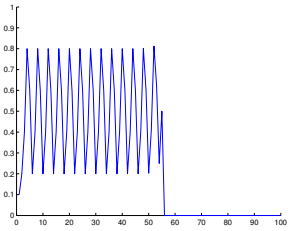
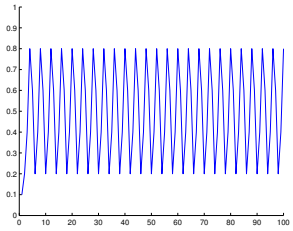
Wetenschappelijk Rekenen

Examen - Bacheloropleiding informatica

Oefeningen – 30 mei 2013

1. Implementeer de functie $x_{n+1} = \text{mod}(2x_n, 1)$ waarbij je gebruik maakt van een voorstelling met reële getallen. Zorg er in je implementatie voor dat je de beginwaarde x_0 en het aantal iteraties als parameter kan meegeven. Pas vervolgens deze functie aan zodat de functie gebruik maakt van een voorstelling met rationale getallen. Plot in beide gevallen de resultaten voor $x_0 = 0.1$ met 100 iteraties. Hieronder kan je de plots zien die wij bekwamen. Los nu volgende problemen op:

   - verklaar welke plot bij welke implementatie hoort;
   - verklaar het gedrag bij beide implementaties in het interval $[50, 60]$: verklaar in detail waarom we rond iteratie 50 fouten beginnen te zien bij één van de implementaties;
   - verklaar waarom er zich geen fouten voordoen bij $x_0 = 0.125$ met 100 iteraties.

Oplossing: De functie $x_{n+1} = \text{mod}(2x_n, 1)$ is gekend als de Bernoulli shift en modelleert op een deterministische manier een chaotisch systeem. Wanneer we deze functie implementeren met behulp van reële getallen stellen we vast dat we na 53 iteraties fouten beginnen te zien, terwijl na 56 iteraties de foutief berekende waarde 0 wordt. De eerste plot komt dan ook overeen met de implementatie waarbij we reële getallen gebruiken. De tweede plot wordt bekomen bij de implementatie met rationale getallen.

Het probleem zit in de afrondingsfout die aan het begin van het algoritme gemaakt wordt. Een getal zoals 0.1 kan niet exact voorgesteld worden in floating point notatie. Deze initiële fout wordt bij elke iteratie uitvergroot (bij elke iteratie wordt de fout gemiddeld verdubbeld). Om dit te verklaren moeten we kijken naar de binaire voorstelling van x_0:

$$x_0 = (1001100110011001100110011001100110011001100110011001100110011010)_2 \cdot 2^{-4}$$

De vermenigvuldigingen van 0.1 naar 0.2 naar 0.4 naar 0.8 veranderen enkel de exponent van respectievelijk -4 naar -3 voor $x_1 = \text{fl}(0.2)$, naar -2 voor $x_2 = \text{fl}(0.4)$ en naar -1 voor $x_3 = \text{fl}(0.8)$, waarbij $\text{fl}(x)$ de floating point voorstelling van x is. Vermenigvuldigen we x_3 opnieuw met 2, dan moeten we ditmaal ook 1 ervan af trekken. Dit vereist dat het getal opnieuw genormaliseerd wordt; m.a.w. we moeten een 1 voorop brengen in de

binaire voorstelling (aangezien we niet het getal 0 willen voorstellen) wat er concreet op neer komt dat we de bits in de fractie 1 plaats naar links verschuiven (en we dus op het einde een nul introduceren). De binaire voorstelling voor $x_4 \approx 0.6$ wordt dan:

$$x_4 = (00110011001100110011001100110011001100110011001100110100)_2 \cdot 2^{-1}$$

Ook in de volgende stap moeten we 1 aftrekken van het bekomen getal. Bij normalisatie betekent dit ditmaal dat we de fractie met 3 plaatsen naar links opschuiven (en we dus 3 nullen op het einde introduceren). Ditmaal wordt ook de exponent gewijzigd. We krijgen voor $x_5 \approx 0.2$ de voorstelling

$$x_5 = (10011001100110011001100110011001100110011001100110100000)_2 \cdot 2^{-3}$$

We hebben, bij de overgang van x_1 naar x_5, in totaal 4 nullen acheraan in de fractie geïntroduceerd. Deze cyclus herhaalt zich, want $x_5 = x_9 = x_{13} = \ldots = x_{49} = x_{53}$ bij exacte arithmetiek. Als we kijken naar de voorstelling van x_{49}, dan zien we dat we hier al 12 keer 4 nullen geïntroduceerd hebben:

