Geheugensteuntje

foutenanalyse.

Absolute fout:
$\Delta x = \mathbf{x} - x$
Relatieve fout:
$\delta x = \Delta x / x = (\mathbf{x} - x)/x \approx \Delta x / \mathbf{x}$
$\mathbf{x} = x (1 + \delta x)$

Genormaliseerde getalvoorstelling:
$x = .c_k c_{k-1} ... E e$
Juist cijfer:
$|\mathbf{x} - x| \leq ½ b^i$ => cijfer op pos. i is juist.
$|\mathbf{x} - x| > ½ b^j$ => cijfer op pos. j is fout.
Laatste j.c. na de komma $p = -j-1$
$p \approx \log_b 1/(2|\Delta x|)$
Aantal j.b.c. $q = k - j$ met k de positie van het 1e b.c.
$q \approx \log_b 1/(2|\delta x|)$ als $\log_b 1/(2|\delta x|) \geq 0$

Machinenauwkeurigheid:
$\varepsilon_{mach} = ½ b^{-p+1}$

Konditie & stabiliteit.

Konditie:

- Een probleem is slecht gekonditioneerd als voor gelijk welke berekeningsmethode de afwijking op het berekende resultaat groot zal zijn.
- Konditie t.o.v. de absolute fout:
  $\Delta_k f = \Sigma_{i=1}^{n} (\delta f/\delta x_i)*\Delta x_i$
  (= |f '(x)|)
  met n het aantal
  veranderlijken van f.
- Konditie t.o.v. de relatieve fout:
  $\delta_k f = |x*(\Delta_k f/f)|$
  (= |x*(f '(x)/f)|)

Stabiliteit

- Een algoritme is stabiel als er in het
  begin van het algoritme mogelijk
  wel kleine afrondingsfouten gemaakt worden, maar voor de rest
  quasi exact gerekend wordt. Het
  resultaat van de berekeningen kan
  ook bij een stabiel algoritme sterk afwijken van de exacte oplossing, nl.
  als de konditie van het probleem slecht is.
- Stabiliteit t.o.v. de absolute fout:
  $\Delta_s f = |\mathbf{f}(x) - f(x)|$

Stabiliteit t.o.v. de relatieve fout:
$\delta_s f = |(\mathbf{f}(x) - f(x))/f(x)|$

Veelterminterpolatie.

Algemeen:

$f(x) = y_n(x) + E_n(x)$

met $y_n(x)$ de interpolerende veelterm van graad n en $E_n(x)$ de gemaakte interpolatiefout.

$E_n(x) = [f^{(n+1)}(\xi)/(n+1)!](x-x_0)(x-x_1)...(x-x_n)$

$|E_n(x)| \leq [|(x-x_0)(x-x_1)...(x-x_n)|/(n+1)!] * \max_{x\in[a,b]} |f^{(n+1)}(x)|$

waarbij [a,b] het interval is waarvoor $y_n(x)$ interpoleert.

Lagrange-interpolatie:

$y_n(x) = l_0(x)*f_0 + l_1(x)*f_1 + ... + l_n(x)*f_n$

$y_n(x_i) = f_i$

$l_i(x) = \Pi(x)/[\Pi'(x_i)(x- x_i)]$

$$= \frac{[(x - x_0)(x - x_1) ... (x - x_n)]}{[(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})... (x_i - x_n)(x- x_i)]}.$$

Iteratieve methoden voor het oplossen van transcendente vergelijkingen.

- Vast punt:
  - x* is een vast punt van een functie f <=> f(x*) = x*
- Consistentie:
  - Een methode heet consistent als alle nulpunten van f ook vaste punten zijn van F (met F de iteratieformule).
  - Een methode heet reciprook consistent als alle vaste punten van F ook nulpunten zijn van f.
  - Een methode heet volledig consistent als ze zowel consistent als reciprook consistent is.
  - Een methode heet zwak consistent als x* een nulpunt is van f en ook een vast punt van F (f(x*) = 0 => F(x*) = x*).

- Convergentie:
  - Convergentiestelling: als x* een vast punt is van F, als F afleidbaar is in de omgeving van x*, en als F zwak consistent is met f, dan zal de rij gedefinieerd door $x^{(k)} = F(x^{(k-1)})$ convergeren naar x* als x* een nulpunt is van f en $x^{(0)}$ voldoende dicht bij x*.
  - Voorwaarden voor convergentie:
    - $|F'(x)| \leq 1$ (als F afleidbaar is).
    - $x^{(0)}$ voldoende dicht bij x*.
    - Minstens zwakke consistentie.
    - Soms treedt er convergentie op ondanks dat (sommige) voorwaarden niet voldaan zijn.
- Convergentie-orde:
  - $p = \sup\{n \in |R: \lim_{k->\infty} \varepsilon^{(k+1)}/[\varepsilon^{(k)}]^n = 0\}$
    $= \inf\{n \in |R: \lim_{k->\infty} \varepsilon^{(k+1)}/[\varepsilon^{(k)}]^n \neq 0\}$

- Voor substitutiemethoden is p steeds een geheel getal.
- Praktisch: bereken F (= $F^{(0)}$), F' (= $F^{(1)}$), F'' = $F^{(2)}$, $F^{(3)}$, ... totdat $F^{(i)}(x^*) \neq 0$.
  i is nu de convergentie-orde.

- Newton-Raphson:

1) f(x) benaderen door een rechte (de eerstegraads Hermite-interpolerende veelterm = de raaklijn in $(x^{(i-1)}, f(x^{(i-1)}))$:
$y_{(i-1)(i-1)}(x) = f(x^{(i-1)}) + f'(x^{(i-1)})(x - x^{(i-1)})$

2) $x^{(i)}$ is het nulpunt v/d rechte voor $x^{(i-1)}$:
$x^{(i)} = x^{(i-1)} - f(x^{(i-1)})/f'(x^{(i-1)})$

3) Stappen 1) en 2) herhalen totdat de gewenste nauwkeurigheid bereikt is (of totdat het maximaal aantal iteratiestappen bereikt is).

4) Pseudo-algoritme:

```
function [res, msg] = NewtonRaphson
                (x0, f, f ', epsilon, Kmax)
for i=1:Kmax,
    f0 = f(x0);
    f '0 = f '(x0);
    res = x0 - f0/f '0;
    if (abs(res - x0) < epsilon),
            msg = 0; % convergentie
            return;
    else
            x0 = res;
    end; % if
end; % for
msg = -1; % geen convergentie voor Kmax
return;
```

5) Consistentie:

$F(x) = x - f(x)/f'(x)$
$f(x)$ heeft m-voudige wortel x^*, dus:
$f(x) = (x-x^*)^m g(x)$ met $g(x^*) \neq 0$

$$f'(x) = m(x-x^*)^{m-1}g(x) + (x-x^*)^m g'(x)$$
$$= (x-x^*)^{m-1}[m\,g(x) + (x-x^*)g'(x)]$$
$$F(x) = x - \frac{(x - x^*)^m g(x)}{m(x-x^*)^{m-1}g(x) + (x-x^*)^m g'(x)}$$
$$= x - \frac{(x - x^*)^m g(x)}{m(x-x^*)^{m-1}g(x)+(x-x^*)^m g'(x)}$$
$$= x - \frac{(x - x^*)g(x)}{m\,g(x)+(x-x^*)g'(x)}$$
$$F(x^*) = x^* - \frac{(x^* -x^*)g(x^*)}{m\,g(x^*)+(x^*-x^*)g'(x^*)}$$
$$F(x^*) = x^* - 0/[m\,g(x^*)]$$

met $g(x^*) \neq 0$ en $m > 0$

$$= x^*$$

Consistent.

Convergentie-orde:

$F'(x) = 1 - f'(x)/f'(x) + f(x)f''(x)/[f'(x)]^2$
$= 1 - 1 + f(x)f''(x)/[f'(x)]^2$
$= f(x)f''(x)/[f'(x)]^2$

$f''(x) = m(m-1)(x-x^*)^{m-2}g(x) + 2m(x-x^*)^{m-1}g'(x)$
$+ (x - x^*)^m g''(x)$
$= (x-x^*)^{m-2}[m(m-1)g(x) + 2m(x-x^*)g'(x)$
$+ (x - x^*)^2 g''(x)]$

$f(x)\, f''(x) = (x-x^*)^m g(x)(x-x^*)^{m-2}$
$[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x- x^*)^2 g''(x)]$
$= (x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]$

$$F'(x) = \frac{(x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{((x-x^*)^{m-1}[m\, g(x) + (x-x^*)g'(x)])^2}$$

$$= \frac{(x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{(x-x^*)^{2m-2}[m\, g(x) + (x-x^*)g'(x)]^2}$$

$$= \frac{g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{m^2(g(x))^2 + (x-x^*)^2(g'(x))^2 + 2m(x-x^*)g(x)g'(x)}$$

$$F'(x^*) = \frac{g(x^*)[m(m-1)g(x^*) + 2m(x^*-x^*)g'(x^*) + (x^*-x^*)^2 g''(x^*)]}{m^2(g(x^*))^2 + (x^*-x^*)^2(g'(x^*))^2 + 2m(x^*-x^*)g(x^*)g'(x^*)}$$

$$= \frac{g(x^*)[m(m-1)g(x^*)]}{m^2(g(x^*))^2}$$

$$= \frac{(g(x^*))^2[m(m-1)]}{m^2(g(x^*))^2}$$

$= (m - 1)/m$
$= 1 - 1/m$

Omdat $m \geq 1$, $m \in |N$ (want x^* is een wortel) geldt:
$m = 1$: $F'(x^*) = 0$ en orde ≥ 2
$m > 1$: $0 < F'(x^*) < 1$ en orde $= 1$.

Iteratieve methoden voor het oplossen van stelsels lineaire en niet-lineaire vergelijkingen.

Niet-lineaire vergelijkingen:

- Newton-Raphson:
  - De Jacobiaan van f:

    $J = [\partial f_i(x^{(0)})/\partial x_j]_{i,\, j=1}^{n}$
    $F(x) = x - [J(x)]^{-1}f(x)$

  - Praktisch: i.p.v. $[J(x)]^{-1}$ te berekenen:

    $J(x^{(0)})h = -f(x^{(0)})$ oplossen.
    $x^{(1)} = x^{(0)} + h$

- Stoppen als J singulier is (ev. herh. met andere startwaarde).
- Convergentie:
  - in het algemeen kwadratisch; lineair als J singulier is.
  - Indien de startwaarde voldoende dicht bij de gezochte waarde x* ligt, zal er convergentie zijn.

○ Voorbeeld: n = 2.

Stelsel:

$u(x, y) = 0$

$v(x, y) = 0$

Formules:

$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$

$y^{(k+1)} = y^{(k)} + \Delta y^{(k)}$

met $J = \begin{bmatrix} u'_x(x^{(k)}, y^{(k)}) & u'_y(x^{(k)}, y^{(k)}) \\ v'_x(x^{(k)}, y^{(k)}) & v'_y(x^{(k)}, y^{(k)}) \end{bmatrix}$

en $J * \begin{bmatrix} \Delta x^{(k)} \\ \Delta y^{(k)} \end{bmatrix} = - \begin{bmatrix} u(x^{(k)}, y^{(k)}) \\ v(x^{(k)}, y^{(k)}) \end{bmatrix}$

waarbij $u'_x = \partial u/\partial x$, $u'_y = \partial u/\partial y$,

$v'_x = \partial v/\partial x$ en $v'_y = \partial v/\partial y$

$$\Delta x^{(k)} = - \frac{uv'_y - vu'_y}{u'_x v'_y - v'_x u'_y}$$

$$\Delta y^{(k)} = - \frac{vu'_x - uv'_x}{u'_x v'_y - v'_x u'_y}$$

● Vereenvoudigde Newton-Raphson:

In de i^{de} vgl.:

x_i = veranderlijk

x_j ($j \neq i$) = vast; stel $x_j = x_j^{(0)}$

(Totale stapmethode;
enkelvoudige stapmethode:
$j < i$: $x_j = x_j^{(1)}$; $j > i$ $x_j = x_j^{(0)}$)

$f_i(x_1^{(0)}, ..., x_{i-1}^{(0)}, x_i, x_{i+1}^{(0)}, ..., x_n^{(0)}) = 0$ is een scalaire vgl. in 1 variabele.
(Enkelvoudige stapmethode: $j < i$: $x_j^{(1)}$ i.p.v. $x_j^{(0)}$)

$$x_i^{(1)} = x_i^{(0)} - \frac{f_i(x_1^{(0)}, ..., x_n^{(0)})}{\partial f_i/\partial x_i\ (x_1^{(0)}, ..., x_n^{(0)})}$$

(Enkelvoudige stapmethode: $j < i$: $x_j^{(1)}$ i.p.v. $x_j^{(0)}$)
Herhalen voor elke i; dan $x_i^{(1)}$ bewaren op de plaats van $x_i^{(0)}$ en het geheel herhalen (totale stapmethode).
(Enkelvoudige stapmethode: de nieuwe waarden worden rechtstreeks op de plaats van de vorige geschreven).

- Voordelen t.o.v. Newton-Raphson:

  ■ Minder rekenwerk (voor elke cyclus slechts n partieel afgeleiden i.p.v. n^2).

- Nadelen t.o.v. Newton-Raphson:
  - Tragere convergentie.
  - Ook met een goed gekozen startwaarde is convergentie niet gegarandeerd; J wordt immers vervangen door zijn diagonaal.
  - Of er convergentie optreedt, hangt mede af van de volgorde van de vergelijkingen.

○ Voorbeeld: n = 2:

Stelsel:

$u(x, y) = 0$

$v(x, y) = 0$

Totale stap:

$$x^{(k+1)} = x^{(k)} - \frac{u(x^{(k)}, y^{(k)})}{u'_x(x^{(k)}, y^{(k)})}$$

$$y^{(k+1)} = y^{(k)} - \frac{v(x^{(k)}, y^{(k)})}{v'_y(x^{(k)}, y^{(k)})}$$

Enkelvoudige stap:

$$x^{(k+1)} = x^{(k)} - \frac{u(x^{(k)}, y^{(k)})}{u'_x(x^{(k)}, y^{(k)})}$$

$$y^{(k+1)} = y^{(k)} - \frac{v(x^{(k+1)}, y^{(k)})}{v'_y(x^{(k+1)}, y^{(k)})}$$

Elementaire substitutiemethoden:

○ Stelsel van de vorm:

$$x_1 = F_1(x_1, ..., x_n)$$
$$...$$
$$x_i = F_i(x_1, ..., x_n)$$
$$...$$
$$x_n = F_n(x_1, ..., x_n)$$

○ Totale stap:

$$x_1^{(k+1)} = F_1(x_1^{(k)}, ..., x_n^{(k)})$$
$$...$$
$$x_i^{(k+1)} = F_i(x_1^{(k)}, ..., x_n^{(k)})$$
$$...$$
$$x_n^{(k+1)} = F_n(x_1^{(k)}, ..., x_n^{(k)})$$

○ Enkelvoudige stap:

$$x_1^{(k+1)} = F_1(x_1^{(k)}, ..., x_n^{(k)})$$
$$...$$
$$x_i^{(k+1)} = F_i(x_1^{(k+1)}, ..., x_{i-1}^{(k+1)}, x_i^{(k)}..., x_n^{(k)})$$
$$...$$
$$x_n^{(k+1)} = F_n(x_1^{(k+1)}, ..., x_{n-1}^{(k+1)}, x_n^{(k)})$$

Lineaire vergelijkingen:

- Jacobi:

  Totale stap:
  $x_j^{(k+1)} = x_j^{(k)} - (\sum_{i=1}^{n} a_{ji} x_i^{(k)} - b_j)/a_{jj}$
  (Enkelvoudige stap heet Gauss-Seidel).

  - Matrixnotatie: A = L + D + U

    $Dx^{(k+1)} = b - (L+U)x^{(k)}$
    Ax = b herschrijven naar
    Dx = - (L+U)x + b
    Dan hierop een directe substitutie toepassen.
    $x^{(k+1)} = D^{-1}[- (L+U)x^{(k)} + b]$

  - Convergentie:

    $e^{(k+1)}$ is de $(k+1)^{de}$ iteratiefout;
    $e^{(k+1)} = Ge^{(k)}$ met $G = -D^{-1}(U+L)$
    $e^{(k+1)} = G^{(k+1)}e^{(0)}$
    $\Rightarrow \| e^{(k+1)}\| \leq \|G\|^{(k+1)} \| e^{(0)}\|$
    $\Rightarrow e^{(k+1)} \dashrightarrow 0$ als $\|G\| < 1$
    $\|G\| \neq 0$ en < 1: lineaire convergentie
    $\|G\| = 0$: kwadratische convergentie
    $\|G\| > 1$: divergentie
    Als A diagonaaldominant is, is Jacobi convergent (voldoende voorwaarde, maar geen nodige vwd.; dit is ook een voldoende vwd. voor een goede conditie).

    A is diagonaaldominant <=>
    $\forall i$ (i1..n):
    $|a_{ii}| > (\sum_{j=1}^{i-1} a_{ij} + \sum_{j=i+1}^{n} a_{ij})$

- Gauss-Seidel:

  Enkelvoudige stap:
  $x_j^{(k+1)} = x_j^{(k)} -(\sum_{i=1}^{j-1} a_{ji} x_i^{(k+1)} + \sum_{i=j}^{n} a_{ji} x_i^{(k)} - b_j)/a_{jj}$
  (Totale stap heet Jacobi).

  - Matrixnotatie: A = L + D + U

    $Dx^{(k+1)} = b - Lx^{(k+1)} - Ux^{(k)}$
    Ax = b herschr. naar (L+D)x = -Ux + b
    Dan hierop een directe substitutie toepassen.
    $x^{(k+1)} = (L+D)^{-1}[-Ux^{(k)} + b]$

  - Convergentie:

    $e^{(k+1)}$ is de $(k+1)^{de}$ iteratiefout;
    $e^{(k+1)} = Ge^{(k)}$ met $G = -(L+D)^{-1}U$
    $e^{(k+1)} = G^{(k+1)}e^{(0)}$
    $\Rightarrow \| e^{(k+1)}\| \leq \|G\|^{(k+1)} \| e^{(0)}\|$

$\Rightarrow e^{(k+1)} \dashrightarrow 0$ als $\|G\| < 1$

$\|G\| \neq 0$ en < 1: lineaire convergentie
$\|G\| = 0$: kwadratische convergentie
$\|G\| > 1$: divergentie
Als A diagonaaldominant is, is Gauss-Seidel convergent (voldoende voorwaarde, maar geen nodige vwd.; dit is ook een voldoende vwd. voor een goede conditie).
A is diagonaaldominant <=>
$\forall i$ (i=1..n):
$|a_{ii}| > (\sum_{j=1}^{i-1} a_{ij} + \sum_{j=i+1}^{n} a_{ij})$

- Matrixrekenen:
  - Inverse nemen:

    A is een nxn-matrix.
    I_n is de identieke matrix met grootte nxn.
    Schrijf $[A|I_n]$.
    Pas hierop Gauss-Jordan toe (Gausseliminatie gevolgd door spillen gelijk aan 1 maken en de rest 0).
    Nu: $[I_n|A^{-1}]$ verkregen.
  - Determinant:

    $$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

    Det(A) i/h algemeen:
    kies een rij (of kolom) i.
    Neem achtereenvolgens elk element op die rij (of kolom) en vermenigvuldig het met de determinant van A zonder rij i (of kolom i) en de kolom (of rij) waarop het element staat.
    Sommeer deze resultaten; wissel dan het teken af
    (1e ele +, 2e ele -, ...).

Nulpunten van veeltermen.

Konditie van het probleem f(x) = 0:
Als x* een wortel is van f(x) = 0, hoe zal de wortel dan wijzigen bij een kleine wijziging op de gegevens?
Stel $\varepsilon g(x)$ een perturbatie op f(x).
We zoeken nu de oplossing van
$h(x, \varepsilon) = 0$ met $h(x, \varepsilon) = f(x) + \varepsilon g(x)$
Noem deze oplossing $x^*(\varepsilon)$.
De gewenste oplossing $x^* = x^*(0)$.
$x^* - x^*(\varepsilon)$ is bij benadering $(\varepsilon g(x))/f'(x)$ als
$f'(x) \neq 0$ en ε klein.

We concluderen dat de konditie slecht is
als $|f'(x)|$ klein is; in het extreme geval:
$|f'(x)| = 0$ --> x^* is een meervoudige
wortel.

Opmerking: kleine $|f'(x)|$ komt vaak voor als er een aantal wortels zijn die dicht bij elkaar liggen.

Newton-Raphson:
$F(x^{(k)}) = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$

Binomium van Newton:
$(a + b)^n = \sum_{j=0}^{n} \binom{n}{j} a^{n-j} b^j$

OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 1: FOUTENANALYSE

1. Genormeerde bewegende kommavoorstelling. Analyseer de volgende algoritmen voor het bepalen van de basis, b, en het aantal cijfers in de mantisse, p:
   - bepaal_b <uit: b >
     1. $A \leftarrow 1$
     2. while $(A+1)-A=1$
        2.1. $A \leftarrow 2*A$
     3. $i \leftarrow 1$
     4. while $(A+i)=A$
        4.1. $i \leftarrow i+1$
     5. $b \leftarrow (A+i)-A$
   - bepaal_p <in: b; uit: p >
     1. $p \leftarrow 1$
     2. $z \leftarrow b$
     3. while $(z+1)-z=1$
        3.1. $p \leftarrow p+1$
        3.2. $z \leftarrow z*b$
2. Beschouw volgende algoritmen voor het berekenen van product en scalair product:
   - PRODUCT <in: $a_1,\ldots,a_n$; uit: $P=\Pi_{i=1}^{n} a_i$ >
     1. $P \leftarrow a_1$
     2. voor $i=2:1:n$
        2.1 $P \leftarrow P*a_i$
   - SCALAIR PRODUCT <in: $a_1,\ldots,a_n,b_1,\ldots,b_n$; uit: $SP=\sum_{i=1}^{n} a_i b_i$ >
     1. $SP \leftarrow a_1 b_1$
     2. voor $i=2:1:n$
     2.1 $SP \leftarrow SP + a_i * b_i$

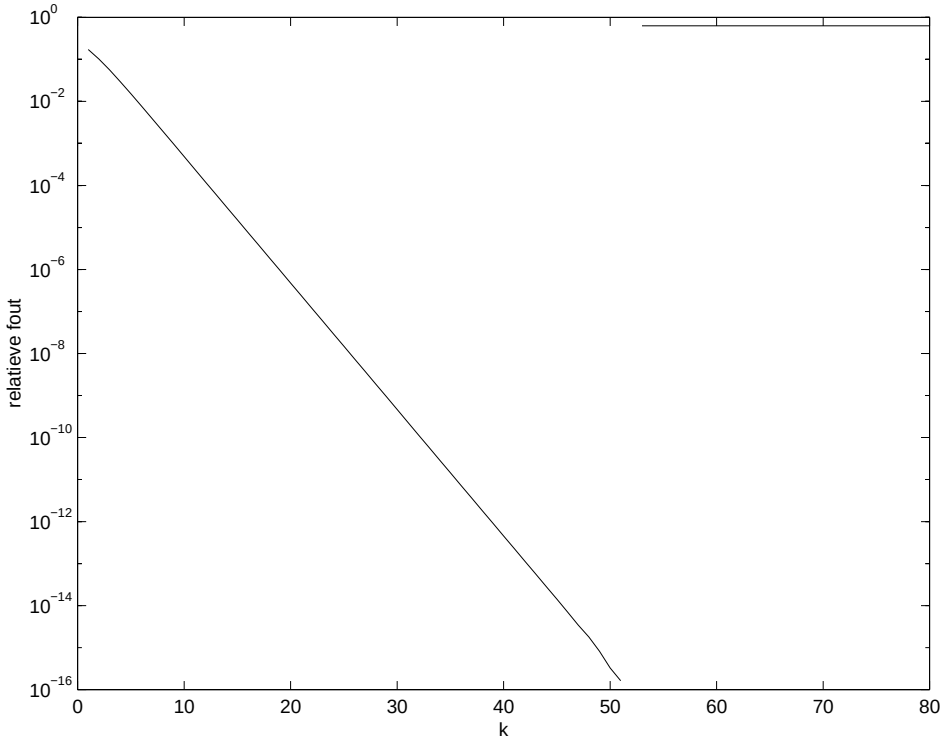
   Maak een afschatting van de fout wanneer de bewerkingen met eindige precisie in bewegende kommavoorstelling gebeuren. Veronderstel dat, met a en b exact voorgesteld in de computer, $\mathrm{fl}(a \circ b) = (a \circ b)(1+\epsilon)$ met $|\epsilon| \leq \epsilon_{\mathrm{mach}}$. Met $\circ$ bedoelen we zowel optelling als vermenigvuldiging.
3. Beschouw de volgende algoritmen voor het evalueren van een veelterm $V=\sum_{i=0}^{n} a_i x^i$ in x:
   - eval1 <in: $a_0,\ldots,a_n,x$; uit: V >
     1. $V \leftarrow a_n x^n$
     2. voor $i=n-1:-1:0$
        2.1 $V \leftarrow V + a_i * x^i$

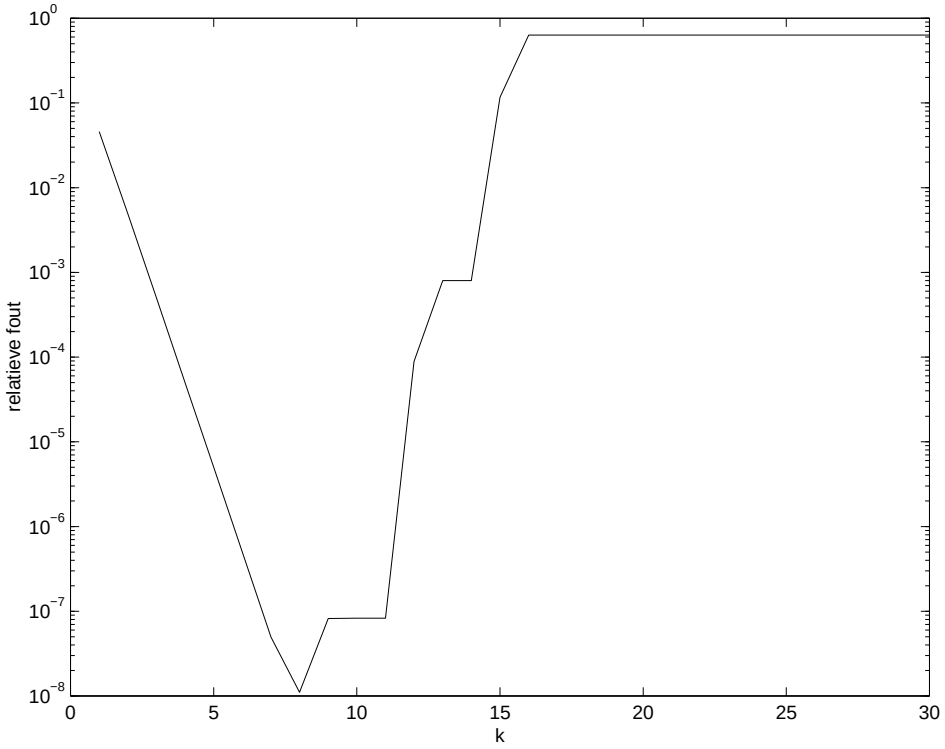
- eval2 <in: $a_0, \ldots, a_n, x$; uit: V >
  1. $V \leftarrow a_n$
  2. voor $i = n-1 : -1 : 0$
     2.1 $V \leftarrow V * x + a_i$
- eval3 <in: $a_0, \ldots, a_n, x$; uit: V >
  1. $p \leftarrow x$
  2. $V \leftarrow a_0$
  3. voor $i = 1 : 1 : n$
     3.1 $V \leftarrow V + a_i * p$
     3.2 $p \leftarrow p * x$

Welke methode is geimplementeerd (achterwaartse/voorwaartse evaluatie, horner)? Bepaal voor de drie algoritmen het aantal bewerkingen in functie van n en maak een afschatting van de fout indien de machineprecisie eindig is. Vergelijk de resultaten.

4. Examenvraag
   We weten dat $\lim_{x\to\infty}(1+1/x)^x = e^1$. We gaan de waarde e benaderen door $\tilde{e}_k = (1+1/x_k)^{x_k}$ uit te rekenen voor
   - $x_k = 2^k$ en
   - $x_k = 10^k$.

   In de figuren hieronder geven we de relatieve fout weer, d.w.z. $\left|\frac{\tilde{e}_k - e}{e}\right|$. De eerste figuur geeft de relatieve fout voor de machten van 2 terwijl de tweede figuur de fouten voor machten van 10 weergeeft.





Waarom zijn de twee grafieken zo verschillend?
Kan men hieruit afleiden met welke basis en met hoeveel beduidende cijfers de computer werkt?
Verklaar in detail je antwoord.

5. Extra oefening: analyseer het volgende recursieve algoritme:
SOM <in: $a_1, \ldots, a_n$; uit: $S = \sum_{i=1}^{n} a_i$ >
1. als $n = 1$
    1.1 $S \leftarrow a_1$
    1.2 anders
        $S \leftarrow S(a_1, \ldots, a_{n\text{div}2}) + S(a_{n\text{div}2+1}, \ldots, a_n)$

De bewerking 'div' levert het quotient van de gehele deling, dus $(2n+1)\text{div}2 = (2n)\text{div}2 = n,\ n \in \mathbb{N}$.
Toon aan dat in eindige precisie, de fout $S - \bar{S}$ voldoet aan:

$$n \leq 2^k,\ k \in \mathbb{N} \rightarrow |S - \bar{S}| \leq k\epsilon_{\text{mach}} \left(|a_1| + |a_2| + \cdots + |a_n|\right).$$

Een recursief algoritme leent zich tot een bewijs door ...

Oefenzitting 1: foutenanalyse.

Herhaling.

Absolute fout:

$\Delta x = \mathbf{x} - x$

Relatieve fout:

$\delta x = \Delta x / x = (\mathbf{x} - x)/x \approx \Delta x / \mathbf{x}$
$\mathbf{x} = x (1 + \delta x)$

Genormaliseerde getalvoorstelling:

$x = .c_k c_{k-1} ... E e$

Juist cijfer:
$|\mathbf{x} - x| \leq \frac{1}{2} b^i$ => cijfer op pos. i is juist.
$|\mathbf{x} - x| > \frac{1}{2} b^j$ => cijfer op pos. j is fout.

Laatste j.c. na de komma $p = -j-1$
$p \approx \log_b 1/(2|\Delta x|)$

Aantal j.b.c. $q = k - j$ met k de positie van het 1e b.c.
$q \approx \log_b 1/(2|\delta x|)$ als $\log_b 1/(2|\delta x|) \geq 0$

Machinenauwkeurigheid:

$\varepsilon_{mach} = \frac{1}{2} b^{-p+1}$

Oplossingen van de oefeningen:

Oefening 1:

Voorbeeld voor bepaal_b:

Stel, je werkt in het tiendelig talstelsel (b = 10) en je hebt 1 cijfer voor de mantisse (p = 1).
Dan: initialisatie: A = 1 = 1E0.

1e herhaling while: (A + 1) - A = 1, want A + 1 is 2 en kan exact voorgesteld worden als 2E0.
2e herhaling while: (A + 1) - A = 1, want A + 1 is 3 en kan exact voorgesteld worden als 3E0.
3e herhaling while: (A + 1) - A = 1, want A + 1 is 5 en kan exact voorgesteld worden als 5E0.
4e herhaling while: (A + 1) - A = 1, want A + 1 is 9 en kan exact voorgesteld worden als 9E0.
5e herhaling while: (A + 1) - A = 0, want A + 1 is 17 en kan niet meer exact voorgesteld worden; A wordt voorgesteld als 1 E 1, evenals (A + 1).

A is dus nu de kleinste macht van 2 waarbij verlies aan preciesie optreedt.
Nu: i = 1.

1e herhaling while 2: (A + i) = A, want (A + i) is 17 en kan niet exact voorgesteld worden.
2e herhaling while 2: (A + i) = A, want
(A + i) is 18 en kan niet exact voorgesteld worden.
3e herhaling while 2: (A + i) = A, want
(A + i) is 19 en kan niet exact voorgesteld worden.
4e herhaling while 2: (A + i) ≠ A, want
(A + i) is 20 en kan exact voorgesteld worden als 2E1.
b = (A + i) - A = 2E1 - 1E1 = 1E1 = 10.

Voorbeeld voor bepaal_p:

Stel: 10-delig talstelsel (b = 10) en 3 cijfers voor de mantisse (p = 3).
Dan: Initialiatie p = 1 en z = 10 = 10E0.

1e herhaling while: (z + 1) - z = 1, want (z + 1) = 11 en kan exact voorgesteld worden als 11E0; p = 2.
2e herhaling while: (z + 1) - z = 1, want (z + 1) = 101 en kan exact voorgesteld worden als 101E0; p = 3.
3e herhaling while: (z + 1) - z = 0, want (z + 1) = 1001 en kan niet meer exact voorgesteld worden; z = 100E1 en

(z + 1) wordt eveneens voorgesteld als 100E1 (er is 1 cijfer verloren gegaan, dus de preciesie is te klein); p blijft 3.

Theoretische uitleg:

Het bepalen van b en p.
Eigenschap: In $[b^t, b^{t+1}]$ moet je b/2 eenheden verdergaan om een verschillende voorstelling te krijgen.

Om b te bepalen moet je het interval proberen te bepalen, dit door het eerste getal g te zoeken zodat fl(g) = fl(g+1); dat getal g zit nu in het interval.

Vanaf 0 tellen duurt te lang => werken met machten van 2.

We zoeken een p zodat $2^p \in [b^t, b^{t+1}]$.
Een dergelijke p bestaat, want:
$2^p \in [b^t, b^{t+1}] \Leftrightarrow p \in [t \log_2 b, (t+1)\log_2 b]$
en $b \geq 2 \Rightarrow \log_2 b \geq 1$ dus
$|(t+1)\log_2 b - t \log_2 b| \geq 1$ dus er ligt een natuurlijk getal in dit interval.
t bepalen door opeenvolgende machten van b te nemen en b^t te vergelijken met b^{t+1} totdat $fl(b^t) \neq fl(b^{t+1})$

Oefening 2:

Opmerking:

ε^2, ε^3, ... zijn zeer klein en mogen verwaarloosd worden.

Product:

n = 2:
$(a1 * a2)(1 + \varepsilon_1) \leq (a1 * a2)(1 + \varepsilon_{mach})$

n = 3:
$((a1 * a2)(1 + \varepsilon_1))* a3\ (1+\varepsilon_2) = a1*a2*a3*(1 + \varepsilon_1)(1+\varepsilon_2)$
$= a1*a2*a3*(1 + \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2)$
$\approx a1*a2*a3*(1+ \varepsilon_1 + \varepsilon_2) \quad \leq \quad (a1 * a2 *a3)(1 + 2\varepsilon_{mach})$

n = 4:
$(a1*a2*a3*(1+\varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2))*a4*(1+\varepsilon_3)$
$= a1*a2*a3*a4*(1+\varepsilon_1+\varepsilon_2+ \varepsilon_1\varepsilon_2+\varepsilon_3+\varepsilon_1\varepsilon_3+\varepsilon_2\varepsilon_3 + \varepsilon_1\varepsilon_2\varepsilon_3)$
$\approx a1*a2*a3*a4*(1+\varepsilon_1+\varepsilon_2+\varepsilon_3) \quad \leq \quad (a1 * a2 *a3 * a4)(1 + 3\varepsilon_{mach})$

Algemeen:

$\mathbf{P} = P*(1 + \sum_{i=1}^{(n-1)}\varepsilon_i) \quad => \quad |P-\mathbf{P}| \leq P*(n-1)\varepsilon_{mach}$

Scalair product:

n = 2:
$((a1*b1)(1+\varepsilon_1) +(a2*b2)(1+\varepsilon_3))(1+\varepsilon_2)$
$= a1*b1(1+\varepsilon_1+\varepsilon_2) + (a2*b2)(1+\varepsilon_3 + \varepsilon_2) \quad \leq \quad (a1*b1 + a2*b2)(1 + 2\varepsilon_{mach})$

n = 3:
$(((a1*b1)(1+\varepsilon_1) + (a2*b2)(1+\varepsilon_4))(1+\varepsilon_2) + (a3*b3)(1+\varepsilon_5))(1+\varepsilon_3)$
$= (a1*b1)(1+\varepsilon_1 + \varepsilon_2 + \varepsilon_3) + (a2*b2)(1+ \varepsilon_2 + \varepsilon_3+ \varepsilon_4) + (a3*b3)(1+ \varepsilon_3+ \varepsilon_5)$
$\leq (a1*b1+a2*b2)(1+3\varepsilon_{mach})+(a3*b3)(1+2\varepsilon_{mach})$

n = 4:
$((((a1*b1)(1+\varepsilon_1) + (a2*b2)(1+\varepsilon_5))(1+\varepsilon_2) + (a3*b3)(1+\varepsilon_6))(1+\varepsilon_3) + (a4*b4)(1+\varepsilon_7))(1+\varepsilon_4)$
$= (a1*b1)(1+\varepsilon_1+\varepsilon_2+\varepsilon_3+\varepsilon_4) + a2*b2)(1+\varepsilon_2+\varepsilon_3+\varepsilon_4+ \varepsilon_5) + (a3*b3)(1+\varepsilon_6+\varepsilon_3+\varepsilon_4) +$
$(a4*b4)(1+\varepsilon_7+\varepsilon_4) \quad \leq \quad (a1*b1+a2*b2)(1+4\varepsilon_{mach})+(a3*b3)(1+3\varepsilon_{mach}) + (a4*b4)(1+ 2\varepsilon_{mach})$

Algemeen:

$\mathbf{SP} = (a1*b1)(1 + \sum_{i=1}^{n}\varepsilon_i) \ + \ \sum_{i=2}^{n} (ai*bi)(1 + (n-i+2)\varepsilon_j)$
$|SP - \mathbf{SP}| \leq (a1*b1)(n\varepsilon_{mach}) \ + \ \sum(ai*bi)((n-i+2)\varepsilon_{mach})$

Oefening 3:

Opmerking:

Een toekenning is GEEN bewerking; dit is een operatie. Je verandert immers niks aan het getal of de voorstelling ervan.

Algoritme 1 ≈ Achterwaartse evaluatie; SP nemen van ai en x^i voor dalende i.

Aantal bewerkingen in functie van n:
$\sum(i+1)$ Verm + n Opt

Fout: $\sum(ai*bi)((n-i+2)\varepsilon_{mach})$

Algoritme 2 ≈ Horner

Aantal bewerkingen in functie van n:
n Verm + n Opt

De fout ≈ $\sum(ai*bi)((n-i+2)\varepsilon_{mach})$

Algoritme 3 ≈ Voorwaartse evaluatie; SP nemen van ai en x^i voor stijgende i.

Aantal bewerkingen in functie van n:
(2(n-1) + 1) Verm + n Opt

Fout: $\sum(ai*bi)((n-i+2)\varepsilon_{mach})$
De fout voor algoritmen 1 en 3 is dus gelijk (zie oef. 2 SP), maar algoritme 3 is efficiënter.

Hornerevaluatie is echter het meest efficiënt (plaatsefficiënter dan de andere algoritmes en een klein beetje meer efficiënt in de tijd dan voorwaartse evaluatie omdat x^i niet in een apart geheugenplaatsje moet berekend worden, dus minder toekenningsoperaties).

Oefening 4:

De benaderingsfout daalt eerst bij beide grafieken; dan neemt de afrondingsfout toe omdat het maximaal aantal voorstelbare cijfers bereikt is (vanaf dan verandert het resultaat niet meer, want 1 + iets dat te klein is om voor te stellen, blijft 1).

De 2e grafiek convergeert sneller omdat $(1 + 1/10^k)10^k$ sneller naar e gaat dan $(1 + 1/2^k)2^k$.
Daarna treden er grotere afrondingsfouten op dan dat de benaderingsfout daalt en stijgt de grafiek.

De PC werkt met een binaire getalvoorstelling; daarom is de fout op de gegevens a.h.v. berekeningen met machten van 2 kleiner (deze getallen kunnen exact voorgesteld worden _ indien geen overflow) en heeft die grafiek ook een vloeiender verloop.

Machten van 2 kunnen exact voorgesteld worden in een binaire voorstelling; machten van 10 niet altijd. Dit zie je aan de meer geleidelijke stijging in de 2e grafiek: de afrondingsfout neemt niet alleen toe omdat het maximaal aantal cijfers in de mantisse bereikt is, maar ook

omdat het resultaat niet exact voorgesteld kan worden en omdat de afrondingsfouten sneller stijgen dan de benaderingsfout daalt.

Het aantal beduidende cijfers kan je afleiden uit de eerste grafiek: de sprong gebeurt op 53 => 53 cijfers.

Oefening 5:

Bewijs door inductie:

n = 1, k = 0:
Geen fout => als n even ok, dan elke n oneven ook ok.

n = 2, k = 1:
$\mathbf{S} = (a1 + a2)(1 + \varepsilon)$
$| S_{(a1a2)} - \mathbf{S_{(a1a2)}} | = (a1 + a2)*\varepsilon \leq 1*\varepsilon_{mach} (|a1| + |a2|)$

$n = 2^k$ ok, $k' = k + 1$, $n' = 2n$:
$S_{(a1..a2n)} = (S_{(a1..an)} + S_{(a(n+1)a2n)})(1 + \varepsilon) \; | S_{(a1..a2n)} - \mathbf{S_{(a1..a2n)}} |$
$= S_{(a1..an)} + S_{(a(n+1)a2n)} - S_{(a1..an)}(1 + \varepsilon) - S_{(a(n+1)a2n)}(1 + \varepsilon)$
$\leq | S_{(a1..an)} - \mathbf{S_{(a1..an)}}| + | S_{(a(n+1)a2n)} - \mathbf{S_{(a(n+1)a2n)}}| + |\varepsilon|*| \mathbf{S_{(a1..an)}} + \mathbf{S_{(a(n+1)a2n)}}|$
$\leq k*\varepsilon_{mach} (|a1| + |a2| + ... + |an|) + k*\varepsilon_{mach} (|a(n+1)| + |a(n+2)| + ... + |a2n|)$
$\quad + \varepsilon_{mach} (|a1| + |a2| + ... + |a2n|)$
$= (k+1)*\varepsilon_{mach} (|a1| + |a2| + ... + |a2n|)$

PC-zitting 1: Foutenanalyse

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Inleiding

Voor het uitwerken van de opgaven moet je '.m'-bestanden gebruiken die je kan vinden op

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

Kopieer de nodige bestanden naar je gebruikers-direcotry `D:\USER`.
Op het einde van deze zitting wordt verwacht dat je het antwoordblad achteraan ingevuld afgeeft. Hierop zal je niet gekwoteerd worden, maar het is voor ons eerder een indicatie om te zien in hoeverre de studenten de stof beheersen.

Veel succes!

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} met de permanente variabele `eps` in MATLAB.
Opgelet: `eps` stelt niet de machineprecisie voor, maar wel het verschil tussen het kleinste getal groter dan 1, voorstelbaar in de floating point-voorstelling, en 1. Controleer dit door $(1 + \mathbf{eps})$ van 1 af te trekken en daarna $(1 + \frac{\mathbf{eps}}{3})$ van 1 af te trekken. Wat gebeurt er als je $(1 + 0.70 * \mathbf{eps})$ van 1 aftrekt en hoe verklaar je dat?

2 Een onschadelijke berekening?

Voer de volgende berekeningen uit voor $x = 100$:

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

In theorie laat dit nochtans 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 zo toe om het verschil tussen deze exacte waarden en de berekende waarden te bepalen en op een grafiek te 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.

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.

4 Een onschadelijke afronding?

4.1 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?

4.2 Jammer maar helaas

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 en afkapt.
- het systeem reeds 1000 uur opstaat.
- een scud 1.61 m/s vliegt.

[1] beschouw enkel de eerste verwaarloosde bit

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?

5 Extra oefeningen

5.1 Veelterm evaluatie

Gegeven zijn drie bestanden `veelt_1.m`, `veelt_2.m` en `veelt_3.m`. In elk van die bestanden is een algoritme geïmplementeerd voor de evaluatie van een veelterm zoals die in de oefenzittingen werden behandeld. Welk algoritme is geïmplementeerd in welk bestand? Voor de functies uit om zo het aantal flops te bepalen. Met de MATLAB-functie `rand` kan je een een matrix met random elementen creëren. Door hiervan de eerste rij te selecteren, bekom je een vector met random getallen. Deze kan kan worden gebruikt als coëfficiëntenvector. Eventueel kan je eerst de vector nog met een getal vermenigvuldigen omdat de random getallen die MATLAB genereert allen kleiner dan één zijn.
De output van het commando `flops` geeft de som van alle bewerkingen, dus zowel optellingen als vermenigvuldigingen. Bepaal het aantal flops voor de drie methodes en kijk na of dit klopt volgens de resultaten uit de oefenzitting. Gebruik hiervoor een veelterm van dimensie 100.
Opmerking: In MATLAB beginnen de indices van vectoren en matrices bij 1. Dus a(1) is het eerste element van vector `a`. In het algoritme zoals wij het hebben gezien in de oefenzitting begint de vector `a` met de coëfficiënten van de veelterm echter met `a(0)`. Hou hier rekening mee bij het berekenen van het aantal flops.
Het bestand `veelt_achter.m` implementeert de achterwaartse evaluatie van een veelterm. Hiermee het aantal flops berekenen geeft een vertekend beeld, vermits MATLAB x^n als één bewerking en dus één flop aanrekent.

5.2 Evaluatie van een functie(2)

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.

5.3 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.

PC-zitting 1: foutenanalyse

Opmerkingen:

De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Niet maken:

oefening 2: lus 60 keer uitvoeren.
Oefening 4.1: reden: flops wordt door recente matlabversies niet meer ondersteund.
Oefening 4.2
Oefening 4.3

Herhaling:

Machinenauwkeurigheid:
εmach = ½ b-p+1

Voorbeeld:

Schrijf een ".m"-file die de functie
f(x) = (1 - cos(x))/x2 evalueert met n cijfers. Gebruik hierbij de functie fl die gedefinieerd is in fl.m (d:\user).

Oplossing is de volgende m-file, op te slaan als voorbeeld.m:

```
function y = voorbeeld(x,n)
y = fl(cos(x),n);
y = fl(1-y, n);
y = y/(fl(x^2,n));
y = fl(y,n);
```

Op de commandolijn de volgende commando's uitvoeren:

```
>> x=1;
>> for i=1:10,
   resultaat(i) = voorbeeld(x,8);
   x = x/4;
   end
>> resultaat'
```

Dit geeft volgende uitvoer:

```
ans =
0.45969769000000
```

0.49740128000000
0.49983744000000
0.49999871000000
0.50003968000000
0.50331648000000
0.50331648000000
0
0
0

Wat loopt er mis?
Limx->0 f(x) = ½ ≠ 0.

Grote fouten treden op tijdens de berekeningen, vooral bij het berekenen van (1 - cos(x)) (grafiek cos: cos(x) gaat naar 1 als x naar 0 gaat => verschil van 2 bijna gelijke getallen => grote fouten).

Je kan dit vermijden door f(x) te herschrijven als f(x) = 2*(sin(x/2))2/x2.
Waarom?
De gevaarlijke aftrekking wordt vermeden. De tweede berekeningsmethode is dus stabieler.

Oplossingen van de oefeningen:

Oefening 1:

Mogelijke reeks commando's:
path(path, 'd:\user')
format long
basis = bepaalb
aantal_cijfers= bepaalp
epsilon_mach = basis^(1-aantal_cijfers)/2
eps
controle1= 1 - (1+eps)
controle2 = 1 - (1+eps/3)
controle3 = 1 - (1 + 0,70*eps)

Vergelijken van eps en epsilon_mach:

eps = 2*epsilon_mach.

Reden:
eps = het verschil tussen het kleinste getal > 1 dat voorstelbaar is in floating point en 1, dus NIET de machinepreciesie, want εmach houdt rekening met afronding en eps niet.
In een interval [a, b] waarbij a en b exact voorstelbaar zijn, geeft eps de grootte van b-a aan.
Afronding: een getal dat midden in het interval [a,b] ligt waarbij a en b exact voorstelbaar zijn en alle getallen ertussen niet, wordt afgerond naar de grens waar het getal het dichtst bij ligt; afronding en eps bepalen samen dus de machinepreciesie.

