Oefeningen Numerieke Wiskunde

Oefenzitting 3 (PC): Bewegende kommavoorstelling en foutenanalyse

Voor het uitwerken van de opgaven heb je de '.m'-bestanden nodig die je kan vinden op Toledo.

1 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 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 dat?

Probleem 2. (Floating Point) Het programma `dumpfp` geeft de binaire floating point voorstelling van een decimaal getal 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[1]? En als je zou werken met een geheugencel waarvoor de mantisse slechts 3 bits telt?

2 Taylorreeks van e^x

De exponentiële functie kan geschreven worden als zijn Taylorreeks

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}\,.$$

[1]Beschouw enkel de eerste verwaarloosde bit.

De reeks convergeert theoretisch gezien voor alle $x \in \mathbb{R}$. In wat volgt gaan we na hoe snel de reeks convergeert en hoe goed de reeks convergeert in de praktijk. We berekenen voor een bepaalde waarde van x de afgebroken Taylorreeks

$$y_n = \sum_{k=0}^{n} \frac{x^k}{k!} .$$

Dit is een goede benadering voor e^x als de termen $\frac{x^k}{k!}$ klein zijn voor $k > n$. Er geldt dan voor de absolute fout dat

$$|e^x - y_n| = O\left(\frac{x^{n+1}}{(n+1)!}\right). \tag{1}$$

Dit geeft de orde-grootte van de fout weer. In MATLAB kan je de termen van de reeks berekenen met

```
t = x.^(0:n)./factorial(0:n)
```

Met het commando `cumsum` kan je nu in één keer $y_0, y_1, \ldots, y_n$ berekenen

```
y = cumsum(t)
```

Maak een nieuw script `probleem_exp.m` waarin je de volgende opdrachten uitvoert:

Probleem 3. Bereken de benaderingen y_k, $k = 0, \ldots, n$ voor $x = 0.1$ en $n = 20$.

(a) Bekijk de vector `y'` (transpose van `y`) met `format long`. Verklaar waarom de waarde van y_k vanaf een bepaalde k niet meer wijzigt. Bekijk hiervoor ook de vector `t'` en gebruik ook eens `format long e`.

(b) Bereken de absolute en de relatieve fout van de uiteindelijke benadering.

(c) Plot de absolute en relatieve fout in functie van k. Welke schaal (`plot`, `semilogx`, `semilogy`, `loglog`) geeft het best weer hoe snel de reeks convergeert?

(d) Leg het verband tussen de grafiek en de convergentie die je theoretisch verwacht. Plot de orde-grootte van de fout in dezelfde figuur met `plot(0:n-1,t(2:n+1),'r--')`.

Probleem 4. Herhaal Probleem 2 voor $x = 0.8$. Wat is er anders? Verklaar waarom de grafiek niet volledig wordt weergegeven tot en met $n = 20$. Is de benadering beter dan voor $x = 0.1$?

Probleem 5. Herhaal Probleem 2 voor $x = 20$ en $n = 100$.

(a) De reeks begint pas na een tijd te convergeren. Verklaar.

(b) Bereken opnieuw de absolute en de relatieve fout van de uiteindelijke benadering.

(c) Is dit numeriek een goede benadering? Zo nee, wat is het probleem?

(d) Bekijk de foutenanalyse van de som van de vorige oefenzitting. Wat is de grootte-orde van de absolute fout in dit geval en komt dit overeen met wat je bekomt?

Probleem 6. Herhaal Probleem 2 voor $x = -20$ en $n = 100$. Ook hier begint de reeks pas na een tijd te convergeren.

(a) Bereken opnieuw de absolute en de relatieve fout van de uiteindelijke benadering.

(b) Is dit numeriek een goede benadering? Zo nee, wat is het probleem?

(c) Bekijk de foutenanalyse van de som van de vorige oefenzitting. Wat is de grootte-orde van de absolute fout in dit geval en komt dit overeen met wat je bekomt?

3 Benadering van een limiet

Maak een script `probleem_limiet.m` waarin je de code van volgende oefening zet.

Probleem 7. We hebben de volgende limiet

$$\lim_{x\to 0} f(x) := \lim_{x\to 0} \frac{1-\cos(x)}{x^2} = \frac{1}{2}\,.$$

(a) Ga de waarde van de limiet na met de regel van Hôpital.

(b) Evalueer $f(x)$ voor $x = 10^k$, waarbij je de exponent k in stapjes van 0.1 laat variëren tussen 0 en -10. Bekijk $f(x)$ als een kolomvector met `format long`. Wat neem je waar?

(c) Plot de absolute fout $|f(x) - 1/2|$ i.f.v. x. Welke schaal (`plot`, `semilogx`, `semilogy`, `loglog`) gebruik je hier best en waarom?

(d) Als we $\cos(x)$ schrijven als zijn Taylorreeks rond $x = 0$, $\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! - \ldots$, dan krijgen we

$$f(x) = \frac{1}{2} - \frac{x^2}{4!} + \frac{x^4}{6!} - \ldots$$

Toon aan dat de fout zich gedraagt als $g(x) = \frac{x^2}{4!}$ en ga dit na door $g(x)$ in dezelfde figuur te plotten met een rode stippellijn.

