Numeriese Metodes / *Numerical Methods* 324

Rekenaaropdrag 01 / *Computer Assignment* 01 - 2018

Sperdatum / *Deadline* - 16/02/2018

Let asseblief op:

- Opdragte word met die aanvang van die tutoriaalsessie op die sperdatum ingehandig. Laat inhandiging sal gepenaliseer word. Handig gedrukte weergawes in asb, geen elektroniese weergawes word aanvaar nie. **Jou opdrag moet beide jou kode en jou resultate insluit.**
- Alle werk ingehandig moet jou eie werk wees. Jy kan idees uitruil met ander studente, maar alle berekeninge en grafieke moet jou eie wees. **Stappe sal geneem word teen plagiaat.**
- Rekenaaropdragte moet in MATLAB of Python voltooi word. As jy 'n alternatiewe program wil gebruik, maak eers seker dat dit reg is met die dosent.
- Wanneer opdragte ingehandig word, is die voorkeurmetode MATLAB se `publish` of `livescript` funksie, maar dis nie verpligtend nie. However, it is perfectly acceptable to paste your code to an MS Word document or to type it up in LaTeX. Vir `publish` instruksies, sien die volgende video by

Please Note:

Computer assignments are handed in at the start of the tutorial session on the due date. There is a penalty if you hand in late. Please hand in hard copy, no electronic versions are accepted. **You should include both your code and your results.**

All work handed in must be your own work. You may exchange ideas with other students, but all calculations and graphs must be your own. **Action will be taken against plagiarism.**

Computer assignments must be completed using MATLAB or Python. If you would like to use an alternative, please clear this in advance with the course lecturer.

The preferred method for submitting assignments is MATLAB's `publish` or `livescript` function, but this is not compulsory. However, it is perfectly acceptable to paste your code to an MS Word document or to type it up in LaTeX. For `publish` instructions, see the video at

http://appliedmaths.sun.ac.za/media/NM262/publish.mp4

Probleme:

1. (a) Skryf kode om die uitdrukkings vir E_1 en E_2 van bladsy 3 van lesing 2 te bereken en skep die tabel van die selfde bladsy. Jy mag 'n `for` of `while` "loop" gebruik as jy wil, maar inplaas daarvan, probeer om vectorised kode te gebruik.

   Wenk: Jy sal dalk MATLAB se `table` funksie wil gebruik. Tik >> `help table/table` vir instruksies.

   (b) Skep 'n soortgelyke tabel vir die funksie

   $$F_1(x) = \frac{1 - \sec x}{\tan^2 x}$$

   deur 'n alternatiewe vorm, F_2, te vind wat nie blootgestel is aan 'n afrondingsvout naby $x = 0$ nie.

   Wenk: Jy sal dalk MATLAB se `table` funksie wil gebruik. Tik >> `help table/table` vir instruksies.

2. Herhaal vir jouself die kwadratiese probleem op bladsy 2 van lesing 2 deur gebruik te maak van MATLAB of Python. Gebruik die ingeboude `roots` funksie om die antwoord op die lesings bladsye te bevestig.

Problems:

1. (a) Write code to evaluate the expressions E_1 and E_2 from slide 3 of Lecture 2 and hence reproduce the table from the same slide. You may use a `for` or `while` loop if you wish, but try instead to use vectorised code.

   Hint: You may want to use MATLAB's `table` function. Type >> `help table/table` for instructions.

   (b) Produce a similar table for the function

   $$F_1(x) = \frac{1 - \sec x}{\tan^2 x}$$

   by finding an alternative form, F_2, which does not suffer from rounding error near $x = 0$.

   Hint: You may want to use MATLAB's `table` function. Type >> `help table/table` for instructions.

2. Repeat for yourself the quadratic problem from slide 2 of Lecture 2 using MATLAB or Python. Verify the answer given on the slides by using the built in `roots` command.

3. Die Bessel funksies, $J_n(x)$, kom dikwels in Toegepaste Wiskunde en Numeriese Analise voor. Hulle is oplossings vir die differensial vergelyking

The Bessel functions, $J_n(x)$, arise often in Applied Mathematics and Numerical Analysis. They are solutions of the differential equation

$$\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} + \Big(1 - \frac{n^2}{x^2}\Big)y = 0, \qquad \text{met/with} \qquad J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{n+2k}}{k!(n+k)!}.$$

(a) Vind 'n benadering tot $J_0(1)$ en $J_1(1)$ deur gebruik te maak van die eerste vier terme in die bostaande reeks (dit wil sê, vervang ∞ met 3). Gebruik die ingeboude funksie (`besselj` in MATLAB en `scipy.special.jn` in Python) om die akuraatheid van die berekeninge te bevestig. Skep 'n tabel en wys die benaderde waardes, die ingeboude waardes en die (absolute) foute tussen hulle.

(a) Find an approximation to $J_0(1)$ and $J_1(1)$ by using the first four terms in the series above (i.e., replace ∞ by 3). Use the built-in function (`besselj` in MATLAB and `scipy.special.jn` in Python) to check the accuracy of your computations. Make a table showing the approximated values, the built in values, and the (absolute) error between them.

