# Lokální extrémy

November 19, 2019

## 1 Teorie úloh bez omezení

### 1.1 Úlohy bez omezení-lokální extrémy

Uvažujme funkci $f : \mathbb{R}^n \to \mathbb{R}$. Je-li $k \in \{0, 1, 2, \ldots\}$, pak má-li funkce $f$ v bodě $x_0 \in \mathbb{R}^n$ derivace až do řádu $k$ včetně, píšeme $f \in D^k(x_0)$.

**Definice.** Řekneme, že funkce $f$ nabývá v bodě $x_0 \in \mathbb{R}^n$ **lokální minimum (maximum)** a píšeme

$$x_0 \in lokmin(f)\ (lokmax(f)),$$

existuje-li $\delta > 0$ tak, že $\forall x \in B(x_0; \delta) := \{x \in \mathbb{R}^n \mid \|x - x_0\| < \delta\}$ platí:

$$f(x_0) \leq f(x), \text{ resp. } f(x_0) \geq f(x).$$

Nabývá-li funkce $f$ v bodě $x_0$ lokální minimum resp. maximum, píšeme také: $x_0 \in lokextr(f)$. Dále píšeme $x_0 \in lokmin(f)$ resp. $x_0 \in lokmax(f)$ pokud funkce $f$ nabývá v bodě $x_0$ lok. minimum resp. maximum.

attachment:image.png

image.png

Derivace se počítá podle vzorce:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$

attachment:image.png

image.png

## 1.2 Funkce jedné proměnné

**Věta** (P.Fermat). Nechť $f : \mathbb{R} \to \mathbb{R}$ je funkce jedné reálné proměnné. Jestliže $x_0 \in locextr(f)$ a $f \in D^1(x_0)$, pak

$$f'(x_0) = 0.$$

Takovýto bod se bude nazývat **stacionární bod.**

**Důkaz** Sporem předpokládejme, že $f'(x_0) \neq 0$. Nechť například $f'(x_0) > 0$. Jelikož $f \in D^1(x_0)$, platí

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \alpha(h),$$

kde

$$\frac{\alpha(h)}{h} \to \text{ pro } h \to 0.$$

Pak $f(x_0 + h) - f(x_0) = (f'(x_0) + \frac{\alpha(h)}{h})h$. Pro dostatečně malé $h$ je

$$f'(x_0) + \frac{\alpha(h)}{h} > 0.$$

Tudíž pak $f(x_0 + h) - f(x_0) > 0$ a $f(x_0 - h) - f(x_0) < 0$, což je ve sporu s předpokladem, že $x_0 \in locextr(f)$. □

**Věta** Nechť $f \in D^2(x_0)$.

a) (nutná podmínka) Jestliže $x_0 \in locextr(f)$, pak

$$f'(x_0) = 0 \text{ a } f''(x_0) \geq 0 \text{ ( pokud jde o minimum )} a$$

$$f'(x_0) = 0 \text{ a } f''(x_0) \leq 0 \text{ ( pokud jde o maximum )}$$

b) (postačující podmínka) Jestliže $f'(x_0) = 0$ a $f''(x_0) > 0$, pak $x_0 \in locmin(f)$.

Jestliže $f'(x_0) = 0$ a $f''(x_0) < 0$, pak $x_0 \in locmax(f)$.

**Důkaz.** Taylorova formule má pak tvar:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \alpha(h),$$

kde $\frac{\alpha(h)}{h^2} \to 0$ pro $h \to 0$. Z Fermatovy věty pak plyne $f'(x_0) = 0 \implies$

$$\begin{aligned} 0 \leq f(x_0 + h) - f(x_0) &= \frac{1}{2}f''(x_0)h^2 + \alpha(h) \\ &= \left(\frac{1}{2}f''(x_0) + \frac{\alpha(h)}{h^2}\right)h^2. \end{aligned}$$

Odtud plyne, že

$$0 \leq \frac{1}{2}f''(x_0) + \frac{\alpha(h)}{h^2} \implies 0 \leq f''(x_0).$$

Postačující podmínka. Předpokládejme, že $f'(x_0) = 0$ $f''(x_0) > .$ Nechť $0 < \varepsilon \leq \frac{1}{2}f''(x_0)$. Pak existuje $\delta > 0$ tak, že pro každé $|h| < \delta$ platí:

$$f(x_0 + h) - f(x_0) = \frac{1}{2}f''(x_0)h^2 + \alpha(h)$$

a

$$|\alpha(h)| \leq \varepsilon h^2 \implies \alpha(h) \geq -\varepsilon h^2.$$

Pak pro $|h| < \delta$ je

$$f(x_0+h) - f(x_0) = \frac{1}{2}f''(x_0)h^2 + \alpha(h) \geq (\frac{1}{2}f''(x_0) - \varepsilon)h^2 \geq 0.$$

