Norwegian University of Science and Technology
Department of Mathematical Sciences

TMA4130/35 Matematikk 4N/D Fall 2018

Exercise set

In these exercises, the answer to one point may sometimes be found in the next point. Do not expect this to happen at the exam!

Solution sets to these exercises will not be made.

Interpolation

$\boxed{1}$ **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.

Integration

$\boxed{2}$ **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 $x_i = a + ih$.

Given the python code:

```
from numpy import *
from matplotlib.pyplot import *

# --- The quadrature rule ---
def Q(f, a, b, n):
    h = (b-a)/n
    xi = linspace(a, b, n+1)
    xi = xi[:-1]                      # xi = [a, a+h, .... ,a+(n-1)*h]
    return h*sum(0.25*f(xi)+0.75*f(xi+2*h/3))

# --- 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
```

d) Write down the quadrature formula described in the function `Q`.

e) Write down the first two lines of the output (do not worry about the number of digits).

f) The output from the code above is:

```
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?

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.

Nonlinear equations

$\boxed{3}$ Newton's method for a scalar equation $f(x) = 0$ is given by

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

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.

$\boxed{4}$ Newton's method for system of equations can be implemented as follows:

```
def my_problem(x):
  f =
  jac =
  return f, jac

x = array([1,1])
for k in range(10):
  f, jac =  my_problem(x)
  delta = solve(jac, -f)  # Solve the linear system jac*delta = -f
  x = x + delta
  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.

There may also be questions about

- **Ordinary differential equations**
- **Numerical differentiation**
- **Partial differential equations**