(e) In de vorige oefenzitting hebben we gezien dat de absolute fout bij het berekenen van $f(x)$ voldoet aan

$$\delta y \approx y\left(-\frac{\cos(x)}{1-\cos(x)}\epsilon_1 + \epsilon_2 - \epsilon_3 + \epsilon_4\right), \qquad |\epsilon_i| \leq \epsilon_{mach}$$

Toon aan dat door het vervangen van $\cos(x)$ door zijn Taylorreeks rond $x = 0$, de dominante term van de fout zich gedraagt als $\sim x^{-2}\epsilon_1$. Ga dit na door $x^{-2}\epsilon_{mach}$ in dezelfde figuur te plotten met een groene stippellijn.

4 Extra oefeningen

Probleem 8. (Een onschadelijke afronding?) Op 25 februari, 1991, tijdens de Golfoorlog, faalt een Amerikaans Patriot raketafweersysteem, operationeel in Dharan, Saudi Arabië, bij het opsporen en onderscheppen van een inkomende Iraakse Scud.[2] De Scud slaat vervolgens in op een legerkazerne, waarbij 28 doden en 98 gewonden vallen. De oorzaak blijkt een onnauwkeurige berekening van de tijd sinds het opstarten, door een afrondingsfout naar een fixed-point getal.

Het raketafweersysteem berekent de positie van de vijandige raket aan de hand van de vorige gemeten positie van de raket, de snelheid van de raket en de tijd. De interne klok van het systeem houdt de tijd sinds het opstarten bij in tienden van seconden als een geheel getal (bv. 250 = 25 sec). Om de nieuwe positie van de raket te berekenen moet de tijd in seconden gekend zijn als een reëel getal. Gewoon een simpele vermenigvuldiging met 0.1 dus...
Er is gegeven dat

- het systeem het getal 0.1 opslaat als een 24-bits binair fixed-point getal. (Hint: 1/8 als 6 bits binair fixed-point getal is 0.00100.)
- het systeem al 100 uur opstaat.
- een scud 1.676 km/s vliegt.

Pas dumpfp nogmaals toe op 0.1.

1. Naar welk getal wordt 0.1 afgekapt en geef een uitdrukking voor de absolute en relatieve fout.
2. Hoe groot is de fout op de berekende tijd? (*antwoord: 0.3433 sec*)
3. Hoe groot is de fout op de berekende positie?

Probleem 9. (Onschadelijke berekening?) Volgende berekeningen uitvoeren voor x

$$
\begin{array}{l}
for\ \ i = 1:40 \\
\quad x = \sqrt{x} \\
end \\
for\ \ i = 1:40 \\
\quad x = x^2 \\
end
\end{array}
$$

laat in theorie elke $x \geq 0$ ongewijzigd, tenminste wanneer er geen afrondingsfouten worden gemaakt.

[2] NAVO-codenaam voor een geleide middellange-afstandsraket van Sovjet makelij.

(a) Voer een foutenanalyse uit. (Hint: doe dit voor beide lussen afzonderlijk, waarbij je voor de tweede lus een relatieve fout δ op het gegeven zet. Deze δ stelt het effect voor van alle afrondingsfouten die gemaakt worden in de eerste lus.)

(b) Vul het script `oefening_forlus.m` aan. Wat is de relatieve fout op het eindresultaat? Komt dit overeen met de foutenanalyse?

(c) Bereken ook de relatieve fouten op de tussenresultaten, waarbij je de op machineprecisie na exacte waarden krijgt in `exacte_x.mat`. Plot deze fouten en verklaar je resultaten. Maak hierbij gebruik van een logaritmische schaal waar nodig.

(d) Stel dat er bij het berekenen van de machten in de tweede lus geen afrondingsfouten zouden optreden. Zou je dan een nauwkeurig eindresultaat krijgen?

Probleem 10. (Evaluatie van een functie) Beschouw de functie

$$f(x) = \frac{e^{x^2} - e^{-x^2}}{2x^2}.$$

(a) Evalueer deze functie voor $x = 10^{-1}, 10^{-2}, \ldots, 10^{-10}$. Vergelijk met de op machineprecisie na exacte waarden in `f_eval_exact.mat`. Plot de relatieve fout en visualiseer dat de fout toeneemt als $\mathcal{O}(x^{-2})$.

(b) Schrijf een Matlab functie om $f(x)$ wel correct te evalueren. (Hint: stelling van Taylor.)

(c) Toon m.b.v. een foutenanalyse aan dat de relatieve fout voor kleine waarden van x toeneeemt als $\mathcal{O}(x^{-2})$.

Probleem 11. (Evaluatie van een functie) Wanneer je de functie

$$f(x) = e^{2x}(1 - \tanh(x))$$

evalueert voor grote waarden van x heb je niet enkel het probleem van gevaarlijke aftrekkingen, maar tevens zal e^{2x} al vlug een overflow genereren. Herschrijf deze uitdrukking zodat beide problemen voorkomen worden.