Odtud plyne $x_0 \in locmin(f)$. □

**Věta.** Nechť $n \in \mathbb{N}$, $n \geq 2$, $f \in D^n(x_0)$. Jestliže $x_0 \in locmin(f)$, resp. $x_0 \in locmax(f)$, pak

$$f'(x_0) = \ldots = f^{(n)}(x_0) = 0$$

nebo

$$f'(x_0) = \ldots = f^{(2m-1)}(x_0) = 0,\ f^{(2m)}(x_0) > 0 \text{ resp. } f^{(2m)}(x_0) < 0$$

pro jisté $m : 2 \leq 2m \leq n$. Je-li poslední podmínka splněna, potom $x_0 \in locmin(max)(f)$.

## 1.3 Funkce více proměnných

**Věta.** Je-li $x_0 = (x_1^0, \ldots, x_n^0) \in locextr(f)$, $f \in D^1(x_0)$, pak

$$\nabla f(x_0) = 0 \iff \frac{\partial f}{\partial x_1}(x_0) = \ldots = \frac{\partial f}{\partial x_n}(x_0) = 0.$$

**Důkaz.** Položme

$$\varphi(x_i) = f(x_1^0, \ldots, x_i, x_{i+1}^0, \ldots, x_n^0).$$

Jestliže $x_i^0 \in locextr(\varphi) \implies \varphi'(x_i^0) = 0$. Dále platí $\varphi'(x_i^0) = 0 \iff \frac{\partial f}{\partial x_i}(x_0) = 0$.

Je-li $f \in D^2(x_0)$, položme

$$A = f''(x_0) = \left(\frac{\partial f}{\partial x_i \partial x_j}(x_0)\right)_{i,j=1}^n = (a_{ij})_{i,j=1}^n.$$

**Definice.** Matice $A = (a_{ij})_{i,j=1}^n$ se nazývá **pozitivně semidefinitní** $(A \geq 0)$, jestliže

$$\langle A \cdot h, h\rangle \geq 0 \ \ \forall h \in \mathbb{R}^n \iff$$

$$\sum_{i,j=1}^n a_{ij}h_ih_j \geq 0 \ \ \forall h = (h_1, \ldots, h_n) \in \mathbb{R}^n.$$

Matice $A = (a_{ij})_{i,j}^n$ se nazývá **pozitivně definitní** $(A > 0)$, jestliže

$$\langle A \cdot h, h\rangle > 0 \ \ \forall h \in \mathbb{R}^n, h \neq 0.$$

Matice $A$ se nazývá **striktně pozitivně definitní**, existuje-li konstanta $\alpha > 0$ tak, že

$$\langle A \cdot h, h\rangle \geq \alpha\|h\|^2, \ \ \forall h \in \mathbb{R}^n.$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 5 \\ 11 \end{pmatrix}.$$

$$\begin{pmatrix} 5 \\ 11 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 5 \cdot 1 + 11 \cdot 2 = 27.$$

**Poznámka.** Platí následující ekvivalence:

$$\langle A \cdot h, h \rangle \geq 0 \ \ \forall h \in \mathbb{R}^n, \ h \neq 0 \iff$$

$$\iff \exists \alpha > 0 : \langle A \cdot h, h \rangle \geq \alpha \|h\|^2, \ \ \forall h \in \mathbb{R}^n.$$

Implikace " $\Leftarrow$ " je zřejmá. Ovšem opačná implikace plyne z věty o nabývání minima spojité funkce na kompaktní množině. Viz Věta **??**.

**Věta.** Nechť $f \in D^2(x_0)$, $x_0 \in \mathbb{R}^n$. \ a) Jestliže $x_0 \in locmin(max)(f)$, potom

$$f'(x_0) = 0, \ \ \langle f''(x_0)h, h \rangle \geq 0, \ \forall h \in \mathbb{R}^n,$$

resp.

$$\langle f''(x_0)h, h \rangle \leq 0, \ \forall h \in \mathbb{R}^n.$$

b) Jestliže $f'(x_0) = 0$, $\langle f''(x_0)h, h \rangle > 0$ resp. $\langle f''(x_0)h, h \rangle < 0$, pro všechna $h \in \mathbb{R}^n$, $h \neq 0$. Pak $x_0 \in locmin(max)(f)$.

