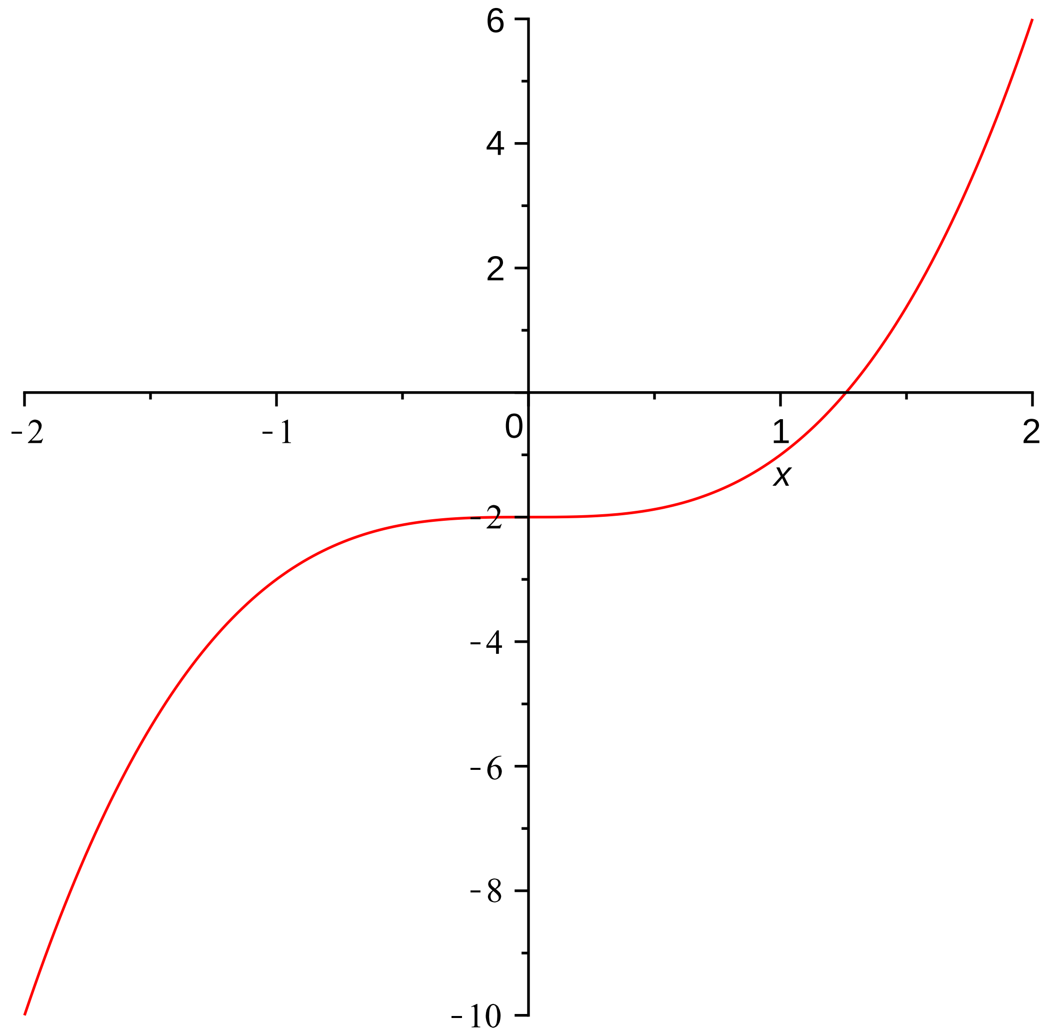
## Opgave 3.2.9

> $f := x \rightarrow x^3 - 2; fp := D(f);$

$$f := x \mapsto x^3 - 2$$
$$fp := x \mapsto 3\,x^2 \tag{1.1}$$

> $plot(f(x), x = -2..2)$




> **Lad os prøve at køre Newtons metode et trin ad gangen:**

> $Newton := x \rightarrow evalf\left(x - \frac{f(x)}{fp(x)}\right)$

$$Newton := x \mapsto evalf\left(x - \frac{f(x)}{fp(x)}\right) \tag{1.2}$$

> $Newton(-1)$

$$0. \tag{1.3}$$

> $Newton(0)$

Error, (in Newton) numeric exception: division by zero

> **Newtons metode får aldrig en x[2] værdi da den bliver ∞ grundet division med 0.**

## Opgave 3.2.23a

> Vi benytter samme fremgangsmåde som i Lecture 3.

