Øving 10 -Interpolasjon og numeriskintegrasjon - LF

Obligatoriske oppgaver

1 We set up the table of divided differences as follows

x_k	f_k	$f[x_k, x_{k+1}]$	$f[x_k, x_{k+1}, x_{k+2}]$
-2	1		
		$1/3$	
1	2		$-1/60$
		$1/5$	
6	3		

Tabell 1: Newton divided differences table used in exercise 1.

This yields the polynomial

$$p_2(x) = 1 + \frac{1}{3}(x+2) - \frac{1}{60}(x+2)(x-1). \tag{1}$$

2 We add one more point to our table and calculate

x_k	f_k	$f[x_k, x_{k+1}]$	$f[x_k, x_{k+1}, x_{k+2}]$	$f[x_k, x_{k+1}, x_{k+2}, x_{k+3}]$
-2	1			
		$1/3$		
1	2		$-1/60$	
		$1/5$		$1/315$
6	3		$-4/315$	
		$6/27$		
$-3/4$	$3/2$			

Tabell 2: Newton divided differences table used in exercise 2.

This yields the polynimial

$$p_3(x) = p_2(x) + \frac{1}{315}(x+2)(x-1)(x-6)$$

where p_2 is given by (1).

3 See the lecture notes.

4 We calculate the Lagrangian polynomials,

$$\begin{aligned} l_0(x) &= \frac{(x-0)(x-1)}{(-1-0)(-1-1)} &=& \frac{1}{2}x(x-1), \\ l_1(x) &= \frac{(x+1)(x-1)}{(0+1)(0-1)} &=& -(x+1)(x-1), \\ l_2(x) &= \frac{(x+1)(x-0)}{(1-(-1))(1-0)} &=& \frac{1}{2}x(x+1). \end{aligned}$$

Hence we approximate,

$$f(x) \approx f(x_0)l_0(x) + f(x_1)l_1(x) + f(x_2)l_2(x).$$

The integral is then approximated by

$$\begin{aligned} \int_{-1}^{1} f(x) \approx f(x_0)\int_{-1}^{1} l_0(x)dx + f(x_1)\int_{-1}^{1} l_1(x)dx + f(x_2)\int_{-1}^{1} l_2(x)dx \\ = f(x_0)A_0 + f(x_1)A_1 + f(x_2)A_2. \end{aligned}$$

Finally, the weights are given by

$$\begin{aligned} A_0 &= \int_{-1}^{1} l_0(x)dx = \int_{-1}^{1} \frac{1}{2}x(x-1)dx = \frac{1}{3}, \\ A_1 &= \int_{-1}^{1} l_1(x)dx = \int_{-1}^{1} -(x+1)(x-1)dx = \frac{4}{3}, \\ A_2 &= \int_{-1}^{1} l_2(x)dx = \int_{-1}^{1} \frac{1}{2}x(x+1)dx = \frac{1}{3}. \end{aligned}$$

This yields

$$\int_{-1}^{1} f(x)dx \approx= \frac{1}{3}(f(-1) + 4f(0) + f(1)),$$

also called the Simpsons rule.

