Wetenschappelijk rekenen – {`python`, `sympy`, `numpy`, `matplotlib`}

Oefeningen

1 Functie-analyse

Gebruik volgend (vereenvoudigd) stappenplan om het verloop van een functie f te onderzoeken:

1. Bepaal het definitiegebied van de functie f;
2. Zoek de nulpunten en onderzoek het teken;
3. Ga na of de functie asymptoten heeft;
   - De rechte $x = a$ is een verticale asymptoot als en slechts als
   $$\lim_{\substack{x \to a \\ <}} f(x) = \pm\infty \text{ of } \lim_{\substack{x \to a \\ >}} f(x) = \pm\infty;$$
   - De rechte $y = b$ is een horizontale asymptoot als en slechts als
   $$\lim_{x \to -\infty} f(x) = b \text{ of } \lim_{x \to +\infty} f(x) = b;$$
   - De rechte $y = a \cdot x + b$ is een schuine asymptoot als en slechts als
   $$\lim_{x \to \pm\infty} f(x)/x = a \text{ en } \lim_{x \to \pm\infty} f(x) - a \cdot x = b;$$
4. Bepaal de afgeleide functie f', bereken de kritieke punten en onderzoek het teken;
   - De functie heeft een kritiek punt in a als en slechts als $f'(a) = 0$ of, equivalent, als de functie f een horizontale raaklijn $y = f(a)$ heeft in het punt $\big(a, f(a)\big)$;
   - Is $f'(x) > 0$ voor alle $x \in]a, b[$, dan is f stijgend op $]a, b[$;
   - Is $f'(x) < 0$ voor alle $x \in]a, b[$, dan is f dalend op $]a, b[$;
   - Een kritiek punt is een extremum als de afgeleide van teken verandert;
5. Bepaal de tweede afgeleide f'', bereken de nulpunten en onderzoek het teken;
   - Is $f''(x) > 0$ voor alle $x \in]a, b[$, dan is f hol op $]a, b[$;
   - Is $f''(x) < 0$ voor alle $x \in]a, b[$, dan is f bol op $]a, b[$;
   - Een nulpunt a van de tweede afgeleide is een buigpunt met buigraaklijn $y = f'(a) \cdot (x - a) + f(a)$ als de tweede afgeleide van teken verandert in a en de eerste afgeleide $f'(a)$ bestaat;
6. Maak een schets van de functie op basis van deze gegevens.

Pas dit stappenplan toe op volgende functies, gebruik `Python` zo veel mogelijk voor het maken van de berekeningen:

$$f : \mathbb{R} \to \mathbb{R} : x \mapsto \frac{x^3}{x^2 - 1};$$

$$g : \mathbb{R} \to \mathbb{R} : x \mapsto x \cdot \exp(-\frac{x^2}{2});$$

$$h : \mathbb{R} \to \mathbb{R} : x \mapsto \frac{\ln(x)}{x^2}.$$

Modelplossing. Verloop van $f:\mathbb{R}\to\mathbb{R}:x\mapsto x^3/(x^2-1)$.

```
from sympy import *
def f(x): return x**3/(x**2-1)
x = symbols('x',real=True)
solve(denom(f(x)),x)
[-1, 1]
```

Een rationale functie bestaat overal behalve in de nulpunten van de noemer: $\operatorname{def}(f)=\mathbb{R}\setminus\{-1,1\}$.

```
solve(numer(f(x)),x)
[0]
solve(f(x)>0,x)
((-1 < x) & (x < 0)) | ((1 < x) & (x < oo))
solve(f(x)<0,x)
((-oo < x) & (x < -1)) | ((0 < x) & (x < 1))
```

Het enige nulpunt van de functie wordt bereikt bij $x=0$, het teken is als volgt:

x		-1		0		1	
$f(x)$	$-$	|	$+$	0	$-$	|	$+$

```
limit(f(x),x,-1,'-')
-oo
limit(f(x),x,-1,'+')
oo
limit(f(x),x,1,'-')
-oo
limit(f(x),x,1,'+')
oo
limit(f(x),x,+oo)
oo
limit(f(x),x,-oo)
-oo
limit(f(x)/x,x,+oo)
1
limit(f(x)/x,x,-oo)
1
limit(f(x)+x,x,+oo)
oo
limit(f(x)+x,x,-oo)
-oo
```

De functie heeft verticale asymptoten $\mathrm{VA}_1: x=-1$ en $\mathrm{VA}_2: x=1$ ter hoogte van de nulpunten van de noemer. Er is geen horizontale asymptoot maar de functie convergeert in beide richtingen naar de schuine asymptoot $\mathrm{SA}: y=-x$.

```
def Df(x0): return diff(f(x),x).subs(x,x0)
simplify(Df(x))
x**2*(x**2 - 3)/(x**4 - 2*x**2 + 1)
solve(Df(x),x)
[0, -sqrt(3), sqrt(3)]
solve(Df(x)<0,x)
((-1 < x) & (x < 0)) | ((0 < x) & (x < 1)) | ((1 < x) & (x < sqrt(3)))
| ((x < -1) & (-sqrt(3) < x))
solve(Df(x)>0,x)
((x < oo) & (sqrt(3) < x)) | ((-oo < x) & (x < -sqrt(3)))
```

