdots + 20!.$$

   Construeer deze veelterm met het commando `poly(1:20)` en bereken de nulpunten. Liggen ze dicht bij wat je verwacht? Verander dan de coëfficiënt van x^{19} in $-210 + 2^{-23}$, bereken opnieuw de nulpunten en vergelijk deze met de nulpunten die je eerst hebt berekend. Wat besluit je?
   Dit is een voorbeeld van een zeer slecht geconditioneerde veelterm. Veeltermen van hoge graad zijn dikwijls zeer gevoelig voor kleine wijzigingen aan de coëfficiënten.
   Als je weet dat de functie `roots` stabiel is, kan je dan verklaren wat er mis gaat bij het berekenen van de nulpunten van de Wilkinson veelterm?