Summary of chapters 1-5 (part 1)

Ole Christian Lingjærde, Dept of Informatics, UiO

6 October 2017

Today's agenda

- Exercise A.14, 5.14
- Quiz

Exercise A.14

Find difference equations for computing sin x

The purpose of this exercise is to derive and implement difference equations for computing a Taylor polynomial approximation to $\sin x$:

$$\sin x \approx S(x;n) = \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}$$

To compute $S(x;n)$ efficiently, write the sum as $S(x;n) = \sum_{j=0}^{n} a_j$, and derive a relation between two consecutive terms in the series:

$$a_j = -\frac{x^2}{(2j+1)2j} a_{j-1}.$$

Introduce $s_j = S(x; j-1)$ and a_j as the two sequences to compute. We have $s_0 = 0$ and $a_0 = x$.

a) Formulate the two difference equations for s_j and a_j.

Hint: Section A.1.8 explains how this task can be solved for the Taylor approximation of e^x.

Exercise A.14 (cont'd)

b) Implement the system of difference equations in a function `sin_Taylor(x, n)` which returns s_{n+1} and $|a_{n+1}|$. The latter is the first neglected term in the sum (since $s_{n+1} = \sum_{j=0}^{n} a_j$) and may act as a rough measure of the size of the error in the Taylor polynomial approximation.

c) Verify the implementation by computing the difference equations for $n = 2$ by hand (or in a separate program) and comparing with the output from the `sin_Taylor` function. Automate this comparison in a test function.

d) Make a table or plot of s_n for various x and n values to illustrate that the accuracy of a Taylor polynomial (around $x = 0$) improves as n increases and x decreases.

Hint: `sin_Taylor(x, n)` can give extremely inaccurate approximations to $\sin x$ if x is not sufficiently small and n sufficiently large. In a plot you must therefore define the axis appropriately.

Key idea

Computing series with many terms can be time-consuming. A common strategy is to see if the nth term can be found faster using the $(n-1)$st term.

Calculating $S(x;n) = a_0 + a_1 + \cdots + a_n$

Suppose in the following that x is fixed and let $s_{n+1} = S(x;n)$.
1) We can find s_{n+1} from s_n and a_n:

$$s_{n+1} = s_n + a_n$$

2) We can also find a_n using a_{n-1} and x:

$$a_{n-1} = (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!} \text{ and } a_n = (-1)^n\frac{x^{2n+1}}{(2n+1)!}$$

Thus we have the relation:

$$a_n = -a_{n-1} \cdot \frac{x^2}{2n(2n+1)}$$

Answer to exercise A.14 a)

Question

Formulate the two difference equations for s_n and a_n.

Answer

Set $s_0 = 0$ and $a_0 = x$. For $n = 1, 2, \ldots$ let

$$\begin{aligned} s_n &= s_{n-1} + a_{n-1} \\ a_n &= -a_{n-1} \cdot x^2/(2n(2n+1)) \end{aligned}$$

Answer to exercise A.14 b)

Question

Implement the system of difference equations in a function `sin_Taylor(x, n)` which returns s_{n+1} and $|a_{n+1}|$.

Answer 1: storing all updates

```
def sin_Taylor(x,n):
    s = [0.0]*(n+2)
    a = [0.0]*(n+2); a[0] = x
    for i in range(1,n+2):
        s[i] = s[i-1] + a[i-1]
        a[i] = -a[i-1]*x**2/(2*i*(2*i+1))
    return s[n+1], abs(a[n+1])
```

Answer 2: storing only last update

```
def sin_Taylor2(x,n):
    s = 0
    a = x
    for i in range(1,n+2):
        s = s + a
        a = -a*x**2/(2*i*(2*i+1))
    return s, abs(a)
```

Answer to exercise A.14 c)

Question

Verify the implementation by computing the difference equations for $n = 2$ and comparing with the output from the `sin_Taylor` function. Automate this comparison in a test function.

