Forside

UNIVERSITY OF OSLO
Faculty of mathematics and natural sciences
Trial exam in IN1900, fall 2020
Attachments: ODESolver.pdf in problem 3.3
Permitted aids: All

This exam is a trial exam intended as preparation for the final exam 2020. The questions are mostly the same as the final exam from 2019, but with some adjustments to account for the fact that this year's exam is a home exam with all aids permitted.

Read the entire exam set before you start answering the questions. The exam contains multiple choice questions, and text questions where you shall write short programs or read programs and write the output from the program. If you are missing information you can make your own reasonable assumptions, as long as they are in line with the "nature" of the question. In text questions you should then specify the assumptions you made, for instance in comments to the code.

All code in the question texts is written in Python 3. You can write your answers in either Python 2 or Python 3, but you should avoid using a mix.

Most of the questions lead to short code with little need for comments, unless you do something complicated or non-standard. (which is not recommended; but in this case the comments shall explain the ideas behind the program to make it easier to evaluate the code).

A question may ask you to write a function. A main program which calls the function is in this case not needed, unless it is specifically asked for in the question text.

1.1 Hvilken påstand er riktig?

One of the following statements is correct. Which one?

Select one alternative:

- A test function returns 0 if the test passes
- A test function must always include a return statement
- A test function can have multiple assert statements
- A test function should always take at least one input argument

Maximum marks: 1

1.2 Hvilket funksjonsargument?

Which of the alternatives below will ensure that **count('GATA', ba)** does not give any error messages and returns somthing different from 0?

```
def count(dna, base):
    i = 0
    for j in range(len(dna)):
        if dna[j] == base:
            i += 1
    return i

test_count()
```

Select one alternative:

- ○ **ba** is a string ('str') with length 1.
- ○ **ba** is an integer ('int')
- ○ **ba** is either a string ('str') or an integer ('int')
- ○ **ba** is a string ('str') with arbitrary length

Maximum marks: 1

1.3 Hva er feil i koden?

This code will stop and print an error message in the line "**for number in e:**". Which of these alternatives gives the most precise description of the cause for this error message?

```
a = [[4,5], [0,1], 2, [2,0.5], 4, 3, [5,6,7]]
b =[]
for e in a:
   s = 0
   for number in e:
      s += number
   b.append(s)
print(b)
```

Select one alternative:

- ○ The elements in the list **a** are not integers.
- ○ The elements in the list **a** do not have the same length.
- ○ All the elements in the **a** do not have length larger than 1.
- ○ All the elements in the list **a** are not lists or arrays.

Maximum marks: 2

1.4 Hvilket funksjonskall?

The function **euler(rhs,u0,T,n)** applies Euler's method to solve an ordinary differential equation (ODE) with right hand side defined by the function rhs and initial condition u0, for the time interval 0 ti T, with n time steps:

```
import numpy as np

def euler(rhs,u0,T,n=100):
    t = np.linspace(0,T,n+1)
    dt = T/n
    u = np.zeros_like(t)
    u[0] = u0
    for i in range(1,n+1):
        u[i] = u[i-1]+dt*rhs(u[i-1],t[i-1])
    return u,t
```

We want to use the function to solve the differential equation y' = -0.5 y, y(0) = 1, for t in the interval 0 til 5. Which function call is correct?

Select one alternative:

- ○ u, t = euler(f = -0.5*y, 1.0, 5)
- ○ u, t = euler(-0.5*y, 1.0, 5)
- ○ u, t = euler(rhs=lambda y, t: -0.5*y, 1.0, 5)
- ○ u, t = euler(lambda y, t: -0.5 y(t), 1, 5)
- ○ u, t = euler(lambda y, t: -0.5*y, 1.0, 5)
- ○ u, t = euler(lambda y: -0.5*y, 1.0, 5)

Recall that a lambda function is a compact way to define a function. For instance, the following line will define a function that returns x^2+y^2:

```
func = lambda x,y: x**2 + y**2
```