De afgeleide van de functie heeft voorschrift $f':\mathbb{R}\setminus\{-1,1\}\to\mathbb{R}:x\mapsto x^2\cdot(x^2-3)/(x^2-1)^2$. De functie f heeft bijgevolg drie kritieke punten: een lokaal maximum $x_{\max}=-\sqrt{3}$, het kritieke punt 0 en een lokaal minimum $x_{\min}=\sqrt{3}$ (zie tabel verderop).

```
def D2f(x0): return diff(Df(x),x).subs(x,x0)
simplify(D2f(x))
2*x*(x**2 + 3)/(x**6 - 3*x**4 + 3*x**2 - 1)
solve(D2f(x),x)
[0]
```

```
solve(D2f(x)<0,x)
((-oo < x) & (x < -1)) | ((0 < x) & (x < 1))
solve(D2f(x)>0,x)
((-1 < x) & (x < 0)) | ((1 < x) & (x < oo))
```

De tweede afgeleide, met voorschrift $f'' : \mathbb{R} \setminus \{-1, 1\} \to \mathbb{R} : x \mapsto 2 \cdot x \cdot (x^2 + 3)/(x^2 - 1)^3$, heeft één enkel nulpunt in $x_{\text{bgp}} = 0$: dit nulpunt is dus een buigpunt.

Samenvattend wordt het verloop van de functie f beschreven als volgt:

x	$-\infty$		$-\sqrt{3}$		-1		0		1		$\sqrt{3}$		$+\infty$
$f'(x)$		$+$	0	$-$	$\vert$	$-$	0	$-$	$\vert$	$-$	0	$+$	
$f''(x)$		$-$	$-$	$-$	$\vert$	$+$	0	$-$	$\vert$	$+$	$+$	$+$	
$f(x)$	SA	$\nearrow$	x_{max}	$\searrow$	VA_1	$\searrow$	x_{bgp}	$\searrow$	VA_2	$\searrow$	x_{min}	$\nearrow$	SA

Het onderzoek van de functies g en h verloopt analoog.

```
def g(x): return x*exp(-x**2/2)
solve(g(x),x)
[0]
limit(g(x),x,+oo)
0
limit(g(x),x,-oo)
0
def Dg(x0): return diff(g(x),x).subs(x,x0)
simplify(Dg(x))
(-x**2 + 1)*exp(-x**2/2)
solve(Dg(x),x)
[-1, 1]
solve(Dg(x)<0,x)
((-oo < x) & (x < -1)) | ((1 < x) & (x < oo))
solve(Dg(x)>0,x)
(-1 < x) & (x < 1)
def D2g(x0): return diff(Dg(x),x).subs(x,x0)
simplify(D2g(x))
x*(x**2 - 3)*exp(-x**2/2)
solve(D2g(x),x)
[0, -sqrt(3), sqrt(3)]
solve(D2g(x)<0,x)
((0 < x) & (x < sqrt(3))) | ((-oo < x) & (x < -sqrt(3)))
solve(D2g(x)>0,x)
((x < oo) & (sqrt(3) < x)) | ((x < 0) & (-sqrt(3) < x))
```

De functie g is overal gedefinieerd, heeft een nulpunt in $x = 0$, de horizontale asymptoot $\text{HA} : y = 0$ in beide richtingen, twee (absolute) extrema $x_{\text{min}} = -1$ en $x_{\text{max}} = 1$ en drie buigpunten $x_{\text{bgp},1} = -\sqrt{3}$, $x_{\text{bgp},2} = 0$ en $x_{\text{bgp},3} = \sqrt{3}$. Het verloop wordt samengevat in de tabel.

x	$-\infty$		$-\sqrt{3}$		-1		0		1		$\sqrt{3}$		$+\infty$
$f'(x)$		$-$	$-$	$-$	0	$+$	$+$	$+$	0	$-$	$-$	$-$	
$f''(x)$		$-$	0	$+$	$+$	$+$	0	$-$	$-$	$-$	0	$+$	
$f(x)$	HA	$\searrow$	$x_{\text{bgp},1}$	$\searrow$	x_{min}	$\nearrow$	$x_{\text{bgp},2}$	$\nearrow$	x_{max}	$\searrow$	$x_{\text{bgp},3}$	$\searrow$	HA

```
def h(x): return ln(x)/x**2
solve(h(x),x)
[1]
limit(h(x),x,0,'+')
-oo
limit(h(x),x,+oo)
0
def Dh(x0): return diff(h(x),x).subs(x,x0)
simplify(Dh(x))
(-2*log(x) + 1)/x**3
solve(Dh(x),x)
```

```
[exp(1/2)]
solve(Dh(x)<0,x)
(x < oo) & (exp(1/2) < x)
solve(Dh(x)>0,x)
(0 < x) & (x < exp(1/2))
def D2h(x0): return diff(Dh(x),x).subs(x,x0)
simplify(D2h(x))
(6*log(x) - 5)/x**4
solve(D2h(x),x)
[exp(5/6)]
solve(D2h(x)<0,x)
(0 < x) & (x < exp(5/6))
solve(D2h(x)>0,x)
(x < oo) & (exp(5/6) < x)
```

De functie h is enkel gedefinieerd voor strikt positieve waarden, wordt links begrensd door de verticale asymptoot VA : $x = 0$ en benadert rechts de horizontale asymptoot HA : $x = 0$. De functie heeft een nulpunt in $x = 1$, een (absoluut) maximum in $x_{\text{max}} = \exp(1/2)$ en een buigpunten in $x_{\text{bgp}} = \exp(5/6)$. Het verloop wordt samengevat in de tabel.

x	0		exp(1/2)		exp(5/6)		$+\infty$
$f'(x)$		+	0	−	−	−	
$f''(x)$		−	−	−	0	+	
$f(x)$	VA	↱	x_{max}	⤵	x_{bgp}	⤷	HA

2 Grafieken

Herneem de functies van vorige les en maak telkens een grafiek:

- Grafiek van de functie zelf als een dikke groene lijn;
- Asymptoten als rode streepjeslijnen;
- Raaklijnen in kritieke punten en buigpunten in het blauw;
- Markeer extrema en buigpunten.

