Norges teknisk–naturvitenskapelige universitet
Institutt for matematiske fag

TMA4320
Introduksjon til vitenskapelige beregninger
Vår 2022

Øving 7: Repetisjonsøving

Under finner dere et lite utvalg oppgaver å bryne seg på før eksamen. Det er også lagt inn noen skisser til mulige programmeringsoppgaver. Oppgavene er hentet og delvis modifisert fra tidligere kurs jeg har gitt, så noen er på engelsk, og noen på norsk.

Det blir ikke gitt ut noe løsningsforslag til disse oppgavene.

Det fins også et godt sett med gjennomarbeidede eksempler i læreboka til Sauer.

Ikke-lineære ligninger

$\boxed{1}$ Newton's method for a scalar equation $f(x) = 0$ is given by

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, 2\dots.$$

a) Apply Newton's method on the equation $\sin(x) - 1/2 = 0$. Do so by filling in the missing line in the code:

```
xk = 1
for n in range(5):
xk =
print('xk = {:.8f}, error = {:.3e}'.format(xk, pi/6-xk))
```

and write down the first to lines of output.

b) If correctly done, the output of the code above would be:

```
xk = 0.36800013, error = 1.556e-01
xk = 0.51831364, error = 5.285e-03
xk = 0.52359079, error = 7.990e-06
xk = 0.52359878, error = 1.843e-11
xk = 0.52359878, error = -1.110e-16
```

Confirm (numerically) that the convergence is quadratic.

c) If instead Newton's method is applied to the equation $f(x) = (\sin(x) - 1/2)^2$, the output of the correspondent code is:

```
xk = 0.68400007, error = -1.604e-01
xk = 0.59891000, error = -7.531e-02
xk = 0.56032263, error = -3.672e-02
xk = 0.54175332, error = -1.815e-02
xk = 0.53262696, error = -9.028e-03
xk = 0.52810092, error = -4.502e-03
```

The convergence is no longer quadratic. Why?

d) Let r be the (unknown) solution of the equation $f(x) = 0$. Prove that the error $e_k = r - x_k$ of Newton's method satisfy:

$$e_{k+1} = -\frac{f''(\xi_k)}{2f'(x_k)} e_k^2,$$

where ξ_k is some point between r and x_k.

$\boxed{2}$ Newton's method for system of equations can be implemented as follows:

```
import numpy as no
def my_problem(x):
f = # Insert code
jac =  # Insert code
return f, jac

x = np.array([1,1])
for k in range(10):
f, jac =  my_problem(x)
# Insert code
print('k ={:3d}, x = '.format(k+1), x)
```

a) Apply the method (fill in the missing lines in the code) on the system of equation `myproblem`:

$$5x_1^2 - x_2^2 = 0$$
$$x_2 - \frac{1}{4}(\sin(x_1) + \cos(x_2)) = 0.$$

and write out the first line of output.

b) Include an appropriate stopping criterium.

Interpolasjon

$\boxed{3}$ **a)** Given the data:

x_i	0	1	-1	3
$f(x_i)$	2	-1	-3	17

Find the interpolation polynomial of lowest possible degree interpolating these points.

b) Show that the polynomials

$$p(x) = 2 - x - 4x^2 + 2x^3 \quad \text{and} \quad q(x) = 2 + 2x - 5x^2 - x^3 + x^4$$

both interpolate the data points from **a)**. How will this fit with the existence and uniqueness theorem for interpolation polynomials?

c) Let $p_n(x)$ be the polynomial interpolating a function $f(x)$ in $n+1$ distinct points x_i, $i = 0, 1, 2, \ldots, n$ in the interval $[-1, 1]$. The interpolation error is

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \qquad \xi(x) \in [-1, 1].$$

Explain why the Chebyshev distributed nodes usually will give smaller errors than equidistributed nodes.

$\boxed{4}$ *Inverse interpolation* This is a strategy that can be used to find approximations to solutions of an equation $f(x) = 0$.