This line is equivalent to the following code:

```
def func(x,y):
    return x**2 + y**2
```

Maximum marks: 2

1.5 Hva returnerer funksjonen?

The function below takes a list of strings as input. The strings have exactly the same length (for instance 1000 characters each), and they contain only the characters A, C, G, and T. The function returns four lists of integers: **A**, **C**, **G** og **T**. What do the numbers in these four lists say?

```
def freq_lists(dna_list):
    n = len(dna_list[0])
    A = [0]*n
    C = [0]*n
    G = [0]*n
    T = [0]*n
    for dna in dna_list:
        index = 0
        for base in dna:
            if base == 'A':
                A[index] += 1
            elif base == 'C':
                C[index] += 1
            elif base == 'G':
                G[index] += 1
            elif base == 'T':
                T[index] += 1
            index += 1
    return A, C, G, T
```

Select one alternative:

- ['G', 'G', 'G', 'G', 0]
- How many times a given letter A, C, G or T is found in a given position in the input-strings.
- How many times each letter A, C, G, or T occurs in the input strings in total.
- Listene har alle lengde n og inneholder verdien 0 i alle indekser
- How many times a given letter A, C, G or T is found in one of the input strings.
- The length of each input string, and how many times each letter occurs.

Maximum marks: 2

1.6 Hva gjør koden?

The file formula_cml.py contains the following code:

```
# PART A:
import sys
from math import *
try:
    formula = sys.argv[1]
    x = [float(x_) for x_ in sys.argv[2:]]
except IndexError:
    print('Missing command line argument')
    exit()

# PART B:
code = f"""
def f(x):
    return {formula}
"""

#  PART C:
try:
    exec(code)
except:
    print('Something wrong in formula')
    exit()

# PART D:
for x_ in x:
    print(f(x_),end=' ')
```

Give a brief explanation of what the code does in *each of the four parts* of the program (PART A, PART B, PART C and PART D) . Also explain what kind of command line arguments the code expects in PART A.

Fill in your answer here

Format

Words: 0

Maximum marks: 2

1.7 Hvilken linje mangler?

The file stars.txt contains the following:

```
Name                distance   brightness   luminosity
Alpha_Centauri_A     4.3         0.26          1.56
Alpha_Centauri_B     4.3        0.077         0.45
Alpha_Centauri_C     4.2        0.00001       0.00006
Sirius_A              8.6         1.00          23.6
```

With the exception of the first line (which is a headline), each line in the file contains information about a star (the name of the star, distance from the earth, brightness (seen from the earth), and luminosity (true brightness). There are no blank lines in the file.

We want the code below to read this file and store the information about each star in a nested dictionary **stars_data.**

```
stars_data = {}

with open('stars.txt') as infile:
    infile.readline()

    for line in infile:
        w = line.split()
        ## Missing line goes here ##
        stars_data[w[0]] = data

print(stars_data['Sirius_A'])
```

We want this code to work and print the following output to the screen:

```
{'dist': 8.6, 'bright': 1.0, 'lum': 23.6}
```

Write the missing line in the code, which makes the program work as desired!

Fill in your answer here

Format

Words: 0

Maximum marks: 2

1.8 Hvilket funksjonskall?

The following function implements a bisection method for finding solutions of the equation f(x)=0 in the interval [a,b].

```
def bisection(f, a, b, eps=1e-5):
    fa = f(a)
    if fa*f(b) > 0:
        print(f'No unique root in [{a},{b}]')
        return None
    while b-a > eps:
        m = (a + b)/2.0
        fm = f(m)
        if fa*fm <= 0:
            b = m  # root is in left half of [a,b]
        else:
            a = m  # root is in right half of [a,b]
            fa = fm
    return m
```

We want to use the function to find a solution of the equation
$x^3 + 2x - 1 = 0$
in the interval [-10,10]. Which function call is correct?
Select one alternative:

- ○ x = bisection(x**3+2*x-1,-10,10, eps=1e-5)
- ○ x = bisection(f(x) = x**3+2*x-1,-10,10, eps=1e-5)
- ○ x = bisection(eval('x**3+2*x-1'),-10,10)
- ○ x = bisection(lambda x: x**3+2*x-1,-10,10)
- ○ x = bisection(f=lambda x: x**3+2*x-1, -10, 10)
- ○ x = bisection(lambda x: x**3+2*x-1, a, b, eps=1e-5)

Recall that a lambda function is a compact way to define a function. For instance, the following line will define a function that returns x^2+y^2:

```
func = lambda x,y: x**2 + y**2
```

This line is equivalent to the following code:

```
def func(x,y):
    return x**2 + y**2
```

Maximum marks: 2

1.9 Hva gjør funksjonen?

Which of the statements below gives the most precise description of what the function **f(arg)** does? The argument **arg** is assumed to be an array or a list.

```
def f(arg):
    n = len(arg)
    for i in range(n):
        for j in range(n-i-1):
            if arg[j] > arg[j+1] :
                arg[j], arg[j+1] = arg[j+1], arg[j]
    return arg
```

Select one alternative:

- ○ The function sorts the elements in arg, from the smallest to the largest value.
- ○ The function returns the larges element in arg.
- ○ The function re-orders the elements in arg, to the opposite order.
- ○ The function returns the smallest element in arg.

Maximum marks: 1

2.1 Funksjon av to variable

Write a python-funksjon **f(x,y)** where **x** is a number and **y** is a list of numbers, which calculates the value of the mathematical expression

$$f(x,y) = \begin{cases} 4x^3y - 2xy, & y \leq 0 \\ 4x^3y + 2xy, & y > 0 \end{cases}$$

for the given value of **x** and for each of the values in **y**, and returns the answer as a list of the same length as the input list **y.**

Fill in your answer here

Maximum marks: 3

2.2 Numerisk derivasjon

The derivative of a mathematical function f(x) can be approximated with the midpoint formula
$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$
for a small number h.

Write a Python function **midpoint(f,x,h),** which uses this formula to estimate the derivative of a function f in the point x. The function shall return the function value f(x) and the estimated derivative. The argument f can be an arbitrary mathematical function implemented as a Python function, which takes one input argument and returns one value.

Include a line where you call the function to estimate the derivative of cos(x) in the point x=0, for h=0.001.

Fill in your answer here

Maximum marks: 5

2.3 Implementasjon av en sum

Write a Python function **cos_approx(x,n)** which computes the sum
$f(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k}}{2k!}$
and returns the value.

x can be a decimal number (float) or a Numpy-array of decimal numbers, while n is a positive integer. Recall that k! is the factorial of k. Include necessary imports.

Fill in your answer here

Maximum marks: 5

2.4 Lesing av fil

The file constants.txt has the following contents:

```
name of constant           value                  dimension
------------------------------------------------------------
light speed                 299792458.0       m/s
gravitational constant    6.67259e-11        m**3/kg/s**2
Planck constant             6.6260755e-34      J*s
elementary charge         1.60217733e-19     C
Avogadro number          6.0221367e23       1/mol
Boltzmann constant       1.380658e-23       J/K
electron mass               9.1093897e-31      kg
proton mass                 1.6726231e-27      kg
```

The file contains no blank lines.

Write a Python program that reads this file and stores its contents in a dictionary. The keys of the dictionary shall be the name of the constant (from the column "name of constant"), and the value shall be a list or tuple of length two that contains the numeric value of the constant (column "value") and its physical units (column "dimension").

Hint: The Python method **join** is used for joining a list of strings into a single string. As an example, the code

string_list = ['Hello','world']

hello = ' '.join(string_list)

will result in a string hello having the value "Hello world".

Fill in your answer here

Maximum marks: 6

2.5 Dictionary og sum

A polynomial p(x) can be represented as a dictionary, so that the keys of the polynomial are the exponents and the values are the coefficients in front of each term. For instance, the polynomial $p(x) = 1 - 2x^2 + 3x^4 + x^5$ can be represented as the dictionary

p = {0:1,2:-2,4:3,5:1}

Write a Python function **poly_eval(p,x)**, which evalueates such a polynomial p for a given x, and returns its value.