Enkele tips:

- Grafieken zijn in wezen punten die worden verbonden door lijnstukken. Dat werkt niet goed bij divergerende functies. Vervang grote waarden door `'nan'`. Deze punten zullen niet worden getekend of verbonden, zodat het verticale bereik beperkt blijft en de grafiek niet wordt samengedrukt.
- Functies die enkel standaard `Python`- of `numpy`-bewerkingen gebruiken, kunnen meteen voor een lijst functiewaarden worden berekend. Functies die gebruik maken van `sympy`-commando's, vereisen `lambdify`. Soms is de snelste manier om gewoon met `numpy`-functies te werken (al kan je die dan niet symbolisch afleiden).
- Het volstaat om het begin- en eindpunt van een lijnstuk te berekenen. Kies voor het tekenen van rechten gepaste begin- en eindpunten: niet te ver van kritieke punten voor het tekenen van raaklijnen, aan de grens van het horizontale bereik voor het tekenen van horizontale of schuine asymptoten.
- Elk object past het grafiekgebied aan zodat het volledig zichtbaar is. Gebruik als laatste commando's `plt.xlim([a,b])` en `plt.ylim([c,d])` om het gewenste bereik in te stellen.
- Het volstaat hier om de code voor het maken van deze grafieken *quick & dirty* te schrijven. Later is het de bedoeling om meer algemene functies te schrijven die de nodige rekenstappen automatiseren.

Uitbreiding:

- Zet de grafiek van de functie, afgeleide en tweede afgeleide onder elkaar en maak duidelijk dat de kritieke punten/buigpunten van de ene functie, de nulpunten/kritieke punten zijn van de afgeleide door voor elk punt een andere kleur te gebruiken.

Modeloplossing. Hieronder code voor de grafiek van $f : \mathbb{R} \to \mathbb{R} : x \mapsto x^3/(x^2-1)$ (zie Figuur 1).

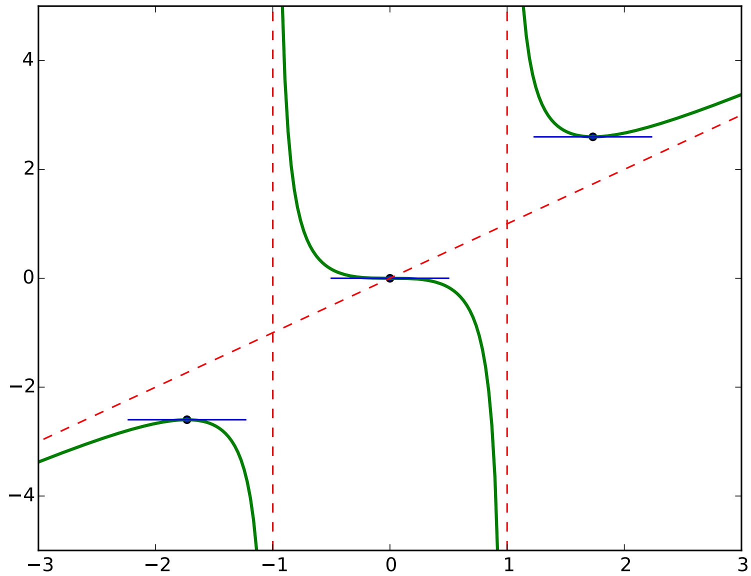
```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
def f(x): return x**3/(x**2-1)
X = np.linspace(-4,4,300)
Y = f(X)
Y[abs(Y)>10] = 'nan'
plt.figure()
plt.plot(X,Y,'g',linewidth=2)
plt.plot((-.5, .5), (0, 0), 'b-')
plt.plot((-np.sqrt(3)-.5, -np.sqrt(3)+.5), (f(-np.sqrt(3)), f(-np.sqrt(3))), 'b-')
plt.plot((np.sqrt(3)-.5, np.sqrt(3)+.5), (f(np.sqrt(3)), f(np.sqrt(3))), 'b-')
plt.plot((-1, -1), (-5, 5), 'r--')
plt.plot((1, 1), (-5, 5), 'r--')
plt.plot((-4,4),(-4,4),'r--')
plt.scatter((-np.sqrt(3),0,np.sqrt(3)),(f(-np.sqrt(3)),f(0),f(np.sqrt(3))))
plt.xlim([-3,3])
plt.ylim([-5,5])
```

Verder de code voor het maken van de grafiek van $g : \mathbb{R} \to \mathbb{R} : x \mapsto x \cdot \exp(-x^2/2)$ (zie Figuur 2).

