Probleem 1:

(a)

(i) Lagrange-vorm van die polinoominterpolant:

$$\begin{aligned}
p_4(x) &= 0 + 0 + \frac{(x+2)(x+1)(x-1)(x-3)}{(0+2)(0+1)(0-1)(0-3)} \cdot 1 \\
&\quad + \frac{(x+2)(x+1)(x-0)(x-3)}{(1+2)(1+1)(1-0)(1-3)} \cdot 2 + \frac{(x+2)(x+1)(x-0)(x-1)}{(3+2)(3+1)(3-0)(3-1)} \cdot 4 \\
p_4(2) &= \frac{4 \cdot 3 \cdot 1 \cdot (-1)}{2 \cdot 1 \cdot (-1) \cdot (-3)} \cdot 1 + \frac{4 \cdot 3 \cdot 2 \cdot (-1)}{3 \cdot 2 \cdot 1 \cdot (-2)} \cdot 2 + \frac{4 \cdot 3 \cdot 2 \cdot 1}{5 \cdot 4 \cdot 3 \cdot 2} \cdot 4 \\
&= -2 + 4 + \frac{4}{5} \;=\; 2.8.
\end{aligned}$$

(ii) Neville se metode:

$$\begin{array}{cccccc}
-2 & 0 & & & & \\
 & & 0 & & & \\
-1 & 0 & & 6 & & \\
 & & 3 & & 2 & \\
0 & 1 & & 3 & & 2.8 \\
 & & 3 & & 3 & \\
1 & 2 & & 3 & & \\
 & & 3 & & & \\
3 & 4 & & & &
\end{array}
\qquad \text{Dus } p_4(2) = 2.8.$$

(iii) Newton: Deelverskiltabel

$$\begin{array}{cccccc}
-2 & 0 & & & & \\
 & & 0 & & & \\
-1 & 0 & & 1/2 & & \\
 & & 1 & & -1/6 & \\
0 & 1 & & 0 & & 1/30 \\
 & & 1 & & 0 & \\
1 & 2 & & 0 & & \\
 & & 1 & & & \\
3 & 4 & & & &
\end{array}$$

$$\begin{aligned}
p_4(x) &= 0 + 0 + \frac{1}{2}(x+2)(x+1) - \frac{1}{6}(x+2)(x+1)(x) + \frac{1}{30}(x+2)(x+1)(x)(x-1) \\
&= (x+2)(x+1)\left[\frac{1}{2} - \frac{1}{6}x + \frac{1}{30}x(x-1)\right] \\
p_4(2) &= 4 \cdot 3\left[\frac{1}{2} - \frac{1}{3} + \frac{1}{30} \cdot 2\right] \;=\; 2.8.
\end{aligned}$$

(iv) Met behulp van `neville31`:

```
>> neville31
This is Nevilles Method.
Choice of input method:
1. Input entry by entry from keyboard
2. Input data from a text file
3. Generate data using a function F
Choose 1, 2, or 3 please
 1
Input n
 4
Input X(0) and F(X(0)) on separate lines.
 -2
 0
Input X(1) and F(X(1)) on separate lines.
 -1
 0
Input X(2) and F(X(2)) on separate lines.
 0
 1
Input X(3) and F(X(3)) on separate lines.
 1
 2
Input X(4) and F(X(4)) on separate lines.
 3
 4
Input point at which the polynomial is to be evaluated
 2
Select output destination
1. Screen
2. Text file
Enter 1 or 2
 1
NEVILLES METHOD
Table for P evaluated at X =   2.00000000 , follows:
Entries are XX(I), Q(I,0), ..., Q(I,I) for each I = 0, ..., N where N =   4

-2.00000000  0.00000000
-1.00000000  0.00000000  0.00000000
 0.00000000  1.00000000  3.00000000  6.00000000
 1.00000000  2.00000000  3.00000000  3.00000000  2.00000000
 3.00000000  4.00000000  3.00000000  3.00000000  3.00000000  2.80000000
```

(v) Met behulp van `polyfit`:

```
>> x = [-2 -1 0 1 3]; y = [0 0 1 2 4];
>> a = polyfit(x,y,4);
>> p = polyval(a,2)

p =

    2.8000
```

(b)

Nee, dit is nie toevallig nie. Daar is in die klas aangetoon dat die polinoominterpolant **uniek** is, solank al die x-waardes verskil, wat hier die geval is. Die vyf metodes hierbo behoort dus almal dieselfde waarde vir $p_4(2)$ te lewer.