5 & 6 **Matlab::**

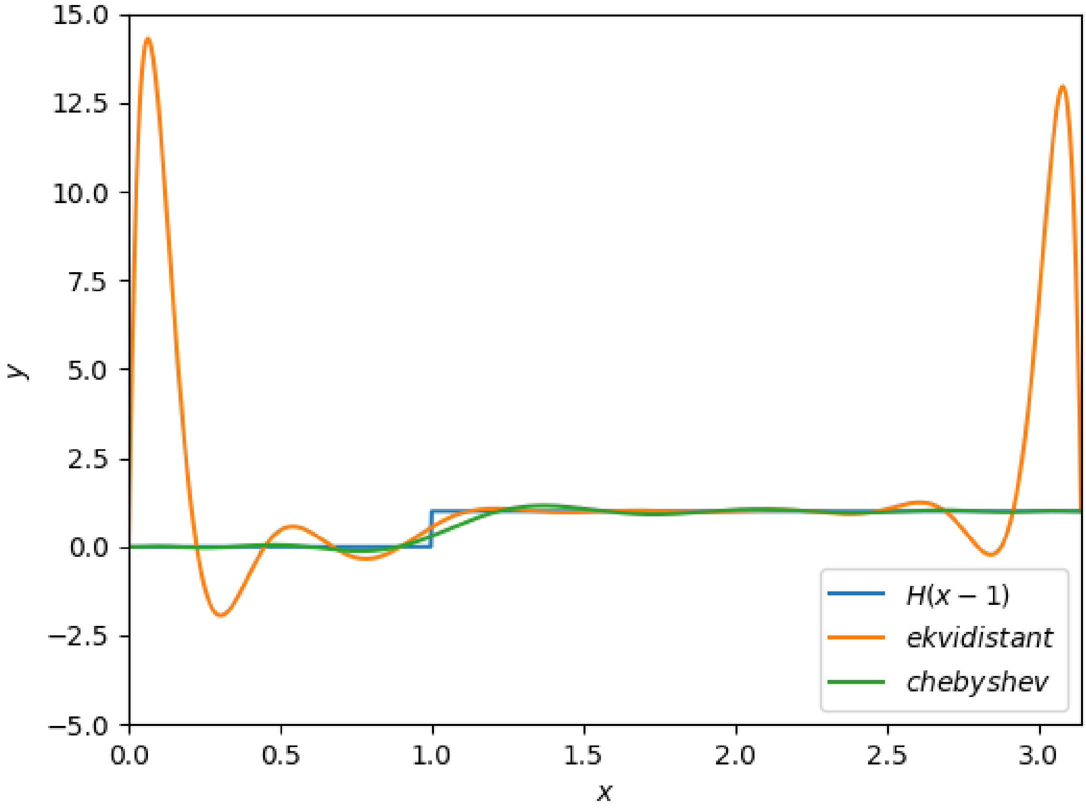
```
clear
figure
pause

% Dette er bare for   tegne x-aksen
x=0:.001:pi;
z=zeros(1,1001);


for I=1:20

    % Husk at hvis det ekvidistante gitteret g  r fra 0 til pi, g  r chebyshev
    % fra 1 til -1.
    x_ekvidistant=0:pi/I:pi;
    x_chebyshev=(cos(x_ekvidistant)+1)*pi/2;
```

```
    %plotte interpolasjon p   ekvidistant gitter
    subplot(1,2,1);
    plot(x,heaviside(x-1))
    hold on
    plot(x,polyval(polyfit(x_ekvidistant, heaviside(x_ekvidistant-1),I),x));
    axis([0 pi -10 10])
    hold off

    %plotte interpolasjon p   chebyshevgitter
    subplot(1,2,2);
    plot(x,heaviside(x-1))
    hold on
    plot(x,polyval(polyfit(x_chebyshev, heaviside(x_chebyshev-1),I),x));
    axis([0 pi -10 10])

    pause(.5)
    hold off
end
```

Python:

```
import numpy as np
import time
import matplotlib.pyplot as plt

#Plottepunkter
N=1000
x=np.linspace(0,np.pi,N)

#Antall interpolasjonspunkter
I=15

#Ekvidistant gitter
x_ekvidistant=np.linspace(0,np.pi,I)
#Chebyshev ekstremalgitter
x_chebyshev=(np.cos(np.linspace(0,I,I)*np.pi/I)+1)*np.pi/2

#Vi bruker np.heaviside for aa definere H(x-1)
f = lambda x: np.heaviside(x - np.ones(len(x)),0)


#Vi evaluerer f i plottepunkter og interpolasjonspunkter
f_val = f(x)
f_ekvidistant = f(x_ekvidistant)
f_chebyshev = f(x_chebyshev)


#Plotter heaviside og de to interpolasjonspolynomene i samme plot.
#Merk bruken av polyval og polyfit. Google disse.
plt.plot(x,f_val,label=r'$H(x-1)$')
plt.plot(x,np.polyval(np.polyfit(x_ekvidistant,f_ekvidistant,I),x),label=r'$ekvidistant$')
plt.plot(x,np.polyval(np.polyfit(x_chebyshev,f_chebyshev,I),x),label=r'$chebyshev$')
plt.axis([0,np.pi,-5,15])
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.legend()
plt.savefig('hei')
```

We get the following plot:



Figur 1: Plot of interpolations

As we can see, the Chebyshev interpolation is much better than the equidistant one!

