OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 3 (PC): FOUTENANALYSE

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

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 (commando **edit**). In `bepaalp` wordt de basis bepaald door `bepaalb` op te roepen. We hadden de basis ook als parameter kunnen doorgeven aan de routine.
Vergelijk de berekende waarde ϵ_{mach} (formule handboek) met de permanente 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. (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 afrondings- en afbrekingsfouten gemaakt worden.

Indien je het commando

```
>> load exacte_x.mat
```

uitvoert, bevat de variabele `exacte_x` de exacte waarden op machine-precisie na, die in theorie tijdens het uitvoeren van de eerste lus bekomen worden. Het scriptje '**oefening2.m**' laat dan toe om het verschil tussen deze exacte waarden en de berekende waarden te bepalen en op een grafiek weer te geven. Ook wordt de omgekeerde lus uitgevoerd. Lees de code en verklaar alle gebruikte commando's. Interpreteer de uitvoer. Verklaar hoe de fout zich gedraagt in de twee gevallen. Probeer ook eens een andere schaalverdeling om de fout weer te geven (informatie vind je met het commando **help plot**). Dit zal toelaten om meer informatie over het gedrag van de fout te bekomen.

Probleem 3. (Evaluatie van een functie) Beschouw de functie

$$f(x) = x\left[\frac{\pi}{2} - \arctan(x)\right].$$

Men kan aantonen dat $\lim_{x\to\infty} f(x) = 1$ (oefening).
Schrijf een .m-bestand om deze functie te evalueren.
Evalueer f voor grote waarden van x (bij voorbeeld $x = 10^1, 10^3, 10^4, \ldots$). Wat loopt er fout bij zeer grote x-waarden? Teken hiertoe de grafiek van $\arctan(x)$.

We kunnen $f(x)$ eenvoudig herschrijven om zo te vermijden dat we twee getallen van elkaar aftrekken die ongeveer even groot en van hetzelfde teken zijn: toon aan dat voor $x \geq 0$:

$$f(x) = x \arcsin \frac{1}{\sqrt{1+x^2}}.$$

(Hint: $\tan^2(x) + 1 = \frac{1}{\cos^2(x)}$.) Implementeer deze formule en experimenteer: nu bekomen we wel een nauwkeurig resultaat.

Probleem 4. (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 3-bits register?

Probleem 5. (Een onschadelijke afronding?) Een raketafweersysteem berekent de positie van een 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 "startup" bij in tienden van seconden (bv. 250 = reeds 25s lopend). Om de nieuwe positie van de raket te berekenen moet de tijd echter gekend zijn als een reëel getal in seconden. Gewoon een simpele vermenigvuldiging met 0.1 dus...
Gegeven dat

- het systeem met een 24-bits register werkt.
- het systeem reeds 1000 uur opstaat.
- een scud 1.61 m/s vliegt.

Pas dumpfp nogmaals toe op 0.1.

(1) met welk getal (decimaal) vermenigvuldigt het systeem ipv 0.1?
(2) hoe groot is de (relatieve en absolute) fout?
(3) hoe groot is de fout op de berekende tijd?
(4) hoe groot is de fout op de berekende positie?

Probleem 6*. (Evaluatie van een functie) Wanneer je, zoals in de derde opgave, 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.

Probleem 7*. (Berekenen van de limiet van een reeks) Analytisch kan men nagaan dat

$$\sum_{k=1}^{\infty} k^{-2} \; = \; \pi^2/6 \approx 1.644934066848.$$

Als we dit niet zouden weten en we deze som numeriek willen berekenen, zouden we dit kunnen doen door de som te berekenen voor toenemende k tot het berekende resultaat niet meer wijzigt. Implementeer deze strategie. Vergelijk het bekomen resultaat dan met de exacte waarde. Om dit wel nauwkeurig te berekenen, kan men de som van achter naar voor berekenen, dus eerst de kleinste getallen optellen. Het probleem hierbij is dat men niet weet hoeveel termen men dan moet nemen.

[1] beschouw enkel de eerste verwaarloosde bit