> $restart;$

> $with(LinearAlgebra):$

> $f1 := 4x_1^2 - x_2^2 = 0; f2 := 4x_1 \cdot x_2^2 - x_1 = 1$

$$f1 := 4x_1^2 - x_2^2 = 0$$
$$f2 := 4x_1 x_2^2 - x_1 = 1 \tag{2.1}$$

> $X := Vector(2, [x_1, x_2]); F := Vector(2, [4x_1^2 - x_2^2, 4x_1 \cdot x_2^2 - x_1 - 1])$

$$X := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$F := \begin{bmatrix} 4x_1^2 - x_2^2 \\ 4x_1 x_2^2 - x_1 - 1 \end{bmatrix} \tag{2.2}$$

> $DF := Matrix(2);$

$$DF := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.3}$$

```
> for i from 1 to 2 do
    DF[i, 1] := diff(F[i], x_1);
    DF[i, 2] := diff(F[i], x_2);
  end do;
```

$$DF_{1,1} := 8x_1$$
$$DF_{1,2} := -2x_2$$
$$DF_{2,1} := 4x_2^2 - 1$$
$$DF_{2,2} := 8x_1 x_2 \tag{2.4}$$

> $DF$

$$\begin{bmatrix} 8x_1 & -2x_2 \\ 4x_2^2 - 1 & 8x_1 x_2 \end{bmatrix} \tag{2.5}$$

> $S := LinearSolve(DF, -F) + X$

$$S := \begin{bmatrix} -\dfrac{-1 - x_1 + 16x_1^3}{32x_1^2 + 4x_2^2 - 1} + x_1 \\ -\dfrac{-8x_1 + 4x_2^4 + 16x_1^2 x_2^2 - 4x_1^2 - x_2^2}{2x_2\left(32x_1^2 + 4x_2^2 - 1\right)} + x_2 \end{bmatrix} \tag{2.6}$$

> $sol := [0, 1]$

$$sol := [0, 1] \tag{2.7}$$

> **for** $i$ **from** 1 **to** 7 **do**
> $sol := evalf(subs(x_1 = sol[1], x_2 = sol[2], S));$
> **end do**;

$$sol := \begin{bmatrix} 0.3333333333 \\ 0.5000000000 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.5416666666 \\ 1.250000000 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.4732764073 \\ 0.9759013283 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.4509380727 \\ 0.9036610198 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.4490917067 \\ 0.8981924697 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.4490804763 \\ 0.8981609527 \end{bmatrix}$$

$$sol := \begin{bmatrix} 0.4490804758 \\ 0.8981609516 \end{bmatrix} \tag{2.8}$$

> $fsolve([f1, f2])$

$$\{x_1 = 0.4490804758, x_2 = -0.8981609516\} \tag{2.9}$$

> **Vi udregnede 7 iterationer for at få et mere korrekt resultat. Dog ved vi stadig ikke hvorfor fsolve giver et negativt resultat for $x_2$.**

# Opgave cp 3.2.1

Vi kan bruge den tidligere skrevne funktion Newton til rekursivt at udregne rødderne nærmest 4.5 og 7.7:

> $restart;$

Vi laver en ny procedure som tager en funktion, et tal den skal bruge til at finde Newtons rod tæt på, og et antal af iterationer den må køre Newtons metode rekursivt.