Fill in your answer here

Maximum marks: 5

2.6 Derivasjon av polynom

A general polynomial can be written on the form
$p(x) = \sum_{j=0}^{n} c_j x^j$
where c_j are constant coefficients.

The derivative of such a polynomial is given by
$\frac{dp}{dx} = \sum_{j=1}^{n} j c_j x^{j-1}$

Write a Python function **poly_diff(p)** that calculates the derivative of a general polynomial p. The argument p is a dictionary representation of a polynomial, as defined in the previous question. The function shall return a dictionary representing the derivative of p.

Fill in your answer here

Maximum marks: 5

2.7 Minste og største verdi

Write a function **min_max(a)** which takes a list or array **a** as argument, and returns the largest and the smallest element in **a.** The function shall not make use of the builtin max- and min-functions of Python or Numpy.

Fill in your answer here

Maximum marks: 5

2.8 Primtall

A prime number is an integer larger than 1 which can only be divided by 1 and the number itself. Write a function **is_prime(n)** which takes a number **n** as input and returns True if the number is a prime and False otherwise. You can assume that the argument **n** is always an integer, so you don't have to test for this inside the function. The efficiency of the algorithm you choose to implement does not matter for the score on the question.

Hint:

In Python the modulo-operator **%** returns the *remainder* in an integer division, as illustrated in the following session:

```
>>> 4%2
0
>>>5%2
1
>>> 6%4
2
```

Fill in your answer here

Maximum marks: 5

3.1 Klasse for en funksjon

Write a Python class **F** which implements the function
$f(x; a, b, c) = ax^2 + bx + c$
The parameters a, b, and c shall be attributes, and the class shall be possible to use in the following way

```
f = F(a=1.0, b=2.0, c=0.0)
x = 2.0
print(f(x))     # prints the function value (8.0)
```

Fill in your answer here

Maximum marks: 5

3.2 ODE-løser, funksjon

Write a Python function **heun3(f, U0, T, n)**, which uses Heun's 3. order method to solve an ordinary differential equation (ODE) given by:

$$u' = f(u), \qquad u(0) = u_0.$$

Heun's 3. order method is given by

$$u_{k+1} = u_k + \tfrac{1}{4}K_1 + \tfrac{3}{4}K_3 K_1 = \Delta t f(u_k, t_k) K_2 = \Delta t f(u_k + \tfrac{1}{3}K_1, t_k + \tfrac{1}{3}\Delta t) K_3 = \Delta t f(u_k + \tfrac{2}{3}K_2, t_k + \tfrac{2}{3}\Delta t)$$

The arguments to the function shall be a callable function f, which defines the right hand side of the ODE, the initial condition U0, the end-time T, and the number of time steps n. The function shall return two numpy-arrays u and t, where u contains the solution and t contains the time points where we have approximated the solution. In this question you can assume that we solve a scalar ODE, where the solution has only one component. The solution array u should therefore be a one-dimensional array. Include necessary imports.

Fill in your answer here

Maximum marks: 5

3.3 ODESolver, arv

Implement the method from the previous question (Heun's 3. order method) as a sub class **Heun3(ODESolver)**. The base class ODESolver is defined in the attached file. Use inheritance to reuse as much code as possible from the base class. The class **Heun3** must support the following use:

```
from numpy import *
from ODESolver import *
rhs = lambda u,t: -0.5*u
solver = Heun3(rhs)
solver.set_initial_condition(1.0)
time = numpy.linspace(0,10,101)
u, t = solver.solve(time)
```

Fill in your answer here

Maximum marks: 5

3.4 Logistisk vekst

The model for logistic growth is defined by the following ordinary differential equation (ODE):

$$\frac{du}{dt} = au(1 - \frac{u}{R})$$

The model describes growth of a population in an environment with limited resources, where u is the size of the population, t is time, and *a* and *R* are constant model parameters. Write a Python function **rhs(u,t)** that defines the right hand side of the equation. The parameters can be local variables in the function, with values *a* = 1 and *R* = 50.