Controles:
De eerste test geeft -eps, zoals verwacht, want (1 + eps) is exact voorstelbaar.
De tweede test geeft 0, want (1 + eps/3) wordt afgerond naar 1 (eps/3 is te klein om voorgesteld te kunnen worden en gaat dus verloren).
De derde test geeft -eps, want (1 + 0,70 eps) wordt afgerond naar (1 + eps).

Oefening 2:

Niet in de oefenzitting, niet thuis maken: 60 herhalingen van de lus.

Mogelijke reeks commando's:

```
x = 100
grens = 40
[res, eindres, eindx] = oefz1oef2_a(x,grens)
load exacte_x.mat
verschil = abs(exacte_x - res')
plot(1:40, verschil, 'b')
x = 100
grens = 40
verschil = abs(flipud(exacte_x) - eindres')
figure
plot(1:40, verschil, 'b')
figure
semilogy(1:40, verschil, 'r')
```

Gedrag van de fout:

De fouten op de worteltrekking blijven klein:
sqrt(sqrt(sqrt(...sqrt(x)))) =
x^(1/240)*(1+ε1)^(1/239)*
(1+ε2)^(1/238)*...*(1+ε40)^(1/20)

De fouten op de machtsverheffing worden echter enorm opgeblazen:
sqr(sqr(sqr(...sqr(y)))) =
y^(240)*(1+ε1)^(239)*
(1+ε2)^(238)*...*(1+ε40)^(20)

Vergelijken met resultaten van een zakrekenmachine:
een zakrekenmachine geeft een grotere fout.

Oefening 3:

limx->∞ f(x) = limx->∞ x[π/2 - arctan(x)]
= limx->∞ (π/2-arctan(x)/x)
= limx->∞ (-1/(1+x2))/(-1/x2)
= limx->∞ x2/(1+x2)
= 1.
(voorwaarde voor keuze van x: x > 1)

Mogelijke commando's:

```
x = 10
grens = 10
afronding = 8
res = oefz1oef3(x, afronding, grens)
```

Dit algoritme loopt fout als x groot wordt: de boogtangens wordt dan π/2 en men trekt dan 2

getallen van ongeveer gelijke grootte (en zelfde teken) van elkaar af => groot verlies aan juiste beduidende cijfers.

Bewijs:

TB komt overeen met TB2:

pi/2 - arctan(x) = arcsin(1/sqrt(1 + x2))
met tan2(x) + 1 = 1/cos2(x)
tan(π/2 - arctan(x)) = cotan(arctan(x))
= 1/tan(arctan(x))
= 1/x

tan(1/cos2(x)) = tan(arcsin(1/sqrt(1+x2)))
= sin(arcsin(1/sqrt(1+x2))) cos(arcsin(1/sqrt(1+x2)))
= (1/sqrt(1+x2)) sqrt(1-sin2(arcsin(1/sqrt(1+x2))))
= (1/sqrt(1+x2)) sqrt(1-sin(bgsin(1/sqrt(1+x2)))2)
= (1/sqrt(1+x2)) sqrt(1 - (1/sqrt(1+x2))2)
= (1/sqrt(1+x2)) sqrt(1 - (1/(1+x2)))
= (1/sqrt(1+x2)) sqrt((1 + x2 - 1)/(1 + x2))
= (1/sqrt(1+x2)) sqrt((x2)/(1 + x2))
= (1/sqrt(1+x2)) (x/sqrt(1 + x2))
= 1/x

Omdat de tangens gelijk is, moeten de argumenten gelijk zijn of antisupplementair; antisupplementair is onmogelijk omdat

1 > 1/sqrt(1+x2) > 0 => arcsin (1/sqrt(1+x2)) ligt in het interval]0, π/2[
x > 0 => arctan(x) zit in]0, π/2[

Beide leden van de gelijkheid liggen in hetzelfde interval, dus 1 van beide leden kan niet π meer zijn dan het andere.

Oefening 4.1:

Niet in de oefenzitting, niet thuis maken.

1= horner, 2 = achterwaarts, 3 = voorwaarts

Vergelijk verkregen waarden met verwachte waarden (zie opl. oefenzitting 1, oef. 3).

achterwaarts:
optellingen: n = 100
vermenigvuldigingen: n*(n+1)/2 = 5050
totaal = 5150

voorwaarts:
optellingen: n = 100
vermenigvuldigingen: 2n -1 = 199
totaal = 299

horner:

optellingen: n = 100
vermenigvuldigingen: n = 100
totaal: n = 200.

Deze oefening lukt niet meer in recente versies van matlab; flops wordt dan niet meer ondersteund.

Oefening 4.2:

Niet in de oefenzitting.

Herschrijven:

f(x) = e2x(1-tanh(x))
= e2x(1-(sinh(x)/cosh(x)))
= e2x((ex + e-x - ex + e-x)/(ex + e-x))
= 2e2xe-x/(ex + e-x)
= 2ex/(ex + e-x)
= 2/(1+e-2x)

Vergelijk de resultaten a.h.v. implementaties voor de originele versie van f(x) en de herschreven versie van f(x).

Oefening 4.3:

Niet in de oefenzitting, niet thuis maken.

OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 3: KONDITIE EN STABILITEIT

1.
   - Beschouw $y = f(x) = 1 + 2x - \frac{1}{1+x}$. Ga na voor welke waarden van x, Δy resp. δy het grootst zijn en illustreer a.h.v. een grafiek.
   - Onderzoek de stabiliteit van de volgende algoritmen t.o.v. de relatieve fout:

   (a) $y = f(x) = 1 + 2x - \frac{1}{1+x}$

   eval1 <in: x; uit: y >

   1. $y \leftarrow 2 * x$
   2. $y \leftarrow 1 + y$
   3. $z \leftarrow 1 + x$
   4. $z \leftarrow \frac{1}{z}$
   5. $y \leftarrow y - z$

   (b) $y = f(x) = \frac{(3+2x)x}{1+x}$

   eval2 <in: x; uit: y >

   1. $y \leftarrow 2 * x$
   2. $y \leftarrow 3 + y$
   3. $y \leftarrow y * x$
   4. $z \leftarrow 1 + x$
   5. $y \leftarrow \frac{y}{z}$

2. Onderzoek de konditie van onderstaande problemen en de stabiliteit van de bijhorende algoritmen.
   - $y = f(x) = x \sin(x)$

     eval <in: x; uit: y >

     1. $y \leftarrow \sin(x)$
     2. $y \leftarrow x * y$

   - $f(a, b) = \sqrt{a} - \sqrt{b}$

     eval1 <in: a, b; uit: f >

     1. $f \leftarrow \sqrt{a} - \sqrt{b}$

     eval2 <in: a, b; uit: f >

     1. $f \leftarrow \frac{a-b}{\sqrt{a}+\sqrt{b}}$

3. Stel dat $g(x)$ goed gekonditioneerd is en dat je een stabiel algoritme hebt ter evaluatie. Toon aan dat de volgende manier van werken onstabiel is:

   $g'(x)$ benaderen door $y = \frac{g(x+h)-g(x)}{h}$

   theoretisch: y beter naarmate $h \to 0$.
4. Herschrijf volgende uitdrukkingen om numerieke problemen te vermijden voor de aangegeven argumenten:
   - $\sqrt{x+1} - 1,\ x \approx 0$
   - $\sin(x) - \sin(y),\ x \approx y$
   - $x^2 - y^2,\ x \approx y$
   - $\frac{1-\cos(x)}{\sin(x)},\ x \approx 0$
5. Extra oefeningen
   - Welke formule is het nauwkeurigst? Voor welke waarden van a en b?

     $$a\sqrt{1-b^2} - b\sqrt{1-a^2} \quad \text{of} \quad \frac{(a+b)(a-b)}{a\sqrt{1-b^2} + b\sqrt{1-a^2}}$$

   - Konditie van veeltermevaluatie:
     Onderzoek de gevoeligheid van

     $$p(x) = \sum_{k=0}^{n} a_k x^k$$

     voor fouten op de coëfficiënten a_k. We kunnen afschatten:

     $$|\Delta p(x)| \leq \sum_{i=0}^{n} |\Delta a_k||x|^k.$$

     Men kan de veelterm $p(x)$ ook schrijven als lineaire combinatie van de Chebychev-veeltermen $T_k(x)$,

     $$p(x) = \sum_{k=0}^{n} c_k T_k(x).$$

     Deze zijn gedefinieerd door de rekursiebetrekking

     $$T_0(x) = 1,\ T_1(x) = x, \quad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad k = 1, 2, \ldots$$

     en voldoen aan

     $$T_k(\cos(t)) = \cos(kt).$$

     Ga dit na! Toon aan dat voor $x \in [-1, 1]$ de gevoeligheid van $p(x)$ voor fouten op de coëfficiënten c_k gegeven wordt door

     $$|\Delta p(x)| \leq \sum_{k=0}^{n} |\Delta c_k|.$$

k	$\bar{a}_{2k}$	$\bar{c}_{2k}$
0	$1.079e+00$	$1.157e+00$
1	$5.211e-03$	$1.005e-01$
2	$2.740e-02$	$1.307e-02$
3	$-3.163e+00$	$-1.007e-02$
4	$3.274e+01$	$1.057e-03$
5	$-1.129e+02$	$1.458e-03$
6	$1.977e+02$	$-9.920e-04$
7	$-1.918e+02$	$2.227e-04$
8	$9.846e+01$	$1.287e-04$
9	$-2.094e+01$	$-1.598e-04$

Beschouw als voorbeeld de veelterm

$$p(x) = \sum_{k=0}^{9} a_{2k} x^{2k} = \sum_{k=0}^{9} c_{2k} T_{2k}(x)$$

met de coëfficiënten, afgerond op 4 beduidende cijfers, gegeven in de tabel. Indien je exact rekent, hoe nauwkeurig kan je dan $p(x)$ evalueren op $[-1, 1]$, enerzijds gebruik makend van de coëfficiënten $\bar{a}_{2k}$ en anderzijds gebruik makend van de coëfficiënten $\bar{c}_{2k}$? Wat is het voordeel van te werken met de basis $\{T_0(x), T_1(x), \ldots, T_n(x)\}$ i.p.v. $\{1, x, x^2, \ldots, x^n\}$?

Opmerking:

$$\begin{aligned} \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_k(x) T_m(x) dx &= 0 \quad \text{als} \quad k \neq m \\ &= \tfrac{\pi}{2} \quad \text{als} \quad k = m \neq 0 \\ &= \pi \quad \text{als} \quad k = m = 0 \end{aligned}$$

Oefenzitting 2: Konditie & stabiliteit.

Herhaling.

Konditie:

Een probleem is slecht gekonditioneerd als voor gelijk welke berekeningsmethode de afwijking op het berekende resultaat groot zal zijn.

Konditie t.o.v. de absolute fout:

$\Delta kf = \Sigma i = 1n \, (\delta f/\delta xi) * \Delta xi$

$(= |f'(x)|)$ met n het aantal veranderlijken van f.

Konditie t.o.v. de relatieve fout:

$\delta kf = |x*(\Delta kf/f)|$

$(= |x*(f'(x)/f)|)$

Stabiliteit:

Een algoritme is stabiel als er in het begin van het algoritme mogelijk wel kleine afrondingsfouten gemaakt worden, maar voor de rest quasi exact gerekend wordt.
Het resultaat van de berekeningen kan ook bij een stabiel algoritme sterk afwijken van de exacte oplossing, nl. als de konditie van het probleem slecht is.

Stabiliteit t.o.v. de absolute fout:

$\Delta sf = |f(x) - f(x)|$

Stabiliteit t.o.v. de relatieve fout:

$\delta sf = |(f(x) - f(x))/f(x)|$

Opmerkingen:

Niet maken: Oef. 5.

Oplossingen van de oefeningen:

Oefening 1:

1) Kond. voor abs. fout: $\Delta ky \approx f'(x)/\Delta x$
$f'(x) = 2 + 1/(1+x)2$
$f'(x)$ is het grootst voor $x = -1$, dus voor $x \approx -1$ is f(x) slecht gekonditioneerd.

2) Kond. voor rel. fout: $\delta ky \approx x*(f'(x)/f(x))$

$x*(f'(x)/f(x)) = x*(2 + 1/(1+x)2) \; 1 + 2x - 1/(1+x)$
$= x*(1+x)*(2*(1+x)2 + 1) \; (1+x)2 \; [(1+2x)(1+x)-1]$
$= x*(2*(1+2x +x2) + 1) \; (1+x) \; (1+3x+2x2 -1)$
$= x*(2x2 + 4x + 3) \; (1+x)*x*(3+2x)$
$= (2x2 + 4x + 3) \; (1+x)(3+2x)$

Dit is het grootst voor $x = -1$ en $x = -3/2$, dus f(x) is slecht gekonditioneerd voor de relatieve fout voor $x \approx -1$ en $x \approx -3/2$.

3) Illustreer met een grafiek:

Schuine asymptoot: $y = 2x + 1$
(ax+b met $a = \lim_{x \to \infty} f(x)/x = 2$
$b = \lim_{x \to \infty} f(x) - ax = 1$)

Verticale asymptoot: $x = -1$
($x = a$ met $\lim_{x \to a} f(x) = \infty$
$a \neq \infty$
Goede kandidaten: nulpunten van de noemer.)

Horizontale asymptoot: /
($y = b$ met $\lim_{x \to \infty} f(x) = b$
$b \neq \infty$)

Slechte konditie waar de afgeleide groot is ($x = -1$) en slechte relatieve konditie voor $x = -3/2$, waar f(x) een nulpunt heeft.

0 is ook een nulpunt van f(x); nochtans is er daar geen slechte konditie voor de relatieve fout (reden: $x \approx f(x)$).

Title:
Maple plot
Creator:
Maple
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

4) Stabiliteit:

a) $f(x) = 1 + 2x - 1/(1+x)$

Opmerkingen:

$1/(1+\varepsilon) = 1-\varepsilon$

Termen $O(\varepsilon i)$ $(i > 1)$

verwaarlozen want zeer klein.

eval1:

$y = [(1 + 2x(1 + \varepsilon 1))(1+ \varepsilon 2) - ([1/((1+x)(1+\varepsilon 3))]*(1+\varepsilon 4))](1+\varepsilon 5)$

$= [(1+2x+2x\varepsilon 1)(1+\varepsilon 2) - 1/(1+x) - \varepsilon 4/(1+x) + \varepsilon 3/(1+x)] (1+\varepsilon 5)$

$= 1+2x -1/(1+x) + 2x\varepsilon 1 + \varepsilon 2(1 + 2x) - \varepsilon 4/(1+x) + \varepsilon 3/(1+x) + \varepsilon 5(1+2x-1/(1+x))$

$= y + 2x\varepsilon 1 + \varepsilon 2 (1+ 2x) + \varepsilon 3(1/(1+x)) - \varepsilon 4/(1+x) + \varepsilon 5y$

$|\Delta sy| = | y- y|$

$= |-2x\varepsilon 1-\varepsilon 2(1+2x)-\varepsilon 3(1/(1+x)) + \varepsilon 4/(1+x) -\varepsilon 5y|$

$\leq |-2x\varepsilon mach - \varepsilon mach(1+2x) -\varepsilon mach(1/(1+x)) + \varepsilon mach/(1+x) - \varepsilon machy|$

$|\delta sy| = |(y - y)/y|$

$= |\varepsilon 1(2x/y) + \varepsilon 2 (1+ 2x)/y + \varepsilon 3(1/(y(1+x))) - \varepsilon 4/(y(1+x)) + \varepsilon 5 |$

$\leq \varepsilon mach [|2x/y| + |(1+2x)/y| + 2/(|1+x||y|) + 1]$

Nu,

$y = 1 + 2x - 1/(1+x)$

$= (1+2x)(1+x)-1 (1+x)$

$= 1 + 3x + 2x2 -1 (1+x)$

$= x*(3+2x) (1+x)$

Dus

$|\delta sy| \leq \varepsilon mach [|(2(1+x))/(3+2x)| + |((1+2x)(1+x))/(x*(3+2x)) | + 2/|x*(3+2x)| + 1]$

Besluit:

eval1 is instabiel voor $x \approx 0$

$|\delta sy| >> |\delta ky|$ (ligt aan volgorde)

eval1 is (voorwaarts) stabiel voor $x \approx -3/2$

$|\delta sy| \approx |\delta ky|$ (groot)

(ligt aan probleem zelf)

eval1 is stabiel voor $x \approx -1$

$|\delta sy| \leq 3\varepsilon mach << |\delta ky|$

b) f(x) = ((3 + 2x)*x)/(1+x)

Opmerkingen:

$1/(1+\varepsilon) = 1-\varepsilon$

Termen $O(\varepsilon i)$ $(i > 1)$

verwaarlozen want zeer klein.

eval2:

$y = [(3 + 2x(1+ \varepsilon 1))\ (1+ \varepsilon 2)*x]\ (1+ \varepsilon 3)\ (1+x)(1+ \varepsilon 4)$

$= [3x+2x2(1+\varepsilon 1)](1+\varepsilon 2+ \varepsilon 3)(1-\varepsilon 4)\ (1+\varepsilon 5)\ (1+x)$

$= (3x+3x\varepsilon 2+3x\varepsilon 3+2x2+2x2\varepsilon 2+2x2\varepsilon 5+2x2\varepsilon 1)\ (\ 1-\varepsilon 4+\varepsilon 5)\ (1+x)$

$= 3x - 3x\varepsilon 4 + 3x\varepsilon 2 + 3x\varepsilon 3 + 3x\varepsilon 5\ (1+x) + 2x2 - 2x2\varepsilon 4 + 2x2\varepsilon 2 + 2x2\varepsilon 5$
$+ 2x2\varepsilon 1 + 2x2\varepsilon 3\ (1+x)$

$= 3x + 2x2 + 2x2\varepsilon 1 + (3x + 2x2)\varepsilon 2\ (1+x)\ (1+x) + (3x + 2x2)\varepsilon 3 + (3x + 2x2)\varepsilon 4$
$+ (3x + 2x2)\varepsilon 5\ (1+x)$

$= y + y[(\varepsilon 1(1+x)2x2)/((3x + 2x2)(1+x)) + \varepsilon 2+\varepsilon 3-\varepsilon 4+\varepsilon 5]$

$= y + y[(\varepsilon 12x)/(3 + 2x) + \varepsilon 2+\varepsilon 3-\varepsilon 4+\varepsilon 5]$

$|\Delta sy| = |y-y|$

$= |y- y + y[(\varepsilon 12x)/(3 + 2x) + \varepsilon 2+\varepsilon 3-\varepsilon 4+\varepsilon 5]|$

$= |\ y[(\varepsilon 12x)/(3 + 2x) + \varepsilon 2+\varepsilon 3-\varepsilon 4+\varepsilon 5]|\ |\Delta sy| \leq |\ y[(\varepsilon mach2x)/(3 + 2x) + 4\varepsilon mach]|$

$\leq \varepsilon mach * |y| * |2x/(2x+3) + 4|$

$\leq \varepsilon mach * |y| * 2|x/(2x+3) + 2|$

$|\delta sy| = |(y - y)/y| = |(\varepsilon 12x)/(3 + 2x) + \varepsilon 2+\varepsilon 3-\varepsilon 4+\varepsilon 5|$

Dus $|\ \delta sy| \leq \varepsilon mach\ (4 + |2x|/|3+2x|)$

Opmerking: 4 en niet 3, want |-a| = |a|

Besluit:

eval2 is stabiel voor $x \approx 0$

$|\delta sy| \approx |\delta ky|$

eval2 is stabiel voor $x \approx -3/2$

$|\delta sy| \approx |\delta ky|$ (groot)

(ligt aan probleem zelf)

eval2 is stabiel voor $x \approx -1$

$|\delta sy| \leq 6\varepsilon mach << |\delta ky|$

($|\delta ky|$ heel groot voor $x \approx -1$)

Oefening 2:

a) $f(x) = x*\sin(x)$

Konditie t.o.v. de absolute fout:

$\Delta ky \approx f'(x)\Delta x$

$f'(x) = (\sin(x) + x*\cos(x))$

Slecht gekonditioneerd voor grote waarden van x.

Konditie t.o.v. de relatieve fout:

$\delta ky \approx x*(f'(x)/f(x))\delta x$

$x*(f'(x)/f(x)) = x*(\sin(x) + x*\cos(x))\ x*\sin(x)$

$= \sin(x) + x*\cos(x)\ \sin(x)$

$= 1 + x*\cos(x)\ \sin(x)$

Slecht gekonditioneerd voor de nulpunten van sin(x), uitgezonderd $x = 0$.

Stabiliteit t.o.v. de absolute fout:

$y = (x*\sin(x)*(1+\varepsilon 1))\ (1+\varepsilon 2)$

$= (x*\sin(x) + \varepsilon 1*x*\sin(x))\ (1+\varepsilon 2)$

$= x*\sin(x) + \varepsilon 1*x*\sin(x) + \varepsilon 2*x*\sin(x)$

$= x*\sin(x) + (\varepsilon 1+\varepsilon 2)*x*\sin(x)$

$|\Delta sy| = |y - y| = |(\varepsilon 1+\varepsilon 2)*x*\sin(x)| \leq 2\varepsilon mach|x*\sin(x)|$

Stabiel.

Stabiliteit t.o.v. de relatieve fout:

$|\delta sy| = |(y - y)/y| = |(\varepsilon 1+\varepsilon 2)| \leq 2\varepsilon mach$

Stabiel.

b) $f(a,b) = \sqrt{a} - \sqrt{b}$

Konditie t.o.v. de absolute fout:

$\Delta kf = (\delta f/\delta a)\Delta a + (\delta f/\delta b)\Delta b$

$= 1/(2\sqrt{a})\Delta a + 1/(2\sqrt{b})\Delta b$

$|\Delta kf| = |1/(2\sqrt{a})\Delta a + 1/(2\sqrt{b})\Delta b| \leq \frac{1}{2}*(|\Delta a/\sqrt{a}| + |\Delta b/\sqrt{b}|)$

Slecht gekonditioneerd als $a \approx 0$ en/of $b \approx 0$.

Konditie t.o.v. de relatieve fout:

$\delta kf = \Delta kf / f$
$= (1/(2\sqrt{a})\Delta a + 1/(2\sqrt{b})\Delta b) (\sqrt{a} - \sqrt{b})$
$= (1/(2\sqrt{a}))\sqrt{a}\sqrt{a}\,\delta a + (\sqrt{a} - \sqrt{b})\, 1/(2\sqrt{b})\sqrt{b}\sqrt{b}\,\delta b\, (\sqrt{a} - \sqrt{b})$
$= (\frac{1}{2} * \sqrt{a}\,\delta a) + (\frac{1}{2} * \sqrt{b}\,\delta b) (\sqrt{a} - \sqrt{b})$
$= \sqrt{a}\,\delta a + \sqrt{b}\,\delta b\, 2(\sqrt{a} - \sqrt{b})$

$|\delta kf| \leq (\sqrt{a}\,|\delta a| + \sqrt{b}\,|\delta b|) /(2|\sqrt{a} - \sqrt{b}|)$

Slecht gekonditioneerd als $a \approx b$.

Stabiliteit v. eval1 t.o.v. de abs. fout:

$f = [(\sqrt{a}(1+k\varepsilon 1)) - (\sqrt{b}(1+k\varepsilon 2))] (1+\varepsilon 3)$
$= \sqrt{a}+\sqrt{a}*k\varepsilon 1+\sqrt{a}*\varepsilon 3-\sqrt{b}-\sqrt{b}*k\varepsilon 2-\sqrt{b}*\varepsilon 3$
$= (\sqrt{a} - \sqrt{b}) + \sqrt{a}(k\varepsilon 1+\varepsilon 3) - \sqrt{b}(k\varepsilon 2+\varepsilon 3)$
$= f + \sqrt{a}(k\varepsilon 1+\varepsilon 3) - \sqrt{b}(k\varepsilon 2+\varepsilon 3)$

Opmerking:
k = 1 als de worteltrekking is
geïmplementeerd als 1 bewerking
(bv. Matlab).
Anders iteratief proces, dus k > 1.

$|\Delta sf| = |f-f| = |\sqrt{a}(k\varepsilon 1+\varepsilon 3)| - |\sqrt{b}(k\varepsilon 2+\varepsilon 3)|$
$|\Delta sf| \leq (k + 1)\, \varepsilon mach\, (\sqrt{a} + \sqrt{b})$

Stabiliteit v. eval1 t.o.v. de rel. fout:

$|\delta sf| = |(f-f)/f|$
$= |(\sqrt{a}(k\varepsilon 1+\varepsilon 3)| - |\sqrt{b}(k\varepsilon 2+\varepsilon 3)) \,|\, |\sqrt{a} - \sqrt{b}|$

$|\delta sf| \leq (k + 1)\, \varepsilon mach\, ((\sqrt{a} + \sqrt{b})/ |\sqrt{a}-\sqrt{b}|)$

Stabiliteit v. eval2 t.o.v. de abs. fout:

$f = (a-b)(1+\varepsilon 3) * (1+\varepsilon 5) (\sqrt{a}(1+k\varepsilon 1) + \sqrt{b}(1+ k\varepsilon 2)) * (1+\varepsilon 4)$
$= (a - b) (1+\varepsilon 3) (1 - \varepsilon 4)(1+ \varepsilon 5) (\sqrt{a}(1+k\varepsilon 1) + \sqrt{b}(1+ k\varepsilon 2))$
$= (a - b) (1 + \varepsilon 3 - \varepsilon 4 + \varepsilon 5) (\sqrt{a}(1+k\varepsilon 1) + \sqrt{b}(1+ k\varepsilon 2))$
$= (a - b) * (\sqrt{a} + \sqrt{b})\ \sqrt{a} + \sqrt{b}\, (\sqrt{a}(1+k\varepsilon 1) + \sqrt{b}(1+ k\varepsilon 2))$
$\quad * (1 + \varepsilon 3 - \varepsilon 4 + \varepsilon 5)$
$= f * (1/[(1+ k\varepsilon 1\sqrt{a}+k\varepsilon 2\sqrt{b})/(\sqrt{a}+\sqrt{b})]) * (1+ \varepsilon 3 - \varepsilon 4+\varepsilon 5)$
$= f + f(\varepsilon 3 - \varepsilon 4+\varepsilon 5 - (k\varepsilon 1\sqrt{a} + k\varepsilon 2\sqrt{b})/(\sqrt{a} + \sqrt{b}))$

$|\Delta sf| = |f-f|$
$= |\, (a-b)/(\sqrt{a} + \sqrt{b}) * (\varepsilon 3 -\varepsilon 4+\varepsilon 5 - (k\varepsilon 1\sqrt{a} + k\varepsilon 2\sqrt{b})/ (\sqrt{a} + \sqrt{b}))|$

Stabiliteit v. eval2 t.o.v. de rel. fout:

$|\delta sf| = |(f-f)/f|$
$= |(\varepsilon 3- \varepsilon 4 + \varepsilon 5 - (k\varepsilon 1\sqrt{a} + k\varepsilon 2\sqrt{b}) / (\sqrt{a} + \sqrt{b}))|$
$\leq \varepsilon mach*(3 + (k\sqrt{a})/|\sqrt{a} + \sqrt{b}| + (k\sqrt{b})/|\sqrt{a} + \sqrt{b}|)$

Oefening 3:

Stel: g(x) goed gekonditioneerd
stabiel algo om g(x) te evalueren.

T.B.: g'(x) = (g(x+h) - g(x))/h (met h --> 0 i/d limiet)
is onstabiel algo.

Uitwerking: ($k \in |N$; $y = g'(x)$)

$g(x) = g(x)*(1 + k*\varepsilon 1)$

$g(x+h) = g(x+h)*(1 + k*\varepsilon 2)$

$$g'(x) = \frac{(g(x+h)(1+k\varepsilon 2) - g(x)(1+k\varepsilon 1))*(1+\varepsilon 3)*(1+\varepsilon 4)}{h}$$

$$= \frac{[g(x+h) - g(x)]*(1+\varepsilon 3 + \varepsilon 4)}{h} + k*\frac{[g(x+h)*\varepsilon 2 - g(x)*\varepsilon 1](1 + \varepsilon 3 + \varepsilon 4)}{h}$$

$|\Delta sy| = |y*(\varepsilon 3+\varepsilon 4)+k*([g(x+h)*\varepsilon 2-g(x)*\varepsilon 1]/h)(1+\varepsilon 3+ \varepsilon 4)|$
$\leq |2\varepsilon mach*y + k\varepsilon machy)| \approx \varepsilon mach*|(k+2)y|$

$|\delta sy| = |(\varepsilon 3+\varepsilon 4)+ k*([g(x+h)*\varepsilon 2-g(x)*\varepsilon 1]/(h*y))|$
$\leq 2\varepsilon mach + k* |[\varepsilon 2g(x+h) - \varepsilon 1g(x)]/[h*y]|$
$\leq 2\varepsilon mach + k* (|\varepsilon 2g(x+h)| + |\varepsilon 1g(x)|)/|h*y|$
$\leq 2\varepsilon mach + k*\varepsilon mach (|g(x+h)| + |g(x)|)/|h*y|$
$\leq 2\varepsilon mach + k*\varepsilon mach (|g(x+h)| + |g(x)|)/|g(x+h)-g(x)|$

Instabiel als h --> 0 (|g(x+h)-g(x)| --> 0)
Instabiel als |g'(x)| --> 0 (Noemer = h*g'(x))
Instabiel als g(x) --> 0 (Noemer --> 0)

Oefening 4:

1) Gevaarlijke aftrekking voor $x \approx 0$.

$\sqrt{(x+1)} - 1 = [(\sqrt{(x+1)} - 1)(\sqrt{(x+1)} + 1)]/(\sqrt{(x+1)}+1)$
$= [(x + 1) - 1] / (\sqrt{(x+1)}+1)$
$= x/(\sqrt{(x+1)}+1)$

2) Gevaarlijke aftrekking voor $x \approx y$

($\sin(x) \approx \sin(y)$)
Als $\sin(x) \approx \sin(y)$, maar $x \neq y$, dan:
$\sin(x) - \sin(y) = 2\cos((x+y)/2)\sin((x-y)/2)$
(Sin is een periodische functie, dus bv. $x = \pi$ en $y = 3\pi$
hebben dezelfde sin, maar (x - y) is geen gevaarlijke aftrekking).

Herschrijven zoals hieronder kan, maar matlab geeft dezelfde resultaten voor het originele algoritme en het herschreven algo.

$\sin(x) - \sin(y) = [\sin(x)-\sin(y)][\sin(x)+\sin(y)] [\sin(x) + \sin(y)]$
$= \sin2(x) - \sin2(y) [\sin(x) + \sin(y)]$
$= \sin2(x) - (1 - \cos2(y)) [\sin(x) + \sin(y)]$
$= \sin2(x) + \cos2(y) - 1 [\sin(x) + \sin(y)]$

3) Gevaarlijke aftrekking voor $x \approx y$

($x2 \approx y2$)

$x2 - y2 = (x - y)(x + y)$

Ook de herschreven uitdrukking kan problemen geven, maar deze formule is stabieler; x en y zijn immers kleiner dan hun kwadraat en (x + y) is stabiel.

4) Gevaarlijke aftrekking voor $x \approx 0$, $x \neq 0$.

(limx->0 cos(x) = 1)

[1- cos(x)]/ sin(x) = [(1- cos(x))*(1+cos(x))] sin(x) *(1+cos(x))
= [1-cos2(x)]/[sin(x) *(1+cos(x))]
= sin2(x)/[sin(x) *(1+cos(x))]
= sin(x)/(1 + cos(x))

limx->π (1+ cos(x)) = 0, maar ook limx->π sin(x) = 0.

Herschrijven zorgt dus niet voor bijkomende problemen.

Alternatieve oplossing:

cos(x) = 1-2sin2(x/2)

sin(x) = 2sin(x/2)cos(x/2)

[1-cos(x)]/sin(x) = [1 - 1 + 2sin2(x/2)]/sin(x)
= [2 sin2(x/2)]/sin(x)
= [2 sin2(x/2)]/[2sin(x/2)cos(x/2)]
= sin(x/2)/cos(x/2)
= tg(x/2)

Oefening 5: NIET.

pc-zitting 2: Konditie en Stabiliteit

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Copieer de nodige bestanden op

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

naar je home-directory en pas het MATLAB pad aan.

1 Vierkantsvergelijking

Gebruik het bestand `vierkant.m` om de wortels van een vierkantsvergelijking te berekenen. Ga na hoe beide algoritmes de wortels berekenenen en pas die dan toe op

$$x^2 - bx + 2 = 0,$$

voor $b = 10^7, 10^8, 10^9$. Verklaar de verschillen. Welk algoritme is het nauwkeurigst?

2 Wortel van een complex getal

Ontwikkel een stabiel algoritme voor het berekenen van een wortel $x + iy$ van een complex getal $a + ib$. Implementeer dit algoritme in een matlab-functie. Hint: $a + ib = (x + iy)^2$ en stel reële en imaginaire delen gelijk aan mekaar.

3 Numerieke differentiatie

Schrijf een .m-file die de afgeleide van de functie $y = 5e^x$ in 0 benadert met behulp van de volgende differentieformule:

$$f'(0) \approx \frac{f(h) - f(0)}{h}.$$

Reken dit uit voor $h = \frac{1}{10}, \frac{1}{100}, \frac{1}{1000}, \ldots$. Zet op een grafiek de fout uit in functie van h en verklaar het gedrag. Daarbij kan het nuttig zijn om logaritmische schalen te kiezen. Hoe zie je dat de bovenstaande formule een eerste orde benadering voor de afgeleide is, i.e.

$$f'(x) = \frac{f(x + h) - f(x)}{h} + O(h).$$

Probeer ook eens de formule

$$f'(0) \approx \frac{f(h) - f(-h)}{2h}.$$

Wat is nu de orde van de benadering?

4 Extra oefeningen

4.1 Konditieonderzoek

Beschouw de functie $\arctan(Kx)$ voor verschillende waarden van K. Onderzoek hoe deze parameter de konditie beïnvloedt voor $x \approx 0$ door de functie te evalueren voor kleine waarden van x en dan een perturbatie op het gegeven aan te brengen. Maak ook de grafiek van de functie en zet de konditiegetallen t.o.v. de absolute en relatieve fouten uit in functie van x. Verklaar de resultaten.

4.2 Evaluatie van een functie

Beschouw de functie

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

Evalueer de functie voor x-waarden dicht bij 0. Voor $x = 0$ is de limietwaarde $= 2$. Schrijf hiervoor een '.m'-bestand. Gebruik hierbij de functie `fl`.
Evalueer de functie voor verschillende waarden van x en voor een verschillend aantal cijfers in de mantisse. Zoek combinaties waarvoor het fout loopt. Waarom loopt het fout?
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. Vermits $\frac{1}{2}e^x(1 - e^{-2x}) = \sinh(x)$ geldt:

$$f(x) = \frac{2\sinh(x)}{xe^x}$$

Nu bekomen we wel een nauwkeurig resultaat.

4.3 Berekening van $(e^x - 1)/x$

Implementeer hiervoor de volgende algoritmen

$$\begin{array}{l} if\ (x = 0) \\ \quad f = 1 \\ else \\ \quad f = (e^x - 1)/x \\ end \end{array}$$

en

$$\begin{array}{l} y = e^x \\ if\ (y = 1) \\ \quad f = 1 \\ else \\ \quad f = (y - 1)/\log y \\ end \end{array}$$

Deze twee algoritmen zijn theoretisch equivalent. Waarom is het tweede algoritme nauwkeuriger voor kleine waarden van x? Merk op dat de 'gevaarlijke' aftrekking uit het eerste algoritme nog steeds aanwezig is.

PC-zitting 2: konditie en stabiliteit.

Opmerkingen:

De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Oplossingen van de oefeningen:

Oefening 1:

Mogelijke reeks commando's:
path(path, 'd:\user')
format long
[y, z] = vierkant(1, -742, 2, 5)
[y2, z2] = vierkant(1, -742, 2, 10)

Vergelijken van de resultaten:

Met een mantisse van 10 cijfers geven beide methoden hetzelfde resultaat. Bij 5 cijfers is dit niet het geval. Vooral als er weinig cijfers zijn in de mantisse is het belangrijk om stabiele algoritmen te gebruiken.

Verklaring stabiel/onstabiel:
De niet-stabiele methode houdt geen rekening met het bestaan van een grote en kleine wortel; beide wortels worden op dezelfde manier berekend. Als de vierkantswortel van de discriminant ongeveer gelijk is aan de term ervoor (-b), komt hier een gevaarlijke aftrekking te staan in de formule.
De stabiele methode berekent de grootste wortel volgens de traditionele formule en de kleinste wortel op een alternatieve en stabielere manier.

Oefening 2:

Theoretische uitwerking stabiel algoritme:
$(x+Iy)2 = a+Ib$
$\Rightarrow x2 - y2 = a$
$2xy = b$
$\Rightarrow y = b/(2x)$
$4x4-4ax2-b2 = 0$
$D = 16(a2 + b2)$
$\Rightarrow x2 = (4a \pm 4\sqrt{(a2+b2)})/8$
$= a/2 \pm \sqrt{(a2+b2)}/2$

We weten dat x reëel is en dus $x2 >= 0$

Hieruit volgt: $x2 = a/2 + \sqrt{(a2+b2)/2}$

Als a > 0 stellen we:

$x = \sqrt{(a/2 + \sqrt{(a2+b2)/2})}$

$y = b/(2x)$

Als a<0 en b ≈ 0 is dit geen stabiele werkwijze (aftrekken van 2 ongeveer gelijke getallen).
We hebben dan:

$$x2 = \frac{(a+\sqrt{(a2+b2)}) * (a-\sqrt{(a2+b2)})}{2 * (a-\sqrt{(a2+b2)})}$$

$$= \frac{b2}{2 * (\sqrt{(a2+b2)} - a)}.$$

P.S. Dit resultaat kan ook gevonden worden door naar y2 op te lossen.

Mogelijke reeks commando's:

```
a_plus_bi = sqrt(-4)
a = real(a_plus_bi)
b = imag(a_plus_bi)
[x, y] = oefz2oef2a(a, b)
[echte_x, echte_y] = oefz2oef2b(a, b)
a_plus_bi = sqrt(a_plus_bi)
a = real(a_plus_bi)
b = imag(a_plus_bi)
[x, y] = oefz2oef2a(a, b)
[echte_x, echte_y] = oefz2oef2b(a, b)
```

Oefening 3:

Mogelijke commando's:

```
n = 5
aantal = 20
h_init = 1/2
[benadering, echte_waarde, h] = oefz2oef3(h_init, aantal, n)
plot(1:aantal, benadering, 'b', 1:aantal, echte_waarde, 'r')
figure
verschil = abs(benadering - echte_waarde)
plot(1:aantal, verschil, 'b')
figure
semilogy(1:aantal, verschil, 'b')
% delen door 1/h komt overeen met
% vermenigv. met h.
fout_inv_h = verschil.*h;
figure
plot(1:aantal, fout_inv_h, 'b', 1:aantal, h, 'r')
```

Foutengedrag:

De totale fout bestaat uit 2 componenten: benaderingsfouten en afrondingsfouten.

Eerst daalt de totale fout omdat de benaderingsfouten sterker dalen dan dat de afrondingsfouten stijgen.

Daarna stijgt de totale fout weer omdat de afrondingsfouten nu sterker stijgen.

x = 1/16 is klein; t.e.m. x = 1/512 geen probleem:
[5*ex - 5]/x = 5*[ex - 1]*(1/x)
met ex voldoende verschillend van 1.

x = 1/1024 (= 1/(210)) te klein:
gevaarlijke aftrekking
=> groot verlies aan j.b.c.
=> grote fout.

Oefening 4:

Mogelijke commando's:

```
K = 5
x_waarden = 0:0.01:1
functie_waarden = atan(K*x_waarden)
figure
plot(x_waarden, functie_waarden, 'b')
x_1 = 1
aantal = 50
perturbatie = 1*10^(-15)
[abskond, relkond, f_x, f_pert_x, x, x_pert] = oefz2oef4(x_1, aantal, K, perturbatie)
figure
semilogy(1:aantal,abskond,'b',1:aantal,relkond,'r')
figure
semilogy(x,abskond,'b',x,relkond,'r')
abs_verschil = abs(f_x - f_pert_x)
rel_verschil = abs_verschil ./ abs(f_x)
figure
semilogy(x, abs_verschil, 'b', x, rel_verschil, 'r')
figure
plot(x, f_x, 'b');
figure
plot(x_pert, f_pert_x, 'r');
neg_x = zeros(size(x)) - x;


figure
plot(neg_x, f_x, 'g');
neg_x_pert = zeros(size(x_pert)) - x_pert;
figure
plot(neg_x_pert, f_pert_x, 'm')
figure
plot(x, f_x, 'b');
hold
plot(neg_x, f_x, 'g');
figure
plot(x_pert, f_pert_x, 'r');
hold
plot(neg_x_pert, f_pert_x, 'm');
figure
semilogy(x, abskond, 'b');
```

```
hold
semilogy(x, relkond, 'r');
semilogy(neg_x, abskond, 'g');
semilogy(neg_x, relkond, 'm');
figure
semilogy(x, abs_verschil, 'b');
hold
semilogy(x, rel_verschil, 'r');
semilogy(neg_x, abs_verschil, 'g');
semilogy(neg_x, rel_verschil, 'm');
K = 10
x_waarden = 0:0.01:1


functie_waarden = atan(K*x_waarden)
figure
plot(x_waarden, functie_waarden, 'b')
x_1 = 1
aantal = 50
perturbatie = 1*10^(-15)
[abskond, relkond, f_x, f_pert_x, x, x_pert] = oefz2oef4(x_1, aantal, K, perturbatie)
figure
semilogy(1:aantal,abskond,'b',1:aantal,relkond,'r')
figure
semilogy(x,abskond,'b',x,relkond,'r')
abs_verschil = abs(f_x - f_pert_x)
rel_verschil = abs_verschil ./ abs(f_x)
figure
semilogy(x, abs_verschil, 'b', x, rel_verschil, 'r')
figure
plot(x, f_x, 'b');
figure
plot(x_pert, f_pert_x, 'r');
neg_x = zeros(size(x)) - x;
figure
plot(neg_x, f_x, 'g');
neg_x_pert = zeros(size(x_pert)) - x_pert;
figure


plot(neg_x_pert, f_pert_x, 'm')
figure
plot(x, f_x, 'b');
hold
plot(neg_x, f_x, 'g');
figure
plot(x_pert, f_pert_x, 'r');
hold
plot(neg_x_pert, f_pert_x, 'm');
figure
semilogy(x, abskond, 'b');
hold
semilogy(x, relkond, 'r');
semilogy(neg_x, abskond, 'g');
semilogy(neg_x, relkond, 'm');
```

```
figure
semilogy(x, abs_verschil, 'b');
hold
semilogy(x, rel_verschil, 'r');
semilogy(neg_x, abs_verschil, 'g');
semilogy(neg_x, rel_verschil, 'm');
K = 20
x_waarden = 0:0.01:1
functie_waarden = atan(K*x_waarden)
figure
plot(x_waarden, functie_waarden, 'b')
x_1 = 1


aantal = 50
perturbatie = 1*10^(-15)
[abskond, relkond, f_x, f_pert_x, x, x_pert] = oefz2oef4(x_1, aantal, K, perturbatie)
figure
semilogy(1:aantal,abskond,'b',1:aantal,relkond,'r')
figure
semilogy(x,abskond,'b',x,relkond,'r')
abs_verschil = abs(f_x - f_pert_x)
rel_verschil = abs_verschil ./ abs(f_x)
figure
semilogy(x, abs_verschil, 'b', x, rel_verschil, 'r')
figure
plot(x, f_x, 'b');
figure
plot(x_pert, f_pert_x, 'r');
neg_x = zeros(size(x)) - x;
figure
plot(neg_x, f_x, 'g');
neg_x_pert = zeros(size(x_pert)) - x_pert;
figure
plot(neg_x_pert, f_pert_x, 'm')
figure
plot(x, f_x, 'b');
hold


plot(neg_x, f_x, 'g');
figure
plot(x_pert, f_pert_x, 'r');
hold
plot(neg_x_pert, f_pert_x, 'm');
figure
semilogy(x, abskond, 'b');
hold
semilogy(x, relkond, 'r');
semilogy(neg_x, abskond, 'g');
semilogy(neg_x, relkond, 'm');
figure
semilogy(x, abs_verschil, 'b');
hold
semilogy(x, rel_verschil, 'r');
```

```
semilogy(neg_x, abs_verschil, 'g');
semilogy(neg_x, rel_verschil, 'm');
```

Verklaring foutengedrag:

Als K kleiner wordt, daalt de absolute fout, maar de rel. fout blijft gelijk.
Het absolute en relatieve konditiegetal dalen, dus de resultaten voor de echte x-waarden en de gepermuteerde x-waarden komen ongeveer overeen.
Als K daalt verbetert de konditie voor $x \approx 0$.
Als K daalt, krijgt atan(Kx) een meer vloeiend verloop (minder trapfunctie-achtig); dicht bij 0 wordt de afgeleide kleiner bij dalende K, dus de konditie (rel. & abs) wordt beter.

Oefening 5.1:

Mogelijke commando's:

```
x_1 = 10;
aantal = 20;
n = 5;
[res, abs_fout] = oefz2oef51_a(x_1, aantal, n)
plot(1:aantal, res, 'b', 1:aantal, abs_fout, 'r')
[res2, abs_fout2] = oefz2oef51_b(x_1, aantal, n)
plot(1:aantal, res2, 'b', 1:aantal, abs_fout2, 'r')
```

Voor welke combinaties van x en n (n = aantal cijfers i/d mantisse) loopt het fout?
Als x klein is en als de mantisse een klein aantal cijfers heeft.
Bij de deling is er een groot verlies aan j.b.c. en ook e(-2*x) zal niet erg nauwkeurig berekend worden.
Het 2e algoritme is stabieler, maar voor x in de buurt van 0 zijn er nog steeds vrij grote fouten.

Oefening 5.2:

Mogelijke commando's:

```
x_init = 10^(-1);
aantal = 15;
[res, x] = oefz2oef52_a(x_init, aantal)
plot(1:aantal, res, 'b')
[res2, x] = oefz2oef52_b(x_init, aantal)
plot(1:aantal, res, 'b')
verschil = res - res2;
plot(1:aantal, verschil, 'b')
```

Waarom is het 2e algo nauwkeuriger voor kleine waarden van x? Nochtans is de gevaarlijke aftrekking uit algo 1 nog steeds aanwezig.
Waarden in de buurt van $x = 0$ kunnen misschien nog voorgesteld worden als $x \neq 0$, terwijl ex wel 1 geeft (door afrondingsfouten).

Probleem:
Aftrekking van 2 getallen die ongeveer even groot zijn en van hetzelfde teken.

Waarom is de 2e versie beter?:
De 2e versie is beter omdat door het nemen van de exponentiele functie de test verbetert; voor een x-waarde in de buurt van 0 maar verschillend van 0, wordt soms e^x ook voorgesteld als 1.

pc-zitting 3: Het oplossen van stelsels lineaire vergelijkingen

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Inleiding

Voor het uitwerken van de opgaven moet je '.m'-bestanden gebruiken die je kan vinden op

`http:\\www.cs.kuleuven.ac.be\~wimm\oefenzittingen\`

1 MATLAB–functies voor deze oefenzitting

1.1 Genereren van matrices

$m =$ **genmat**(n) genereert een matrix $m = [a\ b]$ met a een $n \times n$–matrix en b een $n \times 1$–vector.

$m =$ **genmatc**(n) genereert een matrix $m = [a\ b]$ met a een $n \times n$–matrix en b een $n \times 1$–vector. De matrix a is nu wel slecht geconditioneerd.

Door deze bevelen wordt een stelsel $ax = b$ gegenereerd. De oplossingsvector x bestaat uit natuurlijke getallen.

1.2 Stelsels oplossen

$g =$ **gauss1**(m) waarbij $m = [a\ b]$ met a een $n \times n$–matrix en b een $n \times 1$–vector. Dit bevel maakt de matrix

$$\left(\begin{array}{c|c|c} & & 1 \\ a & b & \vdots \\ & & n \end{array}\right)$$

aan en past Gauss–eliminatie toe (Eerste deel, algoritme 3.4, pag. 61).

$g =$ **gauss2**(m) analoog als gauss1 maar met optimale rij–pivotering (Eerste deel, algoritme 3.5, pag. 69).

$[q, r] =$ **qr**(m) berekent een qr-factorisatie van de matrix m.