**Věta.** (Weierstrassova věta). Nechť $S \subset \mathbb{R}^n$ je omezenou a uzavřenou podmnožinou. Dále nechť $f : S \to \mathbb{R}$ je spojitou funkcí na množině $S$. Pak existují body $x_1, x_2 \in S$ takové, že $x_1 \in absmin_S(f)$ a $x_2 \in absmax_S(f)$. □

Označení $x_1 \in absmin_S(f)$ resp. $x_2 \in absmax_S(f)$ znamená, že funkce $f$ nabývá v bodě $x_1$ resp. $x_2$ své absolutní minimum resp. maximum na množině $S$. Tedy

$$f(x_1) \leq f(x), \ \forall x \in S$$

a

$$f(x_2) \geq f(x), \ \forall x \in S.$$

**Věta.** Mějme dánu spojitou funkci $f : \mathbb{R}^n \to \mathbb{R}$.

a) Jestliže $\lim_{\|x\| \to \infty} f(x) = +\infty$, potom pro libovolnou uzavřenou množinu $S \subset \mathbb{R}^n$ existuje bod $x_0 \in absmin_S(f)$.

b) Jestliže $\lim_{\|x\| \to \infty} f(x) = -\infty$, potom pro libovolnou uzavřenou množinu $S \subset \mathbb{R}^n$ existuje $x_0' \in absmax_S(f)$. □

**Věta.** Uvažujme symetrickou čtvercovou matici:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \in Mat(n \times n).$$

a) Matice $A$ je pozitivně definitní, právě když

$$a_{11} > 0, \ det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} > 0, \dots, det(A) > 0.$$

b) Matice $A$ je negativně definitní, právě když

$$a_{11} < 0,\ det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} > 0,\ det\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} < 0, \ldots,$$

$$(-1)^n det(A) > 0.$$

□

Tzv. hlavním minorem $A_{i_1,\ldots,i_k}$ matice $A$ budeme rozumět determinant

$$A_{i_1,\ldots,i_k} = det\begin{pmatrix} a_{i_1 i_1} & \ldots & a_{i_1 i_k} \\ \vdots & \ddots & \vdots \\ a_{i_k i_1} & \ldots & a_{i_k i_k} \end{pmatrix},$$

kde $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n$.

**Věta.** Uvažujme symetrickou čtvercovou matici $A \in Mat(n \times n)$.

a) Matice $A$ je pozitivně semidefinitní $(A \geq 0)$, právě když všechny její hlavní minory jsou nezáporná čísla, tj.

$$A_{i_1,\ldots,i_k} \geq 0,$$

kdykoliv je $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n, k = 1, \ldots, n$.

b) Matice $A$ je negativně semidefinitivní $(A \leq 0)$, právě když

$$(-1)^k A_{i_1,\ldots,i_k} \geq 0,$$

kdykoliv je $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n, k = 1, 2, \ldots, n$.

## 1.4 Schéma řešení úlohy bez omezení

1) Sestavíme soustavu rovnic:

$$f'(x) = 0 \iff \left\{ \begin{array}{rcl} \frac{\partial f(x)}{\partial x_1} & = & 0 \\ & \vdots & \\ \frac{\partial f(x)}{\partial x_n} & = & 0 \end{array} \right\}$$

tvořící nutnou podmínku 1. řádu (tzv. podmínky stacionarity).

2) Prověříme splnění podmínek 2. řádu v tzv. stacionárních bodech. Sestavíme matice druhých derivací (tzv. Hessovy matice):

$$A = f''(x) = \left( \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{i,j=1}^{n} = (a_{ij})_{i,j=1}^{n}.$$

3) Prověříme splnění postačujících podmínek: spočítáme posloupnost hlavních minorů

$$A_{1\ldots k} = det(a_{ij})_{i,j=1}^{k},\ k = 1, \ldots, n.$$

Pokud $A_{1\ldots k} > 0$ pro $k = 1, 2, \ldots, n$, pak je bod $x_0$ bodem lokálního minima funkce $f$, tj. $x_0 \in locmin(f)$. Jestliže hlavní minory střídají znaménko, pak $x_0 \in locmax(f)$.

4) Pokud ve stacionárním bodě $x_0$ není splněna postačující podmínka, potom se prověří splnění nutné podmínky 2. řádu v bodě $x_0$.

### 1.4.1 Příklad

Vyšetřete, zdali funkce $f(x,y) = x \cdot y$ nabývá v bodě $(0,0)$ lok. extrém.

