Norges teknisk–naturvitenskapelige universitet
Institutt for Matematiske Fag

[S]=T. Sauer, Numerical Analysis, Second International Edition, Pearson, 2014

"Teorioppgaver"

1 Oppgave 6, Avsnitt 1.1, s. 29, [S]

Solution: One can see that the method produces the sequence of intervals $[-2, 1]$, $[-1/2, 1]$, $[-1/2, 1/4]$, ..., $[-1/2^k, 1/2^{k\pm1}]$. Thus the intervals bracket the value 0, which is not a root though. This happens because the function under consideration is not continuous on the initial interval $[-2, 1]$.

2 Oppgave 2, Avsnitt 1.2, s. 40, [S]

Solution:

(a) $(x + 6)/(3x - 2) = x$ iff $x + 6 = 3x^2 - 2x$ (and $3x - 2 \neq 0$) iff $3x^2 - 3x - 6 = (x + 1)(3x - 6)$ iff $x = -1$ or $x = 2$.

(b) $(8 + 2x)/(2 + x^2) = x$ iff $8 + 2x = 2x + x^3$ (and $2 + x^2 \neq 0$) iff $x^3 = 8$ iff $x = 2$ (if we only consider real roots). Otherwise $-1 \pm i\sqrt{3}$ will also do the trick.

(c) $x^5 = x$ iff $x(x^4 - 1) = 0$ iff $x = 0$ or $x = \pm 1$. Complex roots also include $x = \pm i$.

3 Oppgave 20, Avsnitt 1.2, s. 42, [S]

Solution: Let $g(x) = wx + (1 - w)A/x^2$, $0 < w < 1$. Its fixed points are $(1 - w)A + wx^3 = x^3$, $x \neq 0$, or $r = A^{1/3}$ for $A \neq 0$.

The fastest local convergence of the fixed point iteration will be obtained when $g'(r) = w - 2(1 - w)A/r^3 = 3w - 2$ has the smallest absolute value, that is, when $w = 2/3$.

4 Oppgave 1, Avsnitt 1.3, s. 50, [S]

Solution: In all cases, the forward error is $|0.74 - 3/4| = 0.01$.

The backward error is:

(a) $|4*0.74-3| = 0.04$; (b) $|(4*0.74-3)^2| = 0.0016$; (c) $|(4*0.74-3)^3| = 6.4 \cdot 10^{-5}$; (d) $|(4 * 0.74 - 3)^{1/3}| \approx 0.3420$

5 Oppgave 12, Avsnitt 1.4, s. 59, [S]

Solution: The Newton's iteration in this case is $x_{k+1} = x_k - (1/x_k)/(-1/x_k^2) = x_k + x_k = 2x_k$. Thus given $x_0 = 1$, $x_{50} = 2^{50}$.

"Computeroppgaver"

6 Oppgave 7, Avsnitt 1.1, s. 30, [S]

Solution:

Up to six digits $x = 9.70830$. The backward error is $-5.025448{\cdot}10^{-4}$. See `oppgave_1_1_7.py`

7 Oppgave 6 (b), Avsnitt 1.2, s. 43, [S]

Solution:

For example: $x = g_1(x) = \exp(x-2) + x^3$ (converges to $x_1 \approx 1.63823 \cdot 10^{-1}$ with linear rate $S \approx 2.399371 \cdot 10^{-1}$ and $g_1'(x_1) =\approx 2.399381 \cdot 10^{-1}$); $x = g_2(x) = (x - \exp(x-2))^{1/3}$ (converges to $x_2 \approx 7.889405 \cdot 10^{-1}$ with linear rate $S \approx 3.760119 \cdot 10^{-1}$ and $g_2'(x_2) \approx 3.760105 \cdot 10^{-1}$); $x = g_3(x) = -1 - \exp(x-2)/(x^2 - x)$ (converges to $x_3 \approx -1.023482$ with linear rate $S \approx 5.802971 \cdot 10^{-2}$ and $g_3'(x_3) \approx -5.803052 \cdot 10^{-2}$.

See `oppgave_1_2_6.py`

8 Oppgave 1, Avsnitt 1.3, s. 51, [S]

Solution: $f(0) = \sin 0 - 0 = 0$, $f'(0) = \cos 0 - 1 = 0$, $f''(0) = -\sin 0 = 0$, $f'''(0) = -\cos(0) \neq 0$. Therefore the multiplicity of the root is 3.

```
import math
from scipy.optimize import fsolve

def f(x):
    return math.sin(x)-x

r = fsolve(f, 0.1)

fwd_error = abs(r-0)
bckwd_error = abs(f(r))

print('Forward error: %e, backward error: %e' % (fwd_error,bckwd_error))
```

produces the output

```
Forward error: 2.137514e-08, backward error: 0.000000e+00
```

9 **a)** Skriv en Matlab funksjon som gitt startverdien x_0 og toleransen δ løser likningen $x^3 = 1$ ved bruk av Newtons metoden. Tjekk om metoden konvergerer kvadratisk.

b) Likningen $x^3 = 1$ har tre *komplekse* løsninger: 1 og $(-1 \pm i\sqrt{3})/2$. For startverdiene i boksen $-2 \le \mathrm{re}(x_0) \le 2$, $-2 \le \mathrm{im}(x_0) \le 2$, plot punkter med fire forskjellige farver, som avhenger fra hvilken løsning Newtons metoden konvergerer til, eller om den ikke konvergerer. (Til denne oppgaven kan du bruke visualisering kode `coloring.py` fra wiki-siden.)

Solution:

a) By starting the Newton's iteration from random real initial points we can measure the ratios e_{k+1}/e_k^2 for various iterations k. One observes behaviour like this:

```
iter =  1, e1=1.289931e-01, e1/e0^2 =   1.585234104690e+00
iter =  2, e1=1.417680e-02, e1/e0^2 =   8.520114556424e-01
iter =  3, e1=1.972487e-04, e1/e0^2 =   9.814269693620e-01
iter =  4, e1=3.889682e-08, e1/e0^2 =   9.997370660616e-01
iter =  5, e1=1.554312e-15, e1/e0^2 =   1.027330344408e+00
```

Note that the rate predicted by the calculus (formula (1.24) in the book) $f''(x)/(2f'(x)) = 6x/(2 \cdot 3x^2) = 1/x$, which at the real root $x = 1$ evaluates to 1.

b) See `newtonx3.py`