$x =$ **asubst**(g) , resp. $x =$ **asubst**(r) voert achterwaartse substitutie uit op het resultaat van **gauss1** of **gauss2**, resp. **qr**, en geeft de oplossing x van het stelsel $ax = b$.

2 Oefeningen

De LU ontbinding

1. (a) Stel

$$a = \begin{pmatrix} 7 & 8 & 0 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Gebruik het bevel **lu** om de LU ontbinding van a te berekenen.
Wat is $l * u$? Waarmee komt deze ontbinding overeen? Geef de Matlab-bevelen om uit l en u de determinant van a te berekenen.

(b) Stel

$$a = \begin{pmatrix} 1 & 4 & 3 \\ 4 & 3 & 5 \\ 9 & 8 & 0 \end{pmatrix}$$

Bereken van deze matrix a de LU ontbinding.
Wat merk je op als je de LU ontbinding van deze matrix bekijkt?
Geef een verklaring voor wat er gebeurd is.

2. Gauss-eliminatie

Gebruik het bevel **genmat** om een 6×7–matrix $m = [a\ b]$ te genereren. Los het stelsel $ax = b$ op m.b.v. de functies **gauss1**, **gauss2** en **qr**. Je bekomt dus drie oplossingen voor hetzelfde stelsel.
Geef de relatieve fout van de drie oplossingen indien je weet dat de exacte oplossing een vector met gehele getallen is. Bereken ook de residus.
Verklaar de verschillen.

3. Conditie

Genereer m.b.v. het bevel **genmatc** een 6×7–matrix $m = [a\ b]$.
Wat is het conditiegetal van de matrix a?
Los het stelsel op met de functies **gauss2** en **qr**. Bereken de relatieve fout van de oplossingen en de residus, als je weet dat de exacte oplossing uit natuurlijke getallen bestaat. Vergelijk met de resultaten uit de vorige oefening.
Wat is de rol van het conditiegetal van de matrix A?

4. Implementatie-opdracht

Schrijf een functie

```
[q,r]=qrstep(m)
```

die een **qr**-factorisatie berekent van een matrix $m = [a\ b]$ van de volgende structuur:

$$m = \left[\begin{array}{ccccc|ccc} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} & b_{1,1} & \cdots & b_{1,p} \\ 0 & a_{2,2} & & a_{2,n-1} & a_{2,n} & b_{2,2} & & b_{2,p} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{n-1,n-1} & a_{n-1,n} & b_{n-1,1} & \cdots & b_{n-1,m} \\ a_{n,1} & \cdots & a_{n,n-2} & a_{n,n-1} & a_{n,n} & b_{n,1} & \cdots & b_{n,p} \end{array}\right]$$

De implementatie moet gebeuren met maximaal $n-1$ Givens rotaties. Een Givens rotatie kan berekend worden met de standaard Matlab functie **givens.m**.

Test je algoritme (is q orthogonaal, r bovendriehoeks en $a = q * r$?). Vergelijk met het resultaat van de matlab functie **qr**. Merk dat dit niet noodzakelijk hetzelfde is, waaruit nogmaals blijkt dat de qr-factorisatie niet uniek is!

Extra oefening
Door je algoritme $n-1$ keer toe te passen kan je een qr-factorisatie van een willekeurige $n \times (n+p)$-matrix berekenen. Schrijf zulke routine. Nu heb je een alternatief voor het schema uit je cursus, met hetzelfde aantal Givens rotaties.

PC-zitting 3: stelsels

Opmerkingen:
De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Oplossingen van de oefeningen:

Oefening 1:

Mogelijke reeks commando's:
```
path(path, 'd:\user')
format long
a = [7 8 0 ; 1 2 3 ; 4 5 6]
[l,u] = lu(a)
inverse_a = inv(u) * inv(l)
det_a = det(u)
a2 = [1 4 3 ; 4 3 5 ; 9 8 0]
[l2, u2] = lu(a2)
```

Wat is l*u?
L*u geeft opnieuw de matrix a.
(LR-ontbinding zie hb.).

Merk op: afrondingsfouten geven kleine afwijkingen op het resultaat (vergelijk det(u) en det(a)).

Wat merk je bij de lu-ontbinding van de 2e matrix?
L is geen diagonaalmatrix; als de laatste rij boven de andere geplaatst wordt, dan verkrijgen we wel weer een diagonaalmatrix.
De oorzaak is pivotering tijdens de berekening van de lu-ontbinding.

Oefening 2:

Mogelijke reeks commando's:
```
A = genmat(6)
Tussen_opl1 = gauss1(A)
Tussen_opl2 = gauss2(A)
Opl1 = asubst(Tussen_opl1)
Opl2 = asubst(Tussen_opl2)
Echte_opl = fl(Opl1, 2)
% Opl1 en Opl2 hebben 2 cijfers voor de komma
Verschil1 = norm(Opl1 - Echte_opl)
Verschil2 = norm(Opl2 - Echte_opl)
```

Relfout1 = Verschil1/norm(Echte_opl)
Relfout2 = Verschil2/norm(Echte_opl)

Verklaring:
De relatieve fout van de 2e methode is veel kleiner (orde 10^{-16}) dan de relatieve fout voor de 1e methode (orde 10^{-6}) omdat de 2e methode optimale pivotering gebruikt.

Oefening 3:

Opmerkingen:
help cond
help norm

Mogelijke commando's:
M = genmatc(6)
A = M(:, 1:6)
cond(A)
tussenopl1 = gauss2(M)
opl1 = asubst(tussenopl1)
exacte_opl = abs(fl(opl1, 2))
verschil = norm(opl1 - exacte_opl)
rel_fout = verschil/norm(exacte_opl)

Verklaring fout:
De konditie is van grootte-orde 10^{11}, dus zeer slecht. Hierdoor is de fout groot.
A.h.v. onderstaande formules is dit te voorspellen:

$$||\Delta X||/||X|| \leq k(m)*||\Delta A||/||A||$$
$$||\Delta X||/||X|| \leq k(m)*||\Delta B||/||B||$$

waarbij k(m) het konditiegetal is ($||A||*||A^{-1}||$).

Als de konditie slecht is, dan zal een kleine fout op de gegevens (bv. afrondingsfout in het begin) een grote fout op het resultaat geven, ook al is het algoritme stabiel.

Oefening 4: demo Laplace

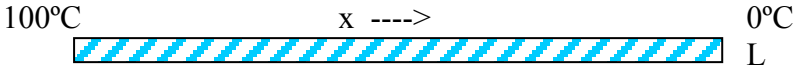
Opmerkingen:
Zie ook: http://www.cs.kuleuven.ac.be/~ade/WWW/NW/
> Illustraties > Convectie-diffusievergelijking).

Zie ook: http://www.cs.utah.edu/~zachary/isp/applets/Plotter/Plotter.html).
Fysisch probleem:

1) Transiente 1D warmtevergelijking.

Er is een staaf met een linkeruiteinde (LU) en een rechteruiteinde (RU).
t = de tijd en T = de temperatuur

.



Voor t ≤ 0: De hele staaf heeft de temperatuur van de omgeving, T_{omg}.
Vanaf t = 0: Aan LU wordt een warmtebron geplaatst (van 100 graden), en aan RU wordt een koudebron geplaatst (van 0 graden).
De warmte (en de koude) gaan zich verspreiden over de hele staaf, en dus de temperatuur tussen beide uiteinden doen wijzigen, totdat een evenwichtstoestand is bereikt.

2) Stationaire 2D warmtevergelijking:

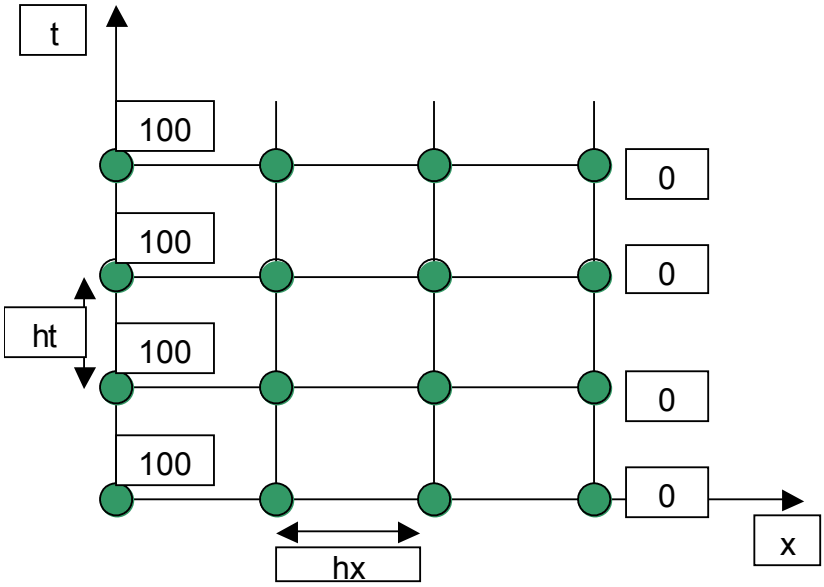
We kijken enkel naar t = ∞ voor de warmtetoestand van een plaat waarbij de randen op een constante temperatuur gehouden worden ($\partial T/\partial t = 0$).

Model voor het probleem:

1) Warmtevergelijking: $\rho * c * \partial T/\partial t = k * \nabla^2 T$,

waarbij:
- ρ de massadichtheid
- c de soortelijke warmte
- $\partial T/\partial t$ de verandering van de temperatuur in de tijd
- k de warmtegeleidingscoëfficiënt
- $\nabla^2 T$ de Laplaciaan van de temperatuur

Opm.: de temperatuur is een functie van de tijd en van de ruimte; daarom gebruiken we de partiële afgeleide in de warmtevergelijking.
De Laplaciaan $\nabla^2 T = \partial^2 T/\partial x^2$, de tweede afgeleide van de temperatuur naar de ruimte.
De oplossing van deze vergelijking kan berekend worden door zowel de ruimte als de tijd te discretiseren (d.w.z. indelen in stapjes).
Op die manier verkrijg je een aantal roosterpunten die berekend kunnen worden in een stelsel.



(∇^2T in het punt $T_i = (T_{i-1} - T_{i+1})/h_x^2$).
($\partial T/\partial x$ in het punt $T_{i-0,5} = (T_{i-1} - T_i)/h_x$)
($\partial T/\partial x$ in het punt $T_{i+0,5} = (T_i - T_{i+1})/h_x$)
(∂^2T/∂^2x in het punt T_i = de afgeleide nemen van de eerste afgeleide, dus de afgeleiden in i-0,5 en i+0,5 van elkaar aftrekken en delen door h_x, de stapgrootte in de ruimte).

Het rechterlid voor het stelsel kan bekomen worden door de punten aan de rand van de staaf in te vullen (van deze punten is de temperatuur gekend vanaf t = 0).

Vb. van een vgl. in het stelsel:
$\rho*c*((T_5 - T_2)/h_t) = k*((T_3 - T_1)/h_x^2)$

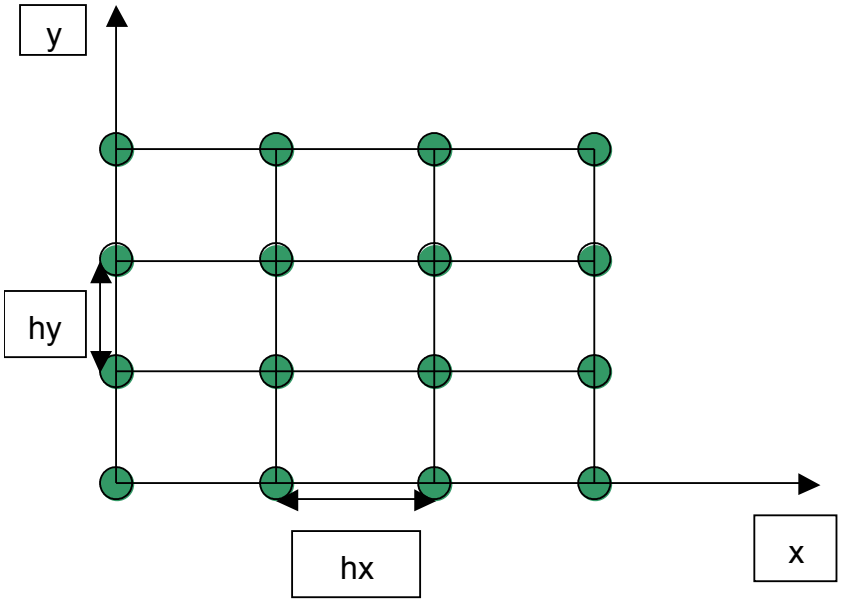
<u>2) Warmtevergelijking: $\nabla^2T = 0$.</u>

$$\nabla^2T = \partial^2T/\partial x^2 + \partial^2T\partial y^2$$

De oplossing van deze vergelijking kan berekend worden door zowel de ruimte in x als de ruimte in y discretiseren (d.w.z. indelen in stapjes). (Een plaat heeft een x- en een y-dimensie).

Op die manier verkrijg je opnieuw een aantal roosterpunten die berekend kunnen worden in een stelsel.

In een punt (i,j):
$((T_{i+1,j} + T_{i-1,j})/h_x^2) + ((T_{i,j+1} + T_{i,j-1})/h_y^2) = 0$



Voor een plaat zijn er meerdere modellen geïmplementeerd:

- 1kant van de plaat warm, de tegenoverliggende kant koud en ertussen lineaire overgangen.

- warmte aan de hoek rechtsonder, warmte aan de linkerbovenhoek en beide andere hoeken koud, met overgangen ertussen.

- Warme linkeronderhoek en de 3 andere hoeken koud.

Ook hier kunnen a.h.v. de randpunten alle andere punten berekend worden.
Resultaten:

1) De grafiek toont de evolutie in de tijd van de verspreiding van de warmte.
Van een grafiek met aan beide uiteinden een piek evolueert de warmtetoestand naar een lineaire grafiek (rechte).
Niet alle parameters mogen willekeurig gekozen worden om een grafiek te krijgen die in realiteit mogelijk is! Daarom laat het bestand demo.m enkel toe het aantal stappen in de tijd en de initiële temperatuur te kiezen.

2) De eerste stap is de gausseliminatie van het stelsel; typisch hiervoor is dat het aantal van 0 verschillende elementen gaat stijgen gedurende de eliminatiefase.
De structuur van het stelsel is typisch voor een differentiatieprobleem: een groot aantal onbekenden (orde n^m waarbij n het aantal discretisatiestappen voor x is en m het aantal discretisatiestappen voor y) (en dus veel vgln.), en een spaarse matrix.

Hoe bekomen:

1) demo.m gebruiken:
y = demo(aantal_tijdsstappen, init_temp)
Vb.: demo(5, 50)

2) gaussel.m gebruiken:
opl = gaussel(n, m, soortrandvwd, plot)

n = aantal stappen voor x
m = aantal stappen voor y
soortrandvwd = 1, 2 of 3 en bepaalt welke hoek(en)/kant warm is.
(1 = bovenkant warm, 2 = linkerbovenhoek en rechteronderhoek warm, 3 = linkeronderhoek warm).
plot = 1 of 0 (al dan niet tekenen van grafiekjes; kies dus steeds 1).
Vb.: opl = gaussel(10, 15,1,1)
Met contour i.p.v. plot kunnen de contourlijnen getekend worden (dit zijn de isotermen; deze verbinden punten met dezelfde temperatuur).

Wat onthouden:

Een eenvoudig model kan soms al een zeer goede benadering geven voor een vrij complex (fysisch) probleem.
Hierbij kunnen problemen optreden voor bepaalde keuzes van de parameters (vb. hieronder); daarom is niet elk model geschikt voor alle gevallen.

Voorbeeld van de staaf:
los_warmtevgl_op(hx, ht, lengte, maxtijd)
(Aantal stappen in x = hx*lengte)
(Aantal stappen in y = ht*maxtijd)

Goede benadering:
los_warmtevgl_op(0.2, 0.05, 10, 20)
los_warmtevgl_op(0.2, 0.05, 10, 30)
los_warmtevgl_op(0.2, 0.05, 15, 20)
los_warmtevgl_op(0.6, 0.05, 10, 20)
los_warmtevgl_op(0.2, 0.02, 10, 20)

Slechte benadering:
los_warmtevgl_op(0.2, 0.5, 10, 30)

Vooruitblik op iteratieve methoden:

Later zullen we ongeveer dezelfde demo geven als hierboven beschreven, maar dan iteratief.
Hiervoor gebruiken we run_laplace.m:
y = run_laplace(soortrandvwd, n, m, meth, preciesie, plot on)
waarbij:
soortrandvwd = 1, 2 of 3 (zie hoger)
n = aantal discretisatiestappen in x
m = aantal discretisatiestappen in y
meth = oplossingsmethode
(kies 1: Gauss-Seidel)
preciesie = criterium om te stoppen met
iteraties (verschil tussen 2 opeenvolgende
stappen)
plot on = 1 (teken grafiekjes)

Bij iteratieve berekeningen is het belangrijk een niet te hoge preciesie te kiezen (je kan in matlab altijd onderbreken met CTRL + C); anders stopt het algoritme misschien nooit.

De demo met iteratieve methode geeft 3 figuren:

de isotermen, het verdelingsprofiel van de warmte en de evolutie van de fout.

Het belangrijkste nadeel van Gauss voor dit probleem is dat de spaarsheid van de matrix niet benut wordt. Dit verbetert door iteratief te gaan werken.

Oefening 5:

Opmerkingen:

De functie "teken" houdt een pauze tussen de eerste en de 2e figuur; druk op de entertoets om verder te gaan.
Voor sommige gegevens krijg je een foutboodschap "division by zero".

Mogelijke commando's:

Voorbeeldje:

```
A = [1 2 ; 1 7]
B = [3 ; 4]
C0 = teken(A, B, 0)
C1 = teken(A, B, 1)
C2 = teken(A, B, 2)
C3 = teken(A, B, 3)
K = cond(A)
Determ = det(A)
Q = A + ones(size(A))
C0 = teken(Q, B, 0)
C1 = teken(Q, B, 1)
C2 = teken(Q, B, 2)
C3 = teken(Q, B, 3)
K = cond(Q)
Determ = det(Q)
```

a) Loodrechte rechten:

```
A = [1 1 ; -1 1]
B = [1 ; 1]
C0 = teken(A, B, 0)
C1 = teken(A, B, 1)
C2 = teken(A, B, 2)
C3 = teken(A, B, 3)
K = cond(A)
Determ = det(A)
```

konditie = 1.000 (goed)
determinant = 2

a2) Loodrechte rechten (beter vb.):
(Geen figuren op de site).

```
A = [(-7/10) -1 ; (-6/13) -1]
B = [-7 ; -6]
C0 = teken(A, B, 0)
C1 = teken(A, B, 1)
C2 = teken(A, B, 2)
C3 = teken(A, B, 3)
K = cond(A)
Determ = det(A)
```

Determinant = 0,2385
Konditie = 11,2463

b) Snijdende rechten die bijna samenvallen:
A = [-0,025 1 ; 0,025 1]
B = [4 ; 3]
C0 = teken(A, B, 0)
C1 = teken(A, B, 1)
C2 = teken(A, B, 2)
C3 = teken(A, B, 3)
K = cond(A)
Determ = det(A)
konditie = 40 (slecht)
determinant = -0,05

b2) Snijdende rechten die bijna samenvallen:
(beter vb.)
(Geen figuren op de site).
A = [1 0; 0 1]
B = [-3 ; -3]
C0 = teken(A, B, 0)
C1 = teken(A, B, 1)
C2 = teken(A, B, 2)
C3 = teken(A, B, 3)
K = cond(A)
Determ = det(A)
konditie = 1
determinant = 1

Verklaring figuren:

Als de coëfficiëntenmatrix slechter gekonditioneerd is, zal het gebiedje waarin de berekende oplossing zich bevindt groter zijn.
Er is dan immers mogelijk een grotere afwijking op het resultaat.
Echter, ook de positie van de rechten t.o.v. elkaar speelt een rol. Rechten die bijna samenvallen hebben een grote mogelijke afwijking in de richting waarin ze lopen en een kleine mogelijke afwijking in de andere richting (+/- parallellogram met lange en korte kant).
Rechten die loodrecht op elkaar staan, hebben in beide richtingen een even grote mogelijke afwijking (gebiedje = vierkant).

Oefeningen Numerieke Wiskunde

Oefenzitting 6: Veeltermінterpolatie

1 Oefeningen

1. Bewijs dat

$$\Pi'(x_i) = \prod_{j=0, j\neq i}^{n} (x_i - x_j),$$

zodat de formule

$$l_i(x) = \frac{\Pi(x)}{\Pi'(x_i)(x - x_i)}$$

geldt.

2. Bewijs

(a)

$$\sum_{i=0}^{n} l_i(x) = 1, \forall x.$$

(b)

$$\sum_{i=0}^{n} l_i(x) {x_i}^k = x^k, k \leq n.$$

3. Bewijs

$$y_n(x) = y_{n-1}(x) + (f_n - y_{n-1}(x_n)) l_n(x).$$

4. Praktische oefening:
Benader de 3^e graadsveelterm

$$f(x) = x^3 + 2x^2 + 3x - 5$$

door een tweedegraadsveelterm op $[-1, 1]$. Maak ook een afschatting voor $E_n(x)$ in dit interval (n=2,3) en maak een grafiek van f, y_2, E_2.
(Hint: Kies als interpolatiepunten -1, 0, 1.)

5. Hieronder staat een tabel met gedeelde differenties van een bepaalde zachtverlopende $f(x_i)$. Eén van de functiewaarden is echter verkeerd. Welke functiewaarde? Corrigeer indien mogelijk.

x_i	f_i					
0	0.3183	0	0	0	0	...
0.100	0.3518	0.3348	0	0	0	
0.130	0.3625	0.3571	0.1719	0	0	
0.150	0.3698	0.3661	0.1807	0.0584	0	
0.160	0.3735	0.3717	0.1843	0.0607	0.0148	
0.165	0.3754	0.3745	0.1865	0.0617	0.0153	...
0.170	0.3773	0.3764	0.1877	0.0623	0.0155	
0.180	0.3811	0.3793	0.1956	0.3961	11.1268	
0.190	0.3849	0.3829	0.1805	-0.6034	-33.3176	
0.200	0.3888	0.3868	0.1975	0.5638	33.3492	
0.220	0.3966	0.3927	0.1950	-0.0604	-12.4839	...

2 Extra oefeningen

1. Iterpolatie volgens Neville.

   (a) Bewijs dat
   $$y_{012} = \frac{x - x_2}{x_0 - x_2} y_{01}(x) + \frac{x - x_0}{x_2 - x_0} y_{12}(x).$$
   de interpolerende veelterm is in de punten $\{(x_i, f_i)\}_{i=0}^{2}$.

   (b) Bewijs de algemene formule:
   $$y_{i,\ldots,j}(x) = \frac{x - x_i}{x_j - x_i} y_{i+1,\ldots,j}(x) + \frac{x - x_j}{x_i - x_j} y_{i,\ldots,j-1}(x).$$

   (c) Bereken het aantal bewerkingen in het algoritme van Neville.

   (d) Hoe zou je het aantal bewerkingen kunnen verminderen en hoeveel bewerkingen moet je dan minder doen?

   (Hint: Ga eens na welke bewerkingen meer dan één keer worden gedaan en hoe je dit kan vermijden.)

Oefenzitting 3: Veelterminterpolatie

Herhaling:

Algemeen:

$f(x) = y_n(x) + E_n(x)$
met $y_n(x)$ de interpolerende veelterm van graad n en $E_n(x)$ de gemaakte interpolatiefout.
$E_n(x) = [f^{(n+1)}(\xi)/(n+1)!](x-x_0)(x-x_1)...(x-x_n)$
$|E_n(x)| \leq [|(x-x_0)(x-x_1)...(x-x_n)|/(n+1)!] * \max_{x\in[a,b]} |f^{(n+1)}(x)|$
waarbij [a,b] het interval is waarvoor $y_n(x)$ interpoleert.

Lagrange-interpolatie:

$y_n(x) = l_0(x)*f_0 + l_1(x)*f_1 + ... + l_n(x)*f_n$
$y_n(x_i) = f_i$
$l_i(x) = \Pi(x)/[\Pi'(x_i)(x-x_i)]$
$= \frac{[(x-x_0)(x-x_1)...(x-x_n)]}{[(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)(x-x_i)]}$.

Oplossingen van de oefeningen:

Oefening 1:

Hint: Ontwikkel naar de laatste kolom.
Ga na of de graad van de veelterm ok is.
Ga na of $y_n(x_i) = f_i$ voor elke i.
=> uit uniciteit van de interpolerende (IP) veelterm zal de gevonden uitdrukking de IP-veelterm zijn.

Notatie:

In het vervolg van de oplossing staat $A_{-i,-j}$ voor de matrix uit de opgave zonder rij i en zonder kolom j.

$A_{1,(n+2)*1,(n+2)}$ staat voor de volledige matrix uit de opgave (met n+2 rijen en kolommen).
Regel van Sarrus voor de determinant van een 3*3-matrix A = [a b c ; d e f ; g h i]:
det(A) = aei + bfg + cdh - afh - bdi – ceg

Grafisch:
Zet de matrix 2 keer naast elkaar, trek eerst 3 lijnen schuin naar rechts naar beneden (aei, bfg, cdh) en dan 3 lijnen schuin naar links naar beneden (afh, bdi, ceg).
De lijnen naar rechts zijn positieve factoren in de determinant, de andere lijnen zijn negatieve factoren in de determinant.

Uitwerken van een determinant in het algemeen:
Kies een rij of kolom i.
Neem achtereenvolgens alle elementen uit die rij of kolom met afwisselend een plus- of minteken.
Vermenigvuldig elk element (met het teken) met de determinant van de matrix die je bekomt door 1 rij

en 1 kolom te schrappen, nl. de rij en de kolom
waarin het gekozen element stond.

Naar laatste kolom:

$$\begin{aligned}\text{Det}(A_{1,(n+2)*1,(n+2)}) = {} & -1^{(1+(n+2))} * y_n(x) * \det(A_{-1,-(n+2)}) + \\ & -1^{(2+(n+2))} * f_0 * \det(A_{-2,-(n+2)}) + \\ & -1^{(3+(n+2))} * f_1 * \det(A_{-3,-(n+2)}) + \\ & \ldots + \\ & -1^{((i+2)+(n+2))} * f_i * \det(A_{-(i+2),-(n+2)}) + \\ & \ldots + \\ & -1^{((n+2)+(n+2))} * f_n * \det(A_{-(n+2),-(n+2)})\end{aligned}$$

De determinanten in de som zijn Vandermonde-determinanten (zie hb. p. 86-87).

$$\begin{aligned}\text{Det}(A_{1,(n+2)*1,(n+2)}) = {} & -1^{(1+(n+2))} * y_n(x) * (x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1}) + \\ & -1^{(2+(n+2))} * f_0 * (x_1-x)(x_2-x)(x_2-x_1)\ldots(x_n-x)(x_n-x_1)\ldots(x_n-x_{n-1}) + \\ & -1^{(3+(n+2))} * f_1 * (x_0-x)(x_2-x)(x_2-x_0)\ldots(x_n-x)(x_n-x_0)\ldots(x_n-x_{n-1}) + \\ & \ldots + \\ & -1^{((i+2)+(n+2))} * f_i * (x_0-x)\ldots(x_{i-1}-x_0) \ldots (x_{i-1}-x_{i-2})(x_{i+1}-x_0) \ldots \\ & \qquad \ldots (x_{i+1}-x_{i-1})\ldots(x_n-x)\ldots(x_n-x_{n-1}) + \\ & \ldots + \\ & -1^{((n+2)+(n+2))} * f_n * (x_0-x)\ldots(x_{n-1}-x_0)\ldots(x_{n-1}-x_{n-2})\end{aligned}$$

$$\begin{aligned}\text{Det}(A_{1,(n+2)*1,(n+2)}) = 0 \quad \Rightarrow \\ & -1^{(1+(n+2))} * y_n(x)*(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1}) + \\ & -1^{(2+(n+2))} * f_0 * (x_1-x)(x_2-x)(x_2-x_1)\ldots(x_n-x)(x_n-x_1)\ldots(x_n-x_{n-1}) + \\ & -1^{(3+(n+2))} * f_1 * (x_0-x)(x_2-x)(x_2-x_0)\ldots(x_n-x)(x_n-x_0)\ldots(x_n-x_{n-1}) + \ldots + \\ & -1^{((i+2)+(n+2))} * f_i * (x_0-x)\ldots(x_{i-1}-x_0)\ldots(x_{i-1}-x_{i-2})(x_{i+1}-x_0)\ldots (x_{i+1}-x_{i-1})\ldots(x_n-x)\ldots(x_n-x_{n-1}) + \\ & \ldots + -1^{((n+2)+(n+2))} * f_n * (x_0-x)\ldots(x_{n-1}-x_0)\ldots(x_{n-1}-x_{n-2}) = 0\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \quad & y_n(x)*(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1}) = \\ & f_0 * (x_1-x)(x_2-x)(x_2-x_1)\ldots(x_n-x)(x_n-x_1)\ldots(x_n-x_{n-1}) - \\ & f_1*(x_0-x)(x_2-x)(x_2-x_0)\ldots(x_n-x)(x_n-x_0)\ldots(x_n-x_{n-1}) + \ldots + \\ & -1^{((i+1))}*f_i *(x_0-x)\ldots(x_{i-1}-x_0)\ldots(x_{i-1}-x_{i-2})(x_{i+1}-x_0)\ldots (x_{i+1}-x_{i-1})\ldots(x_n-x)\ldots(x_n-x_{n-1}) + \\ & \ldots + -1^{(n+1)} * f_n *(x_0-x)\ldots(x_{n-1}-x_0)\ldots(x_{n-1}-x_{n-2})\end{aligned}$$

$$\begin{aligned}\Leftrightarrow \quad y_n(x) = {} & f_0 * \frac{(x_1-x)(x_2-x)(x_2-x_1)\ldots(x_n-x)(x_n-x_1)\ldots(x_n-x_{n-1})}{(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1})} \\ & -f_1*\frac{(x_0-x)(x_2-x)(x_2-x_0)\ldots(x_n-x)(x_n-x_0)\ldots(x_n-x_{n-1})}{(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1})} \\ & + \ldots \\ & + (-1)^{((i+1))}*f_i * \frac{(x_0-x)\ldots(x_n-x_{n-1})}{(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1})}. \\ & + \ldots \\ & + -1^{(n+1)} * f_n * \frac{(x_0-x)\ldots(x_{n-1}-x_0)\ldots(x_{n-1}-x_{n-2})}{(x_1-x_0)(x_2-x_0)(x_2-x_0)\ldots(x_n-x_0)\ldots(x_n-x_{n-1})}. \\ & = f_0 * l_0(x) + f_1 * l_1(x) + \ldots + f_i * l_i(x) + \ldots + f_n * l_n(x)\end{aligned}$$

Invullen van x_i:
Behalve bij $y_n(x)$ en bij f_i zijn alle determinanten gelijk aan 0.

$$\begin{aligned} \det(A_{-(i+2),-(n+2)}) &= -1^{i} * \det(A_{-1,-(n+2)}) \\ & -1^{(1+(n+2))} * y_n(x_i) * \det(A_{-1,-(n+2)}) = -\left(-1^{((i+2)+(n+2))} * f_i * \det(A_{-(i+2),-(n+2)})\right) \\ &= -1^{(i+2)} * f_i * \det(A_{-(i+2),-(n+2)}) \end{aligned}$$

$$y_n(x_i) * \underline{\det(A_{-1,-(n+2)})} = -1^{(i+2)} * f_i * -1^{i} * \underline{\det(A_{-1,-(n+2)})}$$

$$\begin{aligned} y_n(x_i) &= -1^{(2i+2)} * f_i \\ &= f_i \end{aligned}$$

y_n interpoleert in de x_i, dus is dit de interpolerende veelterm.

Voorbeeld: $n = 2, i = 1$.

$$\begin{aligned} \det(A_{1,4*1,4}) = {} & y_3(x)*\det(A_{-1,-4}) \\ & - f_0*\det(A_{-2,-4}) \\ & + f_1*\det(A_{-3,-4}) \\ & - f_2*\det(A_{-4,-4}) \end{aligned}$$

$$\begin{aligned} = {} & y_3(x)*(1x_1x_2^2 + x_0x_1^2 1 + x_0 1^2 x_2 \\ & \quad - 1x_1^2x_2 - x_0 1x_2^2 - x_1x_0^2 1) \\ & - f_0*(1x_1x_2^2 + xx_1^2 1 + x^2 1x_2 \\ & \quad - 1x_1x^2 - x_2x_1^2 1 - x_2^2 1x) \\ & + f_1*(1x_0x_2^2 + xx_0^2 1 + x^2 1x_2 \\ & \quad - 1x_0x_1^2 - xx_0^2 1 - x_2^2 1x) \\ & - f_2*(1x_0x_1^2 + xx_0^2 1 + x^2 1x_1 \\ & \quad - 1x_0x^2 - x_1x_0^2 1 - x_1^2 1x) \end{aligned}$$

Invullen van x_1:

$(\det(A_{1,4*1,4}) = 0)$

$$\begin{aligned} & y_3(x)*(x_1x_2^2 + x_0x_1^2 + x_0^2x_2 - x_1^2x_2 - x_0x_2^2 - x_1x_0^2) \\ & - f_0*(x_1x_2^2 + x_1x_1^2 + x_1^2x_2 - x_1x_1^2 - x_2x_1^2 - x_2^2x_1) \\ & + f_1*(x_0x_2^2 + x_1x_0^2 + x_1^2x_2 - x_0x_1^2 - x_1x_0^2 - x_2^2x_1) \\ & - f_2*(x_0x_1^2 + x_1x_0^2 + x_1^2x_1 - x_0x_1^2 - x_1x_0^2 - x_1^2x_1) \qquad = 0 \end{aligned}$$

Dus:

$$\begin{aligned} & y_3(x)*(x_1x_2^2 + x_0x_1^2 + x_0^2x_2 - x_1^2x_2 - x_0x_2^2 - x_1x_0^2) \\ & - f_0*0 + f_1*(x_0x_2^2 + x_1x_0^2 + x_1^2x_2 - x_0x_1^2 - x_1x_0^2 - x_2^2x_1) \\ & - f_2*0 \qquad = 0 \end{aligned}$$

Dus:

$$\begin{aligned} & y_3(x)*(x_1x_2^2 + x_0x_1^2 + x_0^2x_2 - x_1^2x_2 - x_0x_2^2 - x_1x_0^2) \\ & + f_1*(x_0x_2^2 + x_1x_0^2 + x_1^2x_2 - x_0x_1^2 - x_1x_0^2 - x_2^2x_1) \\ & \qquad = 0 \end{aligned}$$

Dus:

$$\begin{aligned} & y_3(x)*(x_1x_2^2 + x_0x_1^2 + x_0^2x_2 - x_1^2x_2 - x_0x_2^2 - x_1x_0^2) \\ = {} & -f_1*(x_0x_2^2 + x_1x_0^2 + x_1^2x_2 - x_0x_1^2 - x_1x_0^2 - x_2^2x_1) \end{aligned}$$

Oefening 2:

a) $\Pi(x) = \Pi_{k=0}^{n} (x - x_k)$
$= (x - x_0)(x - x_1) \ldots (x - x_n)$

b) $\Pi'(x) =$
$(x - x_1)(x - x_2) \ldots (x - x_n) +$
$(x - x_0)(x - x_2) \ldots (x - x_n) +$
$(x - x_0)(x - x_1)(x - x_3) \ldots (x - x_n) +$
$\ldots +$
$(x - x_0) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_n) +$
$\ldots +$
$(x - x_0) \ldots (x - x_{n-1})$
$= \Sigma_{j=0}^{n} \Pi_{k=0,\, k\neq j}^{n} (x - x_k)$

c) $\Pi'(x_i) = \Sigma_{j=0}^{n} \Pi_{k=0,\, k\neq j}^{n} (x_i - x_k)$
Als $j \neq i$: als $k=i$, dan $(x_i - x_k) = 0$
Als $j = i$: $(x_i - x_k) \neq 0$ want $k \neq j$
dus $k \neq i$
Dus: $\Pi'(x_i) = \Pi_{k=0,\, k\neq i}^{n} (x_i - x_k)$, q.e.d.

Verder uitwerken voor d):
$\Pi'(x_i) = (x_i - x_0) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)$

d) $p(x) = p_0(x)f_0 + p_1(x)f_1 + \ldots + p_n(x)f_n$
$p(x_j) = f_j$? Indien ja: $p(x)$ is de IP veelterm.
(uniciteit).
$p_i(x)$ is dan ook $l_i(x)$
Schrijf $p_i(x)$ als $\Pi(x)/[\Pi'(x_i)(x - x_i)]$:
$p(x) =$
$\Pi(x)/[\Pi'(x_0)(x - x_0)] * f_0 +$
$\Pi(x)/[\Pi'(x_1)(x - x_1)] * f_1 +$
$\ldots$
$\Pi(x)/[\Pi'(x_i)(x - x_i)] * f_i +$
$\ldots$
$\Pi(x)/[\Pi'(x_n)(x - x_n)] * f_n +$

Dus:
$p(x) =$
$\dfrac{[(x - x_0)(x - x_1) \ldots (x - x_n)]}{[(x_0 - x_1) \ldots (x_0 - x_n)(x - x_0)]} * f_0 +$

$\dfrac{[(x - x_0)(x - x_1) \ldots (x - x_n)]}{[(x_1 - x_0)(x_1 - x_2) \ldots (x_0 - x_n)(x - x_1)]} * f_1 +$

$\ldots$

$\dfrac{[(x - x_0)(x - x_1) \ldots (x - x_n)]}{[(x_i - x_0) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)(x - x_i)]} * f_i +$

$\ldots$

$\dfrac{[(x - x_0)(x - x_1) \ldots (x - x_n)]}{[(x_n - x_0) \ldots (x_n - x_{n-1})(x - x_n)]} * f_i +$

$p_i(x_j) = 0$ voor $i \neq j$:

$$\frac{[(x_j - x_0)(x_j - x_1) \ldots (x_j - x_n)]}{[(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)(x_j-x_i)]} * f_i$$

Bevat in de teller $(x_j - x_j) = 0$.
Bevat in teller en noemer de factor $(x_i - x_j)$ die geschrapt mag worden.
Bevat verder in teller en noemer geen gelijke factoren en geen factor = 0
Is dus gelijk aan 0.

$p_i(x_j) = f_i$ voor $i = j$, dus $p_j(x_j) = f_j$:

$$\frac{[(x_j - x_0)(x_j - x_1) \ldots (x_j - x_n)]}{[(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)(x_j-x_i)]} * f_i$$

$$= \frac{[(x_j - x_0)(x_j - x_1) \ldots (x_j - x_n)]}{[(x_j-x_0)\ldots(x_j-x_{j-1})(x_i-x_{j+1})\ldots(x_j-x_n)(x_j-x_j)]} * f_j$$

bevat in teller en noemer $(x_j - x_k)$ voor $k = 0..n$
Is dus gelijk aan $1*f_j = f_j$

$p(x)$ interpoleert dus in de punten x_i, dus is $p(x)$ de interpolerende veelterm en $p_i(x) = l_i(x)$, q.e.d.

Oefening 2:

a) T.B.: $\forall x: \Sigma_{i=0}^{n} l_i(x) = 1$

Bewijs: voor $x = x_j$:

$$\Sigma_{i=0}^{n} l_i(x) = p(x)/f_j$$
$$= f_j/f_j$$
$$= 1 \qquad \text{q.e.d.}$$

voor $x \neq x_j$: kies $f(x) = 1$.

$$E_n(x) = f(x) - y_n(x)$$
$$= \frac{f^{(n+1)}(\xi)}{(n+1)!} * \Pi(x) = 0$$

($f^{(n+1)}(\xi) = 0$, want $f(x)$ is een constante functie).

Dus: $\Sigma_{i=0}^{n} l_i(x) = \Sigma_{i=0}^{n} 1*l_i(x)$
$= 1$ q.e.d.

(De Lagrangeveelterm interpoleert exact voor $f(x)$ een veeltermfunctie met graad $\leq n$).

b) T.B.: $\forall x: \Sigma_{i=0}^{n}[l_i(x)x_i^k] = x^k$, $k \leq n$

Bewijs: voor $x = x_j$:

$$\Sigma_{i=0}^{n} l_i(x) = p(x)/f_j * x_j^k$$
$$= f_j/f_j * x_j^k$$
$$= x_j^k$$
$$= x^k \qquad \text{q.e.d.}$$

voor $x \neq x_j$: kies $f(x) = x^k$

$$E_n(x) = f(x) - y_n(x)$$
$$= \frac{f^{(n+1)}(\xi)}{(n+1)!} * \Pi(x) = 0$$

($f^{(n+1)}(\xi) = 0$, want $f(x)$ is een veeltermfunctie met graad $\leq n$; vb.: x^3 afleiden: $3x^2$, afleiden: $6x$, afleiden: 6, afleiden: 0).

Dus: $\Sigma_{i=0}^{n} [x^k * l_i(x)] = x^k$

q.e.d.

Oefening 3:

T.B.: $y_n(x) = y_{n-1}(x) + (f_n - y_{n-1}(x_n))l_n(x)$

Bewijs:

1) $y_n(x)$ is een veelterm?
   - $y_{n-1}(x)$ is een veelterm
   - $l_n(x)$ is een veelterm
   - $(f_n - y_{n-1}(x_n))$ is een constante
   - Dus: $y_n(x)$ is een veelterm.

2) $y_n(x)$ heeft graad n?
   - $l_n(x)$ heeft graad n
   - $(f_n - y_{n-1}(x_n))$ heeft graad 0 (constante)
   - $y_{n-1}(x)$ heeft graad (n-1)
   - Dus: de graad van $y_n(x)$ is n.

3) $y_n(x)$ interpoleert in de interpolatiepunten?
   - Voor x_i in $[x_0, x_{(n-1)}]$:
     $y_n(x_i) = f_i + (f_n - y_{n-1}(x_n))l_n(x)$
     $l_n(x) = 0$
     $y_n(x_i) = f_i$
   - Voor x_n:
     $y_n(x_i) = y_{n-1}(x_n) + (f_n - y_{n-1}(x_n))l_n(x)$
     $l_n(x) = 1$
     $y_n(x_i) = y_{n-1}(x_n) + f_n - y_{n-1}(x_n)$
     $= f_n$
   - Dus: $y_n(x)$ interpoleert in de interpolatiepunten.

4) De interpolerende veelterm is uniek, dus $y_n(x)$ is **de** interpolerende veelterm van graad n voor f.

Oefening 5:

1) (x_i, f_i) berekenen voor i = 0..2:
   Kies $x_0 = -1$, $x_1 = 0$, $x_2 = 1$.
   $f_0 = f(x_0) = f(-1) = -1 + 2 - 3 - 5 = -7$
   $f_1 = f(x_1) = f(0) = 0 + 0 + 0 - 5 = -5$
   $f_2 = f(x_2) = f(1) = 1 + 2 + 3 - 5 = 1$

2) n = 2 => $y_2(x)$ zoeken:
   $y_2(x) = l_0(x)*f_0 + l_1(x)*f_1 + l_2(x)*f_2$
   $l_0(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)]}{[(x_0-x_1)(x_0-x_2)(x-x_0)]}$

   $= \frac{[(x-x_1)(x-x_2)]}{[(x_0-x_1)(x_0-x_2)]}$

   $= \frac{[(x-0)(x-1)]}{[(-1-0)(-1-1)]}$

   $= \frac{x(x-1)}{2}$

$$-1(-2)$$

$$= [x(x-1)]/2$$

$$l_1(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)]}{[(x_1-x_0)(x_1-x_2)(x-x_1)]}$$

$$= \frac{[(x-x_0)(x-x_2)]}{[(x_1-x_0)(x_1-x_2)]}$$

$$= \frac{[(x-(-1))(x-1)]}{[(0-(-1))(0-1)]}$$

$$= \frac{(x+1)(x-1)}{1(-1)}$$

$$= -[(x+1)(x-1)]$$
$$= -(x^2 - 1)$$
$$= 1 - x^2$$

$$l_2(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)]}{[(x_2-x_0)(x_2-x_1)(x-x_2)]}$$

$$= \frac{[(x-x_0)(x-x_1)]}{[(x_2-x_0)(x_2-x_1)]}$$

$$= \frac{[(x-(-1))(x-0)]}{[(1-(-1))(1-0)]}$$

$$= \frac{(x+1)x}{2*1}$$

$$= [(x+1)x]/2$$

$$\begin{aligned}
y_2(x) &= ([x(x-1)]/2)*f_0 \\
&\quad + (1 - x^2)*f_1 \\
&\quad + ([(x+1)x]/2)*f_2 \\
&= ([x(x-1)]/2)*(-7) \\
&\quad + (1 - x^2)*(-5) \\
&\quad + ([(x+1)x]/2)*1 \\
&= ([-7x(x-1)]/2) \\
&\quad + (5x^2 - 5) \\
&\quad + ([(x+1)x]/2) \\
&= ([-7x(x-1)]/2) \\
&\quad + [10*(x^2 - 1)]/2 \\
&\quad + ([(x+1)x]/2) \\
&= [-7x^2 + 7x + 10x^2 - 10 + x^2 + x]/2 \\
&= [4x^2 + 8x - 10]/2 \\
&= 2x^2 + 4x - 5
\end{aligned}$$

Alternatieve oplossing om $y_2(x)$ te zoeken:

$y_2(x)$ is van de vorm $ax^2 + bx + c$

$f(0) = -5 \Rightarrow c = -5$

$(a*0^2 + b*0 + c = -5)$

Stelsel opstellen:

a) $a*(-1)^2 + b*(-1) -5 = -7$

$\Rightarrow a - b -5 = -7$

$\Rightarrow a - b = -2$

b) $a*(1)^2 + b*(1) -5 = 1$

$\Rightarrow a + b -5 = 1$

$\Rightarrow a + b = 6$

Uit a): $a = b - 2$

Vul a in in de 2e vgl b):

$b -2 + b = 6$

$\Rightarrow 2b -2 = 6$

$\Rightarrow 2b = 8$

$\Rightarrow b = 4$

Vul nu b in in de 1e vgl a):

$a - 4 = -2$

$\Rightarrow a = 2$

$y_2(x)$ is dus $2x^2 + 4x -5$

3) $E_2(x)$ berekenen.

$E_2(x) = f(x) - y_2(x)$

$= (x^3 + 2x^2 + 3x - 5) - (2x^2 + 4x - 5)$

$= x^3 - x$

$= x*(x^2 - 1)$

$\leq |x| * | x^2 - 1|$

Voor $x \in [-1, 1]$ is $E_2(x) \leq 2/(3\sqrt{3})$

4) $E_3(x)$ berekenen.