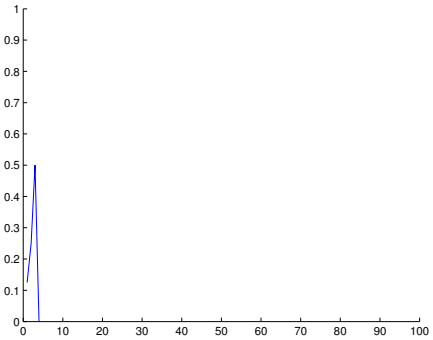
$$x_{49} = (10100000000000000000000000000000000000000000000000000000)_2 \cdot 2^{-3}$$

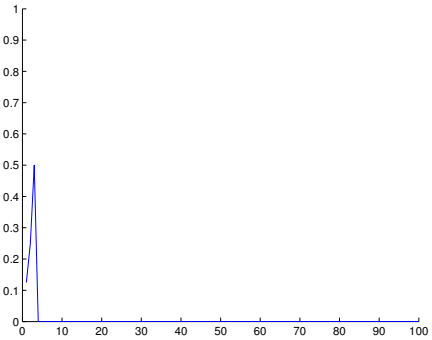
Voor x_{53} krijgen we de voorstelling:

$$x_{53} = (00000000000000000000000000000000000000000000000000000000)_2 \cdot 2^{-3}$$

We hebben dus dat $x_{53} = 0.125$ in plaats van 0.2, waarbij we in de daaropvolgende iteraties bekomen dat $x_{54} = 0.25$, $x_{55} = 0.5$ en $x_{56} = 0$.

Wanneer we de getallen voorstellen als rationale getallen doet dit probleem zich niet voor en krijgen we ook geen uitvergroting van de afrondingsfout doorheen de iteraties. De fout doet zich niet voor wanneer we beginnen bij $x_0 = 0.125$ omdat we dit getal wel correct kunnen voorstellen in floating point notatie, zoals we in de volgende plots kunnen zien.





Een implementatie in MATLAB m.b.h. reële getallen is:

```
function shift_reeel(n, iteraties)
  % maak de matrix klaar om de resultaten in te bewaren
  resultaten = zeros(1, iteraties);
  % het eerste resultaat is het beginpunt
```

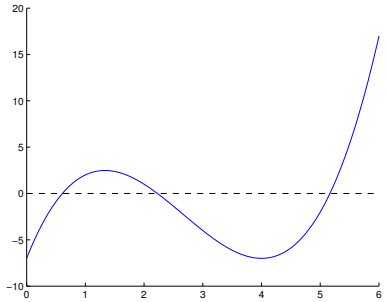
```
  resultaten(1) = n;

  % we berekenen elke volgende stap
  for i = 2:iteraties
      resultaten(i) = mod(2*resultaten(i-1),1);
  end

  % en we plotten de resultaten
  figure;
  plot(resultaten,'-b');
end
```

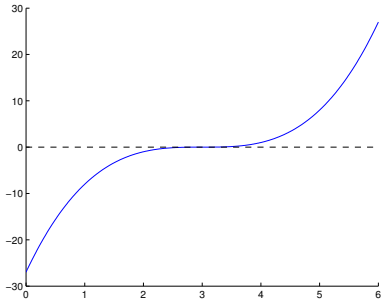
Een implementatie in MATLAB m.b.h. rationale getallen is:

```
function shift_rationaal(teller, noemer, iteraties)
  % maak de matrix klaar om de resultaten in te bewaren
  % merk op dat de noemer ongewijzigd blijft, en we deze
  % dus niet moeten bijhouden voor elk afzonderlijk resultaat
  resultaten = zeros(1, iteraties);
  % het eerste resultaat is het beginpunt
  resultaten(1) = teller;

  % we berekenen elke volgende stap
  for i = 2:iteraties
      resultaten(i) = mod(2*resultaten(i-1),noemer);
  end

  % en we plotten de resultaten, gedeeld door de noemer
  figure;
  plot(resultaten./noemer,'-b');
end
```

2. Beschrijf hoe je in MATLAB een programma zou implementeren dat de Newton-Raphson methode gebruikt om een minimum van een derdegraads functie in een opgegeven interval te bepalen. Denk hierbij goed na over de eigenschappen van een derdegraadsfunctie die belangrijk kunnen zijn voor dit gegeven probleem.