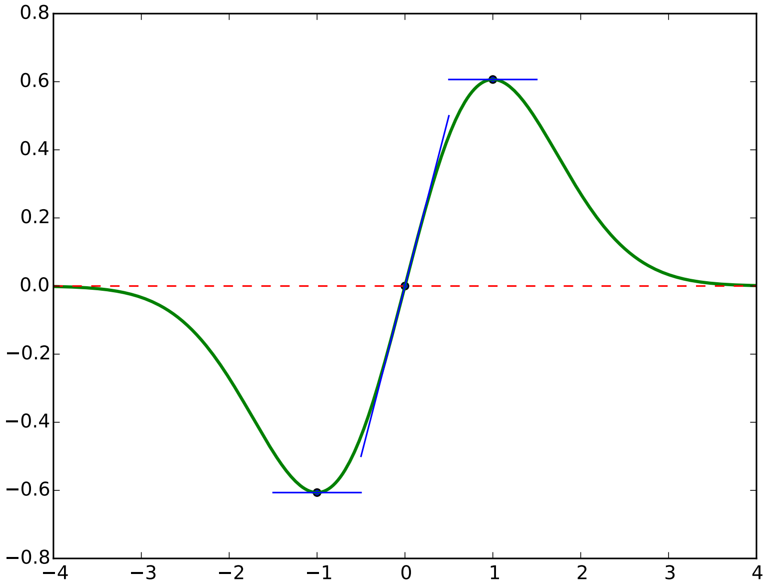
```
def g(x): return x*sp.exp(-x**2/2)
def Dg(x0): return sp.diff(g(x),x).subs(x,x0)
x = sp.symbols('x',real=True)
X = np.linspace(-4,4,300)
Y = sp.lambdify(x,g(x),'numpy')(X)
grafiek = plt.figure()
plt.plot(X,Y,'g',linewidth=2)
plt.plot((-1.5, -0.5), (g(-1), g(-1)), 'b-')
plt.plot((0.5, 1.5), (g(1), g(1)), 'b-')
plt.plot((-0.5,0.5),(Dg(0)*-.5,Dg(0)*.5),'b-')
plt.scatter((-1,0,1),(g(-1),g(0),g(1)))
plt.plot((-4,4),(0,0),'r--')
plt.xlim([-4,4])
```

Tot slot de code voor het maken van de grafiek van $h : \mathbb{R} \to \mathbb{R} : x \mapsto \ln(x)/x^2$ (zie Figuur 3).

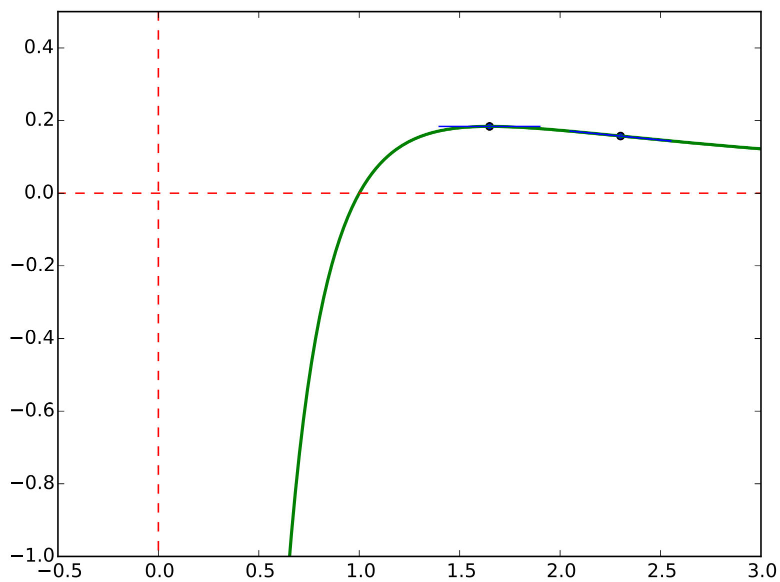
```
def h(x): return sp.ln(x)/x**2
def Dh(x0): return sp.diff(h(x),x).subs(x,x0)
x = sp.symbols('x',real=True)
X = np.linspace(0,3,300)
Y = sp.lambdify(x,h(x),'numpy')(X)
grafiek = plt.figure()
plt.plot(X,Y,'g',linewidth=2)
plt.plot((0, 0), (-10,10), 'r--')
plt.plot((np.exp(1/2)-0.25, np.exp(1/2)+0.25), (h(np.exp(1/2)),h(np.exp(1/2))), 'b-')
plt.plot((np.exp(5/6)-0.25, np.exp(5/6)+0.25), (h(np.exp(5/6))-.25*Dh(np.exp(5/6)),h(np.exp(5/6))+.25*Dh(np
plt.scatter((np.exp(1/2),np.exp(5/6)),((h(np.exp(1/2)),h(np.exp(5/6)))))
plt.plot((-4,4),(0,0),'r--')
plt.xlim([-.5,3])
plt.ylim([-1,.5])
```



Figuur 1: Grafiek van de functie f.



Figuur 2: Grafiek van de functie g.



Figuur 3: Grafiek van de functie h.

3 Vectoren en matrices

1. Bereken de eerste duizend Mersenne-getallen $2^n - 1$ met $n \in \mathbb{N}$, bepaal voor welke waarden n dit priemgetallen zijn en maak een lijst van deze zogenaamde Mersenne-priemgetallen. Maak de grafiek $\big(n, M(n)\big)$, met $M(n)$ het aantal Mersenne-priemgetallen kleiner dan of gelijk aan $2^n - 1$.

   Enkele tips:

   - Gestructureerde lijsten maken, kan met `[... for ... in ... if ...]`.
   - Importeer het commando `isprime()` met `from sympy.ntheory import isprime`.
   - Een trapfunctie voorstellen in `matplotlib` kan met het commando `step()`.

2. Een matrix van de volgende vorm, met $x_i \in \mathbb{R}$, heet een Vandermonde-matrix,

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{pmatrix}.$$

   Maak simultaan meerdere veranderlijken met het commando `symbols('x:n')` en genereer daarmee de Vandermonde-matrix van dimensie vijf. Toon aan dat de determinant van deze matrix verschilt van nul als en slechts als alle getallen x_i onderling verschillend zijn.