Answer (part 1)

```
def sin_two_terms(x):
  s = [0]*4
  a = [0]*4

  a[0] = x

  s[1] = s[0] + a[0]
  a[1] = -a[0]*x**2 / (2*1*(2*1+1))

  s[2] = s[1] + a[1]
  a[2] = -a[1]*x**2 / (2*2*(2*2+1))

  s[3] = s[2] + a[2]
  a[3] = -a[2]*x**2 / (2*3*(2*3+1))

  return s[3], abs(a[3])
```

Answer to exercise A.14 c)

Answer (part 2)

```
def sin_Taylor(x):
    <as before>

def sin_two_terms(x):
    <as before>

def test_sin_Taylor():
    x = 0.63      # Just an arbitrary x-value for validation
    tol = 1e-14   # Tolerance
    s_expected, a_expected = sin_two_terms(x)
    s_computed, a_computed = sin_Taylor(x,2)
    success1 = abs(s_computed - s_expected) < tol
    success2 = abs(a_computed - a_expected) < tol
    success = success1 and success2
    message = 'Output is different from expected!'
    assert success, message
```

Answer to exercise A.14 c)

Answer (part 3)

```
In [10]: sin_two_terms(0.63)
Out[10]: (0.5891525304525, 7.815437776125003e-06)

In [11]: sin_Taylor(0.63, 2)
Out[11]: (0.5891525304525, 7.815437776125003e-06)

In [12]: test_sin_Taylor()

In [13]:
```

Answer to exercise A.14 d)

Question

Make a table or plot of s_n for various x and n values to illustrate that the accuracy of a Taylor polynomial (around $x = 0$) improves as n increases and x decreases.

Answer (part 1)

We first make a plan:

- For a given x and n we can calculate the Taylor approximation s_{n+1} with the statement `s = sin_Taylor(x,n)[0]`.
- To calculate s_n for various x and n, we must decide what x-values and n-values to use.
- We decide here to use a uniform grid of M x-values on $[0, 1]$, and $n = 1, 2, \ldots, N$.
- We write a method producing an M x N table of all the calculated s-values.

Answer to exercise A.14 d)

Answer (part 2)

```
def make_table(M,N):
    import numpy as np
    x = np.linspace(0, 1, M)
    n = np.arange(1,N+1)
    S = np.zeros((M,N))
    for i in range(M):
        for j in range(N):
            S[i,j] = sin_Taylor(x[i], n[j])[0]
    return S, x, n
```

Exercise 5.14

Plot data in a two-column file

The file:

`https://github.com/hplgit/scipro-primer/blob/master/src/plot/xy.dat`

contains two columns of numbers, corresponding to x and y coordinates on a curve. The start of the file looks as this:

```
-1.0000 -0.0000
-0.9933 -0.0087
-0.9867 -0.0179
-0.9800 -0.0274
-0.9733 -0.0374
```

Make a program that reads the first column into a list x and the second column into a list y. Plot the curve. Print out the mean y value as well as the maximum and minimum y values.

Hint: Read the file line by line, split each line into words, convert to float, and append to x and y. The computations with y are simpler if the list is converted to an array.

Filename: read_2columns.