   Implementeer dit vervolgens in MATLAB. Om je wat werk te besparen hebben wij de Newton-Raphson methode reeds geïmplementeerd (deze kan je op de Indianio website vinden onder de naam `vind_min.m`). Probeer je implementatie uit voor de functies $x^3 - 8x^2 + 16x - 7$ en $(x-3)^3$, die je hieronder respectievelijk links en rechts geplot kan zien. In beide gevallen zoek je een minimum in het interval $[0, 6]$. Bij de eerste functie wil je als antwoord krijgen dat er een lokaal minimum is in het punt $(4, -7)$, terwijl je bij de tweede functie als antwoord wil krijgen dat er een minimum is op de grens van het interval. Besteed in je antwoord voldoende aandacht aan de (lokale) minima en maxima, alsook aan de buigpunten.





Oplossing: Een lokaal minimum betekent dat de afgeleide van de functie 0 wordt terwijl de tweede afgeleide van de functie strikt positief is. Gegeven een bepaald interval kunnen we dus, door het zoeken van het nulpunt van de tweede afgeleide, bepalen over welk stuk van het interval de tweede afgeleide positief is. Er zijn natuurlijk speciale gevallen mogelijk. Zo weten we dat er geen minimum is in het opgegeven interval wanner de tweede afgeleide niet van teken verandert en wanneer dit teken negatief is.

Eenmaal we het nauwkeuriger interval bepaald hebben, kunnen we op zoek gaan naar het nulpunt van de eerste afgeleide, wat overeen komt met het minimum van de oorspronkelijke functie. Omdat we weten dat de eerste afgeleide stijgt over dit interval, kunnen we de Newton-Raphson methode gebruiken om betrouwbaar een nulpunt te vinden. Eenmaal we het nulpunt gevonden hebben, volstaat nog een controle om na te gaan of dit nulpunt in het opgegeven interval zit. Tenslotte vergelijken we het resultaat met de functiewaarden in de randpunten van het interval om zo de minimale waarde van de functie te bepalen in het opgegeven interval.

Een implementatie in MATLAB van dit probleem wordt hieronder gegeven:

```
function vind_min(begin_interval, eind_interval)
    %f = @(x) x^3-8*x^2+16*x-7;
    %g = @(x) 3*x^2-16*x+16; % afgeleide van f
    %h = @(x) 6*x-16; % afgeleide van g
    %i = @(x) 6; %afgeleide van h

    f = @(x) (x-3)^3;
    g = @(x) 3 * (x-3)^2; % afgeleide van f
    h = @(x) 6*(x - 3); % afgeleide van g
    i = @(x) 6; %afgeleide van h


    %bepaal welk grenspunt de kleinste functiewaarde levert.
    if (f(begin_interval) < f(eind_interval))
        mingrenspunt = begin_interval;
    else
        mingrenspunt = eind_interval;
    end



    % we gaan er van uit dat er een lokaal minimum is
    minimum_in_interval = true;
    % controleer of het teken van de 2de afgeleide verschilt in het interval
    if (h(begin_interval) * h(eind_interval)) < 0
```

```
        % we hebben enkel in het interesse in het stuk met positief teken
        nulpunt = newton(h,i,begin_interval);
        % controleer of dit nulpunt wel in het interval zit
        if eind_interval > nulpunt && nulpunt > begin_interval
            % pas het interval aan, zodat we enkel zoeken in het
            % interval waarbij de tweede afgeleide positief is
            if sign(h(begin_interval)) == -1
                extra = (eind_interval - nulpunt) / 100;
                begin_interval = nulpunt + extra;
            else
                extra = (nulpunt - begin_interval) / 100;
                eind_interval = nulpunt - extra;
            end
        else
            minimum_in_interval = false;
        end
    elseif sign(h(begin_interval)) == -1
        % er is geen minimum als het teken enkel negatief is in het interval
        minimum_in_interval = false;
    end



    % vervolgens passen we de Newton-Raphson methode toe
    % om het nulpunt van de functie te vinden
    if minimum_in_interval == true
        nulpunt = newton(g,h,(begin_interval + eind_interval)/2);
        if nulpunt > (begin_interval) && nulpunt < (eind_interval)
            % lokaal minimum gevonden in nulpunt.

            if (f(mingrenspunt) < f(nulpunt))
                fprintf('minimum gevonden in grenspunt (%d,%d)\n', mingrenspunt, f(mingrenspunt));
            else
                fprintf('lokaal minimum gevonden in (%d,%d)\n', nulpunt, f(nulpunt));
            end
        else
            fprintf('minimum gevonden in grenspunt (%d,%d)\n', mingrenspunt, f(mingrenspunt));
        end
    else
        fprintf('minimum gevonden in grenspunt (%d,%d)\n', mingrenspunt, f(mingrenspunt));
    end

end



function [nulpunt] = newton(functie, afgeleide, zoek_rond)
    nulpunt = zoek_rond;
    while (abs(functie(nulpunt)) > eps)
        nulpunt = nulpunt - (functie(nulpunt) / afgeleide(nulpunt));
    end
end
```