3. Los volgend stelsel op als verzameling van vergelijkingen (`solve()`) en met matrixrekenen:

$$\mathbf{A} \cdot \vec{x} = \vec{b} \Leftrightarrow \vec{x} = \mathbf{A}^{-1} \cdot \vec{b}.$$

$$\begin{cases} x + 2 \cdot y + 4 \cdot z &= 1 \\ 3 \cdot x + 8 \cdot y + 14 \cdot z &= 2 \\ 2 \cdot x + 6 \cdot y + 11 \cdot z &= 3 \end{cases}$$

4. Voor welke waarden van a en b heeft volgend stelsel geen/één/oneindig veel oplossingen?

$$\begin{cases} x + a \cdot y &= 1 \\ 2 \cdot x + 3 \cdot y &= b \end{cases}$$

5. Volg onderstaand (vereenvoudigd) stappenplan om de lokale extrema van volgende functie van twee veranderlijken $f : \mathbb{R}^2 \to \mathbb{R} : (x, y) \mapsto x^3 - x + y^3 - y$ te vinden.

   - De rol van afgeleide bij een functie van twee veranderlijken wordt gespeeld door de gradiënt ∇f. Definieer deze functie:

$$\nabla f : \mathbb{R}^2 \to \mathbb{R}^2 : (x, y) \mapsto \Big(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\Big).$$

     De partiële afgeleiden $\frac{\partial f}{\partial x}$ en $\frac{\partial f}{\partial y}$ worden bekomen door $f(x, y)$ één keer af te leiden naar x respectievelijk y
   - De kritieke punten van de functie f zijn de punten waarvoor de gradiënt gelijk is aan de nulvector. Los het stelsel op:

$$\begin{cases} \frac{\partial f}{\partial x}(x, y) &= 0 \\ \frac{\partial f}{\partial y}(x, y) &= 0 \end{cases}$$

   - De rol van tweede afgeleide bij een functie van twee veranderlijken wordt gespeeld door de hessiaan Hf. Definieer deze functie:

$$Hf : \mathbb{R}^2 \to \mathbb{R}^{2\times 2} : (x, y) \mapsto \begin{pmatrix} \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) & \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) \\ \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) & \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right) \end{pmatrix}$$

     De partiële afgeleiden van tweede orde worden bekomen door $f(x, y)$ twee keer in de aangegeven volgorde af te leiden naar x of y.
   - De kritieke punten waar de determinant van de Hessiaan positief is, zijn (lokale) extrema. Is $\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)$ negatief, dan gaat het om een maximum. Is $\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)$ positief, dan gaat het om een minimum.

Tabel 1: Exponenten n waarvoor Mersenne-getallen $2^n - 1$ priem zijn.

	n	$2^n - 1$
1	2	3
2	3	7
3	5	31
4	7	127
5	13	8191
6	17	131071
7	19	524287
8	31	2147483647
9	61	2305843009213693951
10	89	618970019642690137449562111
11	107	162259276829213363391578010288127
12	127	170141183460469231731687303715884105727
13	521	$6.86479766013061 \cdot 10^{156}$
14	607	$5.31137992816767 \cdot 10^{182}$

Modeloplossing. In deze oefeningen primeert niet de wiskunde die uitgebreid aan bod komt in andere vakken: focus in eerste plaats op de code.

1. Van de kleinste 1000 Mersenne-getallen zijn er 14 priem, een overzicht in Tabel 1. De grafiek $\big(n, M(n)\big)$, met $M(n)$ het aantal Mersenne-priemgetallen kleiner dan of gelijk aan $2^n - 1$ is afgebeeld in Figuur 4.

```
from sympy.ntheory import isprime
mersenneGetallen = [2**n-1 for n in range(1000)]
mersenneNPriem = [n for n in range(1000) if isprime(mersenneGetallen[n])]
mersennePriemgetallen = [2**n-1 for n in mersenneNPriem]
aantal = len(mersennePriemgetallen)
import matplotlib.pyplot as plt
plt.figure()
plt.step(mersenneNPriem,range(aantal))
plt.savefig('mersenne.pdf')
plt.close()
```

2. De determinant van de Vandermonde-matrix is gelijk aan

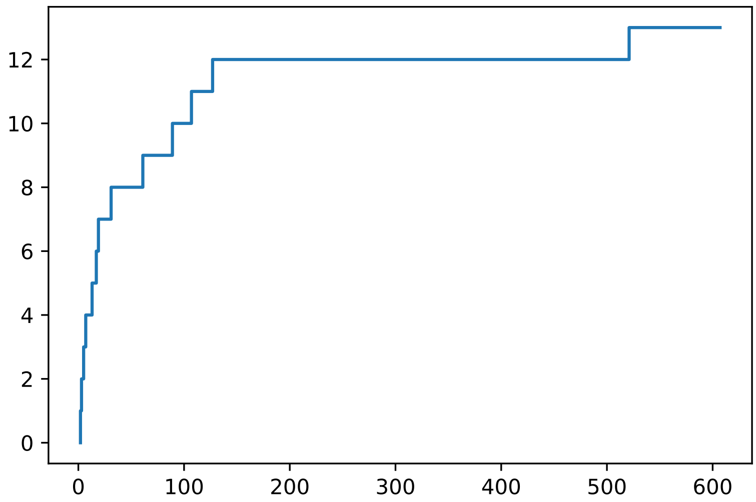
$$\prod_{0 \leq i < j \leq n} (x_i - x_j)$$