$y_3(x) = l_0(x)*f_0 + l_1(x)*f_1 + l_2(x)*f_2 + l_3(x)*f_3$

Kies $x_0 = -1$, $x_1 = -1/3$, $x_2 = 1/3$, $x_3 = 1$

$f_0 = f(x_0) = f(-1) = -1 + 2 -3 -5 = -7$

$f_1 = f(x_1) = f(-1/3) = -1/27 + 2/9 -1 -5$

$= -1/27 + 6/27 -27/27 - 135/27$

$= -157/27$

$f_2 = f(x_2) = f(1/3) = 1/27 + 2/9 + 1 - 5$

$= 1/27 + 6/27 + 27/27 - 135/27$

$= -101/27$

$f_3 = f(x_3) = f(1) = 1 + 2 + 3 -5 = 1$

$l_0(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)(x-x_3)]}{[(x_0-x_1)(x_0-x_2)(x_0-x_3)(x- x_0)]}$

$= \frac{[(x-x_1)(x-x_2)(x-x_3)]}{[(x_0-x_1)(x_0-x_2)(x_0-x_3)]}$

$= \frac{[(x+1/3)(x-1/3)(x-1)]}{[(-1+1/3)(-1-1/3)(-1-1)]}$

$= \frac{(x^3-x^2-1/9x + 1/9)}{-16/9}$

$= -\frac{9*(x^3-x^2-1/9x + 1/9)}{16}$

$= -\frac{9x^3 + 9x^2 + x - 1}{16}$

$l_1(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)(x-x_3)]}{[(x_1-x_0)(x_1-x_2)(x_1-x_3)(x- x_1)]}$

$= \frac{[(x-x_0)(x-x_2)(x-x_3)]}{[(x_1-x_0)(x_1-x_2)(x_1-x_3)]}$

$= \frac{[(x+1)(x-1/3)(x-1)]}{[(-1/3 + 1)(-1/3 - 1/3)(-1/3 - 1)]}$

$= \frac{(x^3 - 1/3x^2 -x + 1/3)}{}$

$$= \frac{9*3*(x^3 - 1/3x^2 - x + 1/3)}{16}.$$

$$= \frac{(27x^3 - 9x^2 - 27x + 9)}{16}.$$

$$l_2(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)(x-x_3)]}{[(x_2-x_0)(x_2-x_1)(x_2-x_3)(x-x_2)]}.$$

$$= \frac{[(x-x_0)(x-x_1)(x-x_3)]}{[(x_2-x_0)(x_2-x_1)(x_2-x_3)]}.$$

$$= \frac{[(x+1)(x+1/3)(x-1)]}{[(1/3+1)(1/3+1/3)(1/3-1)]}.$$

$$= \frac{(x^3 + 1/3x^2 - x - 1/3)}{-16/27}.$$

$$= \frac{-9*3*(x^3 + 1/3x^2 - x - 1/3)}{16}.$$

$$= \frac{-27x^3 - 9x^2 + 27x + 9}{16}.$$

$$l_3(x) = \frac{[(x-x_0)(x-x_1)(x-x_2)(x-x_3)]}{[(x_3-x_0)(x_3-x_1)(x_3-x_2)(x-x_3)]}.$$

$$= \frac{[(x-x_0)(x-x_1)(x-x_2)]}{[(x_3-x_0)(x_3-x_1)(x_3-x_2)]}.$$

$$= \frac{[(x+1)(x+1/3)(x-1/3)]}{[(1+1)(1+1/3)(1-1/3)]}.$$

$$= \frac{(x^3 + x^2 - 1/9x - 1/9)}{16/9}.$$

$$= \frac{9*(x^3 + x^2 - 1/9x - 1/9)}{16}.$$

$$= \frac{9x^3 + 9x^2 - x - 1}{16}$$

$$y_3(x) = [(-9x^3+9x^2+x-1)/16]*(-7) + [(27x^3 - 9x^2 - 27x + 9)/16]*(-157/27) + [(-27x^3 - 9x^2 + 27x + 9)/16]*(-101/27) + [(9x^3 + 9x^2 - x - 1)/16]*1$$

$$= 63/16x^3 - 63/16x^2 - 7/16x + 7/16 - 157/16x^3 + 157/48x^2 + 157/16x - 157/48 + 101/16x^3 + 101/48x^2 - 101/16x - 101/48 + 9/16x^3 + 9/16x^2 - 1/16x - 1/16$$

$$= 16/16x^3 - 189/48x^2 + 157/48x^2 + 101/48x^2 + 27/48x^2 + 48/16x + 21/48 - 157/48 - 101/48 - 3/48$$

$$= x^3 + 96/48x^2 + 3x - 240/48$$

$$= x^3 + 2x^2 + 3x - 5$$

$$= f(x)$$

$$E_3(x) = f(x) - y_3(x) = 0$$

Reden: $f(x) = y_3(x)$

5) Grafieken: $f(x)$, $y_2(x)$, E_2

$f(x)$: nulpunten:

$x \approx 0{,}894558$

afgeleide:

$3x^2 + 4x + 3$

nulpunten afgeleide:

$3x^2 + 4x + 3 = 0$

$D = 16-36 < 0$

Geen reële nulpunten.

asymptoten: /

Schets f(x) a.h.v. een aantal functiewaarden (vnl. [-1,1] belangrijk).

$y_2(x)$: nulpunten:

$2x^2 + 4x - 5 = 0$

$D = 16 + 40 = 56$

$\Rightarrow x_{1,2} = (-4 \pm \sqrt{56})/4$

$= -1 \pm (\sqrt{2}\sqrt{7})/2$

afgeleide: $4x + 4$

nulpunten afgeleide: $x = -1$

(minimumvan f, want 2e afgeleide is positief)

vorm: parabool

Schets $y_2(x)$ a.h.v. een aantal functiewaarden.

E_2: $x*(x^2 - 1)$

nulpunten:

$x = 0$

$x = 1$

$x = -1$

afgeleide:

$3x^2 - 1$

nulpunten afgeleide:

$x = 1/\sqrt{3} \approx 0{,}57735$ (minimum)

$x = -1/\sqrt{3} \approx -0{,}57735$ (maximum)

Schets $y_2(x)$ a.h.v. een aantal functiewaarden.

Grafieken:

<u>Oefening 6:</u>

<u>Algoritme van Neville:</u>

Opmerkingen:

Dit algoritme is geschikt voor het waardenprobleem;
we zoeken dus de functiewaarde voor een bepaalde x, gegeven een aantal interpolatiepunten;
wanneer we $y_{i,\ldots,j}$ willen berekenen:

$y_{i+1,\ldots,j}(x)$ is een berekende constante geworden. $y_{i,\ldots,j-1}(x)$ is een berekende constante geworden.

$y_{i,\ldots,j}(x)$ is een eerstegraadsveeltermfunctie geworden op het moment van evaluatie!

Er wordt een tabel opgesteld; deze tabel wordt diagonaal per diagonaal berekend (vgl. algo Newton).
Om geheugen te sparen kan de tabel opgeslagen worden in 1 kolom (de geheugenplaatsen van de f_i). (Indices in het algoritme aanpassen).
Het aantal bewerkingen zoeken:

- zoek bovengrens voor het aantal optellingen (+ of -)
- zoek bovengrens voor het aantal vermenigvuldigingen (* of /)
- De toekenning is GEEN bewerking; dit is een operatie.

Formule:

$$y_{i,\dots,j}(x) = \frac{x - x_i}{x_j - x_i} y_{i+1,\dots,j}(x) + \frac{x - x_j}{x_i - x_j} y_{i,\dots,j-1}(x)$$

Algoritme:

voor i = 1(1)(n+1)
  voor j = 1(1)i
    $y_{i,\dots,j}(x) = \frac{x - x_i}{x_j - x_i} y_{i+1,\dots,j}(x) + \frac{x - x_j}{x_i - x_j} y_{i,\dots,j-1}(x)$

a) T.B.: y_{012} is de interpolerende veelterm door de punten (x_i, f_i) voor i = 0, 1 en 2.

Bewijs:

1) y_{012} is een veelterm van graad 2?

Ja: de coëfficiënten van y_{01} en y_{12} zijn een veelterm van graad 1 (teller graad 1, noemer is een constante want interpolatiepunten zijn vast gekozen); y_{01} en y_{12} zijn veeltermen van graad 1. Dus: y_{012} is een veelterm van graad 2.

2) y_{012} interpoleert in de interpolatiepunten?

$$y_{012}(x_0) = \frac{x_0 - x_0}{x_2 - x_0} y_{12}(x_0) + \frac{x_0 - x_2}{x_0 - x_2} y_{01}(x_0)$$
$$= 0 + y_{01}(x_0)$$
$$= \left[\frac{x_0 - x_0}{x_1 - x_0} f_1 + \frac{x_0 - x_1}{x_0 - x_1} f_0\right]$$
$$= 0 + f_0$$
$$= f_0$$

$$y_{012}(x_1) = \frac{x_1 - x_0}{x_2 - x_0} y_{12}(x_1) + \frac{x_1 - x_2}{x_0 - x_2} y_{01}(x_1)$$
$$= \left[\frac{x_1 - x_0}{x_2 - x_0} * \left(\frac{x_1 - x_1}{x_2 - x_1} f_2 + \frac{x_1 - x_2}{x_1 - x_2} f_1\right)\right]$$
$$+ \left[\frac{x_1 - x_2}{x_0 - x_2} * \left(\frac{x_1 - x_0}{x_1 - x_0} f_1 + \frac{x_1 - x_1}{x_0 - x_1} f_0\right)\right]$$
$$= \frac{x_1 - x_0}{x_2 - x_0} * f_1 + \frac{x_1 - x_2}{x_0 - x_2} * f_1$$
$$= \left[\frac{-x_1 + x_0 + x_1 - x_2}{x_0 - x_2}\right] * f_1$$
$$= f_1$$

$$y_{012}(x_2) = \frac{x_2 - x_0}{x_2 - x_0} y_{12}(x_2) + \frac{x_2 - x_2}{x_0 - x_2} y_{01}(x_2)$$
$$= y_{12}(x_2) + 0$$
$$= \left[\frac{x_2 - x_2}{x_1 - x_2} f_1 + \frac{x_2 - x_1}{x_2 - x_1} f_2\right]$$
$$= 0 + f_2$$
$$= f_2$$

b) T.B.: $y_{i,\dots,j}(x) = \frac{x - x_i}{x_j - x_i} y_{i+1,\dots,j}(x) + \frac{x - x_j}{x_i - x_j} y_{i,\dots j-1}(x)$

Bewijs: door inductie.

1) Voor 1 interpolatiepunt:

$y_{00}(x) = f_0$

2) Stel: ok voor veeltermen $y_{i,\dots,j}$ die interpoleren in j-i punten:

T.B.: interpoleert in j-i+1 punten.

Bewijs: $y_{i+1,\dots,j}(x)$ interpoleert in $x_{i+1},\dots x_j$

$y_{i,\dots,j-1}(x)$ interpoleert in $x_i,\dots x_{j-1}$

$$y_{i,\dots,j}(x_k) = \frac{x_k - x_i}{x_j - x_i} y_{i+1,\dots,j}(x_k) + \frac{x_k - x_j}{x_i - x_j} y_{i,\dots j-1}(x_k)$$

Voor $i+1 < k < j-1$:

$$y_{i,\dots,j}(x_k) = \frac{x_k - x_i}{x_j - x_i} y_{i+1,\dots,j}(x_k) + \frac{x_k - x_j}{x_i - x_j} y_{i,\dots j-1}(x_k)$$
$$= \left[\frac{x_k - x_i}{x_j - x_i} + \frac{x_k - x_j}{x_i - x_j} \right] * f_k$$
$$= \left[\frac{-x_k + x_i + x_k - x_j}{x_i - x_j} \right] * f_k$$
$$= f_k$$

Voor x_i:

$$y_{i,\dots,j}(x_i) = \frac{x_i - x_i}{x_j - x_i} y_{i+1,\dots,j}(x_i) + \frac{x_i - x_j}{x_i - x_j} y_{i,\dots j-1}(x_i)$$
$$= y_{i,\dots j-1}(x_i)$$
$$= f_i$$

Voor x_j:

$$y_{i,\dots,j}(x_j) = \frac{x_j - x_i}{x_j - x_i} y_{i+1,\dots,j}(x_j) + \frac{x_j - x_j}{x_i - x_j} y_{i,\dots j-1}(x_j)$$
$$= y_{i+1,\dots j}(x_j)$$
$$= f_j$$

Dus: $y_{i,\dots,j}(x)$ interpoleert in alle interpolatiepunten.

$y_{i,\dots,j}(x)$ is een veelterm van de juiste graad (analoog aan $y_{012}(x)$ zie hoger), dus $y_{i,\dots,j}(x)$ is de interpolerende veelterm van graad (j-i).

c) Bereken het aantal bewerkingen van het algoritme van Neville.

Optellingen:

In de binnenste lus: 5

Binnenste lus: i keer uitgevoerd, $i \leq n+1$.

Buitenste lus: n+1keer uitgevoerd.

Exacte uitdrukking in functie van (n+1):

$5*[(n+1)((n+1)+1)/2]$
$= (5/2)(n+1)(n+2)$
$= (5/2)(n^2 + 3n + 2)$
$= (5/2)n^2 + 15/2n + 5$

Dus: $O(n^2)$

Vermenigvuldigingen:

In de binnenste lus: 4

Binnenste lus: i keer uitgevoerd, $i \leq n+1$.
Buitenste lus: n+1 keer uitgevoerd.
Exacte uitdrukking in functie van (n+1):

$4*[(n+1)((n+1)+1)/2]$
$= (4/2)(n+1)(n+2)$
$= 2(n^2 + 3n + 2)$
$= 2n^2 + 6n + 4$

Dus: $O(n^2)$

d) Hoe zou je het aantal bewerkingen kunnen verminderen en hoeveel bewerkingen moet je dan minder doen?
(Hint: ga na welke bewerkingen meerdere keren worden gedaan en hoe je dit kan vermijden).

x_waarde - x_i wordt meerdere keren berekend.
$x_j - x_i = -(x_i - x_j) = -x_i + x_j$); de formule kan dus herschreven worden tot:

$$y_{i,\dots,j}(x) = \frac{x - x_i}{x_j - x_i} y_{i+1,\dots,j}(x) - \frac{x - x_j}{x_j - x_i} y_{i,\dots,j-1}(x)$$
$$= \frac{(x - x_i)\, y_{i+1,\dots,j}(x) - (x - x_j)\, y_{i,\dots,j-1}(x)}{x_j - x_i}$$

Het algoritme wordt nu:
voor i = 1(1)(n+1)
  $x_i^{\#} = x\_waarde - x_i$
voor i = 1(1)(n+1)
  voor j = 1(1)i

$$y_{i,\dots,j}(x) = \frac{x_i^{\#}\, y_{i+1,\dots,j}(x_j) - x_j^{\#}\, y_{i,\dots,j-1}(x_j)}{x_j - x_i}$$

Het aantal bewerkingen hiervoor is:

Optellingen:

Binnenste lus: i keer uitgevoerd, $i \leq n+1$.
Buitenste lus: n+1keer uitgevoerd.
Exacte uitdrukking in functie van (n+1):

$(n+1)+2*[(n+1)((n+1)+1)/2]$
$= (n+1) + (n+1)(n+2)$
$= (n+1) + (n^2 + 3n + 2)$
$= n^2 + 4n + 3$

Dus: $O(n^2)$

Vermenigvuldigingen:

In deel 1: 0
In de binnenste lus: 3
Binnenste lus: i keer uitgevoerd, $i \leq n+1$.
Buitenste lus: n+1 keer uitgevoerd.
Exacte uitdrukking in functie van (n+1):

$0 + 3*[(n+1)((n+1)+1)/2)]$
$= 3*[(n+1)((n+2)/2)]$
$= 3*(n+1)*((n+2)/2)$
$= 3/2 * (n^2 + 3n + 3)$
$= (3/2)n^2 + 9/2n + 9/2$

Dus: $O(n^2)$

De besparing op het aantal bewerkingen is $O(n^2)$:

Optellingen:

$(5/2)n^2 + 15/2n + 5 - [\, n^2 + 4n + 3]$

$= (5/2)n^2 + 15/2n + 5 \; -2/2n^2 \; -8/2n \; -3$

$= 3/2n^2 + 7/2n + 2$

Vermenigvuldigingen:

$2n^2 + 6n + 4 - [(3/2)n^2 + 9/2n + 9/2]$

$= 4/2n^2 + 12/2n + 8/2 \; -(3/2)n^2 \; -9/2n \; -9/2$

$= 1/2n^2 + 3/2n \; -1/2$

De $x_i^{\#}$ kunnen opgeslagen worden op de posities van de x_i; x_i is immers verder niet meer nodig

($x_j - x_i$ = x_waarde - x_i - x_waarde + x_j

= (x_waarde - x_i) - (x_waarde - x_j)

= $x_i^{\#} - x_j^{\#}$).

OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 7: ITERATIEVE METHODEN VOOR HET OPLOSSEN VAN TRANSCENDENTE VERGELIJKINGEN

(1) We willen een wortel vinden van $f(x) = x + \log(x)$.
Verschillende substitutiemethodes die we kunnen gebruiken, zijn:
(a) $x^{(k)} = -\log(x^{(k-1)}) = F(x^{(k-1)})$;
(b) $x^{(k)} = e^{-x^{(k-1)}} = F(x^{(k-1)})$;
(c) $x^{(k)} = \frac{x^{(k-1)} + e^{-x^{(k-1)}}}{2} = F(x^{(k-1)})$;
(d) $x^{(k)} = \sqrt{-x^{(k-1)} \log(x^{(k-1)})} = F(x^{(k-1)})$.
Ga voor elke methode consistentie en convergentie na. Geef, indien mogelijk, ook een grafische interpretatie ($x^0 = 0.9$; $x^* = 0.56714$).

(2) Als x^* een m-voudige wortel is van f, dan geldt voor de Newton-Raphson-methode dat ze lineair is als $m > 1$ en minstens kwadratisch als $m = 1$.
Toon aan dat de volgende aangepaste methode minstens van de tweede orde is (dus minstens kwadratisch)

$$F(x) = x - m\frac{f(x)}{f'(x)}.$$

Hierin is m uiteraard de multipliciteit van de wortel x^* van f.
(Denkvraagje: Als deze methode hogere orde heeft dan NR, waarom wordt ze dan niet verkozen boven NR?)

(3) Iteratie voor $\sqrt{a}$ met behulp van NR.
Er zijn verschillende mogelijke functies f waarvan de wortels gelijk zijn aan $\sqrt{a}$. Bekijk
(a) $f(x) = x^2 - a$;
(b) $f(x) = 1 - \frac{a}{x^2}$;
(c) $f(x) = x - \frac{a}{x}$.
Pas telkens NR toe en bespreek (schets $f(x)$ en $F(x)$, consistentie, convergentie, orde).

Oefenzitting 4: Iteratieve methoden voor het oplossen van transcendente vergelijkingen.

Herhaling.

- Vast punt:
  - x* is een vast punt van een functie f
    <=> f(x*) = x*
- Consistentie:
  - Een methode heet consistent als alle nulpunten van f ook vaste punten zijn van F (met F de iteratieformule).
  - Een methode heet reciprook consistent als alle vaste punten van F ook nulpunten zijn van f.
  - Een methode heet volledig consistent als ze zowel consistent als reciprook consistent is.
  - Een methode heet zwak consistent als x* een nulpunt is van f en ook een vast punt van F (f(x*) = 0 => F(x*) = x*).

- Convergentie:
  - Convergentiestelling: als x* een vast punt is van F, als F afleidbaar is in de omgeving van x*, en als F zwak consistent is met f, dan zal de rij gedefinieerd door $x^{(k)} = F(x^{(k-1)})$ convergeren naar x* als x* een nulpunt is van f en $x^{(0)}$ voldoende dicht bij x*.
  - Voorwaarden voor convergentie:
    - $|F'(x)| \leq 1$ (als F afleidbaar is).
    - $x^{(0)}$ voldoende dicht bij x*.
    - Minstens zwakke consistentie.
    - Soms treedt er convergentie op ondanks dat (sommige) voorwaarden niet voldaan zijn.
- Convergentie-orde:
  - $p = \sup\{n \in |R: \lim_{k->\infty} \varepsilon^{(k+1)}/[\varepsilon^{(k)}]^n = 0\}$
    $= \inf\{n \in |R: \lim_{k->\infty} \varepsilon^{(k+1)}/[\varepsilon^{(k)}]^n \neq 0\}$
  - Voor substitutiemethoden is p steeds een geheel getal.
  - Practisch: bereken F (= $F^{(0)}$), F' (= $F^{(1)}$), F'' = $F^{(2)}$, $F^{(3)}$, ... totdat $F^{(i)}(x^*) \neq 0$.
    i is nu de convergentie-orde.

- Newton-Raphson:

1) f(x) benaderen door een rechte (de eerstegraads Hermite-interpolerende veelterm = de raaklijn in $(x^{(i-1)}, f(x^{(i-1)}))$:
   $y_{(i-1)(i-1)}(x) = f(x^{(i-1)}) + f'(x^{(i-1)})(x - x^{(i-1)})$
2) $x^{(i)}$ is het nulpunt v/d rechte voor $x^{(i-1)}$:
   $x^{(i)} = x^{(i-1)} - f(x^{(i-1)})/f'(x^{(i-1)})$
3) Stappen 1) en 2) herhalen totdat de gewenste nauwkeurigheid bereikt is (of totdat het maximaal aantal iteratiestappen bereikt is).

4) Pseudo-algoritme:

```
function [res, msg] = NewtonRaphson
                (x0, f, f ', epsilon, Kmax)
for i=1:Kmax,
    f0 = f(x0);
    f '0 = f '(x0);
    res = x0 - f0/f '0;
    if (abs(res - x0) < epsilon),
            msg = 0; % convergentie
            return;
    else
            x0 = res;
    end; % if
end; % for
msg = -1; % geen convergentie voor Kmax
return;
```

5) Consistentie:

$F(x) = x - f(x)/f'(x)$

$f(x)$ heeft m-voudige wortel x^*, dus:

$f(x) = (x-x^*)^m g(x)$ met $g(x^*) \neq 0$

$f'(x) = m(x-x^*)^{m-1} g(x) + (x-x^*)^m g'(x)$

$= (x-x^*)^{m-1}[m\, g(x) + (x-x^*)g'(x)]$

$$F(x) = x - \frac{(x - x^*)^m g(x)}{m(x-x^*)^{m-1} g(x) + (x-x^*)^m g'(x)}$$

$$= x - \frac{(x - x^*)^m g(x)}{m(x-x^*)^{m-1} g(x) + (x-x^*)^m g'(x)}$$

$$= x - \frac{(x - x^*) g(x)}{m\, g(x) + (x-x^*) g'(x)}$$

$$F(x^*) = x^* - \frac{(x^* - x^*) g(x^*)}{m\, g(x^*) + (x^*-x^*) g'(x^*)}$$

$F(x^*) = x^* - 0/[m\, g(x^*)]$

met $g(x^*) \neq 0$ en $m > 0$

$= x^*$

Consistent.

6) Convergentie-orde:

$F'(x) = 1 - f'(x)/f'(x) + f(x)f''(x)/[f'(x)]^2$
$= 1 - 1 + f(x)f''(x)/[f'(x)]^2$
$= f(x)f''(x)/[f'(x)]^2$

$f''(x) = m(m-1)(x-x^*)^{m-2}g(x) + 2m(x-x^*)^{m-1}g'(x) + (x - x^*)^m g''(x)$
$= (x-x^*)^{m-2}[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x - x^*)^2 g''(x)]$

$f(x)\, f''(x) = (x-x^*)^m g(x)(x-x^*)^{m-2}$
$[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x- x^*)^2 g''(x)]$
$= (x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]$

$F'(x) = \dfrac{(x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{((x-x^*)^{m-1}[m\, g(x) + (x-x^*)g'(x)])^2}$
$= \dfrac{(x-x^*)^{2m-2}g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{(x-x^*)^{2m-2}[m\, g(x) + (x-x^*)g'(x)]^2}$
$= \dfrac{g(x)[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]}{m^2(g(x))^2 + (x-x^*)^2(g'(x))^2 + 2m(x-x^*)g(x)g'(x)}$

$F'(x^*) = \dfrac{g(x^*)[m(m-1)g(x^*) + 2m(x^*-x^*)g'(x^*) + (x^*-x^*)^2 g''(x^*)]}{m^2(g(x^*))^2 + (x^*-x^*)^2(g'(x^*))^2 + 2m(x^*-x^*)g(x^*)g'(x^*)}$
$= \dfrac{g(x^*)[m(m-1)g(x^*)]}{m^2(g(x^*))^2}$
$= \dfrac{(g(x^*))^2[m(m-1)]}{m^2(g(x^*))^2}$
$= (m - 1)/m$
$= 1 - 1/m$

Omdat $m \geq 1$, $m \in |N$ (want x^* is een wortel) geldt:
$m = 1$: $F'(x^*) = 0$ en orde ≥ 2
$m > 1$: $0 < F'(x^*) < 1$ en orde $= 1$.

Oplossingen van de oefeningen:

Oefening 1:

a) $x^{(k)} = -\log x^{(k-1)} = F(x^{(k-1)})$:

Consistentie:

Stel x* is een vast punt van F:

$F(x^*) = x^* \quad <=> \quad x^* = -\log x^*$

$<=> x^* + \log x^* = 0$

$<=> f(x^*) = 0$

Volledig consistent.

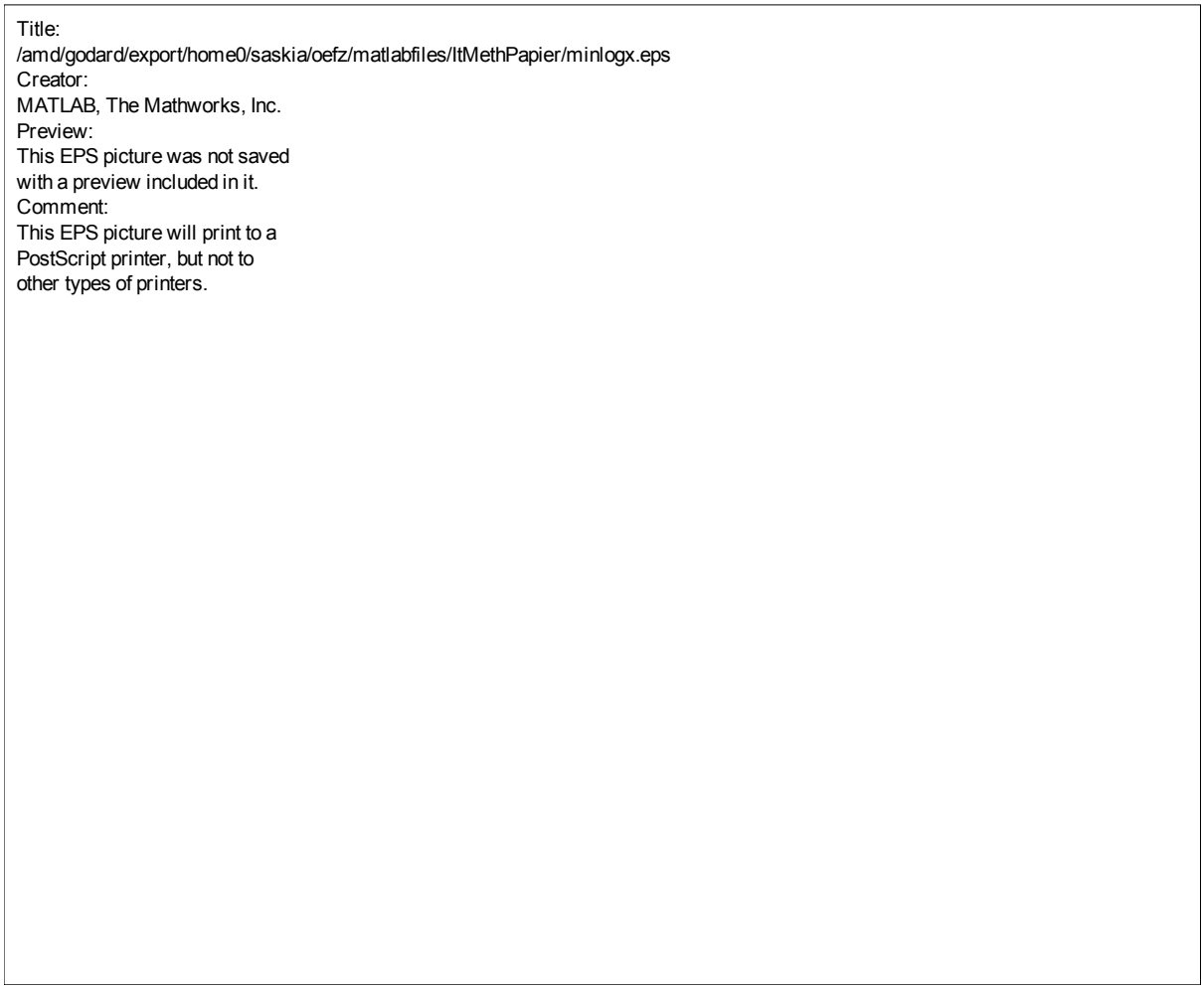
Convergentie:

$f = F'(x^*)$

$F'(x) = -1/x$

$F'(x^*) = -1{,}7632$

Divergent



b) $x^{(k)} = e^{-x(k-1)} = F(x^{(k-1)})$

Consistentie:

Stel x* een vast punt van F:

$F(x^*) = x^* \quad <=> \quad x^* = e^{-x^*}$

$<=> \log x^* = -x^*$

$<=> \log x^* + x^* = f(x^*) = 0$

Volledig consistent.

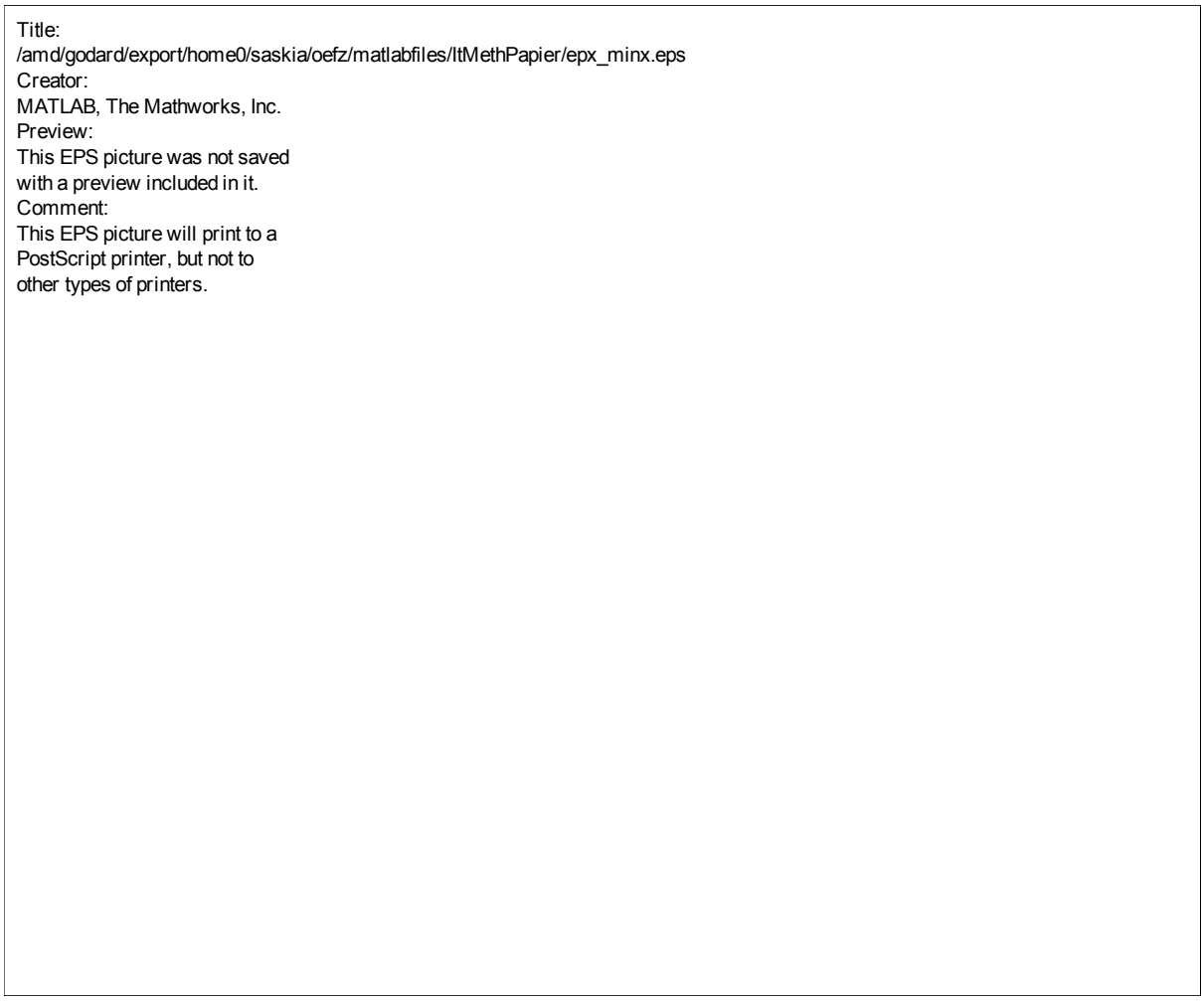
Convergentie:

$f = F'(x^*)$

$F'(x) = -e^{-x}$

$F'(x^*) = -0{,}56714$

Convergent



c) $F(x) = (x + e^{-x})/2$

Consistentie:

Stel x* een vast punt van F:

$F(x^*) = x^* \quad <=> \quad x^* = (x^* + e^{-x^*})/2$

$<=> 2x^* = x^* + e^{-x^*}$

$<=> x^* = e^{-x^*}$

Volledig consistent.

Convergentie:

$f = F'(x^*)$

$F'(x) = (1/2) - [(e^{-x})/2]$

$F'(x^*) = 0{,}21642742$

Convergent, snellere conv. dan b)

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ItMethPapier/x_plusex_door2.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

d) $F(x) = \sqrt{(-x \log x)}$

Consistentie:

Stel x* een vast punt van F:

$F(x^*) = x^* \quad <=> \quad x^* = \sqrt{(-x^* \log x^*)}$

$<=> x^{*2} = -x^* \log x^*$

$<=> x^*(x^* + \log x^*) = 0$

$<=> x^* = 0$ of $f(x^*) = 0$

Consistent, niet reciprook-consistent.

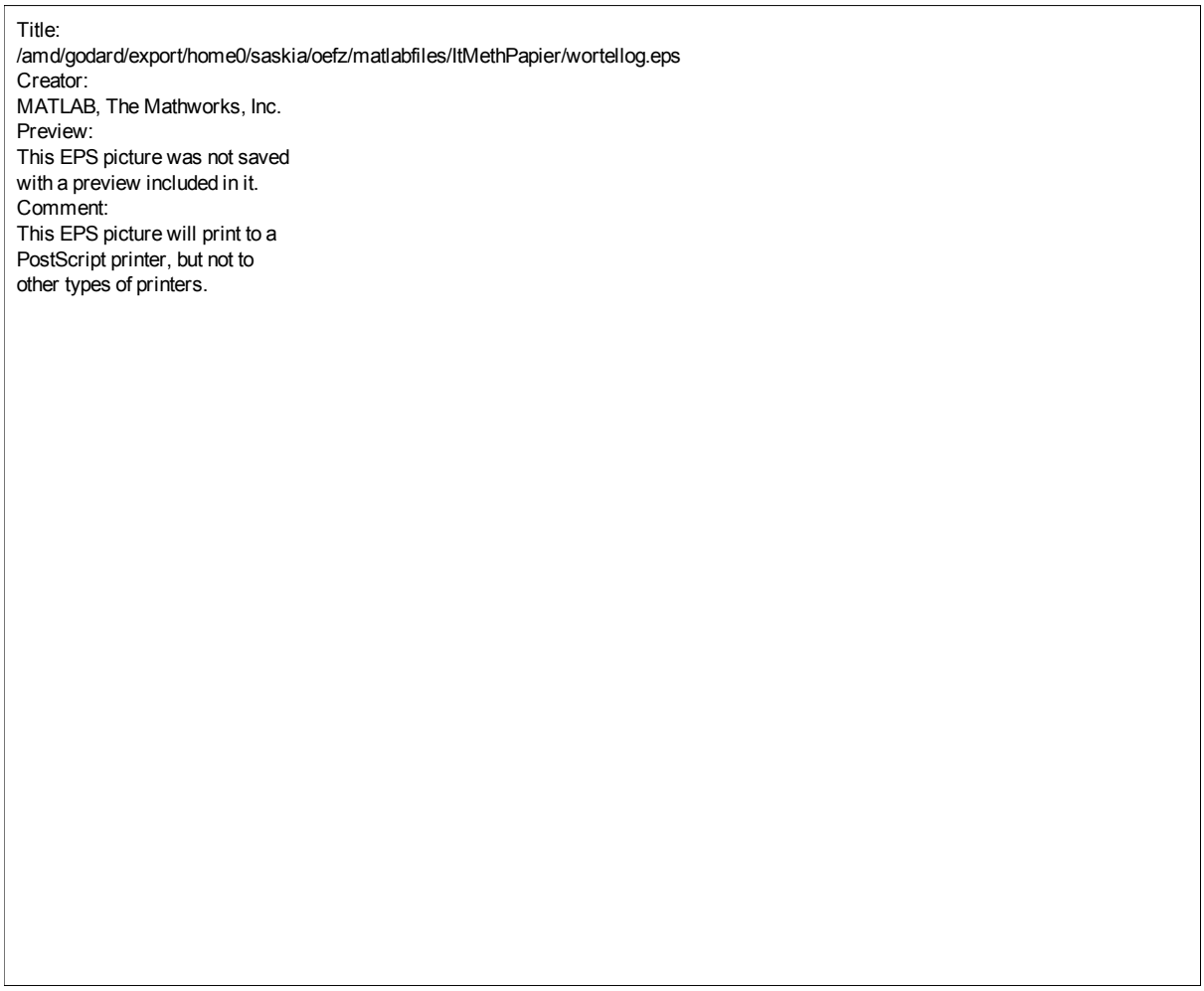
Convergentie:

$f = F'(x^*)$

$F'(x) = 1/2 \, [1/\sqrt{(-x \log x)}] \, (-\log x - 1)$

$F'(x^*) = -0{,}3816$

Convergent



Oefening 2:

$F(x) = x - m[f(x)/f'(x)]$

$F'(x) = 1 - m([(f'(x))^2 - f(x)f''(x)]/[(f'(x))^2])$
$= 1 - m(1 - [f(x)f''(x)]/[(f'(x))^2])$
$= 1 - m + m([f(x)f''(x)]/[(f'(x))^2])$

$f(x) = (x - x^*)^m g(x)$ met $g(x^*) \neq 0$

$f'(x) = m(x - x^*)^{m-1}g(x) + (x - x^*)^m g'(x)$

$f''(x) = m(m-1)(x-x^*)^{m-2}g(x) + m(x-x^*)^{m-1}g'(x)$
$+ m(x-x^*)^{m-1}g'(x) + (x - x^*)^m g''(x)$
$= m(m-1)(x-x^*)^{m-2}g(x) + 2m(x-x^*)^{m-1}g'(x) + (x- x^*)^m g''(x)$
$= (x-x^*)^{m-2}[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)]$

$[f(x)f''(x)] =$
$(x - x^*)^m g(x)(x-x^*)^{m-2}[m(m-1)g(x) + 2m(x-x^*)g'(x) + (x-x^*)^2 g''(x)] =$
$g(x)(x - x^*)^{2m-2}[m(m-1)g(x) + 2m(x - x^*) g'(x) + (x -x^*)^2 g''(x)]$

$[f(x)f''(x)]/[(f'(x))^2] =$

$$\frac{g(x)(x -x^*)^{2m-2}[m(m-1)g(x) + 2m(x -x^*)g'(x) + (x -x^*)^2 g''(x)]}{[m(x - x^*)^{m-1} g(x) + (x - x^*)^m g'(x)]^2} =$$

$$\frac{g(x)(x -x^*)^{2m-2}[m(m-1)g(x) + 2m(x -x^*) g'(x) +(x -x^*)^2 g''(x)]}{[m^2(x-x^*)^{2m-2}(g(x))^2 +(x-x^*)^{2m}(g'(x))^2+2m(x-x^*)^{2m-1}g(x)g'(x)]} =$$

$$\frac{g(x)(x - x^*)^{2m-2}[m(m-1)g(x) +2m(x -x^*)g'(x) + (x -x^*)^2 g''(x)]}{(x-x^*)^{2m-2}[m^2(g(x))^2 + (x-x^*)^2(g'(x))^2 + 2m(x-x^*)g(x)g'(x)]} =$$

$$\frac{g(x)[m(m-1)g(x) + 2m(x - x^*)g'(x) + (x -x^*)^2 g''(x)]}{[m^2(g(x))^2 + (x-x^*)^2(g'(x))^2 + 2m(x-x^*)g(x)g'(x)]}$$

$$\frac{g(x)[m(m-1)g(x) + 2m(x - x^*)g'(x) + (x -x^*)^2 g''(x)]}{[mg(x) + (x-x^*)(g'(x)]^2}$$

$$\lim_{x->x^*} \frac{g(x)[m(m-1)g(x) + 2m(x - x^*)g'(x) + (x -x^*)^2 g''(x)]}{[mg(x) + (x-x^*)(g'(x)]^2} =$$

$$\lim_{x->x^*} \frac{g(x)[m(m-1)g(x)]}{m^2(g(x))^2} =$$

$$\lim_{x->x^*} \frac{g(x^*)[m(m-1)g(x^*)]}{m^2(g(x^*))^2} = (m-1)/m$$

$=> F'(x^*) = 1 - m + m[(m-1)/m]$
$= 1 - m + (m-1)$
$= 1 - m + m - 1$
$= 0$

=> Steeds orde ≥ 2 (onafhankelijk van de waarde van m!)

Antwoord op denkvraagje:

Met die methoden zoek je eigenlijk de nulpunten van f of de vaste punten van F(x); je kent de multipliciteit van de nulpunten dus zeker niet op voorhand.

Oefening 3:

a) $f(x) = x^2 - a$

$f'(x) = 2x$

$F(x) = x - f(x)/f'(x)$

$= x - [x^2 - a]/[2x]$

$= [2x^2 - x^2 + a]/(2x)$

$= (x^2 + a)/(2x)$

$= x/2 + a/(2x)$

$= (1/2)(x + a/x)$

H.A.: /

V.A.: x = 0

S.A.: y = x/2

Consistentie:

$F(x^*) = x^* \quad <=> \quad (1/2)(x^* + a/x^*) = x^*$

$<=> \quad x^* + a/x^* = 2x^*$

$<=> \quad a/x^* = x^*$

$<=> \quad 0 = x^* - a/x^*$

$<=> \quad (x^*)^2 - a = 0$

$<=> \quad f(x) = 0$

Volledig consistent.

Convergentie:

$F'(x) = 1/2(1-(a/x^2))$

$F'(x^*) = F'(\sqrt{a})$

$= 1/2(1-(a/a))$

$= 1/2(1-1)$

$= 0 \quad < 1$

Convergent.

Convergentie-orde:

$F'(x^*) = 0$

$F''(x) = a/x^3$

$F''(x^*) = F''(\sqrt{a})$

$= a/(\sqrt{a})^3$

$= a/a(\sqrt{a})$

$= 1/\sqrt{a} \quad \neq 0$

=> kwadratisch (orde = 2, want 2e afgeleide is de eerste n waarvoor $F^{(n)} \neq 0$).

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ItMethPapier/xkwad_mina.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ItMethPapier/x_plusadoorx_door2.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

b) $f(x) = 1 - a/x^2$

$f'(x) = (2a)/x^3$

$F(x) = x - f(x)/f'(x)$

$= x - [1 - a/x^2]/[(2a)/x^3]$

$= x - (x^3*[1 - a/x^2])/(2a)$

$= x - (x^3 - ax)/(2a)$

$= (2ax - x^3 + ax)/(2a)$

$= (3ax - x^3)/(2a)$

$= [x(3a - x^2)]/(2a)$

Consistentie:

$F(x^*) = x^* \quad <=> \quad [x^*(3a - (x^*)^2)]/(2a) = x^*$

$<=> \quad x^*(3a - (x^*)^2) = 2ax^*$

$<=> \quad x^*(3a - (x^*)^2) - 2ax^* = 0$

$<=> \quad x^*(3a - (x^*)^2 - 2a) = 0$

$<=> \quad x^*(a - (x^*)^2) = 0$

$<=> \quad x^* = 0$ of $f(x^*) = 0$

(Zowel $(a-(x^*)^2)$ als $(1 - a/x^2)$ worden 0 in $\sqrt{a}$)

Consistent, niet reciprook consistent.

Convergentie:

$F'(x) = [(3a - x^2) - 2x^2]/(2a)$

$= [3a - 3x^2]/(2a)$

$F'(x^*) = F'(\sqrt{a})$

$= [3a - 3a]/(2a)$

$= 0 \quad < 1$

Convergent.

Convergentie-orde:

$F'(x^*) = 0$

$F''(x) = (-3x)/a$

$F''(x^*) = F''(\sqrt{a})$

$= (-3\sqrt{a})/a$

$= -3/\sqrt{a} \quad \neq 0$

=> kwadratisch (orde = 2, want 2e afgeleide is de eerste n waarvoor $F^{(n)} \neq 0$).

Grenzen van het convergentie-interval:

Convergentie als $|F(x)| < |x|$.

($|F'(x)| < 1$).

$|F(x)| = |x| \quad <=> \quad |[x(3a-x^2)]/(2a)| = |x|$

$<=> \quad |x(3a-x^2)| = |x|*2a$

$<=> \quad |(3a-x^2)| = 2a$

Opm: $|x| \geq \sqrt{(3a)}$

Dan: $0 = 2a$.

$x^* = \pm\sqrt{(3a)}$

$<=> \quad x^2 - 3a = 2a$

$<=> \quad x^2 = 5a$

$<=> \quad x = \pm\sqrt{(5a)}$

Geen convergentie voor $x = 0$, want $x = 0$ is een vast punt van F maar geen nulpunt van f.

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ltMethPapier/een_min_adoorxkwad.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ltMethPapier/x_maal_3aminxkwad_door2a.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

c) $f(x) = x - a/x$

$f'(x) = 1 + a/x^2$

$F(x) = x - f(x)/f'(x)$

$= x - [x - a/x]/[1 + a/x^2]$

$= x - [x^3 - ax]/[x^2 + a]$

$= [x(x^2 + a) - (x^3 - ax)]/[x^2 + a]$

$= [x^3 + ax - x^3 + ax)]/[x^2 + a]$

$= (2ax)/(x^2 + a)$

Consistentie:

$F(x^*) = x^* \quad <=> \quad (2ax^*)/((x^*)^2 + a) = x^*$

$<=> 2ax^* = x^*((x^*)^2 + a)$

$<=> 0 = x^*((x^*)^2 + a) - 2ax^*$

$<=> 0 = (x^*)^3 + ax^* - 2ax^*$

$<=> 0 = (x^*)^3 - ax^*$

$<=> x^*((x^*)^2 - a) = 0$

$<=> \quad x^* = 0$ of $f(x) = 0$

Consistent, niet reciprook consistent.

Convergentie:

$F'(x) = (2ax^2 + 2a^2 - 4ax^2)/(x^2 + a)^2$

$= (2a^2 - 2ax^2)/(x^2 + a)^2$

$= [2a(a - x^2)]/(x^2 + a)^2$

$F'(x^*) = F'(\sqrt{a})$

$= [2a(a - (\sqrt{a})^2)]/((\sqrt{a})^2 + a)^2$

$= [2a(a - a)]/(a + a)^2$

$= [2a(0)]/(2a)^2$

$= 0/(2a)$

$= 0$

Convergent.

Convergentie-orde:

$F'(x^*) = 0$

$F''(x) = [2a(-2x(x^2+a)^2-(a-x^2)2(x^2+a)2x)]/(x^2+a)^4$

$= [(x^2+a)2a2x(-(x^2+a)-(a-x^2)2)]/(x^2+a)^4$

$= [4ax(-x^2-a-2a+2x^2)]/(x^2+a)^3$

$= [4ax(x^2-3a)]/(x^2+a)^3$

$F''(x^*) = F''(\sqrt{a})$

$= [4a(\sqrt{a})((\sqrt{a})^2-3a)]/((\sqrt{a})^2+a)^3$

$= [4a(\sqrt{a})(a-3a)]/(a+a)^3$

$= [2a*2(\sqrt{a})(-2a)]/(2a)^3$

$= [2(\sqrt{a})(-1)]/(2a)$

$= -\sqrt{a})/a$

$= -1/\sqrt{a} \quad \neq 0$

=> kwadratisch (orde = 2, want 2e afgeleide is de eerste n waarvoor $F^{(n)} \neq 0$).

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ltMethPapier/x_min_adoorx.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/ltMethPapier/tweeax_doorkwadplusa.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

pc-zitting 4: Substitutiemethodes

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Inleiding

Voor het uitwerken van de opgaven gebruik je '.m'-bestanden die je kan vinden in

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

Voor deze oefenzitting heb je ook een Maple-illustratie nodig, die je kan vinden via

`http://www.cs.kuleuven.ac.be/~marc/NumWis/Menu/indexG033.htm`

In de eerste oefening ligt de nadruk op het zelf implementeren van substitutiemethodes in matlab en het interpreteren van de resultaten. De tweede oefening is naast een eerste kennismaking met Maple, bedoeld om het leren werken met de verschillende methodes: falen, convergentiefactor, ... In de derde oefening bekijken we Newton-Raphson, een van de belangrijkste numerieke methodes, meer in detail.