Given a sufficiently continuous function $f(x)$, with a root r in some interval $[a, b]$. Choose this interval sufficiently small to ensure f to be strict monotonically increasing or decreasing at $[a, b]$. Under these conditions f is invertible on the interval, in the sense that

$$y = f(x) \qquad \Rightarrow \qquad x = f^{-1}(y).$$

In particular:

$$f(r) = 0 \qquad \Rightarrow \qquad r = f^{-1}(0).$$

The strategy is then: Choose $n + 1$ distinct data points (x_i, y_i) in the interval $[a, b]$ and find the interpolation polynomial $p_n(y)$ satisfying

$$p_n(y_i) = x_i.$$

In this case, $p_n(y) \approx f^{-1}(y)$, and $r \approx p_n(0)$.

a) Let $f(x) = x^2 - 3$, og $[a, b] = [1, 3]$. As nodes, choose $x_0 = 0, x_1 = 1, x_2 = 3$ and use the idea above to find an approximation to the solution of $f(x) = 0$. How close to the exact solution is the approximation?

To get a better approximation, add the node $x_3 = 3/2$. Will this give a better result?

b) Repeat the example above, but now with $n+1$ uniformly distributed nodes over the interval [0]. Use the functions `divdiff` and `newton_interpolation`. Choose $n = 2$ (to control your hand calculations), $4, 8$ and 16. Find the approximation in each case, as well as the error.

Integrasjon

$\boxed{5}$ **a)** Given the nodes $t_0 = 0$, $t_1 = 2/3$. Construct a quadrature formula

$$Q(0, 1) = a_0 f(t_0) + a_1 f(t_1)$$

approximating the integral

$$I(0, 1) = \int_0^1 f(t)dt.$$

b) Find the degree of precision for the quadrature

$$Q(0,1) = \frac{1}{4}f(0) + \frac{3}{4}f(\frac{2}{3}).$$

In the next question, we are interested in finding an approximation to the integral

$$I(a,b) = \int_a^b f(x)dx.$$

c) Based on the quadrature formula from point **b)**, construct a composite quadrature formula:

$$Q_n(a,b) = h\sum_{i=0}^{n-1}\big(\frac{1}{4}f(x_i) + \frac{3}{4}f(x_i + \frac{2}{3}h)\big)$$

where $h = (b-a)/n$ and $x_i = a + ih$.

d) (Coding) Implement the quadrature formula. Test it on the integral $\int_0^1 e^{x^2}dx$. Calculate the error for different values of n. In practice, fill in the open slots in the following code.

```
from numpy import *

# --- The quadrature rule ---
def Q(f, a, b, n):
# Intput:
#     f: the integrand
#     a, b : the integration interval
#     n: Number of subintervals
# Output:
#     The approximation of the integral by the

# Fill in the rest here.

# --- Test the rule ---
def f(x):
return exp(x**2)

I_exact = 1.4626517459071816088
n = 1
for k in range(5):
Q_eval = Q(f, 0, 1, n)
error = I_exact - Q_eval
print("h = {:.4f}, result = {:.8f}, error = {:.3e}".format(1/n, Q_eval, error
n = 2*n
```

e) The output from the code above should be something like:

```
h = 1.0000, Result = 1.41971762, error = 4.293e-02
h = 0.5000, Result = 1.45554641, error = 7.105e-03
h = 0.2500, Result = 1.46167229, error = 9.795e-04
h = 0.1250, Result = 1.46252515, error = 1.266e-04
h = 0.0625, Result = 1.46263572, error = 1.602e-05
```

From this, what would you expect the order of the composite quadrature to be?

f) Now repeat the experiment on $\int_0^1 \sqrt(x)dx$. What do you observe concerning the error as a function of n? Why do you think this happens (see also the next point).

g) Given (or prove that) the error of the quadrature formula using one step over a small interval is given by

$$I(a,b) - Q_1(a,b) = \frac{1}{216} f^{(3)}(\xi)(b-a)^4, \qquad \xi \in (a,b).$$

Explain how this information can be used to derive an estimate for the error in $Q_1(a,b)$ for some arbitrary function f.

h) Explain the idea of adaptive integration.