Hieruit blijkt inderdaad dat deze nul is van zodra twee getallen x_i gelijk zijn.

```
from sympy import *
V = Matrix([ [k**n for n in range(5)] for k in symbols('x:5') ])
factor(V.det())
(x0 - x1)*(x0 - x2)*(x0 - x3)*(x0 - x4)*(x1 - x2)*(x1 - x3)*(x1 -
x4)*(x2 - x3)*(x2 - x4)*(x3 - x4)
```

3. Het stelsel heeft de unieke oplossing $(x, y, z) = (-2, -5/2, 2)$.

```
from sympy import *
x,y,z = symbols('x y z')
solve([ x+2*y+4*z-1, 3*x+8*y+14*z-2, 2*x+6*y+11*z-3 ], x,y,z)
{x: -2, y: -5/2, z: 2}
A = Matrix([ [1,2,4], [3,8,14], [2,6,11] ])
b = Matrix([1,2,3])
A.inv()*b
```



Figuur 4: Aantal Mersenne-priemgetallen kleiner dan $2^n - 1$.

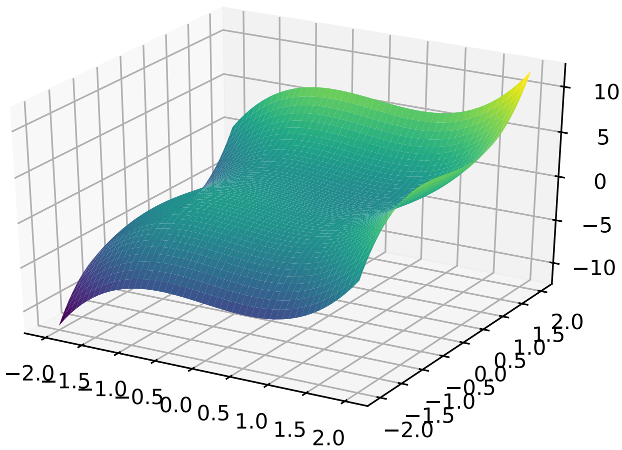
```
Matrix([
[  -2],
[-5/2],
[   2]])
```

4. De determinant verschilt van nul en het stelsel heeft dus een unieke oplossing als $a = 3/2$. In het andere geval zijn er geen oplossingen, tenzij beide vergelijkingen identiek zijn: als $a = 3/2$ en $b = 2$ zijn er oneindig veel oplossingen.

```
x,y,a,b = symbols('x y a b')
A = Matrix([[1,a],[2,3]])
solve(A.det(),a)
[3/2]
A.inv()*Matrix([1,b])
Matrix([
[-a*b/(-2*a + 3) + 3/(-2*a + 3)],
[   b/(-2*a + 3) - 2/(-2*a + 3)]])
solve([x+a*y-1,2*x+3*y-b],x,y)
{y: (-b + 2)/(2*a - 3), x: (a*b - 3)/(2*a - 3)}
solve([x+Rational(3,2)*y-1,2*x+3*y-2],x,y)
{x: -3*y/2 + 1}
```

5. De functie f heeft kritieke punten $(-\sqrt{3}/3, -\sqrt{3}/3)$, $(-\sqrt{3}/3, \sqrt{3}/3)$, $(\sqrt{3}/3, -\sqrt{3}/3)$ en $(\sqrt{3}/3, \sqrt{3}/3)$. Voor het eerste en laatste kritieke punt heeft de Hessiaan een positieve determinant, dit zijn dus lokale extrema: $(-\sqrt{3}/3, -\sqrt{3}/3)$ is een maximum, $(\sqrt{3}/3, \sqrt{3}/3)$ is een minimum. Dit is ook te zien op de grafiek (Figuur 5).

```
from sympy import *
x,y = symbols('x y')
def f(x,y): return x**3-x+y**3-y
def gradient(x0,y0): return Matrix([diff(f(x,y),x), diff(f(x,y),y)]).subs(x,x0).subs(y,y0)
gradient(x,y)
Matrix([
[3*x**2 - 1],
[3*y**2 - 1]])
kritieke = solve(gradient(x,y),x,y)
kritieke
```



Figuur 5: Grafiek van de functie $f : \mathbb{R}^2 \to \mathbb{R} : (x, y) \mapsto (x^3 - x + y^3 - y)$.

```
[(-sqrt(3)/3, -sqrt(3)/3),
 (-sqrt(3)/3, sqrt(3)/3),
 (sqrt(3)/3, -sqrt(3)/3),
 (sqrt(3)/3, sqrt(3)/3)]
def hessiaan(x0,y0): return Matrix([[diff(f(x,y),x,x), diff(f(x,y),x,y)], [diff(f(x,y),y,x), diff(f(x,
hessiaan(x,y)
Matrix([
[6*x,   0],
[  0, 6*y]])
determinant = [hessiaan(punt[0],punt[1]).det() for punt in kritieke]
orientatie = [hessiaan(punt[0],punt[1])[0,0] for punt in kritieke]
Matrix([determinant,orientatie])
Matrix([
[        12,        -12,        -12,        12],
[-2*sqrt(3), -2*sqrt(3), 2*sqrt(3), 2*sqrt(3)]])
plotF = plotting.plot3d(f(x,y),(x,-2,2),(y,-2,2))
plotF.save('grafiek3D.pdf')
```