Deze implementatie geeft voorkeur aan een lokaal minimum boven een minimum in de grenspunten in geval van een gelijkheid. In het geval van de functie $f(x) = x^3 - 8x^2 + 16x - 7$ zijn er twee x-waarden in het interval $[0, 6]$ waarvoor $f(x) = -7$, namelijk $x = 0$ en $x = 4$. Het is voldoende om *een* minimum te geven als resultaat.

3. (a) Bepaal de coëfficiënten van de kwadratuurformule

$$\int_0^1 f(x)\mathrm{d}x \approx Q_2(f) = w_1 f(x_1) + w_2 f(x_2)$$

zodat de graad wordt gemaximaliseerd.

(b) Bepaal de unieke 5-punts kwadratuurformule

$$\int_0^1 f(x)\mathrm{d}x \approx Q_5(f)$$

waarbij men de functieëvaluaties die optreden in de kwadratuurformule Q_2 kan hergebruiken in de evaluatie van Q_5 en waarbij de graad maximaal is.

(c) Bepaal de constanten p en α in de uitdrukking

$$\int_0^1 f(x)\mathrm{d}x - Q_5(f) = \alpha f^{(p)}(\xi)$$

met $\xi \in [0, 1]$. Wat is de graad van Q_5 ?

Oplossing:

(a) De volgende Maple-instructies leveren de oplossing voor de knopen en gewichten:

```
> Q2 := f -> w[1]*f(x[1])+w[2]*f(x[2]):
> solve({seq(w[1]*x[1]^k+w[2]*x[2]^k = int(x^k, x = 0 .. 1)
                                                  ,k = 0 .. 3)}):
> allvalues(%):
> assign(%[1]):
> Q2(f);
```

$Q_2(f) = \frac{1}{2} f\left(\frac{1}{2} - \frac{1}{6}\sqrt{3}\right) + \frac{1}{2} f\left(\frac{1}{2} + \frac{1}{6}\sqrt{3}\right)$

(b) De kwadratuurformule heeft de vorm:

$$Q_5(f) = a_1 f(x_1) + a_2 f(x_2) + a_3 f(z_1) + a_4 f(z_2) + a_5 f(z_3)$$

waarbij x_1 en x_2 de knooppunten zijn uit Q_2 en dus gekende waarden zijn. De onbekenden zijn a_1, a_2, a_3, a_4, a_5, z_1, z_2 en z_3. We hebben 8 vrijheidsgraden en kunnen dus 8 voorwaarden opleggen.

```
> [x[1], x[2]];
```

$\left[\frac{1}{2} - \frac{1}{6}\sqrt{3},\ \frac{1}{2} + \frac{1}{6}\sqrt{3}\right]$

```
> Q5 := f -> a[1]*f(x[1])+a[2]*f(x[2])+a[3]*f(z[1])+
                              a[4]*f(z[2])+a[5]*f(z[3]):

> solutions := solve({seq(a[1]*x[1]^k+a[2]*x[2]^k
       +a[3]*z[1]^k+a[4]*z[2]^k+a[5]*z[3]^k = int(x^k, x = 0 .. 1)
       ,k = 0 .. 7)}):
```

Maple levert verschillende oplossingen, kies daar één van. De verschillende oplossingen leiden trouwens toch naar dezelfde formule.

```
> allvalues(solutions[1]):
> assign(%[1]):
> Q5(f);
```

$$Q_5(f) = \frac{27}{110} f\left(\frac{1}{2} - \frac{1}{6}\sqrt{3}\right) + \frac{27}{110} f\left(\frac{1}{2} + \frac{1}{6}\sqrt{3}\right) + \frac{14}{45} f\left(\frac{1}{2}\right) + \frac{49}{495} f\left(\frac{1}{2} - \frac{1}{14}\sqrt{42}\right) + \frac{49}{495} f\left(\frac{1}{2} + \frac{1}{14}\sqrt{42}\right)$$

(c) De graad van Q_5 moet minstens 7 zijn, we verifiëren dit en testen of de graad al dan niet hoger is.

```
> testGraad := k -> a[1]*x[1]^k+a[2]*x[2]^k+a[3]*z[1]^k
    +a[4]*z[2]^k+a[5]*z[3]^k = int(x^k, x = 0 .. 1):
> [simplify(testGraad(7)), simplify(testGraad(8))]
```