Probleem 2:

(a)

Ons kan die volgende in MATLAB uitvoer:

```
F = inline('exp(x)');
n = input(' n = ? ');
x = -1+2/n*[0:n];
y = F(x);
a = polyfit(x,y,n);
X = linspace(-1,1,501);
P = polyval(a,X);

figure(1)
plot(x,y,'o',X,P); hold on
plot(X,F(X),'LineWidth',2); title('Funksie en Polinoominterpolant')
figure(2)
plot(X,abs(F(X)-P)); title('Absolute fout')
```

Die grafieke wat ons sodoende verkry word hieronder vertoon, vir $n = 2$, $n = 4$, $n = 6$ en $n = 8$.

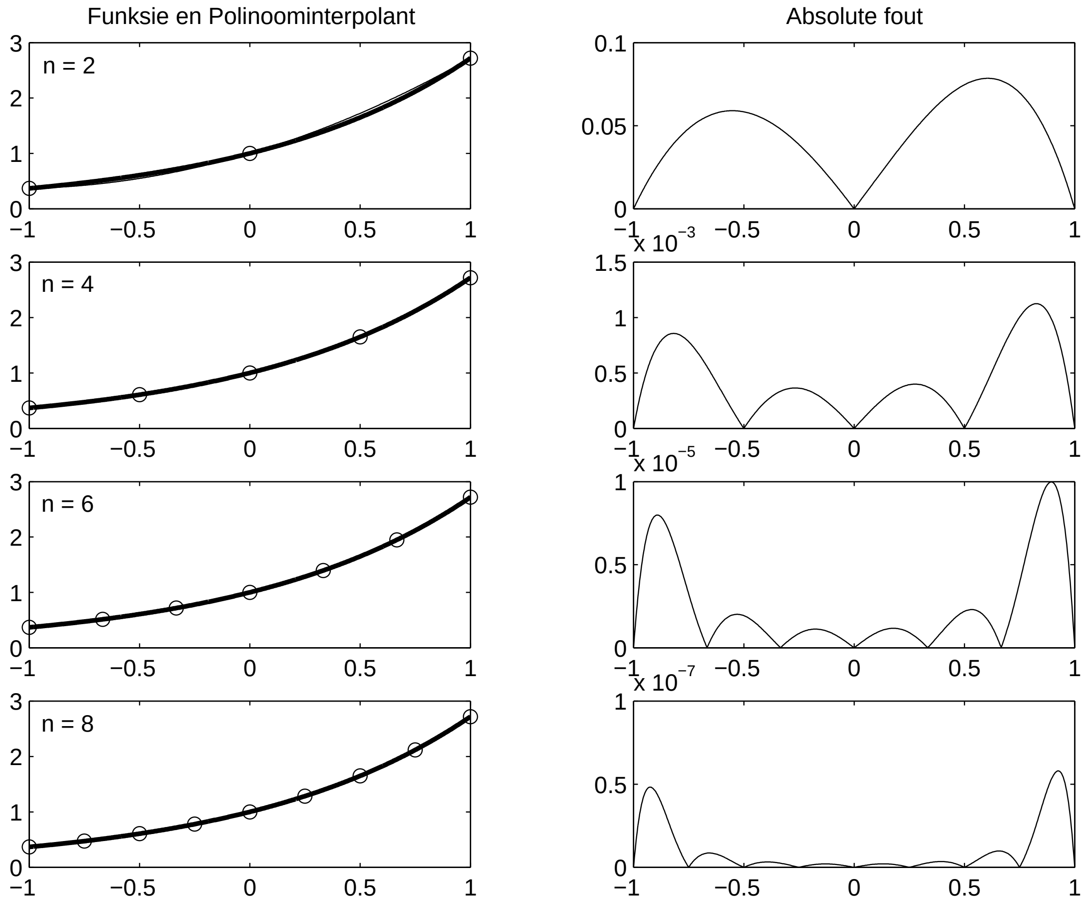

Figure 1: Grafieke van 2(a)