(b) Dit is nie moeilik om te wys dat die Bessel funksie aan die rekursie formule voldoen nie

(b) It is not too difficult to show that the Bessel functions satisfy the recursion formula

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x), \quad n = 1, 2, \ldots.$$

Gebruik jou benaderings tot $J_0(1)$ en $J_1(1)$ van deel (a) en die bostaande rekursie om $J_2(1), J_3(1), \ldots, J_7(1)$ te bereken. Soos in deel (a), gebruik die ingeboude funksie om hierdie waardes te bereken en skep 'n tabel. Lewer kommentaar op die resultate. Is hierdie 'n goeie manier om $J_n(1)$ vir groot n te bereken? [Opsioneel: lei die rekursieformule af]

Use your approximations to $J_0(1)$ and $J_1(1)$ from part (a) and the recursion above to compute $J_2(1), J_3(1), \ldots, J_7(1)$. As in part (a) use the built in function to compute these values and make a similar table. Comment on your results. Is this a good way to evaluate $J_n(1)$ for large n? [Optional: Derive the recursion formula]

(c) 'n Ander manier om hierdie rekursie te gebruik is om dit van agter af te gebruik. Om dit te doen definieer $\tilde{J}_n(x) = \alpha J_n(x)$ met $\alpha = 1/J_7(x)$ sodat $\tilde{J}_7(1) = 1$ en let op dat $\tilde{J}_n(x)$ voldoen aan die herhaling

(c) Another way to use this recursion is to run it in reverse. To do so, define $\tilde{J}_n(x) = \alpha J_n(x)$ with $\alpha = 1/J_7(x)$ so that $\tilde{J}_7(1) = 1$ and notice that the $\tilde{J}_n(x)$ satisfy the recurrence

$$\tilde{J}_{n-1}(x) = \frac{2n}{x}\tilde{J}_n(x) - \tilde{J}_{n+1}(x), \quad n = 7, 6, \ldots, 1.$$

Soos ons in lesing 2 gedoen het, gebruik hierdie herhaling met die (ooglopend verkeerde) benadering $\tilde{J}_8(1) = 0$ om 'n benadering te maak vir $\tilde{J}_6(1), \tilde{J}_5(1), \ldots, \tilde{J}_0(1)$. Gebruik uiteindelik die identiteit

Like we did in Lecture 2, use this recurrence with the (patently wrong) approximation $\tilde{J}_8(1) = 0$ to approximate $\tilde{J}_6(1), \tilde{J}_5(1), \ldots, \tilde{J}_0(1)$. Finally, use the identity

$$\alpha = \tilde{J}_0(x) + 2\tilde{J}_2(x) + 2\tilde{J}_4(x) + 2\tilde{J}_6(x) + \ldots$$

om α te benader en skep 'n tabel vir $J_n(1)$ soos in deel (b). Lewer kommentaar op jou resultate. In besonders, hoe vergelyk hierdie table met die van (b)?

to approximate α and produce a table of $J_n(1)$ as in part (b). Comment on your results. In particular, how does this table compare with that from (b)?

(d) Watter algoritme tussen (b) en (c) is meer stabiel? Hoekom?

(d) Which algorithm, out of (b) and (c) is the more stable? Why?

Opsioneel Problem:

4. Herskryf die halveringsmetode in die lesings as ‘n fuksie-lêer wat `f`, `a`, en `b` as invoer aanvaar. Gebruik jou nuwe funkie-lêer om die wortel van $f(x) = e^x x^2 - 1/\pi^2$ vir $x \in [0, 1]$ te vind. Herhaal die bogenoemde vir die regula-falsi metode. Watter metode konvergeer vinniger vir hierdie probleem? Verduidelik waarom.

Optional problem:

Rewrite the bisection code from lectures as a function file that accepts `f`, `a`, and `b` as inputs. Use your new function file to find the root of $f(x) = e^x x^2 - 1/\pi^2$ for $x \in [0, 1]$. Repeat the above for the regula-falsi method. Which method converges more quickly for this problem? Explain why.

Opsioneel Problem:

5. Laat p en q reële getalle wees met $p^3 + q^2 > 0$, en beskou die kubiese vergelyking

Optional problem:

Let p and q be real numbers with $p^3 + q^2 > 0$ and consider the cubic equation

$$x^3 + 3px + 2q = 0.$$

Volgens Cardano se formule het hierdie vergelyking ’n wortel $x = x_1$, waar

According to Cardano’s formula this equation has a solution $x = x_1$, where

$$x_1 = u - v,$$

met

with

$$u^3 = \sqrt{p^3 + q^2} - q, \qquad v^3 = \sqrt{p^3 + q^2} + q$$

Gebruik Cardano se formule om die wortel x_1 van die volgende vergelyking te bereken

Use Cardano’s formula to compute the root x_1 of the following equation

$$x^3 + 30000x - 0.2 = 0$$

Vertoon x_1 tot alle syfers. As verwysing, bereken x_1 ook met die ingeboude funksie vir wortelbepaling in jou sagteware (bv `roots` in MATLAB). Hoeveel syfers is korrek in die waarde soos bereken met Cardano se formule? Waar het die kansellasie plaasgevind? Herskryf Cardano se formule om die kansellasie te vermy en implementeer jou nuwe formule om die verbetering te demonstreer. Wenk:

Display x_1 to full precision. As reference, also compute x_1 with the built-in rootfinding function in your software (e.g., `roots` in MATLAB). How many digits are correct in the value produced by Cardano’s formula? Where did the cancellation occur? Rewrite Cardano’s formula to avoid the cancellation and implement your new formula to demonstrate the improvement. Hint:

$$u^3 - v^3 = (u - v)(u^2 + uv + v^2)$$

Super Opsioneel Probleme:

6. Implementeer die Illinois en Anderson–Björk weergawes van die Vals-Posisie (Regula Falsi) metode om die vergelyking van Probleem 4 op te los.

Super Optional Problems:

Implement the Illinois and Anderson–Björk versions of the False-Position (Regula Falsi) method to solve the equation from Problem 4.

https://en.wikipedia.org/wiki/False_position_method#Improvements_in_regula_falsi