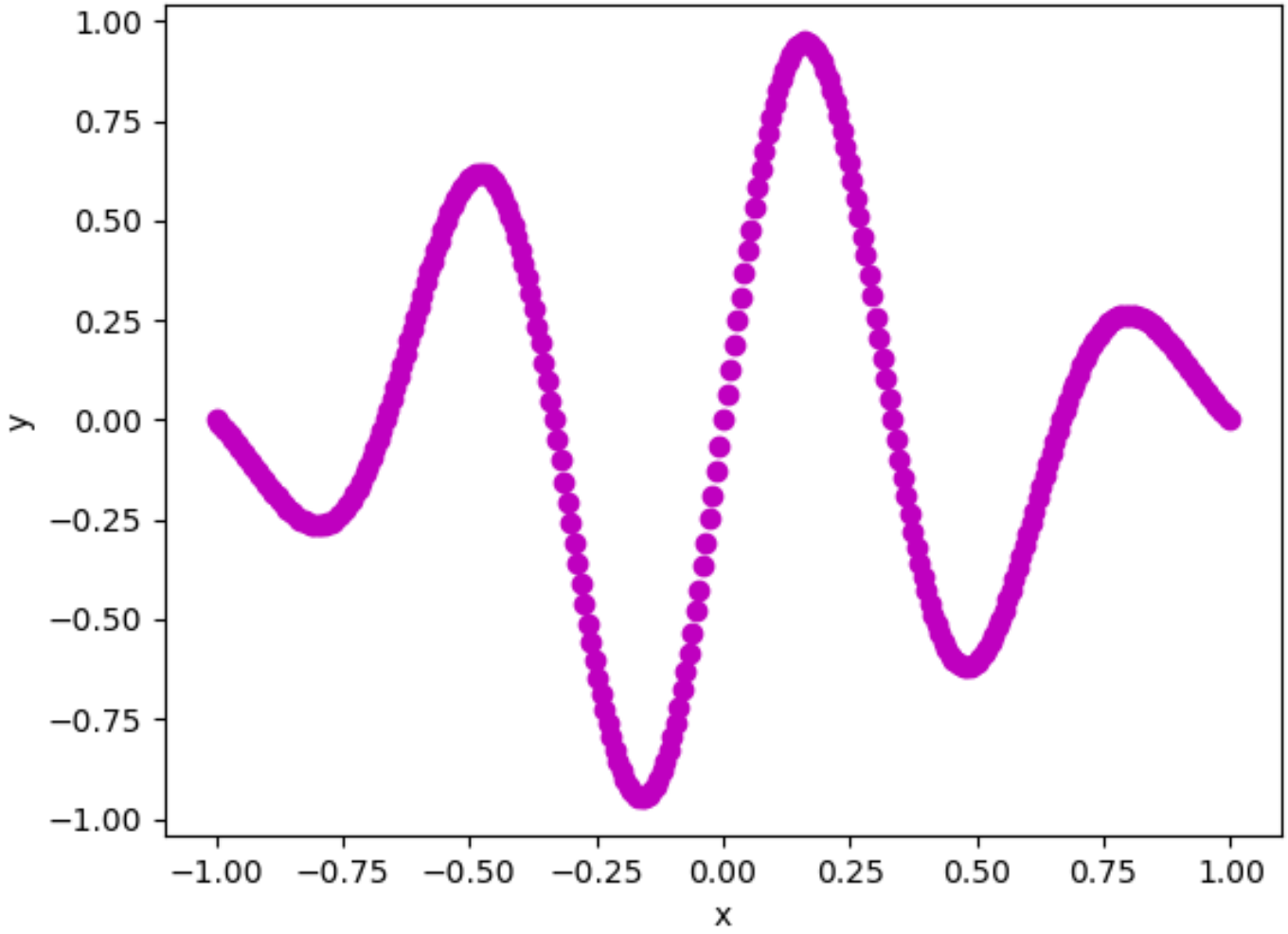
Answer to exercise 5.14

```
# Read file
infile = open('xy.dat12', 'r')
x = []
y = []
for line in infile:
    s = line.split()
    x.append(eval(s[0]))
    y.append(eval(s[1]))
infile.close()

# Plot the points (x[i],y[i])
import matplotlib.pyplot as plt
plt.plot(x, y, 'mo')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

# Print mean, min and max of y
import numpy as np
ya = np.array(y)
print('Mean of y: %g' % np.mean(ya))
print('Min of y: %g' % np.min(ya))
print('Max of y: %g' % np.max(ya))
```

Result


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x

Quiz 1

What is printed out by these programs?

Program A

```
x = []
for i in range(5):
    x.append(i)
print(x)
```

Program B

```
x = []
for i in range(5):
    x = x + [i]
print(x)
```

Program C

```
x = []
for i in range(5):
    x = [i] + x
print(x)
```

Quiz 2

What is printed out by these programs?

Program A

```
x = [0, 1, 2, 3]
for i in range(len(x)):
    x[i] = i * x[i]
print(x)
```

Program B

```
x = [0, 1, 2, 3, 4]
for i in range(len(x)-1):
    x[i] = x[i+1]**2
print(x)
```

Program C

```
x = [0, 1, 2, 3, 4]
for i in range(1,5):
    x[i] = x[i-1]**2
print(x)
```

Quiz 3

What is printed out by these programs?

Program A

```
x = [2*i for i in range(1,4)]
print(x)
```

Program B

```
x = [1, 2, 3, 4, 5, 6, 7]
for i in range(len(x)):
    x[i] = x[6-i]
print(x)
```

Program A

```
x = range(3)*2
y = [range(3)]*2
z = range(1,1)*2
print(x)
print(y)
print(z)
```

Quiz 4

What is printed out by these programs?

Program A

```
x = [[0,1,2],[4,5,6],[8,9,10]]
print(x[1][1])
print(x[-1][1])
print(x[1][-1])
print(x[-1][-1])
```

Program B

```
x = [[0,1,2],[4,5,6],[8,9,10]]
x.reverse()
print(x)
```

Program C

```
x = [[0,1,2],[1,2,3],[2,3,4]]
y = [[1,2,3,4,5][e[-1]] for e in x]
print(y)
```

Quiz 5

What is printed out by these programs?