$[\frac{1}{8} = \frac{1}{8},\ \frac{5227}{47040} = \frac{1}{9}]$
De graad is dus niet hoger dan 7. Hieruit volgt dat $p = 8$, we bepalen α:

```
> p := 8:
> 1/9-5227/47040 = alpha*factorial(p): #geval f := x -> x^8
> solve({%}, alpha);
```

$\left\{\alpha = -\frac{1}{5689958400}\right\}$
Dus

$$\int_0^1 f(x)\mathrm{d}x - Q_5(f) = -\frac{1}{5689958400} f^{(8)}(\xi)$$

met $\xi \in [0, 1]$.

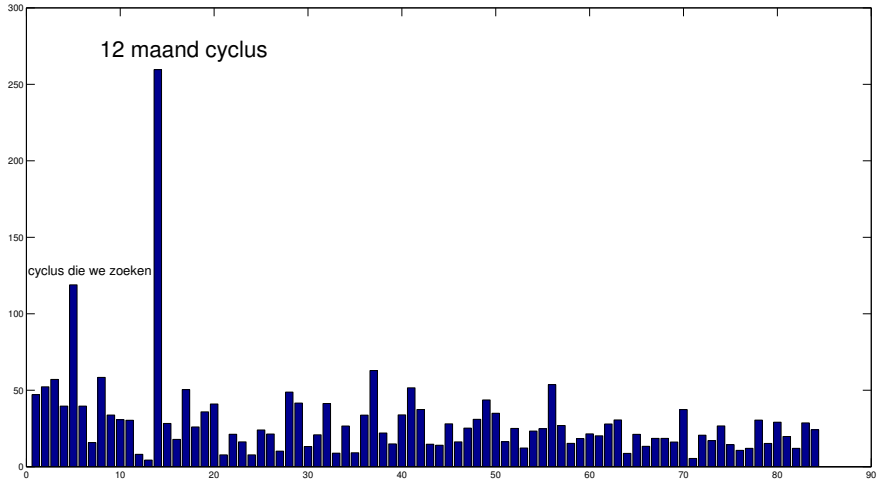
4. Een El Niño is een verschijnsel waarin warm water zich ter hoogte van de evenaar langs de kust en over een groot deel van de Stille Oceaan uitstrekt. De gevolgen van een El Niño voor het weer zijn tot in de wijde omtrek groot. Het bestand `elnino.mat` bevat meetwaarden van luchtdruk die een beeld geven van dit klimatologisch fenomeen. Deze metingen zijn maandelijks genomen over een periode van 14 jaar. Er zijn dus 12×14 meetwaarden. Er is een prominente cyclus aanwezig van 12 maanden en een minder prominente cyclus met een langere periode. Bepaal deze langere periode. De volgende MATLAB instructie laadt de data:

```
>> load 'elnino.mat'
```

Hiervoor moet `elnino.mat` zich in de *current folder* van MATLAB bevinden. Zie instructies: `pwd` en `ls`.

Oplossing:

```
>> load 'elnino.mat'
>> Y = fft(elnino);
>> %Verwijder 'DC' uit de vector:
>> Y(1) = []; %verwijderen van element 1 uit vector Y.
>> %bekijk enkel eerste helft, tweede helft is het complex toegevoegde.
>> midden = ceil(length(Y)/2);
>> bar(abs(Y(1:midden)));
```



Na het verwijderen van het 'DC' component (y_0 in de cursus) correspondeert nu `Y(1)` met een periode van 14 jaar ($\frac{14}{1}$), `Y(2)` met een periode van 7 jaar ($\frac{14}{2}$), `Y(3)` met een periode van 4.666 jaar ($\frac{14}{3}$), `Y(4)` met een periode van 3.5 jaar ($\frac{14}{4}$) , `Y(5)` met een periode van 2.8 jaar ($\frac{14}{5}$), ..., `Y(14)` met een periode van 1 jaar ($\frac{14}{14}$) , ...

```
>> [maximum , index] = max(abs(Y(1:midden)));
>> periode = 14/index %index is 14
periode =

     1
```

De periode is 1 jaar, dit is die prominente cyclus van 12 maanden. Door deze component op nul te zetten en opnieuw de `max` functie toe te passen vinden we de component met de tweede grootste waarde.

```
>> Y(index) = 0;
>> [maximum , index] = max(abs(Y(1:midden)));
>> periode = 14/index %index is 5
periode =

    2.8000
```

De cyclus die we zoeken heeft een periode van 2.8 jaar (33.6 maanden).