> $RecNewton :=$ **proc**$(func, num, iter :: integer)$
> **local** $fp, Newton, t, n;$
> $fp := D(func);$
> $Newton := x \rightarrow evalf\left(x - \frac{func(x)}{fp(x)}\right);$
> $t := num;$
> **for** $n$ **from** 1 **to** $iter$ **do**
> $t := Newton(t);$

```
end do;
return t;
end proc;
```

*RecNewton* := **proc**(*func*, *num*, *iter*::*integer*) **(3.1)**

```
    local fp, Newton, t, n;
    fp := D(func);
    Newton := x→evalf(x − func(x) / fp(x));
    t := num;
    for n to iter do t := Newton(t) end do;
    return t
end proc
```

\> $f := x \to \tan(x) - x;$

$$f := x \mapsto \tan(x) - x \tag{3.2}$$

\> *RecNewton*(*f*, 4.5, 10)

$$4.4934094579090641753 \tag{3.3}$$

\> *RecNewton*(*f*, 7.7, 10)

$$7.7252518369377071642 \tag{3.4}$$

# Opgave 3.4.7

\> *restart*;

\> *RecCos* := **proc**(*val*, *num*)

```
local temp, i;
temp := val;
for i from 1 to num do
temp := evalf(cos(temp));
end do;
return temp;
end proc;
```

*RecCos* := **proc**(*val*, *num*) **(4.1)**

```
    local temp, i;
    temp := val; for i to num do temp := evalf(cos(temp)) end do; return temp
end proc
```

\> *Digits* := 50;

$$Digits := 50 \tag{4.2}$$

\> *RecCos*(1, 290);

$$0.73908513321516064165531208767387340401341175890076 \tag{4.3}$$

Det lader til at man skal tage cos(x) rekursivt på sig selv 290 gange for at finde dottie nummeret præcist til 50 decimaler. Ellers kan man bruge maples fsolve funktion:

\> *fsolve*(cos(*x*) = *x*, *x*)

$$0.73908513321516064165531208767387340401341175890076 \tag{4.4}$$

Vi kigger udelukkende på cos funktionen på intervallet [0,1].
For at vise at cos(x)=x convergerer skal vi finde et lambda < 1 således at:

\> $\mathrm{abs}(\cos(x_0) - \cos(x_1)) \le \lambda \cdot \mathrm{abs}(x_0 - x_1);$

**(4.5)**

$$|\cos(x_0) - \cos(x_1)| \leq \lambda |x_0 - x_1| \quad (4.5)$$

Dette λ finder vi ved hjælp af vores Mean Value Theorem.
For $f(x) = \cos(x)$ og vha. Mean Value Theorem ved vi at
$\cos(x) - \cos(y) = \cos'(t) \cdot (x - y) = (-\sin(t) \cdot (x - y)$
for et t mellem x og y. Dvs. at

> $|\cos(x) - \cos(y)| = |-\sin(t)| \cdot |x - y|$

$$|\cos(x) - \cos(y)| = |\sin(t)|\,|x - y| \quad (4.6)$$

Vi ved at sin stiger på intervallet [0,1] og eftersom sin(1)<1 kan vi bruge det som det lambda vi leder efter:

> $|\cos(x) - \cos(y)| \leq \sin(1)|x - y|$

$$|\cos(x) - \cos(y)| \leq \sin(1)\,|x - y| \quad (4.7)$$

Vi har nu vist at cosinus funktionen er en contractive mapping funktion på intervallet [0,1], hvilket betyder at der findes et unikt punkt i [0,1] som kan estimeres vha. en lang række iterationer, som gjort i den tidligere del af opgaven.

# Opgave 3.5.1

Brug Horner's algoritme til at find p(4) hvor

> $p(z) = 3z^5 - 7z^4 - 5z^3 + z^2 - 8z + 2$

```
> restart;
> with(ArrayTools) :
> Horner :=proc(a, z)
   local b, k, nums;
   nums := Size(a, 2);
   b := a;
   for k from 1 to nums - 1do
   b[k + 1] := a[k + 1] + z·b[k];
   end do;
   return b;
   end proc;
```

```
Horner := proc(a, z)                                                    (5.1)
    local b, k, nums;
    nums := ArrayTools:-Size(a, 2);
    b := a;
    for k to nums - 1 do b[k + 1] := a[k + 1] + z * b[k] end do;
    return b
end proc
```

> $f := [3, -7, -5, 1, -8, 2]$

$$f := [3, -7, -5, 1, -8, 2] \quad (5.2)$$

> $Horner(f, 4)$

$$[3, 5, 15, 61, 236, 946] \quad (5.3)$$

Heraf kan vi se at p(4)=946.