Die waardes van E_n kan van die grafieke afgelees word, en word in die volgende tabel gelys:

n	E_n
2	7.9e−2
4	1.1e−3
6	1.0e−5
8	5.8e−8

Dit lyk dus asof $E_n \to 0$ soos $n \to \infty$, so op grond hiervan wil ons beweer dat die antwoord op die **Vraag** "Ja" is.

(b)

Ons kan dieselfde MATLAB-kode van deel (a) gebruik, deur net die eerste reël te verander na

```
F = inline('1./(1+25*x.^2)');
```

Die grafieke wat sodoende verkry word, word hieronder vertoon, vir $n = 2$, $n = 4$, $n = 6$, $n = 8$ en $n = 10$.

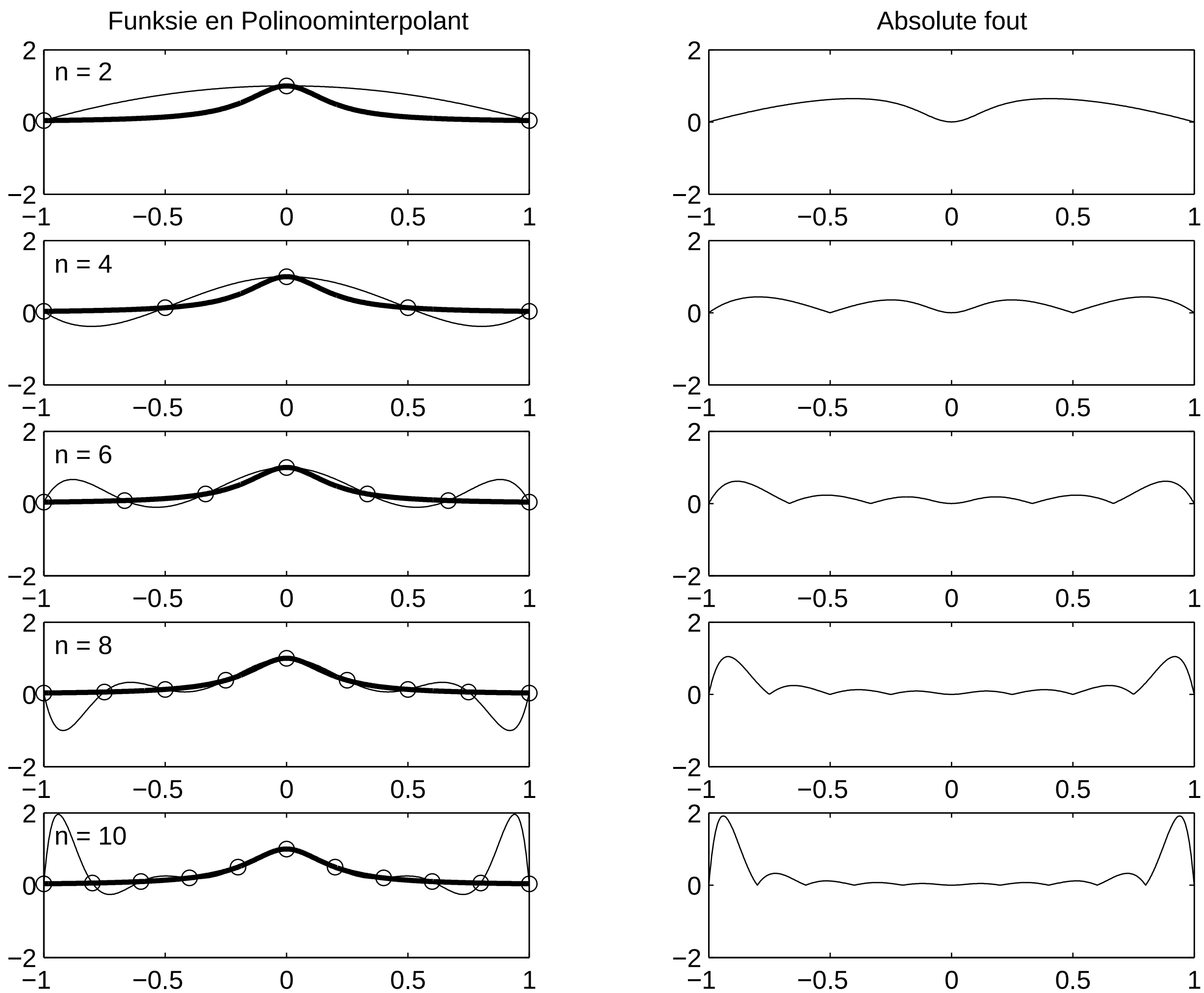

Figure 2: Grafieke van 2(b)

Ons kan weer eens die waardes van E_n aflees, en dit word in die volgende tabel gegee:

n	E_n
2	6.4e−1
4	4.4e−1
6	6.2e−1
8	1.0e+0
10	1.9e+0
12	3.7e+0
14	7.2e+0
16	1.4e+1
18	2.9e+1
20	6.0e+1

Dit lyk nie asof $E_n \to 0$ nie, trouens, $E_n \to \infty$ lyk meer waarskynlik. Ons het dus 'n voorbeeld hier wat wys dat die antwoord op die **Vraag** "Nee" is.

Die moraal van die storie is dat polinoominterpolasie nie vir alle funksies konvergent is nie.

Hierdie voorbeeld word die Runge-voorbeeld genoem, en die groot ossilasies naby $x = -1$ en $x = 1$ word die Runge-ossilasies genoem.

Probleem 3:

(a)

Die volgende kode kan in MATLAB uitgevoer word, om die waardes van $x_0, x_1, \ldots, x_6$ te bereken:

```
>> F = inline('x.^3-x.^2+x-1');
>> x0 = 1.3; x1 = 1.2; x2 = 1.1;
>> y0 = F(x0); y1 = F(x1); y2 = F(x2);
>> a = polyfit([y0 y1 y2],[x0 x1 x2],2); x3 = a(3);
>> y3 = F(x3);
>> a = polyfit([y1 y2 y3],[x1 x2 x3],2); x4 = a(3);
>> y4 = F(x4);
>> a = polyfit([y2 y3 y4],[x2 x3 x4],2); x5 = a(3);
>> y5 = F(x5);
>> a = polyfit([y3 y4 y5],[x3 x4 x5],2); x6 = a(3);
```

Let op die orde van die argumente van die `polyfit`-funksie; inverse interpolasie beteken eers die y's en dan die x'e.

Die resultate word in die volgende tabel gegee:

n	x_n
0	1.30000000000000
1	1.20000000000000
2	1.10000000000000
3	1.00599235709341
4	1.00014573856431
5	1.00000012182777
6	1.00000000000016

(b)

Met $n = 5$ is $e_6 = 1.59 \times 10^{-13}$, $e_5 = 1.22 \times 10^{-7}$ en $e_4 = 1.46 \times 10^{-4}$. Dan

$$\alpha \approx \frac{\log(|e_6|/|e_5|)}{\log(|e_5|/|e_4|)} = 1.91.$$

(Die werklike orde van konvergensie van Brent se metode kan met 'n nie-triviale analise bepaal word, en die resultaat is dat $\alpha = 1.84\ldots$ Let wel, daar is op 'n stadium foutiewelik in die klas berig dat Brent $-$ en ook Müller $-$ se metodes super-kwadraties konvergeer. Dit is inderwaarheid super-lineêr, aangesien $1 < \alpha < 2$.)