Program A

```
import numpy as np
x = np.linspace(0, 1, 11)
y = x + x
print(y)
```

Program B

```
import numpy as np
x = np.array([1,2,3])
y = np.ones(3)
z = y * x ** x
print(z)
```

Program C

```
import numpy as np
x = np.array([1,2,3,4,5])
y = x[0:3]
y += 1
print(x)
print(y)
```

Summary of file reading

You should remember the following functions:

Essentials for file reading

To open a file:

```
infile = open('filnavn.txt', 'r')
```

To read whole file into a string:

```
s = infile.read()
```

To read whole file into a list (each element is a line):

```
a = infile.readlines()
```

To skip a line:

```
infile.readline()
```

To read all remaining lines and split into separate words:

```
for line in infile:
    a = line.split()
    # a[0], a[1], ... are the words on the line
```

To close the file:

```
infile.close()
```

Quiz 6

Suppose the file 'names.txt' looks like this:

```
Kari    Ola     Katrine     Ingrid
Nils    Are     Jonas       Ella
Arne    Elin    Arvid       Kristin
```

Explain what the following program does:

```
X = []
infile = open('names.txt', 'r')
for line in infile:
    X.append(line.split())
infile.close()

outfile = open('names2.txt', 'w')
for i in range(len(X[0])):
    for j in range(len(X)):
        outfile.write(X[j][i] + ' ')
    outfile.write('\n')
outfile.close()
```

Quiz 7

From blood one can measure CRP (C-reactive protein) level. Normal level is CRP < 5, while CRP > 10 is usually a sign of infection. We have a text file 'CRP.txt' with two columns (CRP value and social security number) which starts like this:

```
CRP     ID
4.3     01016663223
1.2     15106364267
13.6    08049886252
7.2     24128763233
3.1     31047920251
46.12   27098360656
...     ...
```

Suppose the file has been read into two lists `crp` (float) and `id` (string). Write code to calculate and print on screen:

- The highest measured CRP value
- How many CRP values are above 10
- The IDs of all patients with CRP values above 10

Answer to Quiz 7

```
import numpy as np

# The highest measured CRP value
acrp = np.array(crp)
max_crp = max(acrp)
print(max_crp)

# How many CRPs above 10
high_crp = sum(acrp > 10)
print(high_crp)

# IDs of patients with CRP > 10
aid = np.array(id)
high_id = aid[acrp > 10]
print(high_id)
```

Quiz 8

Write code to plot the points (i, CRP[i]) with red circles, and with suitable labels on the x-axis and y-axis.

Answer to Quiz 8

```
import matplotlib.pyplot as plt
t = range(len(crp))
plt.plot(t, crp, 'ro')
plt.xlabel('Measurement no')
plt.ylabel('crp')
plt.show()
```

Note

You should remember the import statement for plotting and the most used plot functions, so you can avoid looking up references every time you need to use them (also, no aids are allowed at the midterm exam).

Quiz 9

Consider the following program:

```
import sys
try:
    x = eval(sys.argv[1])
except IndexError:
    print(.....)
    sys.exit(1)
except ValueError:
    print(.....)
    sys.exit(1)
x = x**2 + 1
print('x = %g' % x)
```

Fill in the missing parts (.....) with sensible text strings.

Answer to Quiz 9

```
import sys
try:
    x = eval(sys.argv[1])
except IndexError:
    print('Missing command line argument')
    sys.exit(1)
except ValueError:
    print('Command line argument cannot be interpreted')
    sys.exit(1)
x = x**2 + 1
print('x = %g' % x)
```

Quiz 10

What is printed out in these programs?

Program A

```
n = 5
def f(n):
    n = n+5
    return n**2
n = f(n)
print(n)
```

Program B

```
def f(n):
    if n > 1:
        return n + f(n-1)
    else:
        return 1
print(f(4))
```

Answer to Quiz 10

Program A

```
In [94]: n = 5
    ...: def f(n):
    ...:     n = n+5
    ...:     return n**2
    ...: n = f(n)
    ...: print(n)
    ...:
100
```

Program B

```
In [95]: def f(n):
    ...:     if n > 1:
    ...:         return n + f(n-1)
    ...:     else:
    ...:         return 1
    ...: print(f(4))
    ...:
10
```