# Opgave 3.6.1

Vi skal løse følgende ligningssystem med homotopy metoden:

> $x-2y+y^2+y^3-4=-x-y+2y^2-1=0$

> $restart;$

> $f(a)=Matrix\left(2,1,\left[x-2y+y^2+y^3-4,-x-y+2y^2\right]\right);\ a=(x,y)\in\mathbb{R}^2:$

$$f(a)=\begin{bmatrix} x-2y+y^2+y^3-4 \\ -x-y+2y^2 \end{bmatrix} \tag{6.1}$$

Nu differentierer vi på den specielle måde som man gør med homotopy metoden.

> $h_a := Matrix\left(2,2,\left[\left[1,-2+2y+3y^2\right],\left[-1,-1+4y\right]\right]\right)$

$$h_a := \begin{bmatrix} 1 & -2+2y+3y^2 \\ -1 & -1+4y \end{bmatrix} \tag{6.2}$$

Herfra kan vi let finde h[t]:

> $h_t := Matrix(2,1,[-4,-1]);$

$$h_t := \begin{bmatrix} -4 \\ -1 \end{bmatrix} \tag{6.3}$$

Nu kan vi skrive det op med t inkluderet:

> $h(t,a)=Matrix\left(2,1,\left[x-2y+y^2+y^3-4-(-4)-4t,-x-y+2y2-1-(-1)-t\right]\right)$

$$h(t,a)=\begin{bmatrix} x-2y+y^2+y^3-4t \\ -x-y+2y2-t \end{bmatrix} \tag{6.4}$$

Nu kan differential ligningerne skrives op med determinanter som

> $\begin{bmatrix} -4 & 1 & -2+2y+3y^2 \\ -1 & -1 & -1+4y \end{bmatrix}\begin{bmatrix} t' \\ x' \\ y' \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix}:$

Nu kan vi omskrive disse differential ligninger med $\eta'_j=(-1)^{j+1}det\left(A_j\right)$ :

> $\begin{cases} t'=3y^2+6y-3 & t(0)=0 \\ x'=-3y^2+14y-2 & x(0)=0 \\ y'=5 & y(0)=0 \end{cases}:$

> $y(s) := 5\,s:$

> $t'(s)=3(5\,s)^2+6(5\,s)-3=75\,s^2+30\,s-3:$

> $t(s) := 25\,s^3+15\,s^2-3\,s:$

> $x'(s)=-3\,(5\,s)^2+14(5\,s)-2=-75\,s^2+70\,s-2:$

> $x(s) := -25\,s^3+35\,s^2-2\,s:$

Nu kan vi sætte t(s)=1 og få en værdi for s:

> $s_1,s_2,s_3 := fsolve\left(1=25\,s^3+15\,s^2-3\,s,s\right)$

$$s_1,s_2,s_3 := -0.6898979486,\ -0.2000000000,\ 0.2898979486 \tag{6.5}$$

> $x1 := evalf\left(x\left(s_1\right),10\right);$

$$x1 := 26.24744872 \tag{6.6}$$

> $x2 := evalf\left(x\left(s_2\right), 10\right);$

$$x2 := 2.000000000 \tag{6.7}$$

> $x3 := evalf\left(x\left(s_3\right), 10\right);$

$$x3 := 1.752551287 \tag{6.8}$$

> $y1 := evalf\left(y\left(s_1\right), 10\right);$

$$y1 := -3.449489743 \tag{6.9}$$

> $y2 := evalf\left(y\left(s_2\right), 10\right);$

$$y2 := -1.000000000 \tag{6.10}$$

> $y3 := evalf\left(y\left(s_3\right), 10\right);$

$$y3 := 1.449489743 \tag{6.11}$$

> $t1 := evalf\left(t\left(s_1\right), 10\right);$

$$t1 := 1.000000000 \tag{6.12}$$

> $t2 := evalf\left(t\left(s_2\right), 10\right);$

$$t2 := 1.000000000 \tag{6.13}$$

> $t3 := evalf\left(t\left(s_3\right), 10\right);$

$$t3 := 1.000000000 \tag{6.14}$$

>