4 Programmeerstructuren

Het onderzoek naar het verloop van een functie en het maken van de grafiek met asymptoten en raaklijnen in extrema en buigpunten wordt hieronder geautomatiseerd. Het is principieel onmogelijk om dit in volle algemeenheid te doen, onder meer omdat er geen methoden bestaan om nulpunten van een algemene vergelijking te berekenen. Beperk daarom de code tot *eenvoudige* rationale functies, functies waarvoor de vereiste `solve()`-commando's werken. Test de procedures voor de functies

$$f : \mathbb{R} \to \mathbb{R} : x \mapsto \frac{x^3}{x^2 - 1} \text{ en } g : \mathbb{R} \to \mathbb{R} : x \mapsto \frac{x^3 - x^2 + x - 1}{x^3 + x^2 - x - 1}.$$

`asymptootOneindig` De eerste functie berekent of er een horizontale of schuine asymptoot is. Gebruik de methode `.is_finite` om na te gaan of een limiet al dan niet oneindig is. Hou er rekening mee dat bij rationale functies de aymptoot voor positieve en negatieve waarden identiek is. De functie drukt een passende tekst af (`Er is geen/een horizontale/een schuine asymptoot met vergelijking ...`). De respons is `None` als er geen asymptoot is en een functie zoals $a : x \mapsto x$ in het voorbeeld hieronder als er een horizontale of schuine asymptoot is.

```
asymptoot_f = asymptootOneindig(f)
Er is een schuine asymptoot met vergelijking y = x
asymptoot_f(x)
x
asymptoot_g = asymptootOneindig(g)
Er is een horizontale asymptoot met vergelijking y = 1
asymptoot_g(x)
1
```

`klasseerNullen(f)` Deze functie heeft als argument een rationale functie, berekent de reële nulpunten van teller en noemer en klasseert deze. Gebruik volgende stappen:

- Bereken de nulpunten van teller én noemer. Voeg deze samen tot één lijst. Maak hiervan een verzameling met de functie `set()`. Een verzameling gedraagt zich net zoals een lijst, maar bevat elk element slechts één keer.
- Is een element a van de bekomen verzameling enkel nulpunt van de teller, dan is het een nulpunt van de functie. Is het ook nulpunt van de noemer en is de limiet $\lim_{x \to a} f(x)$ een reëel getal, dan heeft de functie een perforatie in a. Zoniet heeft de functie een verticale asymptoot $x = a$.
- Druk per nulpunt een boodschap "`a - nulpunt`", "`a - perforatie`" of "`a - asymptoot`" (vul uiteraard het correcte getal in) en voeg `a` toe aan de gepaste van drie lijsten `nulpunten`, `perforaties` en `asymptoten`.

De output van deze procedure is een lijst `[nulpunten,perforaties,asymptoten]` van drie (eventueel lege) lijsten.

```
klasseerNullen(f)
0 -- nulpunt
1 -- asymptoot
-1 -- asymptoot
[[0], [], [1, -1]]
klasseerNullen(g)
1 -- perforatie
-1 -- asymptoot
[[], [1], [-1]]
```

`klasseerKritiekePunten(f)` Deze functie heeft opnieuw als argument een rationale functie, berekent nu de reële nulpunten van eerste en tweede afgeleide, en klasseert deze. Gebruik volgende stappen:

- Bereken de nulpunten van de eerste afgeleide. Is de tweede afgeleide in zo een nulpunt strikt positief (negatief), dan heeft de functie een lokaal minimum (maximum) in dat punt.
- Bereken de nulpunten van de tweede afgeleide. Is de derde afgeleide in zo een nulpunt verschillend van nul, dan heeft de functie daar een buigpunt. Bereken voor elk buigpunt de vergelijking van de buigraaklijn.

- Druk opnieuw een boodschap af ("`a - maximum`", "`a - minimum`" of "`a - buigpunt`"). Druk bijkomend de vergelijking van de buigraaklijn af in het geval van een buigpunt. Voeg de punten toe aan de gepaste lijst `minima`, `maxima` en `buigpunten`.

De output van deze procedure is een lijst `[minima,maxima,buigpunten]` van drie (eventueel lege) lijsten.

```
klasseerKritiekePunten(f)
-sqrt(3) -- maximum
sqrt(3) -- minimum
0 -- buigpunt
De buigraaklijn heeft vergelijking y= 0
[[sqrt(3)], [-sqrt(3)], [0]]
klasseerKritiekePunten(g)
2 -- buigpunt
De buigraaklijn heeft vergelijking y= 2*x/27 + 11/27
[[], [], [2]]
```

`functieVerloop(f)` De laatste procedure bundelt de voorgaande en drukt een samenvatting af van het verloop van de functie. Er is verder geen output.

```
functieVerloop(f)
Analyse van de grafiek met vergelijking
y = x**3/(x**2 - 1)
Er is een schuine asymptoot met vergelijking y = x
Merkwaardige punten:
0 -- nulpunt
1 -- asymptoot
-1 -- asymptoot
-sqrt(3) -- maximum
sqrt(3) -- minimum
0 -- buigpunt
De buigraaklijn heeft vergelijking y= 0
functieVerloop(g)
Analyse van de grafiek met vergelijking
y = (x**3 - x**2 + x - 1)/(x**3 + x**2 - x - 1)
Er is een horizontale asymptoot met vergelijking y = 1
Merkwaardige punten:
1 -- perforatie
-1 -- asymptoot
2 -- buigpunt
De buigraaklijn heeft vergelijking y= 2*x/27 + 11/27
```

Giet dit alles tot slot in een `.py`-script met volgende structuur en voeg commentaar toe.

```
import sympy as sp

def main():
def f(x): return x**3/(x**2-1)
functieVerloop(f)
def g(x): return (x**3-x**2+x-1)/(x**3+x**2-x-1)
functieVerloop(g)

def ...
...

main()
```

Bijkomende oefening. Het is zeer interessant om de output van alle voorgaande procedures te gebruiken om de grafiek van de functie te tekenen, samen met gevonden asymptoten, nulpunten, perforaties, extrema, buigpunten, horizontale en buigraaklijnen zoals in sessie 2. Bepaal het bereik van de assen zo dat alle gevonden merkwaardige punten goed zichtbaar zijn.