Write code for solving the equation with the Heun3-klass from the previous question. The initial condition shall be u0 = 0.1, the time interval from t=0 to t=20, and you shall use 201 time steps including the end points (dt = 0.1).

Fill in your answer here

Maximum marks: 5

3.5 SEIS-modell

This question presents a so-called SEIS-model for modeling infectious diseases. The model is a modification of the classical SIR-model, where the population is divided in three groups: those who can be infected (S), those who are infected but have not yet developed the disease, and cannot infect others (E), and those that are sick and can infect others (I). There is no immunity in the model, so those that recover from the disease return to the S-category. Let S(t), E(t), and I(t) be the number of people in each category (measured in millions). The following system of differential equations describes how S(t), E(t), and I(t) evolve over a time interval [0,T]:

$$S'(t) = -p(t)S(t)I(t) + rI(t), E'(t) = p(t)S(t)I(t) - qE(t), I'(t) = qE(t) - rI(t),$$

At time t=0 we have the intitial conditions S(0) = S0, E(0) = E0, I(0) = 0. The function p(t) and the constants q and r are assumed to be known. All constants and functions are >0.

Write a Python-function **SEIS(S0,E0,p,q,r,T)**, which takes initial values S0, E0, the function p(t), parameters q, r, and the end-time T as input arguments. The function shall solve the equations of the SEIS-model and return the solution. Use the Heun3 class from the previous question to solve the differential equations. Let the time be given in days. Use ten time steps per day, so that the total number of time steps for a simulation over the interval [0,T] is 10T+1. The function SEIS shall return 4 arrays:

t, which contains the time points tk where the numerical solution is calculated,

S, which contains S(0),S(t1), ... S(tn),

E, which contains E(0),E(t1), ... E(tn),

I, which contains I(0),I(t1), ... I(tn).

We want to solve the model with the following parameters; S0 = 4.0, E0 = 0.2, q = r = 0.1, and p(t) = 0.0233 (constant).

Write the code to call the function with the given parameters and T=100. Also include code to plot S(t), E(t) and I(t) in the same window, with a legend for each curve.

Fill in your answer here

Maximum marks: 8

Question 20

Attached

```
import numpy as np

class ODESolver:
    """
    Superclass for numerical methods solving scalar and vector ODEs
      du/dt = f(u, t)

    Attributes:
    t: array of time values
    u: array of solution values (at time points t)
    k: step number of the most recently computed solution
    f: callable object implementing f(u, t)
    """
    def __init__(self, f):
        if not callable(f):
            raise TypeError('f is %s, not a function' % type(f))
        self.f = lambda u, t: np.asarray(f(u, t), float)

    def set_initial_condition(self, U0):
        if isinstance(U0, (float,int)):  # scalar ODE
            self.neq = 1
            U0 = float(U0)
        else:                            # system of ODEs
            U0 = np.asarray(U0)          # (assume U0 is sequence)
            self.neq = U0.size
        self.U0 = U0

    def advance(self):
        """Advance solution one time step."""
        raise NotImplementedError

    def solve(self, time_points):
        """
        Compute solution u for t values in the list/array
        """
        self.t = np.asarray(time_points)
        n = self.t.size
        if self.neq == 1:  # scalar ODEs
            self.u = np.zeros(n)
        else:              # systems of ODEs
            self.u = np.zeros((n,self.neq))

        # Assume that self.t[0] corresponds to self.U0
        self.u[0] = self.U0

        # Time loop
        for k in range(n-1):
            self.k = k
            self.u[k+1] = self.advance()

        return self.u, self.t


class ForwardEuler(ODESolver):
    def advance(self):
        u, f, k, t = self.u, self.f, self.k, self.t
        dt = t[k+1] - t[k]
        u_new = u[k] + dt*f(u[k], t[k])
        return u_new
```