1 Berekenen van de gulden snede

In deze opgave worden methoden onderzocht om de gulden snede (i.e. $(\sqrt{5}-1)/2$) numeriek te berekenen waarbij enkel gebruik gemaakt wordt van elementaire bewerkingen (optellen en vermenigvuldigen). De gulden snede kan berekend worden als de positieve oplossing van de vergelijking

$$x^2 + x - 1 = 0.$$

Implementeer hiervoor onderstaande substitutieformules. Ga telkens na of er convergentie of divergentie optreedt. Welke formule convergeert het snelst? Neem als startwaarde $x^{(0)} = 0.7$. Hoe zie je of de convergentie al dan niet monotoon is?

$$x^{(k)} = 1 - \left(x^{(k-1)}\right)^2$$

$$x^{(k)} = \frac{3\left(x^{(k-1)}\right)^2 - x^{(k-1)} + 1}{4x^{(k-1)}}$$

$$x^{(k)} = \frac{4 - x^{(k-1)}}{4x^{(k-1)} + 3}$$

De substitutiemethode

$$x^{(k)} = x^{(k-1)} - \frac{\left(x^{(k-1)}\right)^2 + x^{(k-1)} - 1}{2x^{(k-1)} + 1}$$

convergeert veel sneller dan de voorgaande methoden. Hoe verklaar je dat?

2 Convergentiesnelheid van een aantal algoritmes

Open de Maple-demo "Niet-lineaire vergelijkingen". De nulpunten van een gegeven functie worden gezocht met de methodes: bisectie, secant, newton, muller en halley. Voor alle methodes wordt ook het verloop van de relatieve fout en convergentiefactor gegeven.

- Bekijk rustig het Maple-werkblad. Over welke functie gaat het, wat wordt er precies geïmplementeerd, ...
- Komen de convergentiefactoren overeen met wat in de cursus staat?
- Verander de funtie in $f(x) = x^2 - 2$, $f(x) = (x-1)^2$ en $f(x) = (x-1)^3$
- Wat gebeurt er met de convergentiefactoren?
- Waarom falen sommige methodes?
- Ga de invloed na van de startwaarden.

Opmerking: als je per iteratiestap een constant aantal cijfers wint, dan convergeert de methode lineair. Als het aantal cijfers verdubbelt, dan zegt men dat het algoritme kwadratisch convergeert. Ligt het ertussen, dan spreekt men van een superlineair convergentiegedrag. Het nagaan van de convergentie-orde kan natuurlijk alleen gebeuren indien de getallen met voldoende cijfers worden weergegeven!

Opmerking 2: als je iets verandert aan het Maple-werkblad, begin dan steeds terug bovenaan met restart!

Opmerking 3: indien een wortel meervoudig is, dan zet je zero zelf op de juiste waarde.

3 Newton-Raphson

Gebruik de methode van Newton-Raphson om $\sqrt{2}$ te berekenen. Dit kan bv. door de oplossing te zoeken van één van onderstaande vergelijkingen. Ga de convergentie na van elk van die vergelijkingen. Gebruik hiervoor de startwaarde $x^{(0)} = 2$. Herhaal daarna de berekeningen met startwaarde $x^{(0)} = 6$.

$$x^2 - 2 = 0$$

$$1 - \frac{2}{x^2} = 0$$

$$x - \frac{2}{x} = 0$$

$\sqrt{2}$ voldoet ook aan volgende vergelijking: $x^4 - 4x^2 + 4 = 0$. Bereken deze wortel met behulp van de Newton-Raphson methode en m.b.v. de substitutieformule

$$x^{(k)} = x^{(k-1)} - 2\frac{f(x^{(k-1)})}{f'(x^{(k-1)})}.$$

Waarom convergeert de tweede methode sneller?

4 Extra oefeningen

1. Bereken met behulp van `secant.m`, `bisect.m`, `dekker.m` en je eigen Newton-Raphson implementatie het nulpunt van $f(x) = \tan(x) - x - 1$ gelegen in het interval $[0\ ,\ \frac{\Pi}{2}]$. Wat is de invloed van de startwaarden op de convergentie van de methode? (en naar het juiste nulpunt!)

2. Ontwikkel een iteratieve methode om arctan(2) te berekenen, in de veronderstelling dat naast de elementaire bewerkingen in je rekenmachine (computer) alleen sin en cos zijn ingebouwd. Maak daarbij gebruik van de methode van Newton-Raphson.

3. Onderzoek volgende substitutiemethode voor het berekenen van $\cosh(x) - 2 = 0 \ \ x > 0$:
$$x^{(k)} = \log(4 - e^{-x^{(k-1)}})$$
Geef een substitutieformule die sneller convergeert?

4. De vaste punten van $x^{(k)} = F(x^{(k-1)})$ komen overeen met deze van $x^{(k)} = \frac{1}{1+\alpha}F(x^{(k-1)}) + \frac{\alpha}{1+\alpha}x^{(k-1)}$. Wat is de invloed van α op de convergentie. Ga je bevindingen (numeriek) na aan de hand van voorbeelden uit de andere opgaven.

5. We willen een wortel vinden van volgende vergelijking
$$f(x) = x + \log x = 0.$$
Implementeer hiervoor onderstaande substitutieformules. Ga telkens na of er convergentie of divergentie optreedt. Welke formule convergeert het snelst? Neem als startwaarde $x^{(0)} = 0.9$. De exacte wortel is 0.5671432904...
Hoe zie je of de convergentie al dan niet monotoon is?
$$x^{(k)} = -\log x^{(k-1)}$$
$$x^{(k)} = e^{-x^{(k-1)}}$$
$$x^{(k)} = \frac{x^{(k-1)} + e^{-x^{(k-1)}}}{2}$$
$$x^{(k)} = \sqrt{-x^{(k-1)}.\log x^{(k-1)}}$$

PC-zitting 4: stelsels.

Opmerkingen:
De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~saskia/oefz
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Oplossingen van de oefeningen:

Oefening 1:

Mogelijke reeks commando's:
```
path(path, 'd:\user')
format long
x_init = 0.7;
epsilon = 0.0001;
kmax = 30;
res1 = oefz4oef1a(x_init, epsilon, kmax);
res2 = oefz4oef1b(x_init, epsilon, kmax);
res3 = oefz4oef1c(x_init, epsilon, kmax);
res4 = oefz4oef1d(x_init, epsilon, kmax);
size(res1);
size(res2);
size(res3);
size(res4);
```
Welke formule convergeert het snelst?
De laatste; de grootte van de vierde resultaatvector is immers het kleinst.
Omdat het eerste element de startwaarde is, is er convergentie in stap (size - 1) als er convergentie is.
De eerste methode is divergent.
($|F'(x^*)| = |-2x^*| \approx |-2*0.618| = 1{,}236$, dus > 1).
Van de andere methoden convergeert methode 3 het traagst (in stap 20).
Hoe zie je of convergentie monotoon is?
Als elk element van de resultaatvector groter is dan het vorige (of elk element is kleiner dan het vorige), dan is er monotone convergentie (bv. de 2e methode en de 4e methode).
(De 3e methode convergeert spiraalvormig).
Waarom convergeert de laatste methode sneller dan alle voorgaande?
(Exacte oplossing x* van gulden snede is ongeveer 0,618).
Methode 2:

$$F'(x^*) = (6x^*-1)/(4x^*) - 4(3(x^*)^2-x^*+1)/[(4x^*)^2]$$
$$= [24(x^*)^2 -4x^* -12(x^*)^2 +4x^*-4]/[(4x^*)^2]$$
$$= [12(x^*)^2 - 4]/[(4x^*)^2]$$
$$= [3(x^*)^2 - 1]/[4(x^*)^2]$$
$$\approx 0{,}095$$

Vrij snelle convergentie.
Methode 3:

$$F'(x^*) = -1/(4x^* + 3) - 4*(4 - x^*)/[(4x^* + 3)^2]$$
$$= [-(4x^* +3) - 16 - 4x^*]/[(4x^* +3)^2]$$
$$= [-4x^* -3 -16 -4x^*]/[16(x^*)^2 + 9 + 24x^*]$$

$= [-8x^* - 19]/[16(x^*)^2 + 9 + 24x^*]$
$\approx 0{,}754$

Trage convergentie.

Methode 4:

$F'(x^*) = 1 - (2x^*+1)/(2x^*+1) + 2((x^*)^2 + x^* - 1)/[(2x^* + 1)^2]$
$= 1 - 1 + 2((x^*)^2 + x^* - 1)/[(2x^* + 1)^2]$
$= [2(x^*)^2 + 2x^* - 2]/[4(x^*)^2 + 4x^* + 1]$
$\approx -0{,}000030401$

Zeer snelle convergentie.

Oefening 2:

Mogelijke reeks commando's:

```
beginint = 1;
eindeint = 2;
x_init = 0.5;
epsilon = 0.001;
kmax = 20;
res1 = bisect('fun1', beginint, eindeint, epsilon)
res2 = secant('fun1', beginint, eindeint, epsilon, kmax)
res3 = dekker('fun1', beginint, eindeint, epsilon)
res4 = newton_raphson('fun1', 'dfun1', x_init, epsilon, kmax)
res1 = bisect('fun1b', beginint, eindeint, epsilon)
res2 = secant('fun1b', beginint, eindeint, epsilon, kmax)
res3 = dekker('fun1b', beginint, eindeint, epsilon)
res4 = newton_raphson('fun1b', 'dfun1b', x_init, epsilon, kmax)
beginint = -1;
eindeint = 3;
x_init = 1;
res1 = bisect('fun1b', beginint, eindeint, epsilon)
res2 = secant('fun1b', beginint, eindeint, epsilon, kmax)
res3 = dekker('fun1b', beginint, eindeint, epsilon)
res4 = newton_raphson('fun1b', 'dfun1b',
```

Convergentiesnelheid voor a = 2:
Bisectiemethode: lineair
Secantmethode: superlineair
Dekker-Brent: spiraalvormige convergentie in de eerste stappen, daarna monotoon.
Newton_raphson: kwadratisch
Convergentiesnelheid voor a = 0:
Met als interval [1, 2] zijn bisectie en Dekker-Brent divergent; dit is logisch omdat de wortel buiten het interval ligt en omdat het een dubbele wortel is.
De secantmethode convergeert traag.
Newton-Raphson is ook convergent.
Met als interval [-1, 3]:
Bisectiemethode: divergent
Secantmethode: trage convergentie
Dekker-Brent: divergent
Newton_raphson: trage convergentie

De invloed van de keuze van de startwaarden is als volgt:
voor de secantmethode: de startwaarden mogen niet zodanig gekozen zijn dat de rechte door

de functiewaarden ervan parallel loopt met de X-as; anders is er geen convergentie mogelijk (geval a = 0: niet symmetrisch rond de Y-as kiezen).
voor de bisectiemethode: het interval moet de wortel bevatten en de gezochte wortel mag geen dubbele wortel zijn (geen wortel die een even aantal keer kan afgezonderd worden in het algemeen).
Voor Dekker-Brent: de wortel mag geen even wortel zijn.
Voor Newton-Raphson: deze methode convergeert bijna altijd; de startwaarde moet dicht genoeg bij de wortel liggen.

Oefening 3:

Mogelijke commando's:

```
x_init = 2;
epsilon = 0.0000001;
kmax = 50;
res1 = newton_raphson('fun1', 'dfun1',x_init, epsilon, kmax);
res2 = newton_raphson('fun2', 'dfun2', x_init, epsilon, kmax);
res3 = newton_raphson('fun3', 'dfun3', x_init, epsilon, kmax);
res4 = newton_raphson('fun4', 'dfun4', x_init, epsilon, kmax);
res5 = substit_form('fun4', 'dfun4', x_init, epsilon, kmax);
x_init = 6;
epsilon = 0.0000001;
kmax = 50;
res1b = newton_raphson('fun1', 'dfun1',x_init, epsilon, kmax);
res2b = newton_raphson('fun2', 'dfun2', x_init, epsilon, kmax);
res3b = newton_raphson('fun3', 'dfun3', x_init, epsilon, kmax);
res4b = newton_raphson('fun4', 'dfun4', x_init, epsilon, kmax);
res5b = substit_form('fun4', 'dfun4', x_init, epsilon, kmax);
```

Convergentie:
Voor x0 = 2: de eerste 4 methoden zijn convergent.
Voor x0 = 6: de 2e methode divergeert; de andere methoden convergeren.
Verklaring snellere conv. van 5e methode:
zie theoretische oefenzitting 4 (iteratieve methode): Newton-Raphson kan verbeterd worden door volgende formule:
$F(x) = x - m\, f(x)/f'(x)$
Hier: dubbele wortel => m = 2.

Oefening 4: demo Phaser.

De demo betreft een simulatie van de 1-dimensionale differentiaalvergelijking:
de logistische vergelijking (met trapstapdiagramma) en een simulatie met Newton-Raphson.
Logistische vergelijking: x1' = ax1(1.0 - x1)
Opstarten van de demo: nphaser.exe
Commando's:
u
t
1<enter>
n
d
1<enter>
a
d
w
-0.1<enter>
1.1<enter>
-0.1<enter>
1.1<enter>
i
0.3<enter>
<enter>
g
Resultaten: Eerste figuur:
Eerste beginwaarde:
het evenwichtspunt is onstabiel; in de eerste iteratie kom je toevallig dicht bij het evenwichtspunt, maar dan ga je er weer van weg.
Commando's:
c
u
1
2<enter>
g
Resultaten: Tweede figuur:
Werken met dubbele machten (telkens "2 stappen" uitvoeren in 1 stap: G(x) = F(F(x)) berekenen i.p.v. F(x) zoals eerst).
"Convergentie" naar een vast punt van G(x), maar niet naar de gezochte oplossing.
Commando's:
1
1<enter>
b
1
n
t
100<enter>
300<enter>
g

Resultaten: Derde figuur:
De iteratieformule op scherm is nu weer F(x); tussen stap 100 en stap 300 blijf je verspringen tussen 2 punten, dus is het onstabiel. (Divergent).
Commando's:
c
p a 3.65 <enter>

<enter>
g
Resultaten: Vierde figuur:
Andere waarde voor de parameter a:
Chaotisch gedrag; er is geen convergentie naar een bepaald vast punt en ook geen periodisch iets.
Commando's:
c
t 0<enter>
60<enter>
p a 2.8 <enter>
<enter>
g
Resultaten: Vijfde figuur:
Andere waarde voor de parameter a:
Convergentie.
De parameter a heeft een invloed op de afgeleide van F(x) en dus op het convergentiegedrag.

Commando's:
c
e
newton
w -2<enter>
2<enter>
-4<enter>
4<enter>
g
Resultaten: Zesde figuur:
Eerst een stap spiraalvormige convergentie, dan monotoon convergent.
Commando's:
c
i 3 <enter>
g
Resultaten: Zevende figuur:
Monotone convergentie.

Oefening 5.1:

Het gezochte nulpunt ligt in de buurt van 1.13 (maar is niet 1.13).
Om dit te zien, kun je de grafiek schetsen van tan(x) - x - 1:

```
begin = 0;
einde = pi/2;
aantal = 30;
[x, y] = oefz4oef51graf(begin, einde, aantal);
plot(x, y, 'b');
axis([begin einde -2 6]);
y2 = zeros(size(x));
hold;
plot(x, y2, 'g');
```

Mogelijke commando's:

```
beginint = 0;
eindeint = pi/2;
epsilon = 0.000001;
kmax = 50;
res1 = secant('fun5', beginint, eindint, epsilon, kmax)
res2 = bisect('fun5', beginint, eindint, epsilon)
res3 = dekker('fun5', beginint, eindint, epsilon)
start = 1.3;
res4 = newton_raphson('fun5', 'dfun5', start, epsilon, kmax)

beginint = 1.1;
eindeint = 1.3;
res1b = secant('fun5', beginint, eindint, epsilon, kmax)
res2b = bisect('fun5', beginint, eindint, epsilon)
res3b = dekker('fun5', beginint, eindint, epsilon)
start = 1.2;
res4b = newton_raphson('fun5', 'dfun5', start, epsilon, kmax)

beginint = 1.1;
eindeint = 1.2;
resc = secant('fun5', beginint, eindint, epsilon, kmax)
res2c = bisect('fun5', beginint, eindint, epsilon)
res3c = dekker('fun5', beginint, eindint, epsilon)
start = 1.15;
res4c = newton_raphson('fun5', 'dfun5', start, epsilon, kmax)
```

Invloed van de startwaarde op de convergentie (naar het juiste nulpunt):
Voor het volledige interval convergeert de secantmethode naar 0.
De bisectiemethode is convergent.
De methode van Dekker-Brent stopt omdat er een deling door 0 voorkomt.
De methode van Newton-Raphson is divergent met startwaarde 1.3.

Voor het 2e interval is de secantmethode convergent; de bisectiemethode is ook convergent, evenals Dekker-Brent.
Newton-Raphson is echter nog steeds divergent met startwaarde 1,2.
Zelfs bij de 3e poging zijn alle methoden behalve Newton-Raphson convergent!

Verklaring:
De secantmethode convergeert naar 0 in het volledige interval omdat de afgeleide in de buurt van 0 bijna volledig horizontaal is; het verschil tussen 2 opeenvolgende waarden is dan ook steeds bijzonder klein.

In beide andere intervallen valt het gedeelte waarin de afgeleide zo goed als horizontaal ligt, weg. Ook al is de afgeleide aan het begin van het interval nog steeds min of meer horizontaal, de machineprecisie laat toch toe om een verschil te ontdekken tussen 2 opeenvolgende waarden.
De bisectiemethode maakt geen gebruik van de afgeleide en is dus convergent naar het juiste nulpunt. Het gezochte nulpunt heeft een oneven multipliciteit.
Bij de eerste poging stopt de methode van Dekker-Brent omdat er een deling door 0 voorkomt, m.a.w., de machineprecisie is te klein om een verschil te ontdekken tussen 2 opeenvolgende functiewaarden. De andere keren is deze methode convergent.
Newton-Raphson is divergent in alle pogingen, zelfs dicht in de buurt van het nulpunt. Het gezochte nulpunt is onstabiel.

Oefening 5.2:

Opstellen van de functie f:
x* = Bgtan(2) <=> tan(x*) = tan(Bgtan(2))
<=> tan(x*) = 2
<=> tan(x*) - 2 = 0
<=> sin(x*)/cos(x*) - 2 = 0

f(x) = tan(x) - 2
= sin(x)/cos(x) - 2
Afgeleide van de functie f:
f '(x) = 1/(1 + x^2)
Het gezochte nulpunt: x* ≈ 1.107149
Mogelijke commando's:
epsilon = 0.0001;
kmax = 50;
start = 1.1;
opl = newton_raphson('fun6', 'dfun6', start, epsilon, kmax)
start = 1.107;
opl = newton_raphson('fun6', 'dfun6', start, epsilon, kmax)
start = 1.10715;
opl = newton_raphson('fun6', 'dfun6', start, epsilon, kmax)
start = 1.1072;
opl = newton_raphson('fun6', 'dfun6', start, epsilon, kmax)
Behalve voor 1.10715 is Newton-Raphson divergent... Het nulpunt ligt immers dicht bij een verticale asymptoot van de tangensfunctie, waardoor er een deling door 0 zou gebeuren in de meeste gevallen.

Oefening 5.3:

Het gezochte nulpunt ligt ongeveer in de helft tussen 1 en 1,5.
Dit kun je vinden door de grafiek te tekenen van cosh(x):

```
[x, y]  = oefz4oef53graf(0, pi, 50);
y2 = 2*ones(size(x));
plot(x, y, 'b')
hold
plot(x, y2, 'g')
```

Mogelijke commando's:

```
epsilon = 0.001;
kmax = 200;
start = 1.3158;
res = oefz4oef53(start, epsilon, kmax);
start = 1.3159;
res = oefz4oef53(start, epsilon, kmax);
```

Bevindingen:
Zelfs dicht in de buurt van het gezochte nulpunt is de convergentie zeer traag.
Snellere substitutieformule:
Newton-Raphson:
$x^{(k)} = x^{(k-1)} - [\cosh(x^{(k-1)}) - 2]/\sinh(x^{(k-1)})$
Verklaring verschil in convergentiesnelheid:
(cosh(x) - 2) heeft 2 enkelvoudige wortels (m = 1).
Newton-Raphson convergeert kwadratisch in dat geval; de in de opgave vermelde methode convergeert lineair.

Oefening 5.4:

Eis: alfa ≠ -1 (anders deling door 0).
Wat gebeurt er in de nieuwe formule?
De som van beide gewichten is steeds 1:
1/(1+ alfa) + alfa/(1+alfa) = (1+alfa)/(1+alfa)
alfa > 0:
I.p.v. voor de volgende waarde het resultaat van de substitutiemethode te gebruiken, wordt er een klein aandeel van de nieuwe waarde genomen en een groot aandeel van de oude startwaarde om de nieuwe startwaarde te berekenen.
Naarmate alfa dichter naar 0 nadert, is het gedrag van de nieuwe methode meer hetzelfde als dat van de gebruikte substitutiemethode.
-1 < alfa < 0:
I.p.v. voor de volgende waarde het resultaat van de substitutiemethode te gebruiken, wordt er een zeer groot aandeel van de nieuwe waarde genomen en een zeer klein aandeel van de oude startwaarde om de nieuwe startwaarde te berekenen.
Naarmate alfa dichter naar -1 nadert, zal het aandeel van de oude startwaarde lichtjes toenemen (maar klein blijven) en het aandeel van de nieuwe waarde lichtjes dalen.
alfa < -1:
Het aandeel van de oude startwaarde is zeer groot; het aandeel van de nieuwe waarde is ook groot, maar relatief gezien kleiner dan dat van de oude startwaarde.
Mogelijke commando's:

```
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = 5;
res1 = newton_raphson('fun1', 'dfun1', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun1', 'dfun1', x_init, epsilon, kmax);
```

Bevindingen:

Newton-Raphson is convergent; de andere methode is divergent.
Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = 5;
res1 = newton_raphson('fun2', 'dfun2', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun2', 'dfun2', x_init, epsilon, kmax);

Bevindingen:
Newton-Raphson is convergent; de andere methode geeft problemen (deling door 0 in Newton-Raphson).
Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = 5;
res1 = newton_raphson('fun3', 'dfun3', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun3', 'dfun3', x_init, epsilon, kmax);
Bevindingen:
Newton-Raphson is convergent; de andere methode is divergent.

Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = 5;
res1 = newton_raphson('fun5', 'dfun5', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun5', 'dfun5', x_init, epsilon, kmax);

Bevindingen:
Newton-Raphson is divergent; de andere methode is eveneens divergent.

Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = -5;
res1 = newton_raphson('fun4', 'dfun4', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun4', 'dfun4', x_init, epsilon, kmax);
Bevindingen:
Newton-Raphson is convergent; de andere methode is divergent.
Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = -5;
res1 = newton_raphson('fun5', 'dfun5', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun5', 'dfun5', x_init, epsilon, kmax);

Bevindingen:
Newton-Raphson is divergent; de andere methode is eveneens divergent.

Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = -0.0000001;
res1 = newton_raphson('fun5', 'dfun5', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun5', 'dfun5', x_init, epsilon, kmax);
Bevindingen:
Newton-Raphson is divergent; de andere methode is eveneens divergent.
Mogelijke commando's:
epsilon = 0.00001;
kmax = 50;
x_init = 2;
alfa = -0.0000001;
res1 = newton_raphson('fun1', 'dfun1', x_init, epsilon, kmax);
res2 = oefz4oef54(alfa, 'newton_raphson', 'fun1', 'dfun1', x_init, epsilon, kmax);

Bevindingen:
Newton-Raphson is convergent; de andere methode is eveneens convergent.

Invloed van alfa op de convergentie:
alfa heeft invloed op de convergentiesnelheid; het convergentiegedrag (conv/div) blijft gelijk indien men een onbeperkt aantal stappen veronderstelt.
Noem de gebruikte substitutiemethode F.
Noem de methode uit de opgave G.
Als de convergentiefactor ρ van een methode F negatief is, zal de nieuwe methode G elke gewenste convergentiefactor tussen ρ en 1 kunnen verkrijgen door een gepaste keuze van alfa. Door alfa zodanig te kiezen dat de convergentiefactor van G 0 is, krijgt men kwadratische convergentie. Het convergentieproces kan dus versneld worden.
Omgekeerd, door alfa slecht te kiezen kan de convergentiefactor van G in absolute waarde groter zijn dan de convergentiefactor van F, zodat het convergentieproces trager verloopt. (Het convergentieproces kan convergeren of divergeren in bovenstaande betekenis).

Oefening 5.5:

Opmerking:
De formules in de opgave zijn dezelfde als die in de theoretische oefenzitting 4 (iteratieve methode), oef. 1.
Mogelijke commando's:
start = 0.9;
epsilon = 0.0000000001;
kmax = 100;
res1 = oefz4oef55a(start, epsilon, kmax)
res2 = oefz4oef55b(start, epsilon, kmax)
res3 = oefz4oef55c(start, epsilon, kmax)
res4 = oefz4oef55d(start, epsilon, kmax)
Convergentiegedrag:

a) Divergent, spiraalvormig.
b) Convergentie in stap 42, spiraalvormig.
c) Convergentie in stap 17, monotoon.
d) Convergentie in stap 25, spiraalvormig.

Convergentiesnelheid:
De 3e methode (c)) convergeert het snelst.
Hoe zie je of de convergentie monotoon is?
Kijk of de getallen afwisselend groter dan het vorige en kleiner dan het vorige zijn (zo ja: spiraalvormig); kijk of elk getal groter (resp. kleiner) is dan zijn voorganger (zo ja: monotoon).
a) De divergentie is spiraalvormig.
b) De convergentie is spiraalvormig.
c) De convergentie is monotoon.
d) De convergentie is spiraalvormig.

OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 9: ITERATIEVE METHODEN VOOR HET OPLOSSEN VAN STELSELS LINEAIRE EN NIET-LINEAIRE VERGELIJKINGEN

Niet-lineaire stelsels

(1) Beschouw het stelsel

$$\begin{cases} x^2 + x - y^2 &= 1 \\ y - \sin(x^2) &= 0 \end{cases}$$

Leid formules af voor volgende iteratieve methoden:

(a) elementaire substitutiemethode (vertaal naar vaste-punt probleem)
Maak hierin onderscheid tussen totale en enkelvoudige stapmethodes.

(b) vereenvoudigde Newton-Raphson
Maak hierin onderscheid tussen totale en enkelvoudige stapmethodes.

(c) Newton-Raphson.

Lineaire stelsels

(1) Stel $A = (a_{ij})_{ij} = L + D + U$ met $\forall i : a_{ii} \neq 0$. Hierbij is L de onderdriehoeksmatrix, D de diagonaalmatrix en U de bovendriehoeksmatrix. Werk volgende iteratieformule uit:

$$Mx^{(k+1)} = Nx^{(k)} + B$$

voor de opsplitsingen van $A = M - N$:

(a) $M = D$, $N = -(L + U)$;

(b) $M = L + D$, $N = -U$.

Komen deze formules je bekend voor?

(2) Ga na of de methodes van Jacobi en Gauss-Seidel zullen convergeren voor

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{pmatrix}.$$

Hint: de convergentie wordt bepaald door de eigenwaarden van de 'iteratiematrix' $M^{-1}N$.

Extra:

Je kan dan ook eens proberen het stelsel $Ax = b$ op te lossen met

$$b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

door de methodes van Jacobi en Gauss-Seidel te implementeren in matlab.

(3) Extra oefeningen

- Beschouw het stelsel

$$\begin{cases} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 &= 2 \end{cases}$$

Illustreer voor dit stelsel grafisch de methodes van Jacobi en Gauss-Seidel met de startvector

$$X^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Convergeren de methoden? Hoe snel? Controleer je bevindingen door de eigenwaarden van de iteratiematrices te berekenen.

- Bewijs dat de methode van Jacobi altijd convergeert indien A diagonaaldominant is, wat betekent dat

$$\sum_{j \neq i} |a_{i,j}| < |a_{i,i}|, \quad i = 1, 2, \ldots, n.$$

Hint: een voldoende voorwaarde voor convergentie is dat $\|M^{-1}N\| < 1$ en kies $\|.\| = \|.\|_\infty$.

Oefenzitting 5: Iteratieve methoden voor het oplossen van stelsels lineaire en niet-lineaire vergelijkingen.

Herhaling.

Niet-lineaire vergelijkingen:

- Newton-Raphson:
  - De Jacobiaan van f:

    $J = [\partial f_i(x^{(0)})/\partial x_j]_{i,\,j=1}^{n}$

    $F(x) = x - [J(x)]^{-1} f(x)$
  - Practisch: i.p.v. $[J(x)]^{-1}$ te berekenen:

    $J(x^{(0)})h = -f(x^{(0)})$ oplossen.

    $x^{(1)} = x^{(0)} + h$
    - Stoppen als J singulier is (ev. herh. met andere startwaarde).
    - Convergentie:
      - in het algemeen kwadratisch; lineair als J singulier is.
      - Indien de startwaarde voldoende dicht bij de gezochte waarde x* ligt, zal er convergentie zijn.
  - Voorbeeld: n = 2.

    Stelsel:

    $u(x, y) = 0$

    $v(x, y) = 0$

    Formules:

    $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$

    $y^{(k+1)} = y^{(k)} + \Delta y^{(k)}$

    met $J = \begin{bmatrix} u'_x(x^{(k)}, y^{(k)}) & u'_y(x^{(k)}, y^{(k)}) \\ v'_x(x^{(k)}, y^{(k)}) & v'_y(x^{(k)}, y^{(k)}) \end{bmatrix}$

    en $J * \begin{bmatrix} \Delta x^{(k)} \\ \Delta y^{(k)} \end{bmatrix} = - \begin{bmatrix} u(x^{(k)}, y^{(k)}) \\ v(x^{(k)}, y^{(k)}) \end{bmatrix}$

    waarbij $u'_x = \partial u/\partial x$, $u'_y = \partial u/\partial y$,

    $v'_x = \partial v/\partial x$ en $v'_y = \partial v/\partial y$

    $\Delta x^{(k)} = - \dfrac{uv'_y - vu'_y}{u'_x v'_y - v'_x u'_y}$

    $\Delta y^{(k)} = - \dfrac{vu'_x - uv'_x}{u'_x v'_y - v'_x u'_y}$

- Vereenvoudigde Newton-Raphson:

  In de i^{de} vgl.:

  x_i = veranderlijk

  x_j $(j \neq i)$ = vast; stel $x_j = x_j^{(0)}$

  (Totale stapmethode;
  enkelvoudige stapmethode:
  $j < i$: $x_j = x_j^{(1)}$; $j > i$ $x_j = x_j^{(0)}$)

$f_i(x_1^{(0)}, ..., x_{i-1}^{(0)}, x_i, x_{i+1}^{(0)}, ..., x_n^{(0)}) = 0$ is een scalaire vgl. in 1 variabele.

(Enkelvoudige stapmethode: $j < i$: $x_j^{(1)}$ i.p.v. $x_j^{(0)}$)

$$x_i^{(1)} = x_i^{(0)} - \frac{f_i(x_1^{(0)}, ..., x_n^{(0)})}{\partial f_i / \partial x_i\,(x_1^{(0)}, ..., x_n^{(0)})}$$

(Enkelvoudige stapmethode: $j < i$: $x_j^{(1)}$ i.p.v. $x_j^{(0)}$)

Herhalen voor elke i; dan $x_i^{(1)}$ bewaren op de plaats van $x_i^{(0)}$ en het geheel herhalen (totale stapmethode).

(Enkelvoudige stapmethode: de nieuwe waarden worden rechtstreeks op de plaats van de vorige geschreven).

- Voordelen t.o.v. Newton-Raphson:
  - Minder rekenwerk (voor elke cyclus slechts n partieel afgeleiden i.p.v. n^2).
- Nadelen t.o.v. Newton-Raphson:
  - Tragere convergentie.
  - Ook met een goed gekozen startwaarde is convergentie niet gegarandeert; J wordt immers vervangen door zijn diagonaal.
  - Of er convergentie optreedt, hangt mede af van de volgorde van de vergelijkingen.

- Voorbeeld: n = 2:

  Stelsel:

  $u(x, y) = 0$

  $v(x, y) = 0$

  Totale stap:

  $$x^{(k+1)} = x^{(k)} - \frac{u(x^{(k)}, y^{(k)})}{u'_x(x^{(k)}, y^{(k)})}$$

  $$y^{(k+1)} = y^{(k)} - \frac{v(x^{(k)}, y^{(k)})}{v'_y(x^{(k)}, y^{(k)})}$$

  Enkelvoudige stap:

  $$x^{(k+1)} = x^{(k)} - \frac{u(x^{(k)}, y^{(k)})}{u'_x(x^{(k)}, y^{(k)})}$$

  $$y^{(k+1)} = y^{(k)} - \frac{v(x^{(k+1)}, y^{(k)})}{v'_y(x^{(k+1)}, y^{(k)})}$$

- Elementaire substitutiemethoden:

  - Stelsel van de vorm:

    $x_1 = F_1(x_1, ..., x_n)$

    ...

    $x_i = F_i(x_1, ..., x_n)$

    ...

    $x_n = F_n(x_1, ..., x_n)$

  - Totale stap:

    $x_1^{(k+1)} = F_1(x_1^{(k)}, ..., x_n^{(k)})$

    ...

    $x_i^{(k+1)} = F_i(x_1^{(k)}, ..., x_n^{(k)})$

    ...

    $x_n^{(k+1)} = F_n(x_1^{(k)}, ..., x_n^{(k)})$

  - Enkelvoudige stap:

    $x_1^{(k+1)} = F_1(x_1^{(k)}, ..., x_n^{(k)})$

    ...

    $x_i^{(k+1)} = F_i(x_1^{(k+1)}, ..., x_{i-1}^{(k+1)}, x_i^{(k)}..., x_n^{(k)})$

$$x_n^{(k+1)} = F_n(x_1^{(k+1)}, \ldots, x_{n-1}^{(k+1)}, x_n^{(k)})$$

Lineaire vergelijkingen:

- Jacobi:

  Totale stap:
  $x_j^{(k+1)} = x_j^{(k)} - (\sum_{i=1}^{n} a_{ji} x_i^{(k)} - b_j)/a_{jj}$
  (Enkelvoudige stap heet Gauss-Seidel).

  - Matrixnotatie: A = L + D + U

    $Dx^{(k+1)} = b - (L+U)x^{(k)}$
    Ax = b herschrijven naar
    Dx = - (L+U)x + b
    Dan hierop een directe substitutie toepassen.
    $x^{(k+1)} = D^{-1}[- (L+U)x^{(k)} + b]$

  - Convergentie:

    $e^{(k+1)}$ is de $(k+1)^{de}$ iteratiefout;
    $e^{(k+1)} = Ge^{(k)}$ met $G = -D^{-1}(U+L)$
    $e^{(k+1)} = G^{(k+1)}e^{(0)}$
    $\Rightarrow \| e^{(k+1)}\| \leq \|G\|^{(k+1)} \| e^{(0)}\|$
    $\Rightarrow e^{(k+1)} \dashrightarrow 0$ als $\|G\| < 1$
    $\|G\| \neq 0$ en < 1: lineaire convergentie
    $\|G\| = 0$: kwadratische convergentie
    $\|G\| > 1$: divergentie
    Als A diagonaaldominant is, is Jacobi convergent (voldoende voorwaarde, maar geen nodige vwd.; dit is ook een voldoende vwd. voor een goede conditie).

    A is diagonaaldominant <=>
    $\forall i$ (i1..n):
    $|a_{ii}| > (\sum_{j=1}^{i-1} a_{ij} + \sum_{j=i+1}^{n} a_{ij})$

- Gauss-Seidel:

  Enkelvoudige stap:
  $x_j^{(k+1)} = x_j^{(k)} -(\sum_{i=1}^{j-1} a_{ji} x_i^{(k+1)} + \sum_{i=j}^{n} a_{ji} x_i^{(k)} - b_j)/a_{jj}$
  (Totale stap heet Jacobi).

  - Matrixnotatie: A = L + D + U

    $Dx^{(k+1)} = b - Lx^{(k+1)} - Ux^{(k)}$
    Ax = b herschr. naar (L+D)x = -Ux + b
    Dan hierop een directe substitutie toepassen.
    $x^{(k+1)} = (L+D)^{-1}[-Ux^{(k)} + b]$

  - Convergentie:

    $e^{(k+1)}$ is de $(k+1)^{de}$ iteratiefout;
    $e^{(k+1)} = Ge^{(k)}$ met $G = -(L+D)^{-1}U$
    $e^{(k+1)} = G^{(k+1)}e^{(0)}$
    $\Rightarrow \| e^{(k+1)}\| \leq \|G\|^{(k+1)} \| e^{(0)}\|$
    $\Rightarrow e^{(k+1)} \dashrightarrow 0$ als $\|G\| < 1$
    $\|G\| \neq 0$ en < 1: lineaire convergentie
    $\|G\| = 0$: kwadratische convergentie
    $\|G\| > 1$: divergentie

Als A diagonaaldominant is, is Gauss-Seidel convergent (voldoende voorwaarde, maar geen nodige vwd.; dit is ook een voldoende vwd. voor een goede conditie).
A is diagonaaldominant <=>
$\forall i$ (i=1..n):
$|a_{ii}| > (\sum_{j=1}^{i-1} a_{ij} + \sum_{j=i+1}^{n} a_{ij})$

- Matrixrekenen:
  - Inverse nemen:

    A is een nxn-matrix.
    I_n is de identieke matrix met grootte nxn.
    Schrijf $[A|I_n]$.
    Pas hierop Gauss-Jordan toe (Gausseliminatie gevolgd door spillen gelijk aan 1 maken en de rest 0).
    Nu: $[I_n|A^{-1}]$ verkregen.
  - Determinant:

    $$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

    Det(A) i/h algemeen:
    kies een rij (of kolom) i.
    Neem achtereenvolgens elk element op die rij (of kolom) en vermenigvuldig het met de determinant van A zonder rij i (of kolom i) en de kolom (of rij) waarop het element staat.
    Sommeer deze resultaten; wissel dan het teken af
    (1e ele +, 2e ele -, ...).

Oplossingen van de oefeningen:

Niet-lineaire stelsels:

Oefening 1:

Stelsel:
$x^2 + x - y^2 = 1$
$y - \sin(x^2) = 0$

a) Elementaire substitutiemethoden:
Hint: zoek een iteratieformule
$F(x, y) = (F_1(x, y), F_2(x, y))$ zodanig dat het vaste punt (x^*, y^*) van $F(x, y)$ de oplossing is van het gegeven stelsel.
Omvormen stelsel:
$x = 1 - x^2 + y^2$
$y = \sin(x^2)$
Iteratieformule:
$F_1(x) = 1 - x^2 + y^2$
$F_2(y) = \sin(x^2)$
$F(x,y) = (F_1(x, y), F_2(x, y)) \leq (x, y)$
Exacte oplossing x^* schatten: zie bijlage 1)
1) Totale stap:
$x^{(k+1)} = 1 - (x^{(k)})^2 + (y^{(k)})^2$
$y^{(k+1)} = \sin((x^{(k)})^2)$
Divergentie (test zelf in matlab).

2) Enkelvoudige stap:
$x^{(k+1)} = 1 - (x^{(k)})^2 + (y^{(k)})^2$
$y^{(k+1)} = \sin((x^{(k+1)})^2)$
Convergentie (test zelf in matlab)
$y^{(k+1)} = \sin((x^{(k)})^2)$
$x^{(k+1)} = 1 - (x^{(k)})^2 + (y^{(k+1)})^2$
Convergentie (test zelf in matlab)
b) Vereenvoudigde Newton-Raphson
$u(x, y) = x^2 + x - y^2 - 1$
$v(x, y) = y - \sin(x^2)$
$u'_x(x, y) = 2x + 1$
$v'_y(x, y) = 1$
1) Totale stap:

$$x^{(k+1)} = x^{(k)} - \frac{[x^{(k)2} + x^{(k)} - y^{(k)2} - 1]}{[2x^{(k)} + 1]}$$

$$= \frac{[2x^{(k)2} + x^{(k)} - x^{(k)2} - x^{(k)} + y^{(k)2} + 1]}{[2x^{(k)} + 1]}$$

$$= \frac{[x^{(k)2} + y^{(k)2} + 1]}{[2x^{(k)} + 1]}$$

$$y^{(k+1)} = y^{(k)} - [y^{(k)} - \sin(x^{(k)2})]/1$$
$$= \sin(x^{(k)2})$$

Convergentie (test zelf in matlab).

2) Enkelvoudige stap:

$$x^{(k+1)} = \frac{[x^{(k)2} + y^{(k)2} + 1]}{[2x^{(k)} + 1]}$$

$$y^{(k+1)} = \sin(x^{(k+1)2})$$

Convergentie (test zelf in matlab).

c) Newton-Raphson:

$u(x, y) = x^2 + x - y^2 - 1$

$v(x, y) = y - \sin(x^2)$

$u'_x(x, y) = 2x + 1$

$v'_y(x, y) = 1$

$u'_y(x, y) = -2y$

$v'_x(x, y) = -2x \cos(x^2)$

$$J(x^{(k)})\, h^{(k)} = -f(x^{(k)}) \qquad h^{(k)} = \begin{bmatrix}\Delta x^{(k)}\\ \Delta y^{(k)}\end{bmatrix}$$

$$[u'_x(x^{(k)}, y^{(k)}) \quad u'_y(x^{(k)}, y^{(k)})][\Delta x^{(k)}] = -[u(x^{(k)}, y^{(k)})]$$

$$[v'_x(x^{(k)}, y^{(k)}) \quad v'_y(x^{(k)}, y^{(k)})][\Delta y^{(k)}] = -[v(x^{(k)}, y^{(k)})]$$

Oplossen naar $\Delta x^{(k)}$ en $\Delta y^{(k)}$, dan: (*)

$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$

$y^{(k+1)} = y^{(k)} + \Delta y^{(k)}$

$$J = \begin{bmatrix} 2x + 1 & -2y \\ -2x\cos(x^2) & 1 \end{bmatrix}$$

(Regel van Cramer toepassen).

$$\Delta x^{(k)} = \frac{\begin{vmatrix} -u & -2y \\ -v & 1 \end{vmatrix}}{(2x+1) - 4xy\cos(x^2)}$$

$$= \frac{\begin{vmatrix} -x^2 - x + y^2 + 1 & -2y \\ -y + \sin(x^2) & 1 \end{vmatrix}}{(2x+1) - 4xy\cos(x^2)}$$

$$= \frac{-x^2 - x + y^2 + 1 - (2y^2 - 2y\sin(x^2))}{(2x+1) - 4xy\cos(x^2)}$$

$$= \frac{-x^2 - x + y^2 + 1 - 2y^2 + 2y\sin(x^2)}{2x+1 - 4xy\cos(x^2)}$$

$$= \frac{-x^2 - x - y^2 + 1 + 2y\sin(x^2)}{2x+1 - 4xy\cos(x^2)}$$

$$\Delta y^{(k)} = \begin{vmatrix} 2x+1 & -x^2 - x + y^2 + 1 \\ -2x\cos(x^2) & -y + \sin(x^2) \end{vmatrix}$$

$$(2x+1) - 4xy\cos(x^2)$$

$$= \frac{-2xy +2x\sin(x^2) -y +\sin(x^2) -(2x^3\cos(x^2) +2x^2\cos(x^2) -2x\cos(x^2) -2xy^2\cos(x^2))}{(2x+1) - 4xy\cos(x^2)}$$

$$= \frac{-2xy +2x\sin(x^2) -y +\sin(x^2) -2x^3\cos(x^2) -2x^2\cos(x^2) +2x\cos(x^2) +2xy^2\cos(x^2))}{(2x+1) - 4xy\cos(x^2)}$$

Invullen in (*) geeft de oplossing.
Ook hier kan weer totale stap (divergent) of enkelvoudige stap gebruikt worden (convergent).
Ga de convergentie zelf na in matlab.

Opmerking:

$$\Delta x^{(k)} = \frac{\begin{vmatrix} -u & u'_y \\ -v & v'_y \end{vmatrix}}{\begin{vmatrix} u'_x & u'_y \\ v'_x & v'_y \end{vmatrix}} = \frac{-uv'_y + vu'_y}{u'_x v'_y - v'_x u'_y}$$

$$\Delta y^{(k)} = \frac{\begin{vmatrix} u'_x & -u \\ v'_x & -v \end{vmatrix}}{\begin{vmatrix} u'_x & u'_y \\ v'_x & v'_y \end{vmatrix}} = \frac{-u'_x v + v'_x u}{u'_x v'_y - v'_x u'_y}$$

Regel van Cramer:

Als $\det(A) \neq 0$, dan is de oplossing $\underline{x}$ van het stelsel lin. vgln. $A\underline{x} = \underline{b}$ gegeven door:

$x_i = \det(A_i)/\det(A)$

met A_i = A waarin kolom i vervangen is door de vector b.

Lineaire stelsels:

Oefening 1:

a) $Dx^{(k+1)} = -(L+U)x^{(k)} + \underline{b}$

stap 1: $Dx^{(k+1)}$ uitwerken.

$$\begin{bmatrix} a_{11} & 0 & & & 0 \\ 0 & a_{22} & 0 & & 0 \\ | & \backslash & \backslash & \backslash & | \\ 0 & 0 & a_{n-1n-1} & 0 & \\ 0 & --- & 0 & a_{nn} & \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \dots \\ x_{n-1}^{(k+1)} \\ x_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} a_{11}\, x_1^{(k+1)} \\ a_{22}\, x_2^{(k+1)} \\ \dots \\ a_{nn}\, x_n^{(k+1)} \end{bmatrix}$$

stap 2: $(L+U)x^{(k)}$ uitwerken.

$$\begin{bmatrix} 0 & a_{12} & & & a_{1n} \\ a_{21} & 0 & a_{23} & & a_{2n} \\ | & \backslash & \backslash & \backslash & | \\ a_{(n-1)1} & a_{(n-1)(n-2)} & 0 & a_{(n-1)(n-1)} & \\ a_{n1} & & & a_{n(n-1)} & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \dots \\ x_{n-1}^{(k)} \\ x_n^{(k)} \end{bmatrix} =$$

$$\left[\sum_{j=2}^{n} a_{1j}x_1^{(k)}; \sum_{j=1,j\neq 2}^{n} a_{2j}x_2^{(k)}; \dots; \sum_{j=1}^{(n-1)} a_{nj}x_n^{(k)}\right]$$

stap 3: i^{de} vergelijking beschouwen:

$$a_{ii}\, x_i^{(k+1)} = -\sum_{j=1,\, j\neq i}^{n} a_{ij}x_j^{(k)} + b_i$$
$$= a_{ii}\, x_i^{(k)} - \sum_{j=1}^{n} a_{ij}x_j^{(k)} + b_i$$

$$x_i^{(k+1)} = x_i^{(k)} + \left[-\sum_{j=1}^{n} a_{ij}x_j^{(k)} + b_i\right]/a_{ii}$$

Jacobi.

b) $(L + D)x^{(k+1)} = -U\, x^{(k)} + \underline{b}$

stap 1: $(L+D)x^{(k+1)}$ uitwerken.

$$\begin{bmatrix} a_{11} & 0 & & 0 \\ a_{12} & a_{22} & 0 & 0 \\ | & \backslash & \backslash & | \\ a_{1n-1} & a_{n-1n-1} & 0 & \\ a_{1n} & & a_{nn} & \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \dots \\ x_{n-1}^{(k+1)} \\ x_n^{(k+1)} \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{j=1}^{1} a_{1j}x_1^{(k)} \\ \sum_{j=1}^{2} a_{2j}x_2^{(k)} \\ \dots \\ \sum_{j=1}^{n-1} a_{(n-1)j}x_{(n-1)}^{(k)} \\ \sum_{j=1}^{n} a_{nj}x_n^{(k)} \end{bmatrix}$$

stap 2: $(U)x^{(k)}$ uitwerken.

$$\begin{bmatrix} 0 & a_{12} & & a_{1n} \\ 0 & 0 & a_{23} & a_{2n} \\ | & \backslash & \backslash & | \\ 0 & & 0 & a_{(n-1)n} \\ 0 & & & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \dots \\ x_{n-1}^{(k)} \\ x_n^{(k)} \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{j=2}^{n} a_{1j}x_1^{(k)} \\ \sum_{j=3}^{n} a_{2j}x_2^{(k)} \\ \dots \\ \sum_{j=n}^{n} a_{(n-1)j}x_{(n-1)}^{(k)} \\ 0 \end{bmatrix}$$

stap 3: i^{de} vergelijking beschouwen:

$$\sum_{j=1}^{i} a_{ij}x_j^{(k+1)} = -\sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} + \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} = -\sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} = -\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} = -\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - a_{ii}x_i^{(k)} + a_{ii}x_i^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} = a_{ii}x_i^{(k)} - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - a_{ii}x_i^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} = a_{ii}x_i^{(k)} - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)} + b_i$$
$$a_{ii}x_i^{(k+1)} = a_{ii}x_i^{(k)} - \left[\sum_{j<i} a_{ij}x_j^{(k+1)} + \sum_{j\geq i} a_{ij}x_j^{(k)}\right] + b_i$$
$$x_i^{(k+1)} = x_i^{(k)} - \left(\left[\sum_{j<i} a_{ij}x_j^{(k+1)} + \sum_{j\geq i} a_{ij}x_j^{(k)}\right] + b_i\right)/a_{ii}$$

<u>Gauss-Seidel.</u>

<u>Oefening 2:</u>

Exacte oplossing: zie bijlage 2.

a) Jacobi:

$G = M^{-1}N = -D^{-1}(L+U)$

Matrices uitrekenen en vermenigvuldigen:

$D = 3I_3$ (met I_3 de eenheidsmatrix)

$D^{-1} = (1/3)I_3$

$L + U = [0\ 2\ 2;\ 2\ 0\ 2;\ 2\ 2\ 0]$

G berekenen:

$$G = -(1/3)I_3\ [0\ 2\ 2;\ 2\ 0\ 2;\ 2\ 2\ 0] = \begin{bmatrix} 0 & (-2/3) & (-2/3) \\ (-2/3) & 0 & (-2/3) \\ (-2/3) & (-2/3) & 0 \end{bmatrix}$$

Eigenwaarden zoeken:

λ is eigenwaarde <=>

$$\det \begin{bmatrix} 0-\lambda & (-2/3) & (-2/3) \\ (-2/3) & 0-\lambda & (-2/3) \\ (-2/3) & (-2/3) & 0-\lambda \end{bmatrix} = 0$$

$R_1 <-- R_1 + R_2 + R_3$

$$\det \begin{bmatrix} -\lambda-(4/3) & -\lambda-(4/3) & -\lambda-(4/3) \\ (-2/3) & -\lambda & (-2/3) \end{bmatrix} = 0$$

$$\begin{bmatrix}(-2/3) & (-2/3) & -\lambda \end{bmatrix}$$

-1 voorop zetten.