Ordinære differensialligninger

$\boxed{6}$ Gitt startverdiproblemet

$$\mathbf{y}' = \mathbf{f}(t,\mathbf{y}), \qquad \mathbf{y}'(t_0) = \mathbf{y}_0.$$

hvor $\mathbf{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$. Trapesmetoden for dette problemet er gitt ved

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\big(\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) + \mathbf{f}(t_n, \mathbf{y}_n)\big).$$

med $h = t_{n+1} - t_n$.

a) Vi vil nå anvende trapesmetoden til å løse følgende system:

$$\begin{aligned} y' &= yz^2 + 2\sin(t), & y(0) &= 1, \\ z' &= 2y - z, & z(0) &= -2. \end{aligned}$$

Sett opp systemet av ikke-lineære ligninger som må løses for hvert tidsskritt. Forklar hvordan dette ikke-lineære systemet kan løses vha. Newtons metode. (Det forventes at du skriver ned det linære ligningssystemet som må løses i hver iterasjon, men du behøver ikke forklare hvordan linære ligninger løses.)

b) Anta at $\mathbf{f}$ tilfredstiller Lipschitz-betingelsen

$$\|\mathbf{f}(t,\mathbf{y}) - \mathbf{f}(t,\tilde{\mathbf{y}})\| \le L\|\mathbf{y} - \tilde{\mathbf{y}}\|, \qquad \text{for alle } t \in \mathbb{R},\ \mathbf{y}, \tilde{\mathbf{y}} \in \mathbb{R}^m.$$

Den lokale avbruddsfeilen for trapesmetoden

$$\mathbf{d}_{n+1} = \mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) - \frac{h}{2}\big(\mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1})) + \mathbf{f}(t_n, \mathbf{y}(t_n))\big)$$

tilfredstiller

$$\|\mathbf{d}_{n+1}\| \le \frac{1}{12}h^3 M, \qquad M = \max_{\xi\in\mathbb{R}} \|\mathbf{y}'''(\xi)\|.$$

Bruk dette til å vise at den globale feilen $\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n$ tilfredstiller

$$\|\mathbf{e}_{n+1}\| \le \frac{1 + \frac{1}{2}hL}{1 - \frac{1}{2}hL}\|\mathbf{e}_n\| + \frac{\frac{1}{12}Mh^3}{1 - \frac{1}{2}hL} \qquad \text{for } hL < 2.$$

Til slutt, bruk dette til å bevise følgende estimat av øvre grense for den globale feilen:

$$\|\mathbf{e}_n\| \le \frac{Mh^2}{12L}\left[\left(\frac{1 + \frac{1}{2}hL}{1 - \frac{1}{2}hL}\right)^n - 1\right],$$

under forutsetning av at $\mathbf{e}_0 = 0$.

Numerisk lineær algebra

$\boxed{7}$ Utfør en LU-faktorisering med skalert pivotering for matrisa

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

dvs. finn P, L og U slik at $PA = LU$.

Du kan jo alltids kontrollere svaret med koden fra Øving 5.

$\boxed{8}$ **a)** Er følgende matrise symmetrisk positiv definit?

$$A = \begin{pmatrix} 3.2 & -1.2 & 0.8 \\ -1.2 & 4.6 & 1.2 \\ 0.8 & 1.2 & 3.6 \end{pmatrix}$$

b) Vil dette lineære iterasjonsskjemaet konvergere? Begrunn svaret.

$$\begin{aligned} x_1^{(k+1)} &= \frac{-x_1^{(k)} - x_2^{(k)} + x_3^{(k)} + 2}{4} \\ x_2^{(k+1)} &= \frac{2x_1^{(k)} + x_2^{(k)} - x_3^{(k)} - 1}{6} \\ x_3^{(k+1)} &= \frac{x_1^{(k)} - x_2^{(k)} + x_3^{(k)} + 4}{4} \end{aligned}$$