Anbefalte oppgaver

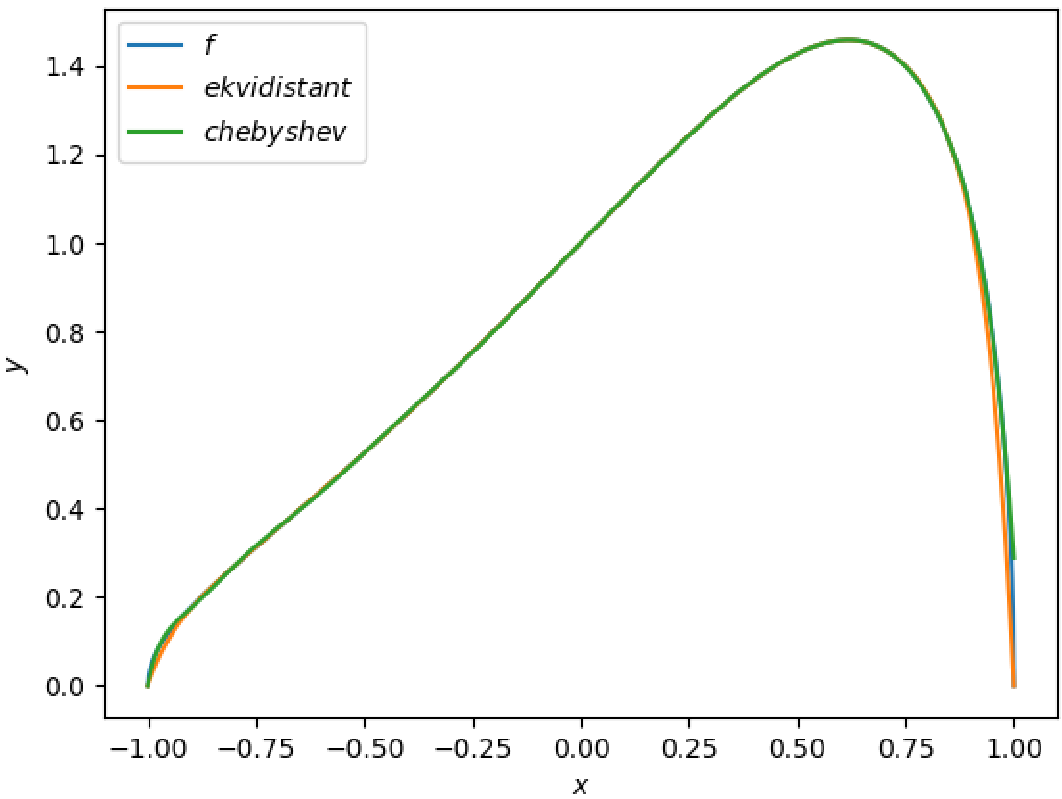
1 **Python:**

Here is the code:

```
import numpy as np
import time
import matplotlib.pyplot as plt


#Plottepunkter
N=1000
x=np.linspace(-1,1,N)

#Antall interpolasjonspunkter
I=15

#Ekvidistant gitter
```

```
x_ekvidistant=np.linspace(-1,1,I)

#Chebyshev ekstremalgitter for intervallet (-1,1)
x_chebyshev=(np.cos(np.linspace(1,I,I)*np.pi / I))

#Kan bruke Lambda-funksjoner for aa definere f:
f = lambda x: np.multiply(np.exp(x), np.sqrt(1-np.multiply(x,x)))

#Vi evaluerer f i plottepunkter og interpolasjonspunkter
f_val = f(x)
f_ekvidistant= f(x_ekvidistant)
f_chebyshev= f(x_chebyshev)

#Vi interpolerer og plotter
plt.plot(x,f_val,label=r'$f$')
plt.plot(x,np.polyval(np.polyfit(x_ekvidistant,f_ekvidistant,I),x),label=r'$ekvidistant$')
plt.plot(x,np.polyval(np.polyfit(x_chebyshev,f_chebyshev,I),x),label=r'$chebyshev$')
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.legend()
plt.savefig('heisann')
```

We get the following plot:



Figur 2: Plot of interpolations

2 Python:

We only use Chebyshev zero point grid (nullpunktgitter) in this case. Here is the code.

```
import numpy as np
import time
import matplotlib.pyplot as plt


#plottepunkter
N=1000
x=np.linspace(-1,1,N)
z=np.zeros(len(x))

#antall interpolasjonspunkter
I=15

#Chebyshev nullpunktgitter:
x_chebyshev=(np.cos((2*np.linspace(1,I,I) - np.ones(I))/(2*I) * np.pi))

#Kan bruke Lambda-funksjoner for aa definere f:
f = lambda x: np.divide(np.exp(x), np.sqrt(1-np.multiply(x,x)))

#Vi evaluerer f i plottepunkter og interpolasjonspunkter
f_val = f(x)
f_chebyshev= f(x_chebyshev)


#Vi interpolerer og plotter
plt.plot(x,f_val,label=r'$f$')
plt.plot(x,np.polyval(np.polyfit(x_chebyshev,f_chebyshev,I),x),label=r'$chebyshev$')
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.legend()
plt.savefig('sveisann')
```