-1 mag weg, want det = 0

$$\det -1 \begin{bmatrix} \lambda+(4/3) & \lambda+(4/3) & \lambda+(4/3) \\ (2/3) & \lambda & (2/3) \\ (2/3) & (2/3) & \lambda \end{bmatrix} = 0$$

Ontwikkelen naar 1e rij:

$$\lambda+(4/3)\left(\det \begin{bmatrix}\lambda & (2/3)\\ (2/3) & \lambda\end{bmatrix} + \det \begin{bmatrix}(2/3) & (2/3)\\ (2/3) & \lambda\end{bmatrix} + \det \begin{bmatrix}(2/3) & \lambda \\ (2/3) & (2/3)\end{bmatrix}\right) = 0$$

$\Leftrightarrow \lambda = -(4/3)$ of

$\lambda^2 +(4/9) +(4/9) -(2/3)\lambda -(2/3)\lambda -(4/9) = 0$

$\Leftrightarrow \lambda = -(4/3)$ of

$\lambda^2 -(4/3)\lambda +(4/9)= 0$

$D = 16/9 - 16/9 = 0$

$\lambda = (4/3)/2 = 2/3$

$\Leftrightarrow \lambda = -(4/3)$ of $\lambda = (2/3)$

λ_{max} v. G berek. en kijken of $|\lambda_{max}| < 1$:

$\lambda_{max} = -4/3$; $|-4/3| > 1$

=> geen convergentie.

(zelf nagaan in matlab)

b) Gauss-Seidel:

$G = -(L+D)^{-1}U$

Matrices uitrekenen en vermenigvuldigen:

$D = 3I_3$ (met I_3 de eenheidsmatrix)

$L = [0\ 0\ 0;\ 2\ 0\ 0;\ 2\ 2\ 0]$

$(L + D) = [3\ 0\ 0;\ 2\ 3\ 0;\ 2\ 2\ 3]$

$$(L + D)^{-1} = \begin{bmatrix} (1/3) & 0 & 0 \\ -(2/9) & (1/3) & 0 \\ -(2/27) & -(2/9) & (1/3) \end{bmatrix}$$

$U = [0\ 2\ 2;\ 0\ 0\ 2;\ 0\ 0\ 0]$

G berekenen:

$$G = \begin{bmatrix} (1/3) & 0 & 0 \\ -(2/9) & (1/3) & 0 \\ -(2/27) & -(2/9) & (1/3) \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -(2/3) & -(2/3) \\ 0 & (4/9) & -(2/9) \\ 0 & (4/27) & (16/27) \end{bmatrix}$$

Eigenwaarden zoeken:

λ is eigenwaarde <=>

$$\det \begin{bmatrix} 0-\lambda & -(2/3) & -(2/3) \\ 0 & (4/9)-\lambda & -(2/9) \\ 0 & (4/27) & (16/27)-\lambda \end{bmatrix} = 0$$

Ontwikkelen naar eerste rij:

$$\begin{aligned}
&<=> -\lambda \det[4/9-\lambda \quad 2/9;\ 4/27 \quad 16/27-\lambda] \\
&\qquad + 2/3 \det[0 \ -2/9;\ 0 \ 16/27-\lambda] \\
&\qquad - 2/3 \det[0 \quad 4/9-\lambda;\ 0 \quad 4/27] = 0 \\
&\qquad \text{Laatste 2 termen vallen weg, want} \\
&\qquad \text{de determinanten zijn 0 daar.} \\
&<=> -\lambda \det[4/9-\lambda \quad 2/9;\ 4/27 \quad 16/27-\lambda] = 0 \\
&<=> -\lambda((64/243) + \lambda^2 -(16/27)\lambda -(4/9)\lambda -(8/243)) = 0 \\
&<=> -\lambda(\lambda^2 -(28/27)\lambda +(56/243)) = 0 \\
&\qquad D = (784/729) - (224/243) \\
&\qquad\quad = (784 - 672)/729 \\
&\qquad\quad = 112/729 = (2^4 7)/(3^6) \\
&\qquad \lambda = (14/27) \pm [(2^2/3^3)\sqrt{7}]/2 \\
&\qquad\quad = (14/27) \pm (2/3^3)\sqrt{7} \\
&\qquad\quad = (14/27) \pm (2/27)\sqrt{7} \\
&<=> -\lambda = 0 \text{ of } \lambda = (14/27) \pm (2/27)\sqrt{7} \\
&<=> \lambda = 0 \text{ of } \lambda = (14/27) \pm (2/27)\sqrt{7} \\
&<=> \lambda = 0 \\
&\qquad \text{of } \lambda = 0{,}32253694 \\
&\qquad \text{of } \lambda = 0{,}7145001
\end{aligned}$$

λ_{max} v. G berek. en kijken of $|\lambda_{max}| < 1$:
$\lambda_{max} = 0{,}7145001$; $|0{,}7145001| < 1$
=> convergentie.
Zelf de convergentie nagaan in matlab.

<u>Oefening 3:</u>

a) Grafisch illustreren van Jacobi en Gauss-Seidel:

$A = [2 \ -1;\ -1 \ 2]$
$\underline{b} = [1;\ 2]$
$x^{(0)} = [0;\ 0]$
$D = [2 \ 0;\ 0 \ 2]$
$L = [0 \ 0;\ -1 \ 0]$
$U = [0 \ -1;\ 0 \ 0]$

Jacobi:

$G_j = -D^{-1}(U + L)$
$D^{-1} = [(1/2) \ 0;\ 0 \ (1/2)]$
$(U + L) = [0 \ -1;\ -1 \ 0]$
$G_j = [0 \ (1/2);\ (1/2) \ 0]$
$||G_j|| = 1/2 < 1$
=> convergentie.

Gauss-Seidel:

$G_{gs} = -(L + D)^{-1}U$

$(L + D) = [2\ 0; -1\ 2]$

$(L + D)^{-1} = [(1/2)\ 0; (1/4)\ (1/2)]$

$G_{gs} = [0\ (1/2); 0\ (1/4)]$

$||G_{gs}|| = 0{,}55901699437... < 1$

=> convergentie.

Convergentiesnelheid:

$||G_j|| = 1/2 \neq 0$

=> Jacobi convergeert lineair.

$||G_{gs}|| = 0{,}55901699437... \neq 0$

=> Gauss-Seidel convergeert lineair.

Eigenwaarden:

Jacobi:

$G = -D^{-1}(L+U) = G_j$ (zie hoger)

$G = G_j = [0\ (1/2); (1/2)\ 0]$

λ is eigenwaarde <=>

$\det[0-\lambda\ (1/2); (1/2)\ 0-\lambda] = 0$

<=> $\lambda^2 - 1/4 = 0$

<=> $\lambda^2 = 1/4$

<=> $\lambda = 1/2$ of $-(1/2)$

$|\lambda_{max}| = 0{,}5 < 1$

=> convergentie.

Gauss-Seidel:

$G = -(L+D)^{-1}U = G_{gs}$ (zie hoger)

$G = G_{gs} = [0\ (1/2); 0\ (1/4)]$

λ is eigenwaarde <=>

$\det[0-\lambda\ (1/2); 0\ (1/4)-\lambda] = 0$

<=> $\lambda^2 - 1/8 = 0$

<=> $\lambda^2 = 1/8$

<=> $\lambda = \text{sqrt}(1/8)$ of $-\text{sqrt}(1/8)$

<=> $\lambda = 1/(2\sqrt{2})$ of $-1/(2\sqrt{2})$

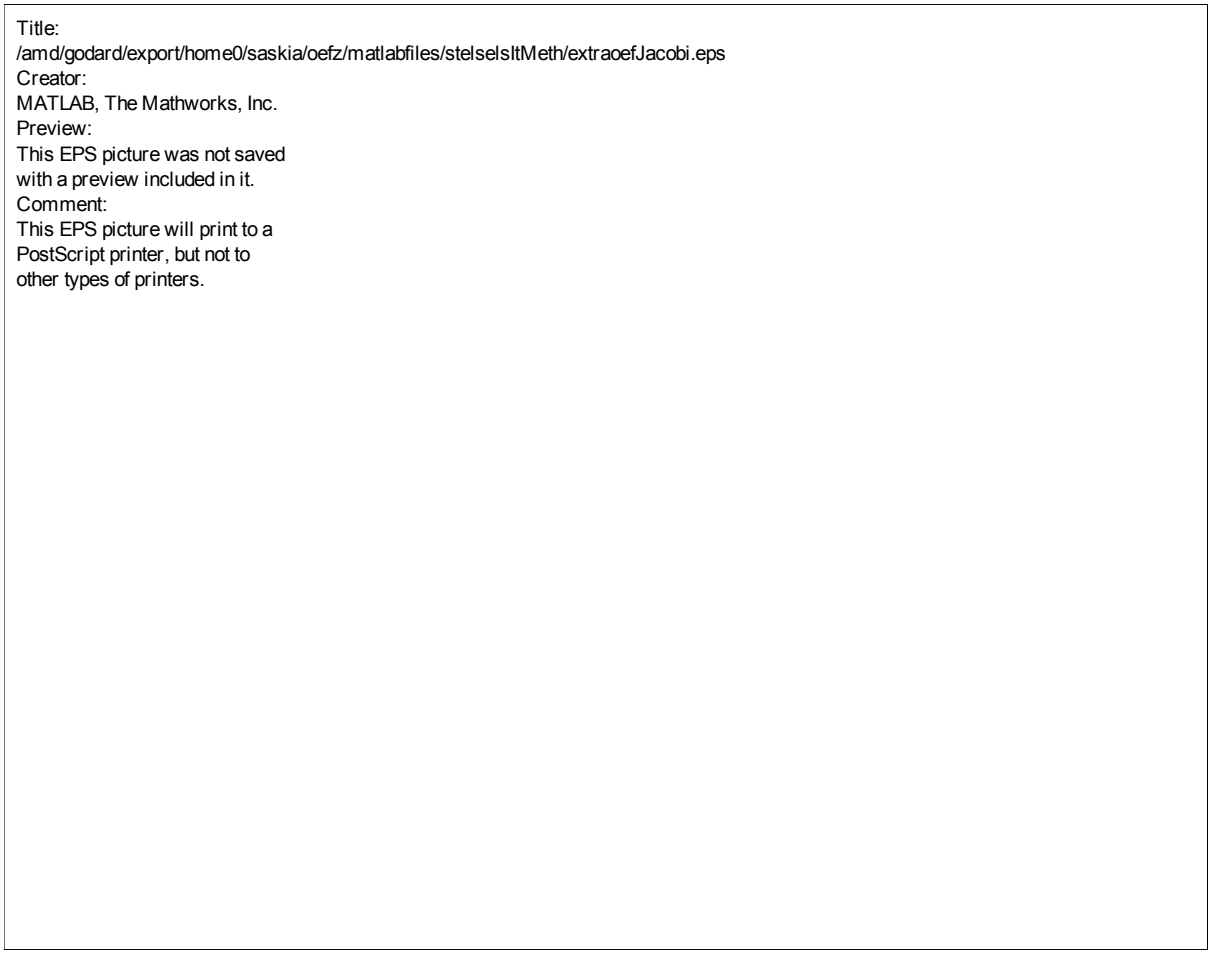
<=> $\lambda = 0{,}35355339$ of $-0{,}35355339$

$|\lambda_{max}| = 0{,}35355339 < 1$

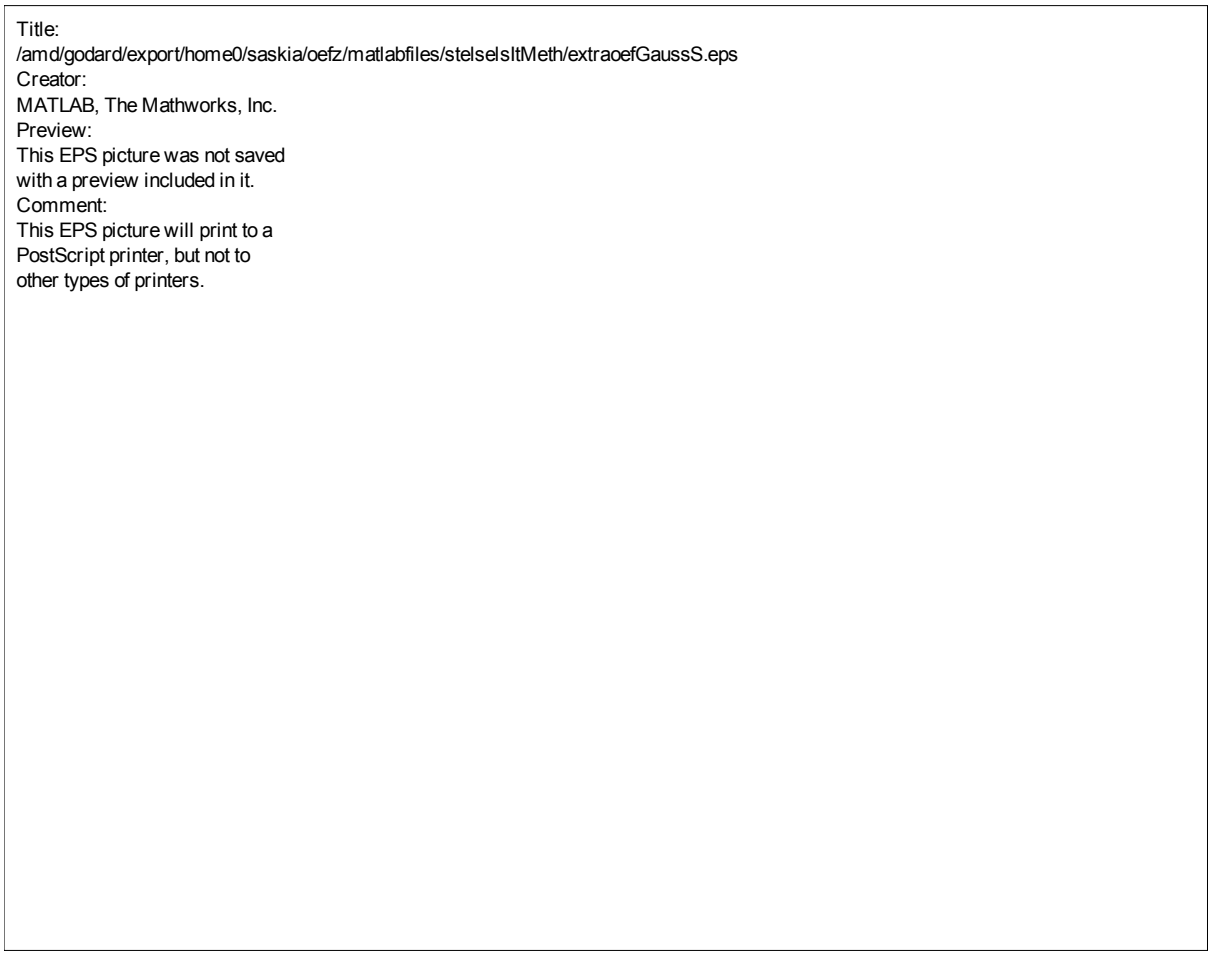
=> convergentie.

Grafisch:

Jacobi:



Gauss-Seidel:



b) Jacobi is steeds convergent als A diagonaaldominant is:

Voldoende voorwaarde voor convergentie:
$||M^{-1}N|| < 1$
A = (L + D + U), een vierkante matrix van orde n
M = D;
N = -(L + U)
Dus: convergentie als $||D^{-1}[-(L+U)]|| < 1$
D is een diagonaalmatrix (d_{ij}) van orde n
noem diag(D) d;
D^{-1} is ook een diagonaalmatrix;
noem diag(D^{-1}) d2
Nu geldt: $\forall i=1..n$: d2(i) = 1/d(i).
Noem [-(L+U)] Som = (Som_{ij})
Noem (D^{-1}Som) Product = $(Product_{ij})$.
$\forall i=1..n$: $\forall j=1..n$:
$Product_{ij} = 0$ als i = j
$Product_{ij} = Som_{ij}/d(i)$ als $i \neq j$
Omdat $|d(i)| > \sum_{j=1,\, j\neq i}^{n} |Som_{ij}|$ is ook:
$|d(i)| > |Som_{ij}|$.
Dus: $|Product_{ij}| < 1$ $\forall i=1..n$: $\forall j=1..n$
Neem $||Product_{ij}|| = ||Product_{ij}||_{\infty}$:
$||Product_{ij}|| = \max_{1\leq i\leq n}|Product_{ij}| < 1$.
Dus: $||M^{-1}N|| < 1$ en convergentie.

(q.e.d.)

Bijlagen:

1) exacte oplossing van het stelsel niet-lineaire vergelijkingen uit oef.1 schatten:
Stelsel:
$x^2 + x - y^2 = 1$
$y - \sin(x^2) = 0$

$y^2 = x^2 + x - 1$
$y = \sin(x^2)$

$y = \pm \text{sqrt}(x^2 + x - 1)$
Dus: $x^2 + x - 1$ moet een opl. ≥ 0 hebben.
$D = 1 + 4 = 5$
$x = (-1 \pm \sqrt{5})/2$
$= -1/2 \pm \sqrt{5}/2$
Dus: $x \leq -1/2 - \sqrt{5}/2$ of $x \geq -1/2 + \sqrt{5}/2$
Dus: $x \leq -1.61803399$ of $x \geq 0.618034$
Afgeleide van $(x^2 + x - 1)$: $2x + 1$
$x = -1/2$ is een extremum (minimum)
van $(x^2 + x - 1)$ want $f''(x) = 2 > 0$.
Dus: $0 \leq y$

$y = \sin(x^2)$ ($\Rightarrow x^2 = \text{Bgsin}(y)$
$\Rightarrow x = \text{sqrt}(\text{Bgsin}(y))$
Dus: $0 \leq y \leq \pi/2$ (verloop Bgsin)
Dus: $0 \leq x \leq \text{sqrt}(\text{Bgsin}(\pi/2))$
Dus: $0 \leq x \leq 1.256980$
$0 \leq y \leq \pi/2$

2) Exacte oplossing van oef. 2 (lin. vgln.):

```
[3 2 2 | 1]          [1 -1 0 | 0]          [1 -1 0 | 0]
[2 3 2 | 1]   -->    [2  3 2 | 1]   -->    [0  5 2 | 1]   -->
[2 2 3 | 1]          [2  2 3 | 1]          [0  4 3 | 1]

[1 -1  0 | 0]        [1 -1  0 | 0]         [1 -1  0 |     0]
[0  1 -1 | 0]  -->   [0  1 -1 | 0]   -->   [0  1 -1 |     0]
[0  4  3 | 1]        [0  0  7 | 1]         [0  0  1 | (1/7)]

        [1 -1 0 |     0]          [1 0 0 | (1/7)]
-->     [0  1 0 | (1/7)]  -->     [0 1 0 | (1/7)]
        [0  0 1 | (1/7)]          [0 0 1 | (1/7)]
```

Dus: $\underline{x}$ = [(1/7); (1/7); (1/7)] is de exacte oplossing.

pc-zitting 5: Het iteratief oplossen van stelsels lineaire en niet-lineaire vergelijkingen

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Inleiding

Voor het uitwerken van de opgaven moet je '.m'-bestanden gebruiken die je kan vinden in op

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

1 Demo

Op het einde van de oefenzitting zal een demo gegeven worden om het nut van iteratieve methodes aan te tonen voor het oplossen van grote spaarse stelsels lineaire vergelijkingen: de 2D- warmtevergelijking. De volgende twee oefeningen zijn slechts driedimensionaal maar geconstrueerd om U (uiteraard) meer inzicht te verschaffen in het verband tussen convergentie en de eigenstructuur van de iteratiematrices.

2 Lineaire stelsels

2.1 Jacobi $\longleftrightarrow$ Gauss-Seidel

Gebruik de methodes van Jacobi en Gauss-Seidel, geïmplementeerd in resp. `jacobi.m` en `gs.m`, om het volgende stelsel op te lossen. Gebruik als startwaarden $x_1 = x_2 = x_3 = 1$.

$$\begin{cases} 4x_1 - x_2 = 1 \\ -x_1 + 4x_2 - x_3 = 0 \\ - x_2 + 4x_3 = 1 \end{cases}$$

Ga na welke methode het snelst convergeert en vergelijk dit met de spectraalradius van de iteratiematrix.
Verklaring van de parameters van de Matlab-routines:

- a = de matrix van het stelsel
- b = het rechterlid van het stelsel
- x0 = een vector met de startwaarden
- xe = de exacte oplossing, indien je een meer betrouwbaar beeld van de fout wil dan zou volgen uit de residus (Verklaar!)
- n = het aantal iteratiestappen

Ga in de code zelf na wat de twee outputparameters betekenen.

2.2 Gevoeligheid voor initiële condities?

Doe dertig jacobi-iteraties om het volgende stelsel op te lossen:

$$\left\{\begin{array}{rcrcrcr} 3x_1 & - & 2x_2 & + & 2x_3 & = & 3 \\ -2x_1 & + & 3x_2 & - & 2x_3 & = & -1 \\ 2x_1 & - & 2x_2 & + & 3x_3 & = & 3 \end{array}\right.$$

Neem als startwaarden $\left[\begin{array}{ccc} 3 & 2 & 0 \end{array}\right]^T$ en $\left[\begin{array}{ccc} 3.0001 & 2 & 0 \end{array}\right]^T$. Verklaar wat er gebeurt? Hint: wat zegt dit over de eigenstructuur van de iteratiematrix?

2.3 Het zoeken van een optimale ω voor SOR

Gauss-Seidel convergeert zeer traag indien de spectraalradius $\rho(M_{GS}^{-1}N_{GS})$ dicht bij 1 ligt. In dat geval kan een kleine wijziging aan de methode van Gauss-Seidel soms leiden tot snellere convergentie. Beschouw daartoe het volgende schema:

$for\ i = 1 : n$

$$x_i^{(k+1)} = \omega \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii} + (1-\omega) x_i^{(k)}$$

end

Dit noemt men de SOR-methode (Successive Over-Relaxation). Bewijs dat dit overeenkomt met het volgende iteratieve schema:

$$M_\omega x^{(k+1)} = N_\omega x^{(k)} + b$$

met $M_\omega = L + \frac{D}{\omega}$ en $N_\omega = (\frac{1-\omega}{\omega})D - U$.
ω noemt men de relaxatieparameter die steeds moet voldoen aan $0 < \omega < 2$. Er bestaat steeds een optimale ω, dus een ω die de spectraalradius minimaal maakt, maar in de meeste gevallen is deze optimale waarde zeer moeilijk of niet analytisch te berekenen.
Implementeer de SOR-methode en zoek de optimale ω (met één cijfer na de komma) voor een stelsel met als matrix de hilbertmatrix van dimensie 3. Deze matrix kan je genereren met het commando `hilb(3)`. Vergelijk de resultaten met Gauss-Seidel.

3 Niet-lineaire stelsels

3.1 Oplossen van niet-lineaire stelsels

Implementeer de methode van Newton-Raphson om de complexe nulpunten van onderstaande vergelijkingen te vinden. Beschouw een complex getal $z = x + iy$ als twee variabelen en genereer een stelsel.

$$(1)\ \ 2z^2 + z + 1 = 0$$

$$(2)\ \ z - e^z = 0$$

3.2 Uitbreiding van Newton-Raphson voor het zoeken van complexe nulpunten van een niet-lineare vergelijking

Wanneer de complexe functie $f : \mathbb{C} \to \mathbb{C}$ analytisch is, zal de iteratieformule

$$z^{(k+1)} = z^{(k)} - \frac{f(z^{(k)})}{f'(z^{(k)})}, \quad z^{(0)} \in \mathbb{C} \tag{1}$$

dezelfde eigenschappen behouden als de Newton-Raphson formule voor reële nulpunten. Formule (1) kan direct geïmplementeerd worden omdat matlab complex rekent. Zoek de complexe wortels van de functies uit de vorige opgaven met behulp van deze methode. Merk wel dat je daarbij alleen naar een complex nulpunt kan convergeren indien de startwaarde complex is. Bewijs m.b.v. de Cauchy-Riemann voorwaarden dat beide methoden in feite identiek zijn.

4 Extra oefeningen

4.1 Nulpunten van $x + \log(x)$

Tijdens vorige zittingen werd aangetoond de iteratie-formule

$$x^{(k+1)} = -\log(x^{(k)})$$

niet convergeert naar de reële oplossing 0.5671 van $x + \log(x) = 0$. Doe een (groot) aantal iteraties met $x^{(0)} = 0.3$ en verklaar wat er gebeurt? Hint: wat is het domein van de logaritmische functie in $\mathbb{C}$, waar is ze analytisch?

4.2 Kiezen van startwaarden

Met formule (1) kan men theoretisch alle n nulpunten van een willekeurige veelterm $p(x)$ van graad n berekenen. Daarbij heeft men echter goede startwaarden nodig (anders kan de methode divergeren of vanuit verschillende startwaarden naar hetzelde nulpunt convergeren). Men kan dit probleem omzeilen via een voortzettingsmethode: wanneer $q(x)$ een n-de graadsveelterm is met gekende wortels, kan men de wortels volgen van de veelterm

$$z(x) = \lambda p(x) + (1 - \lambda)q(x)$$

waarbij men de parameter λ in kleine stapjes varieert van 0 tot 1. Waarom werkt deze methode? Wat gebeurt er indien het aantal reële nulpunten van $p(x)$ en $q(x)$ niet hetzelfde is? Welke moeilijkheden verwacht je bij de implementatie? Een andere methode om alle wortels te bepalen van een veelterm is gebaseerd op deflatie, het wegdelen van een gevonden wortel. Welke methode is het stabielst?

PC-zitting 5: stelsels iteratief oplossen.

Opmerkingen:
De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~saskia/oefz
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

NIET maken: Oef. 4, Oef. 5

Oplossingen van de oefeningen:

Oefening 1: demo Laplace.

Opmerkingen:
Zie ook: http://www.cs.kuleuven.ac.be/~ade/WWW/NW/
> Illustraties > Convectie-diffusievergelijking).
Zie ook: http://www.cs.utah.edu/~zachary/isp/applets/Plotter/Plotter.html).
Herhaling: Fysisch probleem: (zie pc zitting 4)

1) Transiente 1D warmtevergelijking.

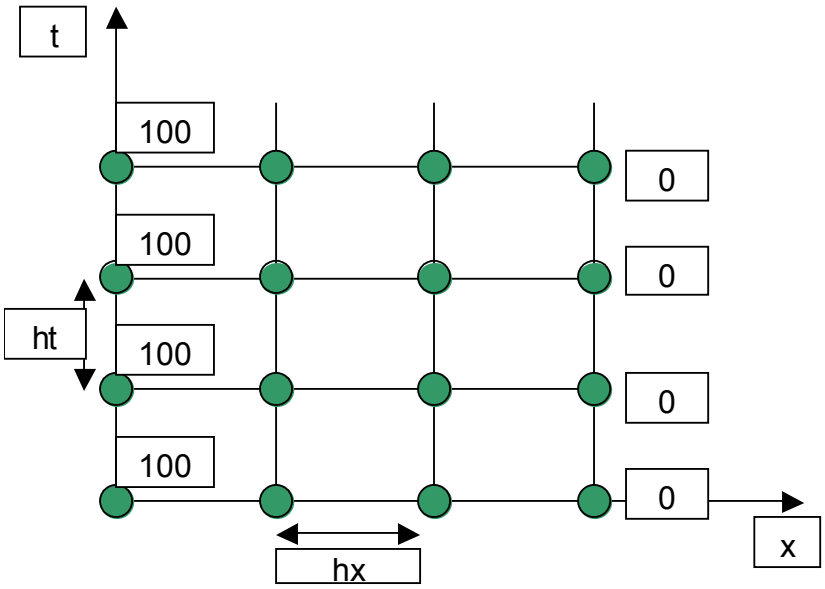
Er is een staaf met een linkeruiteinde (LU) en een rechteruiteinde (RU).
t = de tijd en T = de temperatuur.
100ºC x ----> 0ºC
L
Voor $t \leq 0$: de hele staaf heeft de temperatuur van de omgeving, T_{omg}.
Vanaf t = 0: Aan LU wordt een warmtebron geplaatst (van 100 graden), en aan RU wordt een koudebron geplaatst (van 0 graden).
De warmte (en de koude) gaan zich verspreiden over de hele staaf, en dus de temperatuur [illegible] evenwichtstoestand is bereikt.

2) Stationaire 2D warmtevergelijking:

We kijken enkel naar $t = \infty$ voor de warmtetoestand van een plaat waarbij de randen op een constante temperatuur gehouden worden ($\partial T/\partial t = 0$).

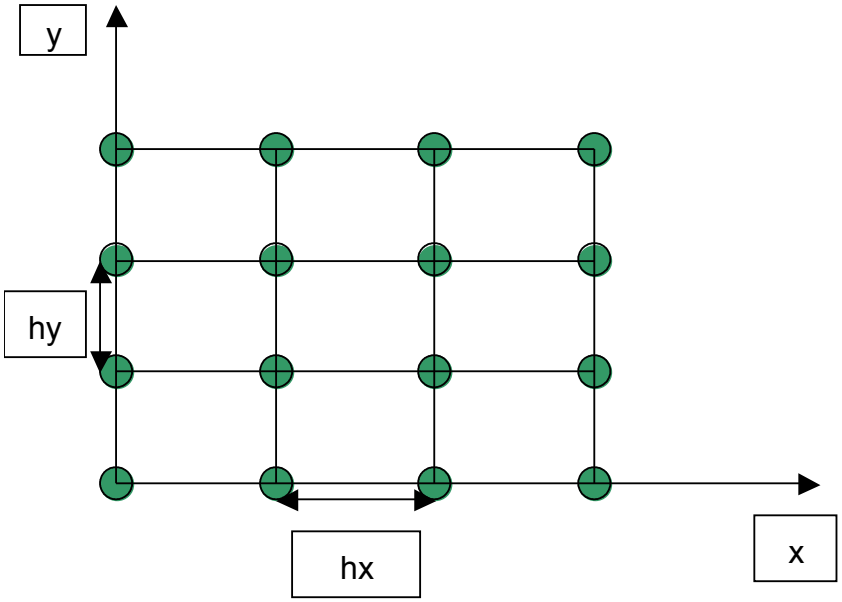
Model voor het probleem:
1) Warmtevergelijking: $\rho * c * \partial T/\partial t = k * \nabla^2 T$,
waarbij: ρ de massadichtheid
c de soortelijke warmte
$\partial T/\partial t$ de verandering van de temperatuur in de tijd
k de warmtegeleidingscoëfficiënt
$\nabla^2 T$ de Laplaciaan van de temperatuur
Opm.: de temperatuur is een functie van de tijd en van de ruimte; daarom gebruiken we de partiële afgeleide in de warmtevergelijking.
De Laplaciaan $\nabla^2 T = \partial^2 T/\partial x^2$, de tweede afgeleide van de temperatuur naar de ruimte.
De oplossing van deze vergelijking kan berekend worden door zowel de ruimte als de tijd te discretiseren (d.w.z. indelen in stapjes).

Op die manier verkrijg je een aantal roosterpunten die berekend kunnen worden in een stelsel.



(∇^2T in het punt $T_i = (T_{i-1} - T_{i+1})/h_x^2$).
($\partial T/\partial x$ in het punt $T_{i-0,5} = (T_{i-1} - T_i)/h_x$)
($\partial T/\partial x$ in het punt $T_{i+0,5} = (T_i - T_{i+1})/h_x$)
(∂^2T/∂^2x in het punt T_i = de afgeleide nemen van de eerste afgeleide, dus de afgeleiden in i-0,5 en i+0,5 van elkaar aftrekken en delen door h_x, de stapgrootte in de ruimte).
Het rechterlid voor het stelsel kan bekomen worden door de punten aan de rand van de staaf in te vullen (van deze punten is de temperatuur gekend vanaf t = 0).
Vb. van een vgl. in het stelsel:
$\rho*c*((T_5 - T_2)/h_t) = k*((T_3 - T_1)/h_x^2)$
2) Warmtevergelijking: $\nabla^2T = 0$.
$\nabla^2T = \partial^2T/\partial x^2 + \partial^2T\partial y^2$

De oplossing van deze vergelijking kan berekend worden door zowel de ruimte in x als de ruimte in y discretiseren (d.w.z. indelen in stapjes). (Een plaat heeft een x- en een y-dimensie).
Op die manier verkrijg je opnieuw een aantal roosterpunten die berekend kunnen worden in een stelsel.
In een punt (i,j):
$((T_{i+1,j} + T_{i-1,j})/h_x^2) + ((T_{i,j+1} + T_{i,j-1})/h_y^2) = 0$



Voor een plaat zijn er meerdere modellen geïmplementeerd:

_ 1kant van de plaat warm, de tegenoverliggende kant koud en ertussen lineaire overgangen.

_ warmte aan de hoek rechtsonder, warmte aan de linkerbovenhoek en beide andere hoeken koud, met overgangen ertussen.

_ Warme linkeronderhoek en de 3 andere hoeken koud.

Ook hier kunnen a.h.v. de randpunten alle andere punten berekend worden.

Resultaten:

1) De grafiek toont de evolutie in de tijd van de verspreiding van de warmte. Van een grafiek met aan beide uiteinden een piek evolueert de warmtetoestand naar een lineaire grafiek (rechte).
Niet alle parameters mogen willekeurig gekozen worden om een grafiek te krijgen die in realiteit mogelijk is! Daarom laat het bestand demo.m enkel toe het aantal stappen in de tijd en de initiële temperatuur te kiezen.

2) De eerste stap is de gausseliminatie van het stelsel; typisch hiervoor is dat het aantal van 0 verschillende elementen gaat stijgen gedurende de eliminatiefase.
De structuur van het stelsel is typisch voor een differentiatieprobleem: een groot aantal onbekenden (orde n^m waarbij n het aantal discretisatiestappen voor x is en m het aantal discretisatiestappen voor y) (en dus veel vgln.), en een spaarse matrix.

Hoe bekomen:

1) demo.m gebruiken:
y = demo(aantal_tijdsstappen, init_temp)
Vb.: demo(5, 50)

2) gaussel.m gebruiken:
opl = gaussel(n, m, soortrandvwd, plot)
n = aantal stappen voor x
m = aantal stappen voor y
soortrandvwd = 1, 2 of 3 en bepaalt welke hoek(en)/kant warm is.
(1 = bovenkant warm, 2 = linkerbovenhoek en rechteronderhoek warm, 3 = linkeronderhoek warm).
plot = 1 of 0 (al dan niet tekenen van grafiekjes; kies dus steeds 1).
Vb.: opl = gaussel(10, 15,1,1)
Met contour i.p.v. plot kunnen de contourlijnen getekend worden (dit zijn de isotermen; deze verbinden punten met dezelfde temperatuur).

Wat onthouden:
Een eenvoudig model kan soms al een zeer goede benadering geven voor een vrij complex (fysisch) probleem.
Hierbij kunnen problemen optreden voor bepaalde keuzes van de parameters (vb. hieronder); daarom is niet elk model geschikt voor alle gevallen.
Voorbeeld van de staaf:
los_warmtevgl_op(hx, ht, lengte, maxtijd)
(Aantal stappen in x = hx*lengte)
(Aantal stappen in y = ht*maxtijd)
Goede benadering:
los_warmtevgl_op(0.2, 0.05, 10, 20)
los_warmtevgl_op(0.2, 0.05, 10, 30)
los_warmtevgl_op(0.2, 0.05, 15, 20)
los_warmtevgl_op(0.6, 0.05, 10, 20)
los_warmtevgl_op(0.2, 0.02, 10, 20)
Slechte benadering:
los_warmtevgl_op(0.2, 0.5, 10, 30)

Iteratieve methoden:
Nu zullen we ongeveer dezelfde demo geven als hierboven beschreven, maar dan iteratief.
Hiervoor gebruiken we run_laplace.m:
y = run_laplace(soortrandvwd, n, m, meth, preciesie, plot on)
waarbij:
soortrandvwd = 1, 2 of 3 (zie hoger)
n = aantal discretisatiestappen in x
m = aantal discretisatiestappen in y
meth = oplossingsmethode
(kies 1: Gauss-Seidel)
preciesie = criterium om te stoppen met
iteraties (verschil tussen 2 opeenvolgende
stappen)
plot on = 1 (teken grafiekjes)
Bij iteratieve berekeningen is het belangrijk een niet te hoge preciesie te kiezen (je kan in matlab altijd onderbreken met CTRL + C); anders stopt het algoritme misschien nooit.
De demo met iteratieve methode geeft 3 figuren:
de isotermen,
het verdelingsprofiel van de warmte

en de evolutie van de fout.

Het belangrijkste nadeel van Gauss voor dit probleem is dat de spaarsheid van de matrix niet benut wordt. Dit verbetert door iteratief te gaan werken.

Oefening 2.1:

Exacte oplossing van het stelsel:

```
[ 4 -1 0 | 1]            [ 1 11 -3 | 1]
[-1 4 -1 | 0]    -->     [-1  4  -1 | 0]   -->
[0 -1  4 | 1]            [ 0 -1   4 | 1]

[1 11 -3 | 1]            [1 11  -3 |   1]
[0 15 -4 | 1]    -->     [0   1 52 | 15]  -->
[0  -1  4 | 1]           [0  -1   4 |   1]

[1 11  -3       |   1    ]              [1 11  -3       |         1]
[0   1 52       | 15     ]      -->     [0   1 52       |        15] -->
[0   0 56       | 16     ]              [0   0   1      |(16/56)]

[1   0  -575    | -164]          [1 0 -575       |  -164]
[0   1     52   |    15] -->     [0 1      0     | (1/7)]
[0   0      1   |(2/7)]          [0 0      1     | (2/7)]

[1 0 0  | (2/7)]
[0 1 0  | (1/7)]
[0 0 1  | (2/7)]
```

De exacte oplossing is dus:

```
[ 2/7]           [0,2857]
[ 1/7]   ≈       [0,1429]
[ 2/7]           [0,2857]
```

Mogelijke reeks commando's:

```
path(path, 'd:\user')
format long
x_init = [1; 1; 1];
b = [1; 0; 1];
A = [4 -1 0; -1 4 -1; 0 -1 4]
epsilon = 0.0000001;
kmax = 50;
resJ = jacobi_opl(A, x_init, b, epsilon, kmax);
xe = resJ(:,18)          % Kolom 18 is de laatste.
n = 36;
[nx, nr] = jacobi(A, b, x_init, xe, n)
                  % Vanaf n= 37 treden er zichtbaar
                  % afrondingsfouten op.
resGS = gauss_seidel(A, x_init, b, epsilon, kmax);
[nx, nr] = gs(A, b, x_init, xe, n)
                  % Vanaf n= 20 zijn er geen
                  % verbeteringen meer.
size(resJ)
size(resGS)
[rhoJ, lambdasJ, eigenvectJ] = spectraalradius(A, 0);
[rhoGS, lambdasGS, eigenvectGS] = spectraalradius(A, 1);
```

Welke formule convergeert het snelst?

Gauss-Seidel convergeert het snelst.
Spectraalradius van de iteratiematrix:
Jacobi: ρ(G) = 0,35355339
Gauss-Seidel: ρ(G) = 0,12500000
Hoe kleiner de spectraalradius, hoe sneller de convergentie.
Wat betekenen de 2 resultaten van de methoden?
Het eerste resultaat, nx, is de norm van het verschil tussen de berekende oplossing en de exacte oplossing (dus: de absolute fout).
Het tweede resultaat, nr, is de norm van het residu. Dit komt overeen met de gemaakte fout (verschil tussen b (Ax = b) en b moet 0 zijn; de rest is de gemaakte fout) maar is afhankelijk van de konditie van het probleem.

Oefening 2.2:

Exacte oplossing van het stelsel:

```
[  3 -2  2    |  3    ]          [  3  -2  2    |  3    ]
[ -2  3 -2    | -1    ]   -->    [ -2   3 -2    | -1    ]   -->
[  2 -2  3    |  3    ]          [  0   1  1    |  2    ]

[  1  1  0    |  2    ]          [1 1  0 | 2    ]
[ -2  3 -2    | -1    ]   -->    [0 5 -2 | 3    ]   -->
[  0  1  1    |  3    ]          [0 1  1 | 2    ]

[1  1    0    |      2    ]      [1 1     0    |   2    ]
[0  1 -(2/5)  |      (3/5)  ] -> [0 1 -(2/5)   | (3/5)] -->
[0  1    1    |      1    ]      [0 0  (7/5)   | (7/5)]

[1  1    0    |      2    ]           [1 1 0  | 2]
[0  1 -(2/5)  |      (3/5)  ]   -->    [0 1 0  | 1]    -->
[0  0    1    |      1    ]           [0 0 1  | 1]
```

```
[1 0 0 | 1]
[0 1 0 | 1]
[0 0 1 | 1]
```

De exacte oplossing is dus:

```
[ 1 ]
[ 1 ]
[ 1 ]
```

Mogelijke reeks commando's:

```
epsilon = 0.0001;
kmax = 30;
A = [3 -2 2; -2 3 -2; 2 -2 3];
b = [3; -1; 3];
x0 = [3; 2; 0];
x1 = [3.0001; 2; 0];
resJ = jacobi_opl(A, x1, b, epsilon, kmax);
xe = resJ(:, 24)
        % Kolom 24 is de laatste van resJ.
[nx, nr] = jacobi(A, b, x0, xe, n);
resJ2 = jacobi_opl(A, x2, b, epsilon, kmax);
[nx, nr] = jacobi(A, b, x1, xe, n)
```

[rhoJ, lambdasJ, eigenvectJ] = spectraalradius(A, 0);
[rhoGS, lambdasGS, eigenvectGS] = spectraalradius(A, 1);

Wat gebeurt er?

Convergentie voor startwaarde [3; 2; 0]; geen convergentie voor startwaarde [3.0001; 2; 0] (vanaf de 14e iteratie treedt er divergentie op).
In het divergente geval is de fout te schrijven met een component in functie van de spectraalradius; deze component gaat overheersen naarmate er meer iteraties gebeuren en zorgt ervoor dat er divergentie is.
Eigenstructuur van de iteratiematrix:
lambda1 = -1.333333
lambda2 = 0.666667
lambda3 = 0.666667
e1 = [0.577350; -0.577350; 0.577350]
e2 = [0.805006; 0.284290;-0.520716]
e3 = [-0.136501; -0.765406;-0.628905]
Er zijn 3 lineair onafhankelijke eigenvectoren; elke vector in $|R^3$ kan dus geschreven worden als een lineaire combinatie van deze 3 eigenvectoren:
vect = $\sum_{i=1}^{3} c_i \lambda_i E_i$
Convergentie kan gemeten worden a.h.v.
$M^{-1}N$: $e^{(k)} = M^{-1}Ne^{(k-1)}$

$e^{(0)} = \sum_{i=1}^{3} c_i E_i$

$e^{(1)} = \sum_{i=1}^{3} c_i \lambda_i E_i$

...

$e^{(k)} = \sum_{i=1}^{3} c_i (\lambda_i)^k E_i$

($e^{(k)}$ is de fout na stap k).
Opdat $e^{(k)}$ naar 0 zou convergeren, moeten alle λ_i in absolute waarde < 1 zijn (zodat $c_i(\lambda_i)^k E_i$ naar 0 convergeert; c_i en E_i blijven constant voor alle iteraties).

Verklaring:

Voor de startwaarde [3; 2; 0] kan de initiële fout geschreven worden als een lineaire combinatie van de eigenvectoren behorende bij de eigenwaarden $\lambda_1 = 2/3$ en $\lambda_2 = 2/3$; de in absolute waarde grootste eigenwaarde
$\lambda_3 = -4/3$ heeft hier dus geen invloed op het convergentiegedrag => convergentie ($|\lambda_1| < 1$ en $|\lambda_2| < 1$)
Voor de startwaarde [3.0001; 2; 0] kan de initiële fout niet meer geschreven worden als een lineaire combinatie van de eigenvectoren behorende bij de eigenwaarden $\lambda_1 = 2/3$ en $\lambda_2 = 2/3$; de in absolute waarde grootste eigenwaarde $\lambda_3 = -4/3$ heeft hier dus wel invloed op het convergentiegedrag => divergentie ($|\lambda_3| > 1$). De fout gaat immers niet meer dalen naar 0, maar naarmate het aantal iteraties stijgt, gaat de component die verband houdt met λ_3 overheersen en gaat de fout in absolute waarde stijgen. Initieel is de component die verband houdt met λ_3 nog zeer klein (omdat ook de perturbatie klein is), zodat de andere 2 componenten overheersen; pas na een aantal stappen wordt de divergentie zichtbaar.

Oefening 2.3:

Eis: $0 < \omega < 2$

T.B.: $\forall i=1..n: x_i^{(k+1)} = \omega (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}) \approx M_\omega x^{(k+1)} = N_\omega x^{(k)} + b$

met $M_\omega = L + (D/\omega)$

$N_\omega = [(1-\omega)/\omega]D - U$

Bewijs:

$[L + (D/\omega)] x^{(k+1)} = ([(1-\omega)/\omega]D - U)x^{(k)} + b$

$\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + (a_{ii}/\omega)x_i^{(k+1)} = [(1-\omega)/\omega]a_{ii}x_i^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + b_i$

$(a_{ii}/\omega)x_i^{(k+1)} = [(1-\omega)/\omega]a_{ii}x_i^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)}$

$(a_{ii}/\omega)x_i^{(k+1)} = - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + b_i + [(1-\omega)/\omega]a_{ii}x_i^{(k)}$

$x_i^{(k+1)} = -(\omega/a_{ii})(\sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - b_i) + (\omega/a_{ii})[(1-\omega)/\omega]a_{ii}x_i^{(k)}$

$x_i^{(k+1)} = -\omega(\sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - b_i)/a_{ii} + [(1-\omega)]x_i^{(k)}$

Exacte oplossing:

$$\begin{bmatrix} 9b_1 & -36b_2 & 30b_3 \\ -36b_1 & 192b_2 & -180b_3 \\ 30b_1 & -180b_2 & 180b_3 \end{bmatrix}$$

Inverse van de matrix volgens matlab:

```
[9          -36          30]
[-36   192   -180]
[30    -180  180]
```

Mogelijke reeks commando's:

```
A = hilb(3);
nkh = 0.01;
[optimale_omega, rhoSOR] = optimaal(A, nkh)
[rhoGS, lambdasGS, eigenvectGS] = spectraalradius(A, 1);
[rho, lambdasSOR, eigenvectSOR] = spectraalradius2(A, optimale_omega,2);
```

Vergelijking van de methodes:

SOR zal sneller convergeren dan GS, want de spectraalradius van SOR is kleiner.
Spectraalradius voor GS op de hilbertmatrix: 0.980859
Optimale waarde voor ω: 1,628141 Spectraalradius voor SOR op de hilbermatrix: 0,852786
Eigenstructuur van de iteratiematrix voor SOR: eigenvectoren:

```
[ 0.604178      ]    [0.221000          -0.014357i ]
[-0.578556      ]    [-0.786350                    ]
[0.547943       ]    [0.575974          + 0.029348i]
        [ 0.221000          -0.014357i ]
        [-0.786350                     ]
        [ 0.575974          + 0.029348i]
```

Eigenwaarden:
0,340792

-0.852446 + 0.024116i
-0.852446 - 0.024116i
Merk op dat niet alle eigenwaarden en eigenvectoren reëel zijn!