Modeloplossing.

```python
# Het script functieverloop.py geeft een samenvatting van het verloop van rationale functies.
import sympy as sp

def main():
  def f(x): return x**3/(x**2-1)
  functieVerloop(f)
  def g(x): return (x**3-x**2+x-1)/(x**3+x**2-x-1)
  functieVerloop(g)
  return None

## functieVerloop(f) berekent asymptoten van een rationale functie;
## klaseert nulpunten van teller en noemer, kritieke punten en buigpunten;
## berekent de buigraaklijn in elk buigpunt.
## Er is geen andere uitvoer dan een afgedrukte samenvatting.
# @param f definitie van een rationale functie
# @return geen
def functieVerloop(f):
  x, a = sp.symbols('x a',real=True)
  print('Analyse van de grafiek met vergelijking')
  print('y =',f(x))
  asymptootOneindig(f)
  print('Merkwaardige punten:')
  klasseerNullen(f)
  klasseerKritiekePunten(f)
  return None

## asymptootOneindig(f) gaat na of er een horizontale of schuine asymptoot is;
## drukt een passende boodschap af en geeft de functie die de asymptoot beschrijft terug.
# @param f definitie van een rationale functie
# @return definitie van de functie die de asymptoot beschrijft als er een is, None in het andere geval
def asymptootOneindig(f):
  x = sp.symbols('x',real=True)
  limiet = sp.limit(f(x),x,sp.oo)
  if limiet.is_finite:
    def asymptoot(x): return limiet
    print('Er is een horizontale asymptoot met vergelijking y =',limiet)
  else:
    a = sp.limit(f(x)/x,x,sp.oo)
    b = sp.limit(f(x)-a*x,x,sp.oo)
    if a.is_finite and b.is_finite:
      def asymptoot(x): return a*x+b
      print('Er is een schuine asymptoot met vergelijking y =',asymptoot(x))
    else:
      asymptoot = None
      print('Er is geen horizontale of schuine asymptoot')
  return asymptoot

## klasseerNullen(f) berekent nulpunten van teller en noemer van een rationale functie;
## gaat na of deze corresponderen met een nulpunt, perforatie of verticale asymptoot
# @param f definitie van een rationale functie
# @return lijst van drie lijsten met respectievelijk nulpunten, perforaties en asymptoten
def klasseerNullen(f):
  x = sp.symbols('x',real=True)
  nulpuntenTeller = sp.solve(sp.numer(f(x)),x)
  nulpuntenNoemer = sp.solve(sp.denom(f(x)),x)
  nullen = set(nulpuntenTeller + nulpuntenNoemer)
  nulpunten = []
  perforaties = []
  asymptoten = []
```

```python
  for k in nullen:
    if k not in nulpuntenNoemer:
      print(k,'-- nulpunt')
      nulpunten += [k]
    elif k in nulpuntenTeller and sp.limit(f(x),x,k).is_finite:
      print(k,'-- perforatie')
      perforaties += [k]
    else:
      print(k,'-- asymptoot')
      asymptoten += [k]
  return [nulpunten,perforaties,asymptoten]

## raaklijn(f,a) berekent de raaklijn aan een functie f in a
## en geeft de definitie van de functie die de raaklijn beschrijft terug
# @param f definitie van een rationale functie
# @param a reëel getal in definitiegebied van de functie
# @return definitie van de functie die de raaklijn beschrijft
def raaklijn(f,a):
  x = sp.symbols('x',real=True)
  def Df(x0): return sp.diff(f(x),x).subs(x,x0)
  def rkl(x): return Df(a)*(x-a)+f(a)
  return rkl

## klasseerKritiekePunten(f) berekent kritieke punten en buigpunten van een rationale functie;
## gaat na of deze corresponderen met een maximum of een minimum
# @param f definitie van een rationale functie
# @return lijst van drie lijsten met respectievelijk minima, maxima en buigpunten
def klasseerKritiekePunten(f):
  x = sp.symbols('x',real=True)
  def Df(x0): return sp.diff(f(x),x).subs(x,x0)
  def D2f(x0): return sp.diff(Df(x),x).subs(x,x0)
  def D3f(x0): return sp.diff(D2f(x),x).subs(x,x0)
  kritiekePunten = sp.solve(Df(x),x)
  nulpuntenTweedeAfgeleide = sp.solve(D2f(x),x)
  maxima = []
  minima = []
  buigpunten = []
  for k in kritiekePunten:
    if D2f(k)>0:
      print(k,'-- minimum')
      minima += [k]
    elif D2f(k)<0:
      print(k,'-- maximum')
      maxima += [k]
  for k in nulpuntenTweedeAfgeleide:
    if D3f(k)!=0:
      print(k,'-- buigpunt')
      print('De buigraaklijn heeft vergelijking y=',raaklijn(f,k)(x))
      buigpunten += [k]
  return [minima,maxima,buigpunten]

# main()
```