

Verslag project numerieke wiskunde

Academiejaar 2017-2018

Thomas Bamelis en Michiel Provoost

Inleiding

Dit verslag behandelt de vragen en antwoorden gesteld in het eind-project numerieke wiskunde 2017. De hoofdvraag in dit project is hoe wortels/nulpunten van een willekeurige veelterm kunnen gevonden worden met de methode van Newton-Raphson en de methode van Bairstow. Verder behandelt dit verslag de performantie en fouten op beide algoritmen en sluit af met deze met elkaar te vergelijken. Beide methoden werden geïmplementeerd in matlab en getest op 10 gegeven veeltermen, met complexe en reële nulpunten. Verder werden verschillende opgedragen ease-of-use features geïmplementeerd, zoals een figuur plotten indien enkel de coëfficiënten gegeven.

1 Newton-Raphson

Dit hoofdstuk bespreekt de implementatie van Newton-Raphson, waarin onder andere de methode van Horner, samen met een analyse van de performantie en fouten.

1.1 Evaluatie veelterm en zijn afgeleide m.b.v. het Hornerschema

Hier wordt opdracht 1 besproken.

Met het uitgebreide Hornerschema, besproken in [1], kan een veelterm geëvalueerd in een gegeven punt. Daarna kan met een deel van de oplossing verder gewerkt worden om 1-per-1 de afgeleiden van de veelterm te evalueren in hetzelfde punt.

Dit werd als volgt geïmplementeerd:

```
1 function y = my_polyval( p, x, m)
2 %my_polyval Geeft de functiewaarde in c van de eerste m afgeleiden van p
3 %terug, via het uitgebreid schema van Horner.
4 %
5 %   Signatuur: y = my_polyval( p, x, m)
6 %
7 %   @param p
8 %       De veelterm die geevalueerd zal worden, voorgesteld door zijn
9 %       coëfficiënten als vector, met de hoogste graad term als eerste
10 %       element enzovoort. Dit is een 1 x n vector.
11 %
12 %   @param x
13 %       Het punt waarin de functies zullen geevalueerd worden.
14 %
15 %   @param m
16 %       Het aantal keer dat p moet afgeleid worden.
17 %
18 %   @return y
19 %       De vector die de functiewaarden in x van de 0'de tot de m'de
20 %       afgeleide van p bevat.
21 %
22
23
24
25
```

```

26  graadStartPlusEen = size( p );
27  graadStartPlusEen = graadStartPlusEen(2);
28
29
30  y = zeros( m + 1, 1);
31
32  % p gebruiken, want minder geheugen gebruikt (call by value)
33
34
35  for afgeleide = 1:m+1
36
37      % 1ste niet kopiëren want die zit al in p.
38
39      for coefficient = 2:(graadStartPlusEen - afgeleide+1)
40
41          %De coëfficiënten aanpassen. Alle elementen van p voor
42          %p(coefficient) zijn al aangepast, dus de 2de term
43          %in onderstaande formule ook.
44
45          p(coefficient) = p(coefficient) + p(coefficient - 1)*x;
46
47      end
48
49      %De functiewaarde in x van deze afgeleide berekenen en opslaan.
50      %y begint bij 1 en eindigt bij m + 1, dus 1 verder zetten.
51
52      y(afgeleide) = factorial(afgeleide-1) * p(graadStartPlusEen - afgeleide
53          +1);
54
55
56  end
57  end

```

Zoals in [1] wordt aangetoond, worden de m eerste afgeleid (inclusief $f^{(0)}$) gevonden, door m keer het schema van Horner toe te passen op p waarna de functiewaarde van i 'de afgeleide gegeven wordt door $k!b_{n-k}^{k+1}$. Merk op dat in de 'afgeleide' iteratie, de 'afgeleide' - 1 'ste afgeleide behandeld wordt om beter matlabs vector nummering in te kunnen spelen.

Het 1'ste element van p , dus de coëfficiënt van de hoogste graad term, wordt nooit aangepast omdat volgens Horner $a_0 = b_0$. Daarna worden de rest van de coëfficiënten aangepast volgens $b_i = a_i + b_{i-1} * x$. Er wordt rekening gehouden met het feit dat de graad van de veelterm bij iedere iteratie van de afgeleide met 1 verlaagd wordt.

Een bijkomende opdracht was om zo weinig mogelijk geheugen te gebruiken. Om hier mee om te gaan werd in de vector p zelf gewerkt. Dit is geen enkel probleem omdat je nooit de oude vorige elementen nodig hebt en omdat matlab call-by-value is. Dit wil zeggen dat matlab p kopieert en de kopie meegeeft aan de functie waar mee gewerkt mag worden zonder dat de originele p wordt aangepast. Het enige geheugen dat extra wordt aangemaakt, is het geheugen gebruikt om de functie waarden terug te geven een de oproeper van de functie. Dit wil zeggen dat het geheugengebruik van de functie van orde $\Theta(n+m)$ is, met n de graad + 1 van p en m het aantal keer dat afgeleid moet worden.

Besluit

Afsluitende tekst

Referenties

- [1] Adhemar Bultheel. *Inleiding tot de numerieke wiskunde*. Acco, 2006.