Oefening 3:

Getalvoorstelling: $z = x+yi \in \mathbb{C} = (x, y)$
Stelsels:
1) $2z^2 + z + 1 = 0$
2) $z - e^z = 0$
Uitwerken stelsels:
1) $2(x+yi)^2 + (x+yi) + 1 = 0$
2) $(x+yi) - e^{(x+yi)} = 0$

$e^{yi} = \cos(y) + i*\sin(y)$
(reeksontwikkeling)

1) $2(x^2 - y^2 +2xyi) + (x+yi) + 1 = 0$
2) $(x+yi) - e^x(\cos(y) + i*\sin(y)) = 0$

1) $2x^2 - 2y^2 +4xyi + x+yi + 1 = 0$
2) $x+yi - e^x\cos(y) - e^x*i*\sin(y) = 0$

1) $2x^2 - 2y^2 + x + 1 + yi(4x + 1) = 0$
2) $x - e^x\cos(y) + yi - e^x*i*\sin(y) = 0$

0 is als complex getal te schrijven als
het koppel (0, 0).

1) $2x^2 - 2y^2 + x + 1 = 0$ en $yi(4x + 1) = 0$
2) $x - e^x\cos(y) = 0$ en $i(y- e^x\sin(y)) = 0$

Resultaat zijn nu 2 stelsels transcendente vergelijkingen.
1) $2x^2 - 2y^2 + x + 1 = 0$
$yi(4x + 1) = 0$

$$J = \begin{bmatrix} 4x+1 & -4y \\ 4y & 4x+1 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Delta x = \frac{-b_1(4x+1) - b_2(4y)}{(4x+1)^2 + 16y^2}$$

$$\Delta y = \frac{-b_2(4x+1) + b_1(4y)}{(4x+1)^2 + 16y^2}$$

2) $x - e^x\cos(y) = 0$
$i(y- e^x\sin(y)) = 0$

$$J = \begin{bmatrix} 1-e^x\cos(y) & e^x\sin(y) \\ -e^x\sin(y) & 1- e^x\cos(y) \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Delta x = \frac{-b_1(1- e^x\cos(y)) + b_2(e^x\sin(y))}{(1- e^x\cos(y))^2 + (e^x\sin(y))^2}$$

$$\Delta y = \frac{-b_2(1- e^x\cos(y)) - b_1(e^x\sin(y))}{(1- e^x\cos(y))^2 + (e^x\sin(y))^2}$$

Pas op deze stelsels Newton-Raphson toe.

Mogelijke commando's:

```
x_init = 1;
y_init = 1;
epsilon = 0.0000001;
kmax = 30;
[res, deltax, deltay] = nr_2linvgln(x_init, y_init, 'oefz5oef3f1', 'oefz5oef3f2', 'oefz5oef3df1x',
'oefz5oef3df1y', 'oefz5oef3df2x', 'oefz5oef3df2y', epsilon, kmax);
x1 = res(size(res), 1);
y1 = res(size(res), 2);

x_init = -1;
[res, deltax, deltay] = nr_2linvgln(x_init, y_init, 'oefz5oef3f3', 'oefz5oef3f4', 'oefz5oef3df3x',
'oefz5oef3df3y', 'oefz5oef3df4x', 'oefz5oef3df4y', epsilon, kmax);
x2 = res(size(res), 1);
y2 = res(size(res), 2);

x_init = 0.5;
y_init = 1.5;
[res, deltax, deltay] = nr_2linvgln(x_init, y_init, 'oefz5oef3f3', 'oefz5oef3f4', 'oefz5oef3df3x',
'oefz5oef3df3y', 'oefz5oef3df4x', 'oefz5oef3df4y', epsilon, kmax);
x3 = res(size(res), 1);
y3 = res(size(res), 2);
```

Oplossingen:

1) -0.250000 - 0.661438i =>
   -1/4 ± √(7/16)
2) 0,318132 - 1.337236i =>
   0,318132 ± 1.337236i

Opmerking: om met complexe getallen te kunnen rekenen in matlab mag geen enkele variabele die je intikt in je commandovenster i heten; anders zal matlab de imaginaire eenheid vervangen door een concreet getal. Indien nodig: clear.

Oefening 3.2:

Mogelijke commando's:

```
x_init = 1 + 1i;
kmax = 30;
epsilon = 0.0001;
res1 = uitgebr_nr(x_init, 'oefz5oef32fun1', 'oefz5oef32dfun1', epsilon, kmax)
res2 = uitgebr_nr(x_init, 'oefz5oef32fun2', 'oefz5oef32dfun2', epsilon, kmax)
```

Oplossingen:
1) -0.250000 - 0.661438i =>
$-1/4 \pm \sqrt{(7/16)}$
2) 0,318132 - 1.337236i =>
$0{,}318132 \pm 1.337236i$

Een functie heet analytisch als de afgeleiden in alle richtingen gelijk zijn (m.a.w. als er 1 unieke afgeleide is).
Cauchy-Riemann:

$\partial u/\partial x = \partial v/\partial y$

$\partial u/\partial y = -\partial v/\partial x$

Bewijs:

We zoeken $f(z) = 0$ met $z = x + iy$
$f(z) = u(x, y) + iv(x, y)$ met u het reële deel en v het imaginaire deel.
Oef. 3: formule:

$$\begin{bmatrix} \partial u/\partial x(x^{(k)}, y^{(k)}) & \partial u/\partial y(x^{(k)}, y^{(k)}) \\ \partial v/\partial x(x^{(k)}, y^{(k)}) & \partial v/\partial y(x^{(k)}, y^{(k)}) \end{bmatrix} \begin{bmatrix} x^{(k+1)} - x^{(k)} \\ y^{(k+1)} - y^{(k)} \end{bmatrix}$$

$$= - \begin{bmatrix} u(x^{(k)}, y^{(k)}) \\ v(x^{(k)}, y^{(k)}) \end{bmatrix}$$

Oef. 3.2 (klassieke N-R op complexe waarden): formule:

$z^{(k+1)} = z^{(k)} - f(z^{(k)})/f'(z^{(k)})$
Dus: $f'(z^{(k)})(z^{(k+1)} - z^{(k)}) = - f(z^{(k)})$
$f(z) = u(x, y) + iv(x, y)$
$f'(z) = \partial u/\partial x + i\partial v/\partial x$ (z = x+iy; f is analytisch)
$[\partial u/\partial x(x^{(k)}, y^{(k)}) + i\partial v/\partial x(x^{(k)}, y^{(k)})](x^{(k+1)} - x^{(k)} + i(y^{(k+1)} - y^{(k)})) = - u(x, y) - iv(x, y)$

(Splitsen in reëel en imaginair deel)

$[\partial u/\partial x(x^{(k)}, y^{(k)}) \quad -\partial v/\partial x(x^{(k)}, y^{(k)})][x^{(k+1)} - x^{(k)}] = - [u(x^{(k)}, y^{(k)})]$

$[\partial v/\partial x(x^{(k)}, y^{(k)}) \quad \partial u/\partial x(x^{(k)}, y^{(k)})][y^{(k+1)} - y^{(k)}] = - [v(x^{(k)}, y^{(k)})]$

(Cauchy-Riemann toepassen)

$$\begin{bmatrix} \partial u/\partial x(x^{(k)}, y^{(k)}) & \partial u/\partial y(x^{(k)}, y^{(k)}) \\ \partial v/\partial x(x^{(k)}, y^{(k)}) & \partial v/\partial y(x^{(k)}, y^{(k)}) \end{bmatrix} \begin{bmatrix} x^{(k+1)} - x^{(k)} \\ y^{(k+1)} - y^{(k)} \end{bmatrix} = - \begin{bmatrix} u(x^{(k)}, y^{(k)}) \\ v(x^{(k)}, y^{(k)}) \end{bmatrix}$$

(q.e.d.)

Oefening 4 (4.1): NIET

Mogelijke commando's:
x_init = 0.3;
kmax = 30;
res = oefz5oef41(x_init, kmax)
Resultaat:
De laatste iteraties geven complexe getallen; res(i) verspringt dan telkens tussen
c = (-0.722816... + 1.929267...i) en het complex toegevoegde hiervan.
Verklaring:
In het domein van de complexe getallen is
-log(c) = de complex toegevoegde van c en omgekeerd; eens de iteratieformule dus bij c of zijn complex toegevoegde belandt, zal er een periodisch iets ontstaan.
(-log(-log(c)) is dus convergent naar c).

De logaritmische functie is overal analytisch behalve in 0.

Oefening 5: NIET

pc-zitting 6: Het berekenen van eigenwaarden

Numerieke wiskunde
2de kand. Informatica - 2de kand. Wiskunde

Inleiding

Voor het uitwerken van de opgaven moet je '.m'-bestanden gebruiken die je kan vinden op

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

Beschrijving van de Matlab-bestanden

$m =$ **machten**$(a, x0, N)$ past de methode van de machten met scalering toe. Hierbij is m het tupel $[x,\ mu,\ e]$. Na oproep van $[x,\ mu,\ e] =$ **machten**$(a, x0, N)$ worden N iteraties uitgevoerd met de methode van de machten op de matrix a met $x0$ als startvector. Als resultaat krijg je eerst een tekening te zien, met daarop de opeenvolgende benaderingen van de eigenwaarde in modulus (aangeduid met een +) en de fouten (aangeduid met een *) t.o.v. de modulus van de laatst berekende benadering voor de eigenwaarde. Tenslotte krijg je de numerieke waarden voor de benaderde eigenvector x, de modulus mu van de bijhorende eigenwaarde en de foutenvector e te zien.

$l =$ **invmachten**$(a, x0, lambda, N)$ past de methode der inverse machten toe. Hierbij is l een vector met de opeenvolgende benaderingen voor de eigenwaarde die wordt gevonden door de methode der inverse machten toe te passen op de matrix $(\sigma I - a)$ met $x0$ als startvector en waarbij σ wordt ingegeven door de parameter $lambda$. Men kan zo de eigenwaarde vinden waarvoor σ een benadering is. Hoe beter de schatting σ voor de eigenwaarde, hoe sneller de methode zal convergeren.

1 Methode van de machten

Pas de methode van de machten toe om de dominante eigenwaarde te berekenen van onderstaande matrices A en B. Gebruik hierbij als startvector $x0$ achtereenvolgens $[1\ 0\ 0]^T$ en $[1\ 1\ 1]^T$.

$$A = \begin{pmatrix} 1002 & -2001 & 1000 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & -3 & 5 \\ -1 & -7 & 11 \\ -1 & -9 & 13 \end{pmatrix}$$

Verklaar telkens het convergentiegedrag. Lees hiertoe paragraaf 5 pag 101 in het tweede deel van de cursus. Verwijder de normalisatiestap uit het programma **machten.m** en doe een aantal iteraties met matrix A en startvector $[1\ 1\ 1]^T$. Wat concludeer je?

2 Methode der inverse machten

Gebruikt `invmachten` om van bovenstaande matrix B met startvector $x0 = [1\ 0\ 0]$ sneller de eigenwaarden te vinden door aan σ een benadering van de gezochte eigenwaarde toe te kennen. Doe een aantal experimenten voor verschillende waarden van σ en verklaar telkens het convergentiegedrag.

3 Rayleigh quotiënt iteratie

Bewijs de volgende stelling (oef 6.1 p 103):

> Als λ een eigenwaarde is van $A \in \mathbb{R}^{n\times n}$ en $X \in \mathbb{R}^{n\times 1}$ een bijhorende eigenvector, dan is
> $$\lambda = \frac{X^T AX}{X^T X}.$$
> Men noemt deze uitdrukking een Rayleigh quotiënt.

Bij Rayleigh quotiënt iteratie kiest men in elke stap van de inverse machtmethode

$$\lambda = \lambda_k = \frac{q_k^T A q_k}{q_k^T q_k}.$$

Implementeer dit algoritme en bereken een eigenwaarde van

$$A = \begin{pmatrix} 1 & 0.5 & 2 \\ 2 & 0 & 3 \\ -1 & 2 & 4 \end{pmatrix}$$

met startvector $x0 = [1\ 1\ 1]^T$ en initiële schatting voor $\lambda = 5$. Vergelijk de resultaten die u bekomt met de inverse machtmethode. Ga na dat de Rayleigh quotiënt iteratie kwadratisch convergeert. Voor hermitische matrices convergeert de methode zelfs kubisch.
Bij het uitvoeren van je programma zal matlab een waarschuwing geven wanneer het aantal iteratiestappen te hoog is. Hoe verklaar je dit?

4 Deelruimte-iteratie

Schrijf een functie

```
lambda=deelruimte(A,x0,N)
```

die de methode van de machten met meerdere startvectoren implementeert. Hier is $A \in \mathbb{R}^{n\times n}$ de matrix waarvan je $m \leq n$ dominante eigenwaarden wil berekenen, $x_0 \in \mathbb{R}^{n\times m}$ een matrix waarvan de kolommen overeen komen met de startvectoren en N het aantal iteratiestappen. De output is een matrix $\lambda \in \mathbb{R}^{N\times m}$, waarvan de rijen de benaderende dominante eigenwaarden na elke iteratiestap weergeven.
Test je algoritme op een aantal voorbeelden.

Praktische info:

- Als een matrix A meer rijen heeft dan kolommen, berekent het commando `[q,r]=qr(A,0)` een 'enconomy size' QR-factorisatie, waarbij de dimensies van q en A gelijk zijn.

- Een testmatrix T met gekende eigenwaarden kan je als volgt aanmaken: $T = WDW^{-1}$ met W een willekeurige matrix (commando `rand`) en D een diagonaalmatrix met de eigenwaarden (commando `diag`).

5 Extra oefeningen

- Pas de methode van Rayleig toe op matrix $\begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$ met startwaarden $x0 = [5 \ -3]^T$ en $\lambda = 4$. Vermits de eigenwaarden van de opgegeven matrix 1 en 5 zijn, convergeert de methode blijkbaar niet naar de dichtstbijgelegen eigenwaarde (zoals de methode van de inverse machten). Verklaar! Opmerking: de startvector is *geen* eigenvector!

- Werkt de methode van de machten nog steeds wanneer de matrix niet diagonaliseerbaar is, i.e. wanneer de eigenvectoren geen basis vormen. Probeer bij voorbeeld eens
$$A = \begin{bmatrix} 5 & \sqrt{2} & 1 \\ 0 & 6 & 0 \\ 1 & \sqrt{2} & 5 \end{bmatrix}.$$
Voor de durvers(?), geef een algemeen bewijs!

6 Iteratief oplossen van stelsels lineaire vergelijkingen

Zoals gezien in vorige oefenzittingen komen een aantal klassieke iteratieve methoden voor het oplossen van een stelsel lineaire vergelijkingen $Ax = B$ neer op een iteratieschema van de vorm:
$$Mx^{(k+1)} = Nx^{(k)} + B,$$
waarbij $A = M - N$. Wanneer men de fout in de k-de iteratiestap definieert als $e^{(k)}$, leidt men hieruit af:
$$e^{(k)} = (M^{-1}N)e^{k-1} \rightarrow e^{(k)} = (M^{-1}N)^k e^{(0)}, \tag{1}$$
wat het verband met de methode van de machten duidelijk maakt. Uit (1) volgt dat een nodige en voldoende voorwaarde voor het convergent zijn van het iteratieschema gegeven wordt door $\rho(M^{-1}N) < 1$ met $\rho(.)$ de spectraalradius, i.e. de modulus van de grootste eigenwaarde. Deze grootste eigenwaarde bepaalt tevens het asymptotische convergentiegedrag.

PC-zitting 6: eigenwaarden.

Opmerkingen:
De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~saskia/oefz
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Oef. 5 is niet echt een oefening, gewoon een herhaling van een klein stukje theorie.

Oplossingen van de oefeningen:

Oefening 1:

Mogelijke commando's:

```
path(path, 'd:\user')
format long
A = [1002 -2001 1000; 1 0 0; 0 1 0];
B = [1 -3 5; -1 -7 11; -1 -9 13];
x_init = [1; 0; 0];
x_init2 = [1; 1; 1];
n = 30;
[xkA1, mukA1, errorA1] = machten(A, x_init, n);
[xkA2, mukA2, errorA2] = machten(A, x_init2, n);

[xkB1, mukB1, errorB1] = machten(B, x_init, n);
[xkB2, mukB2, errorB2] = machten(B, x_init2, n);
[VA, DA] = eig(A);
[VB, DB] = eig(B);

[xkA3, mukA3, errorA3] = machten2(A, x_init, n);
[xkA4, mukA4, errorA4] = machten2(A, x_init2, n);

[xkB3, mukB3, errorB3] = machten2(B, x_init, n);
[xkB4, mukB4, errorB4] = machten2(B, x_init2, n);
```

Convergentiegedrag + verklaring:
Opmerking: eigenvectoren zijn tot op een constante na uniek bepaald => vermenigvuldigen met een constante mag steeds.
Matrix A, startwaarde [1; 0; 0]:
Convergentie van xk naar [10^{0}; 10^{-3}; 10^{-6}], de eigenvector behorende bij de grootste eigenwaarde, zoals verwacht.
Matrix A, startwaarde [1; 1; 1]:
Convergentie van xk naar [1; 1; 1], de gegeven startvector is een vast punt van de methode (indien er exact gerekend wordt).

Matrix B, startwaarde [1; 0; 0]:
Convergentie van xk naar [1; 1; 1], de eigenvector behorende bij de grootste eigenwaarde, zoals verwacht.
Matrix B, startwaarde [1; 1; 1]:
Convergentie van xk naar [1; 1; 1].

Matrix A, startwaarde [1; 0; 0], zonder normalisatie:
In de limiet zal er overflow optreden; in elke stap wordt de norm van de eigenvector groter.
Matrix A, startwaarde [1; 1; 1], zonder normalisatie:
Convergentie van xk naar [1; 1; 1], de gegeven startvector is een vast punt van de methode (indien er exact gerekend wordt).

Matrix B, startwaarde [1; 0; 0], zonder normalisatie:
In de limiet zal er overflow optreden; in elke stap wordt de norm van de eigenvector groter.
Matrix B, startwaarde [1; 1; 1], zonder normalisatie:
In de limiet zal er overflow optreden; in elke stap wordt de norm van de eigenvector groter.

A geeft snellere convergentie dan B, omdat de verhouding van de kleinste eigenwaarde tot de grootste (in abs. waarde) bij A (1/1000) kleiner is dan bij B (2/3).
Het effect van de normalisatie is dat er grote afrondingsfouten gemaakt worden: tijdens het berekenen van de norm en tijdens de deling. Die zorgen ervoor dat er een component kan ontstaan in de richting van de dominante eigenwaarde indien die er niet is in de startvector, zodat de methode van de machten in praktijk bijna altijd convergeert.

Oefening 2:

Mogelijke reeks commando's:

```
n = 30;
s1 = 3.1;
l = invmachten(B, x_init, s1, n);
s2 = 2.9;
l = invmachten(B, x_init, s2, n);
s3 = 2.1;
l = invmachten(B, x_init, s3, n);
s4 = 1.9;
l = invmachten(B, x_init, s4, n);
l = invmachten2(B, x_init, s1, n);
l = invmachten2(B, x_init, s2, n);
l = invmachten2(B, x_init, s3, n);
l = invmachten2(B, x_init, s4, n);
```

Hint:
A heeft als eigenwaarden $\lambda_1 \ldots \lambda_n$;
A heeft als eigenvectoren $E_1 \ldots E_n$.
Methode van de machten: $\max_i\{\lambda_i\}$ van A zoeken.

A^{-1} heeft als eigenwaarden $1/\lambda_1 \ldots 1/\lambda_n$;
A^{-1} heeft als eigenvectoren $E_1 \ldots E_n$.
Methode van de machten: $\max_i\{1/\lambda_i\}$ van A^{-1} zoeken = $\min_i\{\lambda_i\}$

Methode van de inverse machten in deze oef.:

$\min_i\{\gamma_i\}$ van $(\sigma I - A)^{-1}$ zoeken:
$(\sigma I - A)$ heeft als eigenwaarden
$\gamma_1 = \sigma - \lambda_1 \ldots \gamma_n = \sigma - \lambda_n$.
$(\sigma I - A)$ heeft als eigenvectoren
$E_1 \ldots E_n$, want:
$(\sigma I - A)X = \gamma_i X = (\sigma - \lambda_i)X$ (met $X = E_i$).
λ_i ligt verst van σ: $\max_i\{|\sigma - \lambda_i|\}$
$(\sigma I - A)^{-1}$ heeft als eigenwaarden
$1/\gamma_1 = 1/(\sigma - \lambda_1) \ldots 1/\gamma_n = 1/(\sigma - \lambda_n)$.
$(\sigma I - A)^{-1}$ heeft als eigenvectoren
$E_1 \ldots E_n$,
λ_i ligt dichtst bij σ: $\min_i\{|\sigma - \lambda_i|\}$

<u>Convergentiegedrag:</u>
Convergentie naar de eigenwaarde die het dichtst bij de benadering ligt.
Verklaring convergentiegedrag:
De methode van de inverse machten wordt in zijn normale vorm gebruikt om de kleinste eigenwaarde van een matrix A te zoeken door de grootste eigenwaarde van de inverse van A te berekenen.
In deze oefening wordt de methode gebruikt om de kleinste eigenwaarde van $(\lambda I - A)$ te berekenen waarbij λ een benadering is van de eigenwaarde λ_i die je zoekt.
Als γ de kleinste eigenwaarde is van $(\lambda I - A)$, dan: $\gamma = \lambda - \lambda_i$ dus $\lambda_i = \lambda + \gamma$.
Aanpassing van de methode van de inverse machten:
I.p.v. de eigenwaarden van $(\sigma I - A)^{-1}$ te zoeken, zoek je de eigenwaarden van $(\sigma I - A)$.
Levert de 2e methode altijd de dominante eigenwaarde?
Nee, deze methode berekent de eigenwaarde die het dichtst bij de schatting $1/\sigma$ ligt; er is convergentie naar de dominante eigenwaarde van A indien de niet-aangepaste methode convergeerde naar de kleinste eigenwaarde.
Aanpassing 2:
Vergelijken van de resultaten van de originele en de eerste aangepaste methode; hieruit de (in abs. waarde) grootste eigenwaarde selecteren.
Resultaten:
Ook hier is convergentie naar de dominante eigenwaarde niet gegarandeerd; indien 1 van beide methoden convergeert naar de dominante eigenwaarde, zal er convergentie zijn. Echter, voor het vb. waarbij $\lambda_1 = -1$,
$\lambda_2 = 5$ en $\lambda_3 = 10$ zal een schatting $\lambda = 6$ convergeren naar -1 resp. 5, en dus niet naar de dominante eigenwaarde 10.
Aanpassing 3:
Vervang de matrix $(\sigma I - A)^{-1}$ door de matrix
A en inverteer de (absolute waarde van) de gevonden eigenwaarde.
Resultaten:
Dit convergeert steeds naar de dominante eigenwaarde indien er convergentie is.

Oefening 3:

Uitleg bij Rayleigh quotiënt:

De convergentiesnelheid (methode van de inverse machten) is evenredig met $|\lambda_1|/|\lambda_2|$. Hoe groter deze verhouding, hoe sneller de convergentie (veronderstel $|\lambda_1| > |\lambda_2|$).
Zoek een schatting σ die dicht bij $|\lambda_1|$ ligt en ver van $|\lambda_2|$.
Dan: $1/(|\sigma - \lambda_1|) > 1/(|\sigma - \lambda_2|)$
De convergentiesnelheid wordt nu:
$[1/(|\sigma - \lambda_1|)] / [1/(|\sigma - \lambda_2|)] = (|\sigma - \lambda_2|)/(|\sigma - \lambda_1|)$; de noemer hiervan is klein en de teller relatief groot.
Dus: de convergentie is sneller dan voor de methode van de inverse machten.

Bewijs:
λ is eigenwaarde van A en X een bijbehorende eigenvector

$<=> AX = \lambda X$
$<=> X^TAX = X^T\lambda X$
$<=> X^TAX = \lambda X^TX$
$<=> X^TAX/X^TX = \lambda X^TX/X^TX$
$<=> X^TAX/X^TX = \lambda$

(q.e.d.)

Mogelijke reeks commando's:

```
C = [1 0.5 2; 2 0 3; -1 2 4];
x_init3 = [1; 1; 1];
lambda_init = 5;
aantal_stappen = 10;
[lambdas, q] = oefz6oef3rayleigh(C, x_init3, lambda_init, aantal_stappen);
foutRayleigh = 2*abs(lambdas - lambdas(size(lambdas, 1)));
[l, q] = invmachten(C, x_init3, lambda_init, aantal_stappen);
aantal_stappen = 6;
[lambdas, q] = oefz6oef3rayleigh(C, x_init3, lambda_init, aantal_stappen);
[l, q] = invmachten(C, x_init3, lambda_init, aantal_stappen);
```

Vergelijken met inverse machten:
De methode van de inverse machten convergeert trager.
Convergentieorde van Rayleigh:
De methode van Rayleigh berekend voor elke nieuwe benadering van de eigenwaarde ook een betere benadering van de eigenvector behorende bij de gezochte eigenwaarde; daardoor verloopt de convergentie sneller.
De gemaakte fout daalt zeer snel (steeds meer nullen): kwadratische convergentie.
Waarom waarschuwing bij groot aantal iteratiestappen?
Je zoekt de eigenwaarde λ_i die het dichtst bij de schatting λ ligt; naarmate λ_i naar λ gaat, zal $\det(\lambda_i I - A)$ naar 0 gaan.
De stap (l(i)*eye(size(a, 1))-a = $\lambda_i I - A$) is dus slecht geconditioneerd.

Oefening 4: NIET: algemeen bewijs.

Mogelijke commando's:
D = [3 2; 2 3];
start = [5; -3]
lambda_start = 4;
aantal_stappen = 7;
[eigenw, eigenvect] = oefz6oef3rayleigh(D, start, lambda_start, aantal_stappen)
Verklaring convergentiegedrag:

1) Zoeken van de eigenwaarden van de matrix:

$$\det \begin{bmatrix} 3-\lambda & 2 \\ 2 & 3-\lambda \end{bmatrix} = (3-\lambda)^2 - 4$$

$$= 9 - 6\lambda + \lambda^2 - 4$$
$$= \lambda^2 - 6\lambda + 5$$
$$D = 36 - 20$$
$$= 16$$
$$\lambda = (6 \pm 4)/2$$
$\lambda_1 = 5$ en $\lambda_2 = 1$

2) Eigenvectoren zoeken behorende bij de eigenwaarden:

a) $AX = \lambda X \Rightarrow$ $[3\ 2][x_1] = [5x_1]$
$[2\ 3][x_2] = [5x_2]$

$$x_1 = \frac{\begin{vmatrix} 5x_1 & 2 \\ 5x_2 & 3 \end{vmatrix}}{\begin{vmatrix} 3 & 2 \\ 2 & 3 \end{vmatrix}} \Rightarrow x_1 = (15x_1 - 10x_2)/(9 - 4)$$

$x_1 = 5(3/5)x_1 - 5(2/5)x_2$

$$x_2 = \frac{\begin{vmatrix} 3 & 5x_1 \\ 2 & 5x_2 \end{vmatrix}}{\begin{vmatrix} 3 & 2 \\ 2 & 3 \end{vmatrix}} \Rightarrow x_2 = (15x_2 - 10x_1)/(9 - 4)$$

$x_2 = 3x_2 - 2x_1$

Eigenvectoren zijn bepaald tot op een constante na; 1 van de elementen mag dus vrij gekozen worden.
Kies dus bv. $x_1 = 1$.

$x_2 = 3x_2 - 2x_1$
$= 3x_2 - 2$
$\Leftrightarrow x_2 - 3x_2 = -2$
$\Leftrightarrow -2x_2 = -2$
$\Leftrightarrow x_2 = 1$

De eigenvector e_1 behorende bij $\lambda_1 = 5$ is dus [1; 1].

b) $AX = \lambda X \Rightarrow$ $[3\ 2][x_1] = [x_1]$
$[2\ 3][x_2] = [x_2]$

$$x_1 = \frac{\begin{vmatrix} x_1 & 2 \\ x_2 & 3 \end{vmatrix}}{\begin{vmatrix} 3 & 2 \\ 2 & 3 \end{vmatrix}} \Rightarrow x_1 = (3x_1 - 2x_2)/(9 - 4)$$

$x_1 = (3/5)x_1 - (2/5)x_2$

$$x_2 = \frac{\begin{vmatrix} 3 & x_1 \\ 2 & x_2 \end{vmatrix}}{\begin{vmatrix} 3 & 2 \\ 2 & 3 \end{vmatrix}} \Rightarrow x_2 = (3x_2 - 2x_1)/(9 - 4)$$

|3 2|
|2 3|

$x_2 = (3/5)x_2 - (2/5)x_1$

Eigenvectoren zijn bepaald tot op een constante na; 1 van de elementen mag dus vrij gekozen worden.

Kies dus bv. $x_1 = 1$.

$x_2 = (3/5)x_2 - (2/5)x_1$

$= (3/5)x_2 - (2/5)$

$\Leftrightarrow (5/5)x_2 - (3/5)x_2 = -(2/5)$

$\Leftrightarrow (2/5)x_2 = -(2/5)$

$\Leftrightarrow x_2 = -1$

De eigenvector e_2 behorende bij $\lambda_2 = 1$ is dus [1; -1].

3) Zet beide eigenvectoren en de startvector uit op een assenstelsel: oefz6oef4.fig

Title:
/amd/godard/export/home0/saskia/oefz/matlabfiles/oefz6/oefz6oef4.eps
Creator:
MATLAB, The Mathworks, Inc.
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

4) Conclusie:

De eigenvectoren zijn orthogonaal en vormen dus een basis.
De startvector ligt niet in het verlengde van een eigenvector (en is dus geen van beide eigenvectoren, omdat eigenvectoren tot op een constante na bepaald zijn).
De startvector ligt echter wel zeer dicht in de buurt van 1 van beide eigenvectoren, nl. dicht bij $e_2 = [1; -1]$, de eigenvector horende bij $\lambda_2 = 1$.
Omdat de methode van Rayleigh gebruik maakt van de startvector om een nieuwe schatting voor de eigenwaarde te bepalen, zal reeds in de eerste iteratiestappen convergentie naar $\lambda_2 = 1$ optreden i.p.v. naar de dominante eigenwaarde $\lambda_1 = 5$.

Mogelijke commando's:

```
E = [5 sqrt(2) 1; 0 6 0; 1 sqrt(2) 5];
startvect = [1; 1; 1];
```

```
aantal_stappen = 30;
res = machten(E, startvect, aantal_stappen);
```

Convergentiegedrag:

Ook voor niet-diagonaliseerbare matrices is de methode van de machten convergent; echter, Matlab is niet geschikt om eigenwaarden en/of eigenvectoren van dit soort matrices te berekenen met het commando eig.

"Oefening" 5:

Convergentie van iteratieve methodes voor stelsels lineaire vergelijkingen:

1) Criterium uit de cursustekst:

$e^{(k)} = k^{de}$ iteratiefout;

$e^{(k)} = Ge^{(k-1)}$

met $G = -D^{-1}(U+L)$ voor Jacobi

en $G = -(L+D)^{-1}U$ voor Gauss-Seidel

$e^{(k)} = G^k e^{(0)}$

Dus: $||e^{(k)}|| \leq ||G||^k e^{(0)}$

Dus: $||e^{(k)}|| \dashrightarrow 0$ als $||G|| < 1$

2) Nauwkeuriger:

$Ax = B$ oplossen komt neer op een iteratieschema van de vorm

$Mx^{(k+1)} = Nx^{(k)} + B$, met $A = M-N$.

$e^{(k)} = k^{de}$ iteratiefout;

$e^{(k)} = (M^{-1}N)e^{(k-1)}$

$e^{(k)} = (M^{-1}N)^k e^{(0)}$

Dus: $||e^{(k)}|| \dashrightarrow 0$ als $\rho(M^{-1}N) < 1$ met

ρ de spectraalradius (= de modulus van de grootste eigenwaarde).

Immers:

$e^{(k)} = (M^{-1}N)^k e^{(0)}$

$\Rightarrow ||e^{(k)}||/||e^{(0)}|| = (M^{-1}N)^k$

$\Rightarrow ||e^{(k)}||/||e^{(0)}|| < 1$

$\Rightarrow$ Grootste eigenwaarde van

$(M^{-1}N)^k < 1$ in abs. waarde.

De grootste eigenwaarde bepaalt ook het asymptotisch convergentiegedrag

pc-zitting 8: Splines

2de kand. Wiskunde - 2de kand. Informatica

Voor het uitwerken van de opgaven moet je '.m'-bestanden gebruiken die je kan vinden op

`http://www.cs.kuleuven.ac.be/~wimm/oefenzittingen/`

1 Inleiding

Genormaliseerde B-splines worden gedefinieerd aan de hand van de recursiebetrekking

$$N_{i,k}(x) = (x - t_i)\frac{N_{i,k-1}(x)}{t_{i+k} - t_i} + (t_{i+k+1} - x)\frac{N_{i+1,k-1}(x)}{t_{i+k+1} - t_{i+1}} \ ,$$

met

$$\begin{aligned} N_{i,0}(x) &= 1 \quad \text{als} \quad x \in [t_i, t_{i+1}) \\ N_{i,0}(x) &= 0 \quad \text{als} \quad x \notin [t_i, t_{i+1}) \ . \end{aligned}$$

Genormaliseerde B-splines voldoen aan 4 eigenschappen:

a) lokaliteit: $N_{i,k}(x) = 0$ als $x \notin [t_i, t_{i+k+1}]$.

b) positiviteit: $N_{i,k}(x) \geq 0$, of, in het bijzonder:

$$\begin{aligned} N_{i,k}(x) &= 0 \quad \text{als} \quad x \notin [t_i, t_{i+k+1}] \\ N_{i,k}(x) &> 0 \quad \text{als} \quad x \in (t_i, t_{i+k+1}) \ . \end{aligned}$$

c) sommatie tot 1: $\sum_{i=-k}^{n-1} N_{i,k}(x) = 1$ voor $x \in [t_0, t_n)$.

d) $N_{i,k}(x)$ is C^{k-1}–continu.

2 Matlab commando's

2.1 Het manipuleren van interpolerende veeltermen

- *y = veelterm(xi,yi,x)*

  geeft het resultaat van de evaluatie in x van de veelterm die interpoleert in de punten {*xi,yi*}. *De in- en uitvoerparameters zijn rijvectoren.*

2.2 Het manipuleren van B-splines

- *y = spline(xi,yi,x)*

  geeft het resultaat van de evaluatie in x van de kubische spline die interpoleert in de punten {xi,yi}.

- sp = spmak(knots,coef)

  berekent de spline *sp* (=lineaire combinatie van B-splines) waarvan de knooppunten de elementen zijn van de rij *knots* en de coëfficiënten gespecifieerd worden door de vector *coef*. De graad van de spline wordt zodanig gekozen dat het aantal opgegeven coëfficiënten overeenkomt met het aantal verschillende B-splines van die graad mogelijk op de gegeven knooppunten.

- *fnplt(sp)*

  tekent de B-splinefunktie *sp* zoals ze werd berekend door *spmak*

3 Oefeningen

3.1 Interpolatie met veeltermen en splines

Beschouw de functie $y = \frac{x}{1+x^2}$. Teken deze functie over het interval $[-5,\ 5]$ evenals de veelterm en de kubische spline die interpoleren in de punten $\{i, y(i)\},\ i = -5 : 1 : 5$. Vergelijk. Is dit type curve geschikt voor spline/veeltermbenadering?

3.2 Convergentiegedrag

Herneem vorige oefening en zet de benaderingsfout van de interpolerende veelterm uit in functie van x over het interval $[-5,\ 5]$, waarbij je de interpolatiepunten equidistant kiest met een tussenafstand van 2, 1 en 0.5. Maak één grafiek met logaritmische y-as. Verklaar. Doe hetzelfde voor de interpolerende kubische spline en vergelijk.

3.3 Transformatie

Benader de functie $y = 20 \tanh\left(\frac{1}{20(x-1)^2}\right)$ over het interval $[0,\ 2]$ met de interpolerende veelterm en de kubische spline door de punten $\{i, y(i)\},\ i = 0 : 0.2 : 2$. Bepaal ook een benadering gebaseerd op de interpolerende veelterm van de getransformeerde curve $\frac{1}{y}$. Vergelijk de drie resultaten.

3.4 Basisfuncties

3.4.1 Genormaliseerde B-splines

Teken met behulp van de commando's *spmak* en *fnplt* een genormaliseerde B-spline van graad 1, 2 en 3. Eeen B-spline van graad k is op $k+1$ intervallen verschillend van nul.

3.4.2 Samenvallende knooppunten

De recursiebetrekking voor genormaliseerde B-splines blijft geldig voor samenvallende knooppunten indien we volgende conventie respecteren:

- indien $t_i = t_{i+1} = \ldots = t_{i+k}$ wordt in de recursiebetrekking de eerste

term nul gesteld

- indien $t_{i+1} = t_{i+2} = \ldots = t_{i+k+1}$ wordt de tweede term nul gesteld.

Bepaal en teken met behulp van matlab $N_{0,2}(x)$ voor:

a) $t_0 = t_1 < t_2 < t_3$

b) $t_0 = t_1 = t_2 < t_3$

c) $t_0 < t_1 = t_2 < t_3$.

Wat gebeurt er met de continuïteitseigenschap? Is dit een motivatie om *altijd* separate knooppunten te nemen? Denk daarbij aan het soort curve dat je wil benaderen. Ga na dat $N_{i,k}(t_i) = 1$ als t_i een meervoudigheid $k+1$ heeft.

3.4.3 Combinatie

Voer volgend commando uit: *fnplt*(*spmak*(0:10,[1 1 1 1 1 1])). Wat is de graad van de spline? Verklaar de verschillende delen op de grafiek?

3.5 Extra

Op het formule 1 circuit van Monza ligt de zogenaamde parabolica bocht: een aaneensluiting van een rechte baan en een paraboolvormige bocht. Welk soort discontinuïteit zit er in de vorm van de weg? Wat heeft dit met splines te maken? Welk fysisch fenomeen grijpt op een abrupte (i.e. discontinue) manier aan bij het ingaan van zulke bocht?

PC-zitting 7: Splines.

Opmerkingen:
De bestanden die nodig zijn om deze oefenzitting te kunnen oplossen, staan NIET in de map "m:\extern\matlab\numwisiw"; je kunt ze afhalen van het web op volgende URL:
http://www.cs.kuleuven.ac.be/~saskia/oefz
Na kopiëren in de map "d:\user" kun je intikken in Matlab:
path(path, 'd:\user')
De bestanden zullen dan gebruikt kunnen worden tijdens de oefeningen.

"format long" of "format long e" kunnen gebruikt worden om met voldoende cijfers in de mantisse te kunnen werken.

Oplossingen van de oefeningen:

Oefening 1 + 2:

Mogelijke commando's:

```
path(path, 'd:\user')
format long
begin = -5;
einde = 5;
stapgrootte = 0.01;
% fout bij de functie zelf wordt
% 0 verondersteld.
[x1, y1] = oefz7oef1fun(begin, einde, stapgrootte);
intervalgrootte1 = 1;
intervalgrootte2 = 2;
intervalgrootte3 = 0.5;
xi1 = begin:intervalgrootte1:einde;
xi2 = begin:intervalgrootte2:einde;
xi3 = begin:intervalgrootte3:einde;
yi1 = xi1./(1 + xi1.^2);
yi2 = xi2./(1 + xi2.^2);
yi3 = xi3./(1 + xi3.^2);
yveelt1 = polyval(intpol([xi1; yi1]), x1);
yveelt2 = polyval(intpol([xi2; yi2]), x1);
yveelt3 = polyval(intpol([xi3; yi3]), x1);
yspline1 = spline(xi1, yi1, x1);
yspline2 = spline(xi2, yi2, x1);
yspline3 = spline(xi3, yi3, x1);
plot(x1, y1);
hold on;
plot(xi1, yi1, '*');
plot(x1, yveelt1, 'g');
plot(x1, yspline1, 'r');
figure;
plot(x1, y1);
hold on;
plot(xi2, yi2, '*');
plot(x1, yveelt2, 'g');
plot(x1, yspline2, 'r');
figure;
```

```
plot(x1, y1);
hold on;
plot(xi3, yi3, '*');
plot(x1, yveelt3, 'g');
plot(x1, yspline3, 'r');
figure;
eveelt1 = abs(y1 - yveelt1);
espline1 = abs(y1 - yspline1);
eveelt2 = abs(y1 - yveelt2);
espline2 = abs(y1 - yspline2);
eveelt3 = abs(y1 - yveelt3);
espline3 = abs(y1 - yspline3);
plot(x1, eveelt1, 'b');
hold on;
plot(x1, eveelt2, 'g');
plot(x1, eveelt3, 'r');
figure;
plot(x1, espline1, 'b');
hold on;
plot(x1, espline2, 'g');
plot(x1, espline3, 'r');
figure;
```

Vergelijken van de resultaten:

De splinecurve is een goede benadering. De veeltermінterpolatie geeft problemen in de buurt van de grenzen van het interpolatie-interval; dit komt omdat de gevraagde functie niet veeltermachtig is, een veeltermfunctie heeft immers nooit horizontale asymptoten maar zal steeds naar oneindig gaan in de limiet op oneindig.
Geschikt voor veeltermінterpolatie? Nee.
Geschikt voor spline-interpolatie? Ja.
Het nemen van meer interpolatiepunten is voor de splinecurve overal een verbetering; voor veeltermінterpolatie wordt het resultaat aan de uiteinden van het interpolatie-interval nog slechter terwijl er in het midden wel een verbetering is. De reden is, dat bij meer interpolatiepunten ook de graad van de veelterm stijgt, waardoor de veelterm nog meer naar oneindig gaat gaan aan de randen.

Oefening 3:

Mogelijke reeks commando's:

```
begin = 0;
einde = 2;
stapgrootte = 0.01;
intervalgrootte = 0.2;
[x1, y1] = oefz7oef3fun(begin, einde, stapgrootte);
[xi, yi] = oefz7oef3fun(begin, einde, intervalgrootte);
yveelt1 = polyval(intpol([xi; yi]), x1);
yspline1 = spline(xi, yi, x1);
y_inverse = 1./y1;
yi_inverse = 1./yi;
yveelt_inverse = polyval(intpol([xi; yi_inverse]), x1);
yveelt2 = 1./yveelt_inverse;

plot(x1, y1, 'b');
hold on;
plot(xi, yi, '*');
plot(x1, yveelt1, 'g');
plot(x1, yspline1, 'r');
plot(x1, yveelt2, 'y');
```

Gedrag:
De veelterminterpolatie sluit slecht aan aan de randen; de splinecurve sluit beter aan (verklaring cfr. oef. 1 en 2).
De veelterm gebaseerd op de inverse van de functie gedraagt zich zeer goed; dit komt omdat de inverse van de functie wel een veeltermachtig gedrag heeft. De inverse van de functie kan dus goed door een veeltermfunctie benaderd worden en de inverse van die benadering geeft een goede interpolerende veelterm voor de oorspronkelijke functie.

Oefening 4.1:

Mogelijke commando's:

```
knopen1 = [1 2];
knopen2 = [1 2 3];
knopen3 = [1 2 3 4];
coef = 1;
fnplt(spmak(knopen1, coef));
figure;
fnplt(spmak(knopen2, coef));
figure;
fnplt(spmak(knopen3, coef));
figure;
```

Resultaten:
Graad 0: continu, discontinu in de afgeleide.
Graad 1: continu, continu in de afgeleide, discontinu in de 2e afgeleide.
Graad 2: continu, continu in de afgeleide, continu in de 2e afgeleide, discontinu in de 3e afgeleide.
De som van alle B-Splines van een zelfde graad is steeds 1; alle B-Splines die verschillend van 0 zijn in een interval, vormen in dat interval een basis.

Oefening 4.2:
Mogelijke commando's:

```
knopenA = [1 1 2 3];
knopenB = [1 1 1 2];
knopenC = [1 2 2 3];
coef = 1;
fnplt(spmak(knopenA, coef));
figure;
fnplt(spmak(knopenB, coef));
figure;
fnplt(spmak(knopenC, coef));
figure;
```

Wat met continuïteit?

Normaal is $f^{(n)}$ continu voor B-Splines van graad n, maar voor elk samenvallend knooppunt (tel het knooppunt waarmee het samenvalt niet mee) treedt er verlies aan continuïteit op (1 samenvallend => n-1, 2 samenvallende => n-2, ...).
Voor opgave a is dus de 2e afgeleide niet meer continue, opgave b heeft een discontinue afgeleide en opgave c ook een discontinue 2e afgeleide.
Altijd separate knooppunten?
Nee; het laten samenvallen van knooppunten kan het mogelijk maken om voorwerpen te tekenen die een discontinuïteit hebben (bv. een hartje heeft onderaan een discontinue afgeleide).
Nagaan dat $N_{i,k}(t_i) = 1$ als t_i meervoudigheid k+1 heeft:
De som van alle B-Splines die verschillen van 0 in een interval is steeds 1; bij meervoudigheid k+1 is dit interval gereduceerd tot 1 punt, nl. het knooppunt. De waarde van de spline is dus 1 daar.

Oefening 4.3:

Mogelijke commando's:

```
knopenD = 0:10;
coeffic = [1 1 1 1 1 1];
fnplt(spmak(knopenD, coeffic));
```

Graad van de spline:
4; je vraagt 6 coëfficiënten voor 11 knooppunten => 10 intervallen.
Verschillende delen op de grafiek:
Waar de grafiek als functiewaarde 1 heeft, vormen de B-Splines een basis; alle B-Splines van graad 4 worden daar gebruikt om de grafiek samen te stellen. Op de andere delen zijn sommige voor de grafiek geselecteerde
B-Splines 0, tot aan de uiteinden er geen enkele B-Spline verschillend van 0 meer overblijft. Aan de uiteinden zijn er dus
B- Splines verschillend van 0 waarvan het niet-nul deel niet volledig in het beschouwde interval valt.

Oefening 5:

Welk soort discontinuïteit?
Discontinu in 2e afgeleide (de kromming); dit is een evenredigheid met de kracht.
Relatie met splines:
De meervoudigheid van de interpolatiepunten en de graad van de B-Splines bepalen welke afgeleiden continu zullen zijn.
Fysisch fenomeen dat ingrijpt bij het ingaan van een dergelijke bocht:
Centrifugale/centripetale krachten.

OEFENINGEN NUMERIEKE WISKUNDE

OEFENZITTING 12: NULPUNTEN VAN EEN VEELTERM

1. Conditie van een nulpunt van een veelterm

1.1. Een formule i.v.m. conditie van nulpunten van veeltermen.

- Enkelvoudig nulpunt
  Stel

$$\begin{aligned} p(x) &= \sum_{i=0}^{n} a_i x^{n-i} \\ \bar{p}(x) &= \sum_{i=0}^{n} \bar{a}_i x^{n-i} \\ \bar{a}_i &= a_i + \Delta a_i \\ p(c) &= 0 \\ \bar{p}(\bar{c}) &= 0 \\ \bar{c} &= c + \Delta c. \end{aligned}$$

  Bewijs (door verwaarlozen van tweede-orde-termen):

$$\Delta c \approx \frac{-\sum_{j=0}^{n} \Delta a_j c^{n-j}}{p'(c)}.$$

  Waarom geldt deze formule niet meer voor meervoudige nulpunten?

- Meervoudig nulpunt
  Stel $p(x) = \sum_{i=0}^{n} a_i x^{n-i}$. Stel dat α een m-voudige wortel van p is.

  Bewijs dat $p^{(i)}(\alpha) = 0$ voor $i = 0, \ldots, m-1$.

  Stel $\bar{p}(x) = p(x) + \Delta p(x)$ met $\Delta p(x) = \sum_{i=0}^{n} \Delta a_i x^{n-i}$. Stel $\alpha + \Delta\alpha$ een nulpunt van $\bar{p}(x)$.