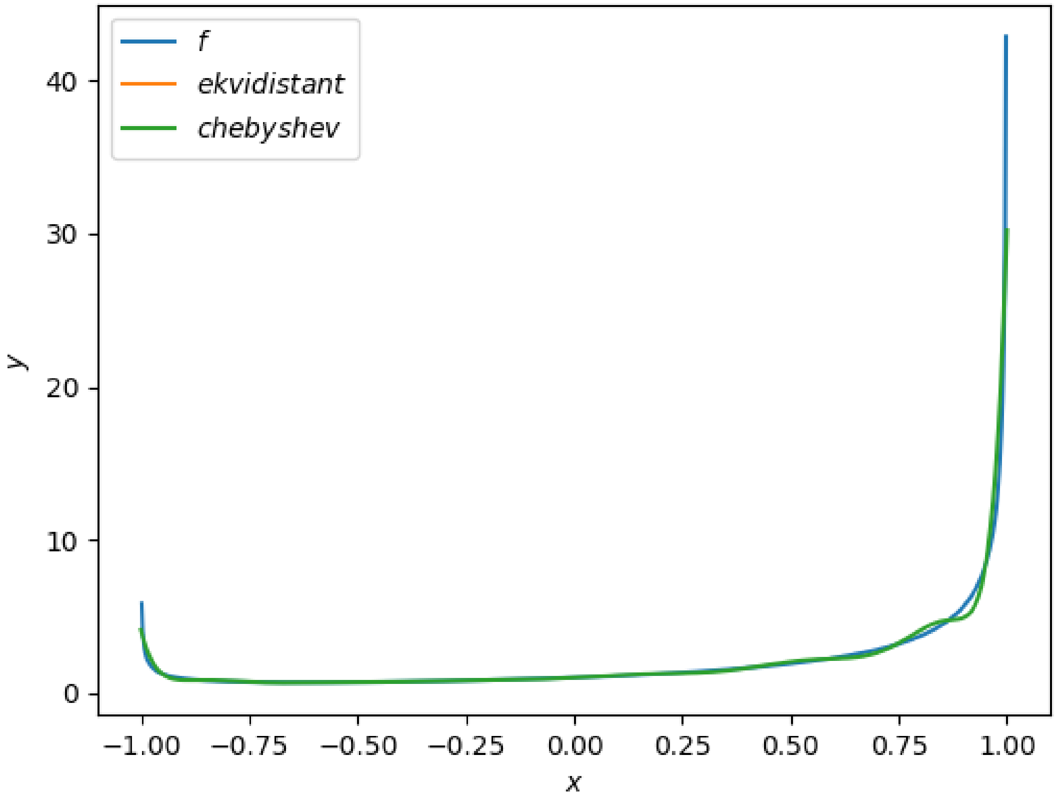
The figure is given below.

3 We have

$$f(x) = \ln(x),$$

and thus

$$\begin{aligned}
f'(x) &= (-1)^0 \frac{1}{x} \\
f''(x) &= (-1)^1 \frac{1}{x^2} \\
f'''(x) &= (-1)^2 2 \frac{1}{x^3} \\
f''''(x) &= (-1)^3 2 \cdot 3 \frac{1}{x^4} \\
&\dots \\
f^{(n)}(x) &= (-1)^{n-1}(n-1)! \frac{1}{x^n}.
\end{aligned}$$



Figur 3: Plot of interpolations

By the interpolation error formula for Chebyshevs zero point grid, we get

$$\max_{x\in[1,2]} |f(x) - p_n(x)| \leq \frac{(2-1)^{n+1}}{2^{2n+1}} \max_{x\in[1,2]} |f^{(n+1)}(x)|$$
$$= \frac{1}{2^{2n+1}} |f^{(n+1)}(1)| = \frac{n!}{2^{2n+1}}.$$

This function increase as n increase, so it seems that we have less and less control over the error, the more points we include.