  Bewijs dat

$$\Delta\alpha \approx \left(\frac{-\sum_{i=0}^{n} \Delta a_i \alpha^{n-i} m!}{p^{(m)}(\alpha)} \right)^{1/m}. \tag{1}$$

Stel dat $|\Delta a_i| < \varepsilon$. Dan zal

$$|\Delta\alpha| \leq \left(\frac{\varepsilon m!}{|p^{(m)}(\alpha)|} \sum_{i=0}^{n} |\alpha|^{n-i} \right)^{1/m}. \tag{2}$$

Wat kun je uit (1) en (2) halen i.v.m. de konditie?

1.2. **Illustraties met matlab.**

- Controle van formule (1)
  Construeer een veelterm met een zesvoudig nulpunt $x = 2$. Gebruik hiervoor het commando `poly`. Bereken dan de nulpunten met het commando `roots`. Ga de geldigheid van de formule na die werd afgeleid. Teken de berekende nulpunten in het complexe vlak (gebruik hiervoor de commando's `real` en `imag`).
- Veelterm van Wilkinson
  De veelterm van Wilkinson is gekend als

  $$p(x) = (x-1)(x-2)\ldots(x-20) = x^{20} - 210x^{19} + \ldots + 20!.$$

  Construeer deze veelterm met het commando `poly`. Verander dan de coëfficiënt van x^{19} in $-210 + 2.10^{-8}$ en bereken dan de nulpunten.
  Dit is een voorbeeld van een zeer slecht gekonditioneerde veelterm. Veeltermen van hoge graad zijn dikwijls zeer gevoelig voor kleine wijzigingen aan de coëfficiënten.

2. Bepalen van alle wortels van een veelterm

2.1. **Deflatie.** Deflatie of het wegdelen van reeds gevonden nulpunten
Construeer een veelterm met nulpunten $x = 1, x = 10, x = 100, x = 1000, x = 10000$ en $x = 100000$. Bereken hiervan de nulpunten. Deel vervolgens met de routine `horner.m` het grootste berekende nulpunt weg. Herhaal dit tot je enkel het kleinste nulpunt overhoudt. Doe nu hetzelfde maar deel de nulpunten weg van klein naar groot. Welke resultaten bekom je? Waarom?

2.2. **Omzetten naar een eigenwaardeprobleem.** Het wegdelen van wortels (deflatie) is een onstabiele methode. Men kan echter het probleem omzetten tot het berekenen van alle eigenwaarden van een matrix, waarvoor stabiele algoritmen bestaan. Stel $p(x) = x^n + a_1 x^{n-1} + \ldots + a_n$ met nulpunten $\alpha_1,\ \alpha_2, \ldots,\ \alpha_n$. Beschouw de Companion-matrix C_p van de polynoom $p(x)$:

$$C_p = \begin{pmatrix} -a_1 & -a_2 & \ldots & -a_{n-1} & -a_n \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}.$$

(1) Bewijs dat de nulpunten van de polynoom $p(x)$ de eigenwaarden zijn van C_p.
(2) Wat zijn de bijhorende eigenvectoren? $(C_p X_i = \alpha_i X_i)$

2.3. **Voortzettingsmethodes.** Om met een iteratieve methode zoals Newton-Raphson alle n nulpunten van een willekeurige veelterm $p(x)$ van graad n te berekenen heeft men goede startwaarden nodig. Men kan dit probleem omzeilen via een voortzettingsmethode: wanneer $q(x)$ een n-de graadsveelterm is met gekende wortels, kan men de wortels volgen van de veelterm

$$z(x) = \lambda p(x) + (1 - \lambda)q(x)$$

waarbij men de parameter λ in kleine stapjes varieert van 0 tot 1.

3. Extra oefeningen

3.1. **Berekenen van eigenwaarden.** Een primitieve methode voor het berekenen van alle eigenwaarden van een matrix A bestaat uit het berekenen van alle nulpunten van de karakteristieke vergelijking $\det(\lambda I - A) = 0$. Deze methode is echter onstabiel. Ga na dat voor de diagonaalmatrix met de getallen $1, 2, \ldots, 20$ op de diagonaal (`A = diag(1:20)`) het commando `roots(poly(A))` onnauwkeurige resultaten oplevert (het commando `poly(.)` met als argument een matrix berekent de karakteristieke veelterm). Nochtans is het probleem goed geconditioneerd. Waaraan zijn de fouten te wijten als je weet dat zowel `poly` als `roots` stabiele algoritmen gebruiken.

Oefenzitting 6: Nulpunten van veeltermen.

Herhaling.

Konditie van het probleem $f(x) = 0$:

Als x^* een wortel is van $f(x) = 0$, hoe zal de wortel dan wijzigen bij een kleine wijziging op de gegevens?
Stel $\varepsilon g(x)$ een perturbatie op $f(x)$.
We zoeken nu de oplossing van
$h(x, \varepsilon) = 0$ met $h(x, \varepsilon) = f(x) + \varepsilon g(x)$
Noem deze oplossing $x^*(\varepsilon)$.
De gewenste oplossing $x^* = x^*(0)$.
$x^* - x^*(\varepsilon)$ is bij benadering $(\varepsilon g(x))/f'(x)$ als
$f'(x) \neq 0$ en ε klein.
We concluderen dat de konditie slecht is als $|f'(x)|$ klein is; in het extreme geval:
$|f'(x)| = 0$ --> x^* is een meervoudige wortel.

Opmerking: kleine $|f'(x)|$ komt vaak voor als er een aantal wortels zijn die dicht bij elkaar liggen.

Newton-Raphson:
$F(x^{(k)}) = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$

Binomium van Newton:
$(a + b)^n = \sum_{j=0}^{n} \binom{n}{j} a^{n-j} b^j$

Oplossingen van de oefeningen:

Oefening 1.1:

Hints: schrijf $p^-(c^-)$ uit
Pas binomium van Newton toe.
Verwaarloos 2e ordetermen.

Enkelvoudig nulpunt:

Gegeven:
$p(x) = \sum_{i=0}^{n} a_i x^{n-i} = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n$
$p(c) = 0$
$p^-(x) = \sum_{i=0}^{n} \bar{a}_i x^{n-i} = \bar{a}_0 x^n + \bar{a}_1 x^{n-1} + \ldots + \bar{a}_{n-1} x + \bar{a}_n$
$p^-(c^-) = 0$
$c^- = c + \Delta c$
$\forall i = 0..n: a_i^- = a_i + \Delta a_i$
$\sum_{i=0}^{n} \Delta a_i x^{n-i} = \Delta p(x)$
Uitwerking:
$p^-(c^-) = 0 \quad \Rightarrow \quad \sum_{i=0}^{n} (a_i + \Delta a_i) c^{-n-i} = 0$
$\Rightarrow \quad \sum_{i=0}^{n} (a_i + \Delta a_i)(c + \Delta c)^{n-i} = 0$

Opmerking: $(a + b)^n = \sum_{j=0}^{n} \binom{n}{j} a^{n-j} b^j$

$\Rightarrow \quad \sum_{i=0}^{n} (a_i + \Delta a_i) \sum_{j=0}^{n-i} \binom{n-i}{j} (c^{n-i} \Delta c^j) = 0$
$\Rightarrow \quad \sum_{i=0}^{n} (a_i + \Delta a_i) [\binom{n-i}{0} (c^{n-i} \Delta c^0) +$
$\binom{n-i}{1} (c^{n-i} \Delta c^1)$ (+ hogere ordetermen)]$= 0$

Eerste ordebenadering:

$\Rightarrow \quad \sum_{i=0}^{n} (a_i)(c^{n-i}) + \sum_{i=0}^{n} (\Delta a_i)(c^{n-i})$
$+ \sum_{i=0}^{n} (a_i + \Delta a_i)(c^{n-i-1})(n-i)(\Delta c) \approx 0$
$p(c) = 0$ (geg.)

2e ordetermen verwaarlozen.

$\Rightarrow \sum_{i=0}^{n} (\Delta a_i)(c^{n-i}) + \sum_{i=0}^{n} (a_i)(c^{n-i-1})(n-i)(\Delta c) \approx 0$
$\Rightarrow \sum_{i=0}^{n} (\Delta a_i)(c^{n-i}) + (\Delta c)\sum_{i=0}^{n} (n-i)(a_i)(c^{n-i-1}) \approx 0$
$p'(c) = \sum_{i=0}^{n} (n-i)(a_i)(c^{n-i-1})$
$\Rightarrow \Delta c = - \sum_{i=0}^{n} (\Delta a_i)(c^{n-i})/p'(c) = -\Delta p(c)/p'(c)$

(q.e.d.)

c moet een enkelvoudig nulpunt zijn, omdat anders $p'(c) = 0$ in de noemer.

Meervoudig nulpunt met meervoudigheid m:

Gegeven:
$p(\alpha) = 0, p'(\alpha) = 0, p''(\alpha) = 0, ..., p^{(m-1)}(\alpha) = 0, p^{(m)}(\alpha) \neq 0$ (want α is een m-voudig nulpunt).
$\bar{p}(x) = p(x) + \Delta p(x)$
$= \sum_{i=0}^{n} a_i x^{n-i} + \sum_{i=0}^{n} \Delta a_i x^{n-i}$
$\bar{p}(\bar{\alpha}) = 0$
$\bar{\alpha} = \alpha + \Delta\alpha$
Uitwerking:
$\bar{p}(\alpha + \Delta\alpha) = p(\alpha + \Delta\alpha) + \Delta p(\alpha + \Delta\alpha) = 0$
$\Rightarrow p(\alpha) + (p'(\alpha)\Delta\alpha)/1! + (p''(\alpha)\Delta\alpha^2)/2! + ... +$
$(p^{(m-1)}(\alpha)\Delta\alpha^{m-1})/(m-1)! + (p^{(m)}(\alpha)\Delta\alpha^m)/(m)!$
$+ 0(\Delta\alpha^{m+1}) + \Delta p(\alpha + \Delta\alpha) = 0$
Geg.: $p(\alpha) = 0, p'(\alpha) = 0, p''(\alpha) = 0, ...,$
$p^{(m-1)}(\alpha) = 0$
$\Rightarrow (p^{(m)}(\alpha)\Delta\alpha^m)/(m)! + \Delta p(\alpha + \Delta\alpha) = 0$
$\Rightarrow \Delta\alpha^m = - [\Delta p(\alpha + \Delta\alpha)*m!]/p^{(m)}(\alpha)$
Uitw.: $\Delta p(\alpha + \Delta\alpha) = \sum_{i=0}^{n} \Delta a_i(\alpha + \Delta\alpha)^{n-i}$
Verwaarlozen hogere ordetermen.
$\approx \sum_{i=0}^{n} \Delta a_i(\alpha)^{n-i}$
$\approx \Delta p(\alpha)$
$\Rightarrow \Delta\alpha^m \approx - [\Delta p(\alpha)m!]/p^{(m)}(\alpha)$
$\Rightarrow \Delta\alpha \approx - [(\Delta p(\alpha)m!)/p^{(m)}(\alpha)]^{1/m} = \delta$

Interpretatie van de formules:

$|\Delta c| \leq -\varepsilon|\sum_{i=0}^{n} c^{n-i}|/|p'(c)|$ met $|\Delta a_i| \leq \varepsilon$
Dus een slechte konditie als:
de afgeleide klein is in abs. w.
$(p'(c) \approx 0 << 1)$
$|\sum_{i=0}^{n} c^{n-i}|$ groot wordt.
$|\Delta\alpha|^m = \delta$ $\delta > 0$: m wortels uit δ
$\Rightarrow$ 2 reële wortels.
$\delta < 0$: m wortels uit δ
$\Rightarrow$ geen reële wortels.
Bovengrens van de fout $|\Delta\alpha| < \varepsilon$
$\Delta p(\alpha) = \sum_{i=0}^{n} \Delta a_i(\alpha)^{n-i}$
$\Rightarrow |\Delta p(\alpha)| \leq \varepsilon\sum_{i=0}^{n} |\alpha|^{n-i}$ met $|\Delta a_i| \leq \varepsilon$
$\Rightarrow |\Delta p(\alpha)| \leq \varepsilon\sum_{i=0}^{n}|\alpha|^i = [\varepsilon(1-|\alpha|^{n+1})]/(1-|\alpha|)$
$\Rightarrow |\Delta\alpha| \leq [(\varepsilon m!/p^{(m)}(\alpha))\ [(1-|\alpha|^{n+1})/(1-|\alpha|)]]^{1/m}$
Dus een slechte konditie als:
m groot wordt ($\Rightarrow$ m! zeer groot)
$|p^{(m)}(\alpha)| << 1$

Oefening 1.2: Illustraties met matlab.

Benodigde matlabcommando's:
roots, poly, real, imag
Meervoudig nulpunt:

```
nulpt = [2 2 2 2 2 2];
p1 = poly(nulpt);
p1 = p1';
res = roots(p1);
x_res = real(res);
y_res = imag(res);
plot(x_res, y_res, 'b*');
xlabel('Real');
ylabel('Imag');
title('Zesvoudig nulpunt 2.');
resb = [2; 2; 2; 2; 2; 2];
verschil = (res - resb);
absverschil = abs(verschil);
plot(1:6, absverschil, rb*');
title('Absolute fout.');
```

Figuren:

Wortels van de veelterm:

Opmerkingen:

1) Het probleem bij het controleren van de formule is, dat je niet weet welke fouten matlab maakt.
2) Je ziet wel dat delta alfa een m-de machtswortel is; de m resultaten van zo'n worteltrekking liggen steeds op een regelmatige m-hoek. Dit zie je duidelijk op de figuur van de wortels.

Veelterm van Wilkinson:

```
nulpt = 1:20;
p2 = poly(nulpt);
p2 = p2'
```

Merk op: de coëfficiënten van de polynoom verschillen in grootte-orde!

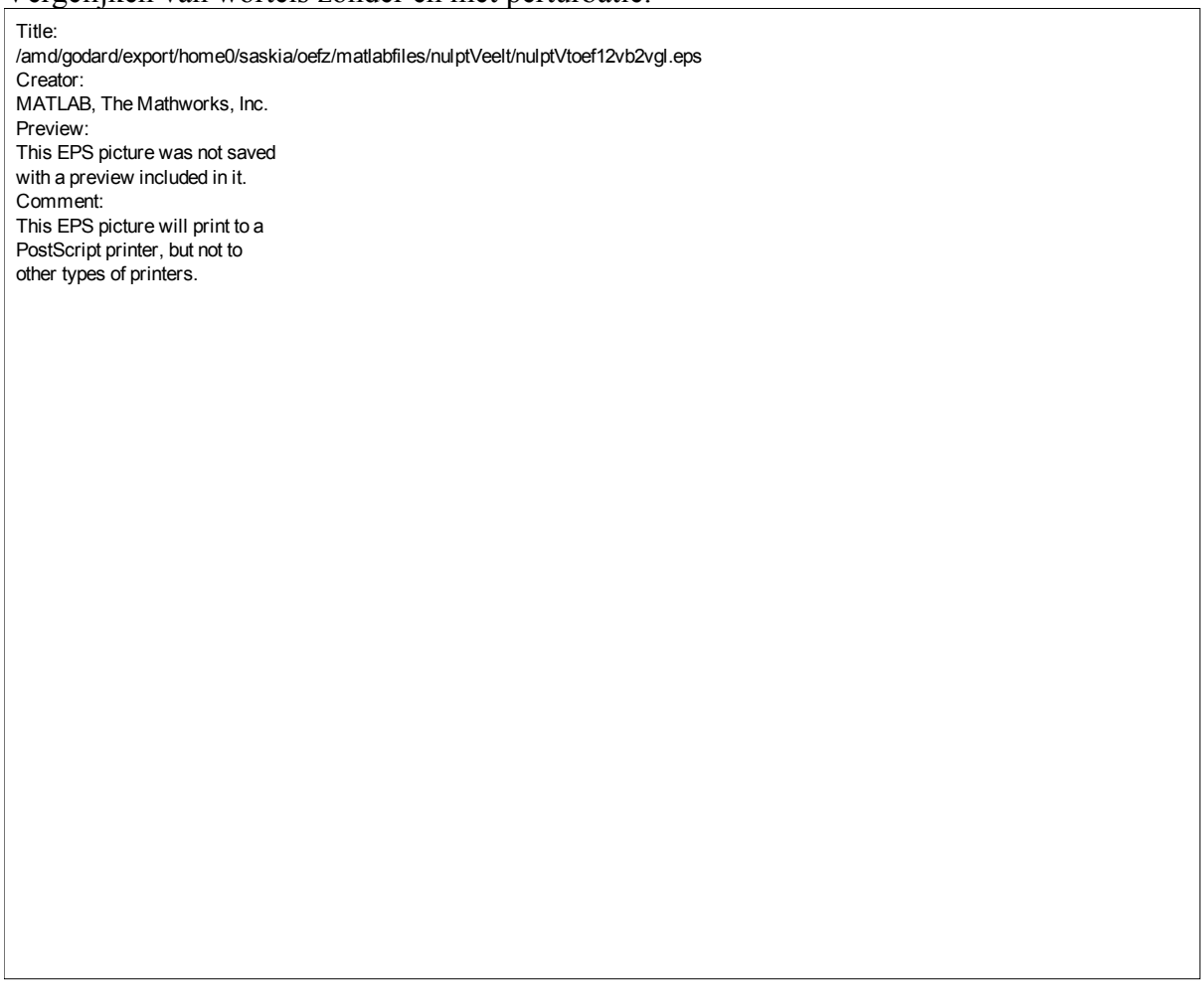
```
res2 = roots(p2);
x_res2 = real(res2);
y_res2 = imag(res2);
plot(x_res2, y_res2, 'b*');
xlabel('Real');
ylabel('Imag');
title('Veelterm van Wilkinson.');
```

Perturbatie op de polynoom:

```
p2b = p2;
p2b(2) = p2b(2) + 2.0e-8;
res2b = roots(p2b);
x_res2b = real(res2b);
y_res2b = imag(res2b);
plot(x_res2, y_res2, 'b*');
xlabel('Real');
ylabel('Imag');
title('Veelterm van Wilkinson met perturbatie.');
plot(x_res2, y_res2, 'r*');
hold;
plot(x_res2, y_res2, 'b+');
title('Veelterm van Wilkinson: vergelijken.');
```

Figuren:

Vergelijken van wortels zonder en met perturbatie:



Opmerkingen:

Ook hier is het weer een probleem dat je niet weet welke fouten matlab maakt bij de berekeningen.

Oefening 2.1: deflatie.

Nulpunten: x = 1, x = 10, x = 100, x = 1000,
x = 10000 en x = 100000.
Veelterm v/d vorm f(x) = (x - 1)(x-10)
(x-100)(x-1000)(x - 10000)
(x - 100000)g(x) met g(x) een restfunctie.

Nulpunten berekenen, grootste nulpunt wegdelen:

```
nulpt = [1 10 100 1000 10000 100000];
p3 = poly(nulpt);
res3 = roots(p3);
```

Horner toepassen op p3 met res3(1) als wortel => een aantal gevonden coëfficiënten voor p3b.

```
[p3b, rest] = nulptVthorner(p3, res3(1));
resb = roots(p3b);
```

horner toepassen op p3b met resb(1) als wortel => een aantal gevonden coëfficiënten voor p3c.

```
[p3c, rest] = nulptVthorner(p3b, res(1));
resc = roots(p3c);
```

horner toepassen op p3c met resc(1) als wortel => een aantal gevonden coëfficiënten voor p3d.

```
[p3d, rest] = nulptVthorner(p3c, res(1));
resd = roots(p3d);
```

horner toepassen op p3d met resd(1) als wortel => een aantal gevonden coëfficiënten voor p3e.

```
[p3e, rest] = nulptVthorner(p3d, res(1));
rese = roots(p3e);
```

horner toepassen op p3e met rese(1) als wortel => een aantal gevonden coëfficiënten voor p3f.

```
[p3f, rest] = nulptVthorner(p3e, res(1));
resf = roots(p3f);
wortels = [res(1); resb(1); resc(1); resd(1); rese(1); resf];
exact = flipud(nulpt');
absfout = abs(exact - wortels);
plot(1:6, absfout, 'b*');
title('Absolute fout.');
hold;
plot(1:6, absfout, 'r');
```

Kleinste nulpunt wegdelen:

```
p3 = poly([1 10 100 1000 10000 100000]);
res = roots(p3);
```

horner toepassen op p3 met res3(size(res3)) als wortel => een aantal gevonden coëfficiënten voor p3b.

```
[p3b, rest] = nulptVthorner(p3, res3(size(res3)));
res3b = roots(p3b);
```

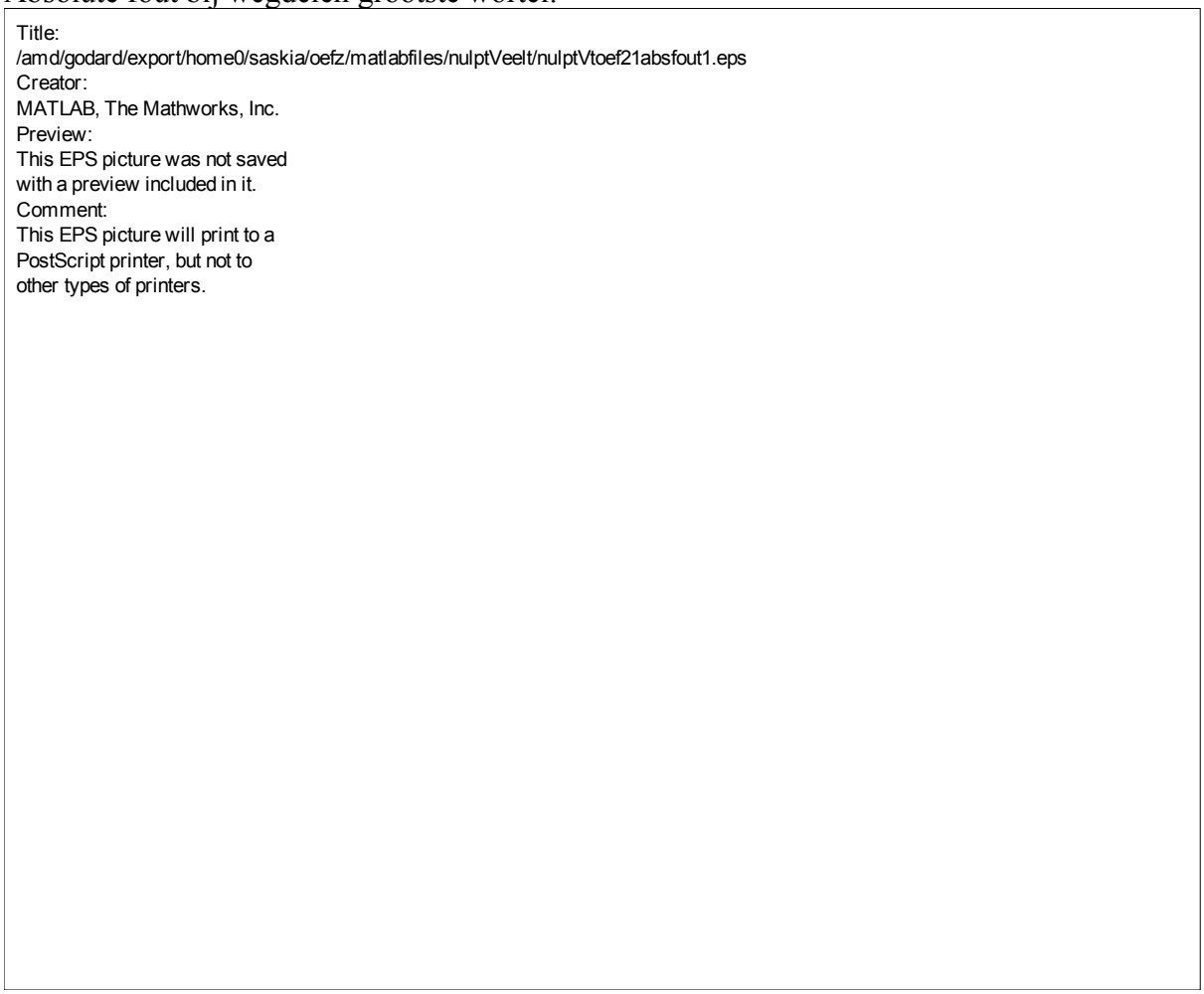
horner toepassen op p3b met res3b(size(res3b)) als wortel => een aantal gevonden coëfficiënten voor p3c.

```
[p3c, rest] = nulptVthorner(p3b, res3b(size(res3b)));
res3c = roots(p3c);
```

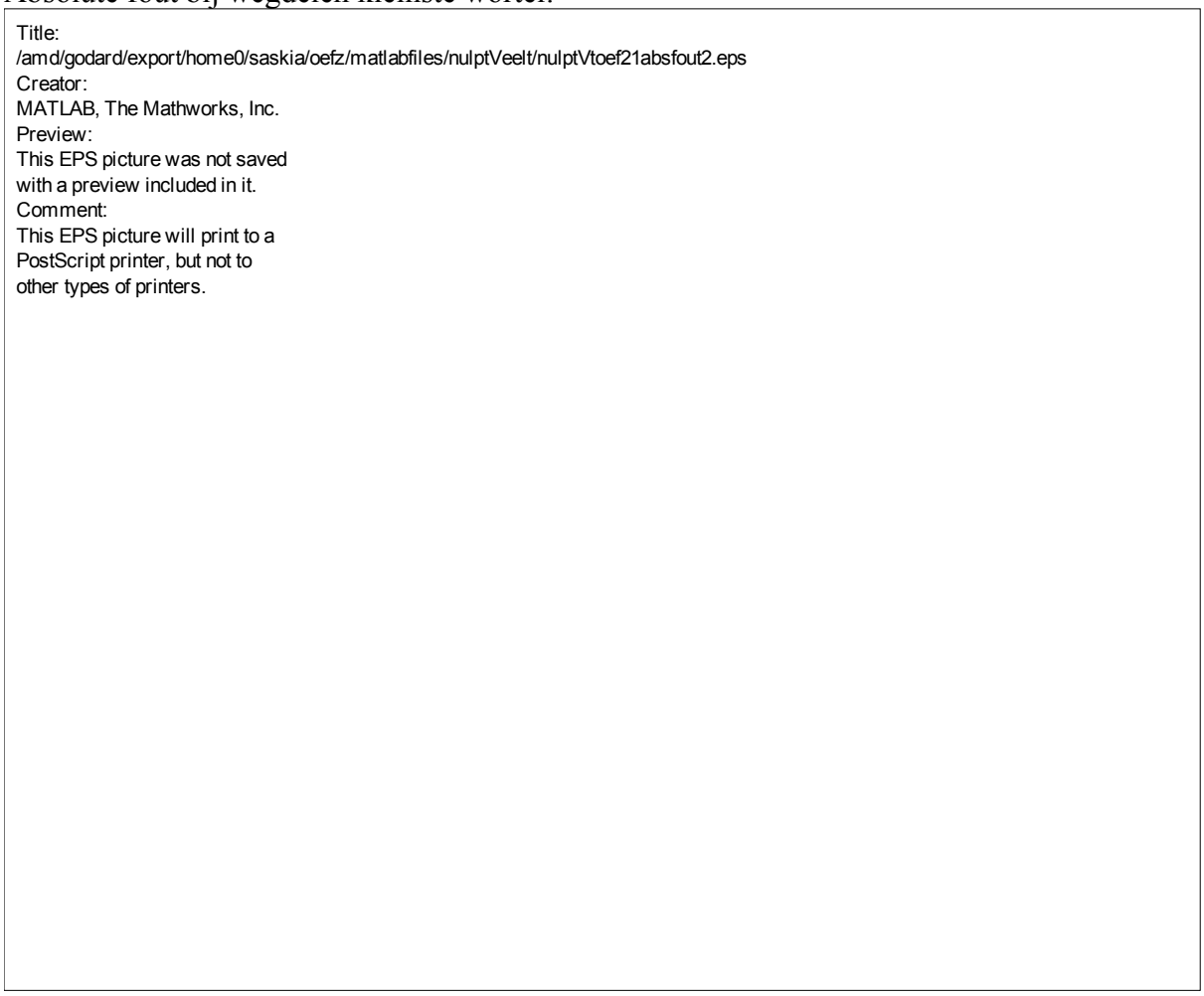
```
horner toepassen op p3c met res3c(size(res3c)) als wortel => een aantal gevonden
coëfficiënten voor p3d.
[p3d, rest] = nulptVthorner(p3c, res3c(size(res3c)));
resd = roots(p3d);
horner toepassen op p3d met res3d(size(res3d)) als wortel => een aantal gevonden
coëfficiënten voor p3e.
[p3e, rest] = nulptVthorner(p3d, res3d(size(res3d)));
res3e = roots(p3e);
horner toepassen op p3e met res3e(size(res3e)) als wortel => een aantal gevonden
coëfficiënten voor p3f.
[p3f, rest] = nulptVthorner(p3e, res3e(size(res3e)));
resf = roots(p3f);
wortels2 = [res3(size(res3, 1));
res3b(size(res3b, 1)); res3c(size(res3c, 1)); res3d(size(res3d, 1)); res3e(size(res3e, 1)); res3f];
exact = (nulpt');
absfout2 = abs(exact - wortels2);
plot(1:6, absfout2, 'b*');
title('Absolute fout.');
hold;
plot(1:6, absfout2, 'r');
verschil = absfout - flipud(absfout2);
plot(1:6, verschil, 'b*');
title('Verschil in absolute fout per wortel.')
relfout = absfout./flipud(exact)
relfout2 = absfout2./exact
procentuele_fout = 100*relfout
procentuele_fout2 = 100*relfout2
semilogx(exact, flipud(procentuele_fout), 'r', exact, procentuele_fout2, 'b')
title('Deflatie: vergelijken')
ylabel('Procentuele fout (= 100*relatieve fout)')
xlabel('Wortels van klein naar groot.')
```

Figuren:

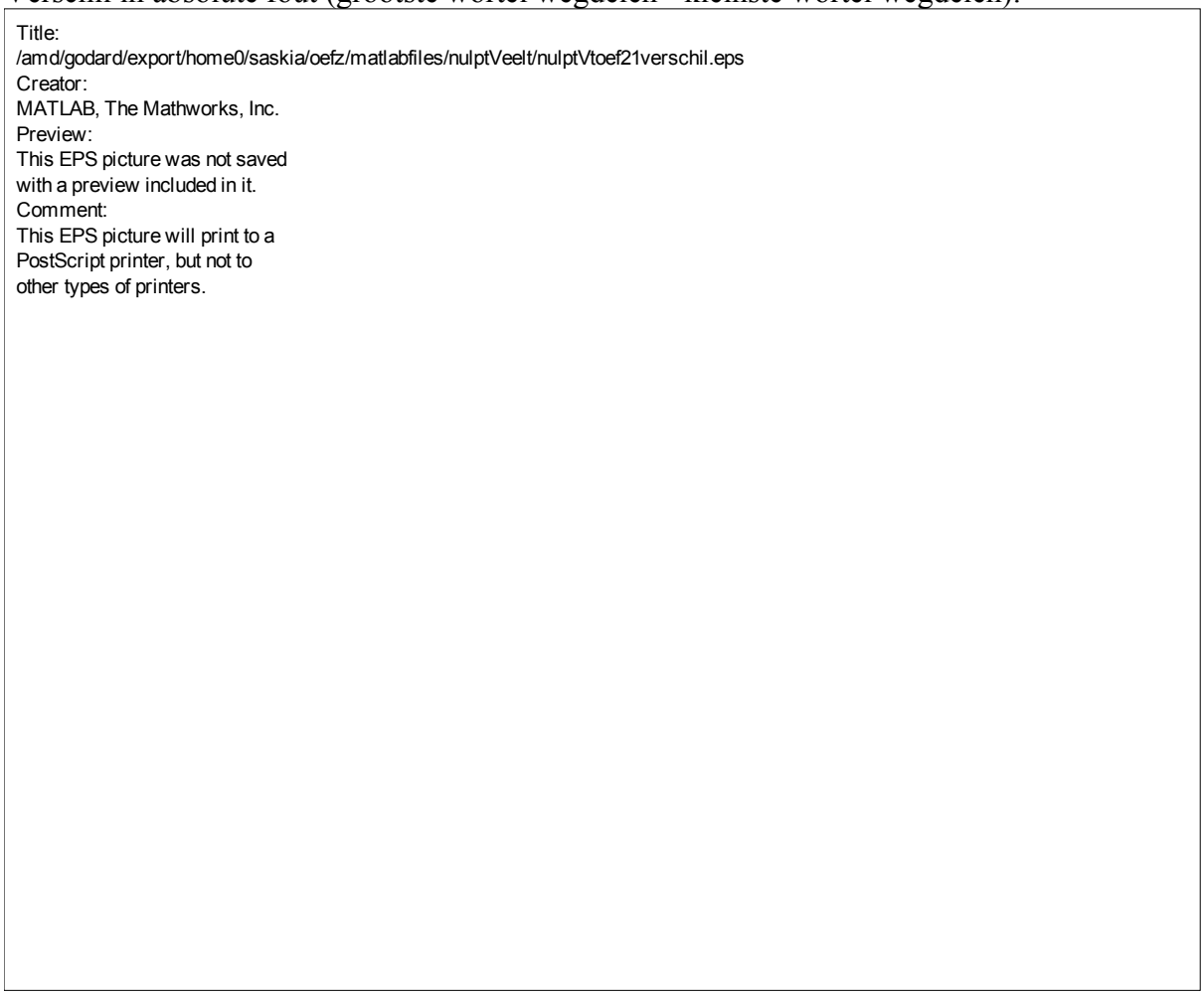
Absolute fout bij wegdelen grootste wortel:



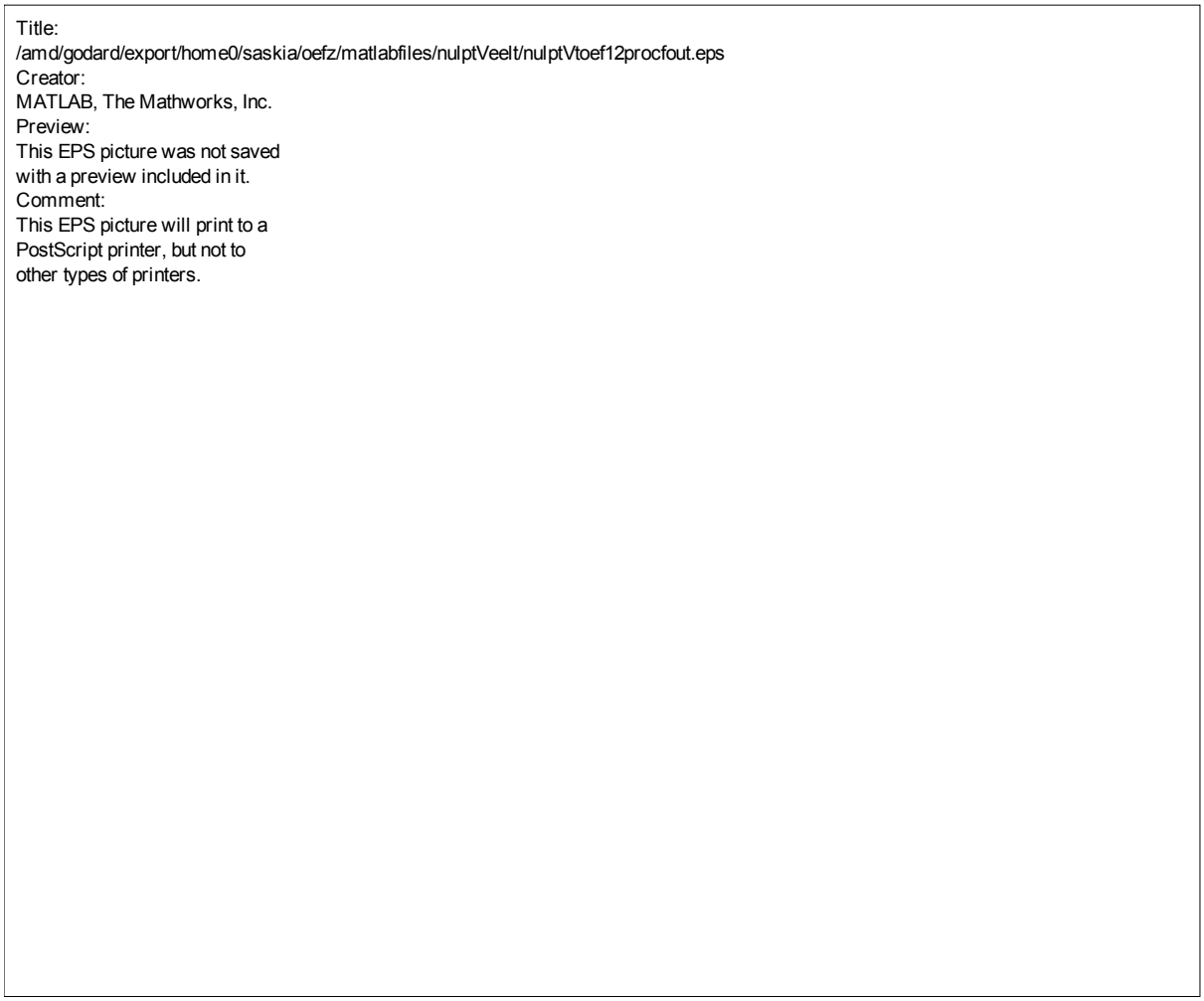
Absolute fout bij wegdelen kleinste wortel:



Verschil in absolute fout (grootste wortel wegdelen - kleinste wortel wegdelen):



Procentuele fout:



Verklaring:

Vooral voor de kleinere wortels is de tweede methode beter (het verschil in fout is > 0, dus de abs. fout op de wortel is groter als de grootste eerst weggedeeld wordt).
Dit komt omdat het wegdelen van kleinere wortels kleinere (abs.) fouten op de polynoom van 1 graad lager introduceert; kleinere wortels worden bij wegdelen van groot naar klein overschaduwd door de grote (abs.) fout op de grotere wortels (er kan een (abs.) verschuiving van de kleinere wortels optreden).

Oefening 2.2.1:

Voorbeeld: n = 4:

$$C_p = \begin{bmatrix} -a_1 & -a_2 & -a_3 & -a_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\det(C_p - \lambda I) = 0$

$$\Leftrightarrow \begin{vmatrix} -a_1-\lambda & -a_2 & -a_3 & -a_4 \\ 1 & 0-\lambda & 0 & 0 \\ 0 & 1 & 0-\lambda & 0 \\ 0 & 0 & 1 & 0-\lambda \end{vmatrix} = 0$$

Ontwikkelen naar 1e rij.

$$\Leftrightarrow (-a_1-\lambda)\begin{vmatrix} -\lambda & 0 & 0 \\ 1 & -\lambda & 0 \\ 0 & 1 & -\lambda \end{vmatrix} + a_2\begin{vmatrix} 1 & 0 & 0 \\ 0 & -\lambda & 0 \\ 0 & 1 & -\lambda \end{vmatrix} - a_3\begin{vmatrix} 1 & -\lambda & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\lambda \end{vmatrix} + a_4\begin{vmatrix} 1 & -\lambda & 0 \\ 0 & 1 & -\lambda \\ 0 & 0 & 1 \end{vmatrix} = 0$$

$\Leftrightarrow (-a_1-\lambda)(-\lambda)^3 + a_2(\lambda)^2 - a_3(-\lambda) + a_4 * 1 = 0$

$\Leftrightarrow \lambda^4 + a_1\lambda^3 + a_2\lambda^2 + a_3\lambda + a_4 = 0$

Hernoem λ naar x:

$\Leftrightarrow x^4 + a_1x^3 + a_2x^2 + a_3x + a_4 = 0$

$\Leftrightarrow p(x) = 0$ (q.e.d.)

Algemeen:

$$C_p = \begin{bmatrix} -a_1 & -a_2 & -a_3 & \dots & -a_n \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}$$

$\det(C_p - \lambda I) = 0 \iff$

$$\begin{vmatrix} -a_1-\lambda & -a_2 & -a_3 & \dots & -a_n \\ 1 & 0-\lambda & 0 & \dots & 0 \\ 0 & 1 & 0-\lambda & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0-\lambda \end{vmatrix} = 0$$

Ontwikkelen naar 1e rij.

$$\begin{aligned} \iff & (-1)^{1+1}(-a_1-\lambda)(-1)^{n-1}\lambda^{n-1} + \\ & (-1)^{1+2}(-a_2)(-1)^{n-2}\lambda^{n-2} + \\ & (-1)^{1+3}(-a_3)(-1)^{n-3}\lambda^{n-3} + \\ & \dots \\ & (-1)^{1+n}(-a_n)(-1)^{0}\lambda^{0} = 0 \end{aligned}$$

n oneven:

$$\begin{aligned} & \iff -\lambda^n - a_1\lambda^{n-1} - a_2\lambda^{n-2} - a_3\lambda^{n-3} - \dots - a_n = 0 \\ & \iff \lambda^n + a_1\lambda^{n-1} + a_2\lambda^{n-2} + a_3\lambda^{n-3} + \dots + a_n = 0 \\ & \iff p(\lambda) = 0 \end{aligned}$$

n even:

$$\begin{aligned} & \iff \lambda^n + a_1\lambda^{n-1} + a_2\lambda^{n-2} + a_3\lambda^{n-3} + \dots + a_n = 0 \\ & \iff p(\lambda) = 0 \end{aligned}$$

Hernoem λ naar x:

$\iff p(x) = 0$

(q.e.d.)

Oefening 2.2.2:

Opmerking: Eigenvectoren zijn bepaald op een constante factor na.

$C_p X = \alpha X$ (met $\alpha = \alpha_i$ en $X = X_i \; \forall i = 1..n$) <=>

$$\begin{bmatrix} -a_1 & -a_2 & -a_3 & \dots & -a_n \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & & 0 \\ | & \backslash & \backslash & \backslash & | \\ | & & \backslash & \backslash & | \\ | & & & \backslash & | \\ | & & & & | \\ | & & & & | \\ | & & & & | \\ 0 & \dots & & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \alpha x_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \alpha x_n \end{bmatrix}$$

$$\begin{bmatrix} -a_1x_1 - a_2x_2 - a_3x_3 - \dots - a_nx_n \\ x_1 \\ x_2 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \alpha x_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \alpha x_n \end{bmatrix}$$

Kies $x_n = 1$

Dan: $x_1 = -(\sum_{i=2}^{n} a_i \alpha^{n-i})/(a_1+\alpha)$, $x_2 = \alpha^{n-2}$,
$x_3 = \alpha^{n-3}$, $x_4 = \alpha^{n-4}$, ..., $x_{n-1} = \alpha^1$

Dus:

$$X = \begin{bmatrix} \alpha^n \\ \alpha^{n-2} \\ \alpha^{n-3} \\ \dots \\ \dots \\ \alpha \\ 1 \end{bmatrix}$$

$$\text{en } (\sum_{i=2}^{n} a_i \alpha^{n-i})/(a_1+\alpha^n) = 0$$

(met $a_1+\alpha^n \neq 0$)

Als $\alpha_i = 0$ en $a_n = 0$, dan kies $x_n = 1$ en X is de eenheidsvector $(0 \dots 0 \; 1)^T$.
Als $\alpha_i \neq 0$ en $x_n = 0$, dan is de oplossing van het stelsel de nulvector $(0 \dots 0)^T$; dit is geen eigenvector.

Oefening 2.3:

Opmerking:
Een veeltermfunctie van graad n heeft n (complexe) nulpunten. Matlab kan rekenen met complexe getallen => Newton-Raphson toepassen kan deze n nulpunten vinden.
Werking:
q(x) is een veelterm van dezelfde graad n als p(x); je kent de wortels van q(x).
Het toepassen van een voortzettingsmethode is het geleidelijk overgaan van q(x) naar p(x) volgen; hierbij kijk je dan naar de wortels. Omdat je de wortels van q(x) kent, liggen de wortels van $z(x) = \lambda p(x) + (1 - \lambda)q(x)$ in elke iteratiestap dicht bij de waarden in de vorige iteratiestap. Hierdoor heb je in elke stap goede startwaarden om de wortels te berekenen (a.h.v. Newton-Raphson). Wanneer uiteindelijk $\lambda = 1$, zul je de nulpunten van p(x) hebben gevonden.
Voorbeelden: zie kopies slides.
Uitwerking in matlab hoef je niet te doen!!! Aan dit onderwerp is immers een heel doctoraat gewijd om een fatsoenlijke implementatie te maken.

Opmerkingen:
Als de startwaarden complex zijn, kan deze methode ook convergeren naar reële nulpunten; je moet er dan wel op letten dat je implementatie 2 verschillende reële wortels gaat berekenen en niet langs beide paden naar dezelfde wortel convergeert.
Zelfs indien p en q beide n verschillende wortels hebben, kan het zijn dat niet alle wortels van p worden gevonden; in de tussenstappen kan het gebeuren dat er wortels samenvallen zodat meerdere keren dezelfde wortel van p gevonden zal worden.

Oefening 3.1:

Uitwerking op papier:
$\det (A - \lambda I) = 0 \iff$
$(1-\lambda)(2-\lambda)(3-\lambda)(4-\lambda)(5-\lambda)(6-\lambda)(7-\lambda)$
$(8-\lambda)(9-\lambda)(10-\lambda)(11-\lambda)(12-\lambda)(13-\lambda)$
$(14-\lambda)(15-\lambda)(16-\lambda)(17-\lambda)(18-\lambda)(19-\lambda)$
$(20-\lambda) = 0$

(De determinant van een diagonaalmatrix is het product van de diagonaalelementen).

Dus: $\lambda_1 = 1, \lambda_2 = 2, ..., \lambda_i = i, ..., \lambda_{20} = 20$ zijn de eigenwaarden van A.

Mogelijke commando's:
```
A = diag(1:20);
p = poly(A);
res = roots(p);
resUD = flipud(res);
fout = diag(A) - resUD;
x = 1:20;
plot(x, fout, 'r');
[eigvect, eigw] = eig(A);        % test.
```

Verklaring voor de fouten:

Er gebeuren 2 deelstappen bij het oplossen:

a) het berekenen van de karakteristieke veelterm gegeven een matrix.
b) het zoeken van de wortels van de karakteristieke veelterm.

De tweede deelstap is slecht gekonditioneerd:
Stel de matrix A heeft eigenwaarden $\lambda_1, ..., \lambda_n$ en stel σ_i een benadering voor 1 van de eigenwaarden λ_i van A.
Als σ_i --> λ_i dan det(σ_iI -A) --> det(λ_iI -A) en det(λ_iI - A) = 0.
Ook al zijn beide algoritmes stabiel, toch kunnen er kleine fouten gemaakt worden (def. van stabiliteit laat toe dat er eventueel wel een paar kleine foutjes kunnen gemaakt worden in de eerste stappen van het algoritme).
Deze kleine afrondingsfoutjes uit stap 1 worden wegens de slechte konditie van stap 2 opgeblazen zodat er uiteindelijk vrij grote fouten zitten op de berekende resultaten.

Bijlage : resultaten in matlab van oef. 3.1:

```
>> format long
>> A = diag(1:20);
>> p = poly(A);
>> res = roots(p);
>> resUD = flipud(res)

resUD =

   0.99999999999988
   2.00000000001614
   2.99999999915563
   4.00000002599858
   4.99999951140530
   6.00000582206711
   6.99995355567939
   8.00026197591442
   8.99890749294903
  10.00353481281963
  10.99108456608065
  12.01861449606569
  12.97005012199360
  14.03978739416597
  14.95972711707775
  16.03020258823751
  16.98285390795503
  18.00632543231030
  18.99853914055759
  20.00015203955076

>> fout = diag(A) - resUD

fout =

   0.00000000000012
  -0.00000000001614
   0.00000000084437
  -0.00000002599858
   0.00000048859470
  -0.00000582206711
   0.00004644432061
  -0.00026197591442
   0.00109250705097
  -0.00353481281963
   0.00891543391935
  -0.01861449606569
   0.02994987800640
  -0.03978739416597
   0.04027288292225
```

```
  -0.03020258823751
   0.01714609204497
  -0.00632543231030
   0.00146085944241
  -0.00015203955076

>> x = 1:20;
>> plot(x, fout, 'r');
>> title('Oef. 3.1: fout op de eigenwaarden.')
>> absfout = abs(fout);
>> plot(x, absfout, 'b');
>> title('Oef. 3.1: abs. fout op de eigenwaarden.')
>>
```

Figuren:

Fout op de gevonden eigenwaarden (oef. 3.1):

Abs. fout op de gevonden eigenwaarden (oef. 3.1):