[36]:
```
from sympy import *
x, y = symbols('x y')
f = x * y
gradf = derive_by_array(f, [x, y])
display(gradf)
```

$$\begin{bmatrix} y & x \end{bmatrix}$$

[37]:
```
stacBody = solve(gradf)
stacBody[x], stacBody[y]
```

[37]:
```
(0, 0)
```

Funkce má jediný stac bod (S = (0,0).) Najděme Hessovu matici (f''(S)) funkce (f) v bodě (S_1).

[38]:
```
hess_f = derive_by_array(gradf, [x, y])

display(hess_f)
```

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

[39]:
```
h1, h2 = symbols('h1 h2')
A = hess_f
h = Matrix([[h1], [h2]])
h
```

[39]:
$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

[40]:
```
A = Matrix([[0, 1], [1, 0]])
h.T*A*H
```

[40]:
$$\begin{bmatrix} 2h_1h_2 \end{bmatrix}$$

Tedy v tomto případě je ($\langle A \cdot h, h \rangle = 2h\_1h\_2.$)
Tato kvadratická je zřejmě indefinitní a tudíž funkce (f(x,y)) nenabývá v bodě S lokální extrém.

[ ]:
```

```

## 2 Testování definitnosti kvadr. forem

[1]:
```
######### Zde je hlavička ################
import sympy as sym
from IPython.display import Math, display, Latex
import numpy as np
sym.init_printing()
```

## 2.1 Vytvoření objektu typu matice

[2]:
```
# Ukažme, jak lze v kniovně SymPy vytvořit "objekt" matice:
from sympy import *
A = Matrix([[1, 2, 3], [5, 7, 8]])
# pretty_print(A)
A
display(Math("\\text{ hodnost matice: }\
            h(A) = %s" %A.rank()))
             # metoda rank() vrací tzv. hodnost matice
```

hodnost matice: $h(A) = 2$

### 2.1.1 Matice se transponuje pomocí metody T

[3]:
```
from sympy import *
init_printing()
A = Matrix([[1, 2, 3], [5, 7, 8]])
# pretty_print(A)
display(A)
```

$$\begin{bmatrix}1 & 2 & 3\\5 & 7 & 8\end{bmatrix}$$

[4]:
```
from sympy import *
init_printing()
A = Matrix([[1, 2, 3], [5, 7, 8]])
display(A)
display(Math(" A^T = %s" %sym.latex(A.T)))
```

$$\begin{bmatrix}1 & 2 & 3\\5 & 7 & 8\end{bmatrix}$$

$$A^T = \begin{bmatrix}1 & 5\\2 & 7\\3 & 8\end{bmatrix}$$

[5]:
```
M = Matrix([[1, 2], [-5, 0]])
N = Matrix([[0, 7], [0, 2]])
# pretty_print(M**2); pretty_print(M * N)
display(Math("%s + %s = %s"\
             %(sym.latex(M), sym.latex(N), sym.latex(M + N))))
```

$$\begin{bmatrix}1 & 2\\-5 & 0\end{bmatrix} + \begin{bmatrix}0 & 7\\0 & 2\end{bmatrix} = \begin{bmatrix}1 & 9\\-5 & 2\end{bmatrix}$$

[9]:
```
# Násobení matic
display(Math("M * N = %s" %sym.latex(M*N)))
```

$$M * N = \begin{bmatrix}0 & 11\\0 & -35\end{bmatrix}$$

[41]:
```
A = Matrix([[1, 2], [3, 4]])
h = Matrix([[1, 2]])
```

```
h_T = Matrix([[1, 2]]).T
display(Math("A = %s,\ \ \ h = %s" %(sym.latex(A), sym.latex(h_T))))
print("************************************")
display(Math("\\text{Najděte hodnotu: } h^TAh = ?"))
pokracovat = input("Pro pokračování stiskneme enter ")
display(Math("\\langle A\\cdot h,h\\rangle = \
            h^T*A*h = \\langle Ah,h\\rangle =\
              %s " %sym.latex((h * A * h_T)[0,0])))
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad h = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

**********

Najděte hodnotu: $h^T Ah =?$

```
Pro pokračování stiskneme enter
```

$\langle A \cdot h, h \rangle = h^T * A * h = \langle Ah, h \rangle = 27$

[38]:
```
h1, h2 = sym.symbols('h1 h2')
A = Matrix([[1, 2], [3, 4]])
h = Matrix([[h1, h2]])
h_T = Matrix([[h1, h2]]).T
display(Math("A = %s,\ \ \ h^T = %s" %(sym.latex(A), sym.latex(h_T))))
print("************************************")
display(Math("\\text{Najděte hodnotu: } hAh^T = ?"))
pokracovat = input("Pro pokračování stiskneme enter ")
vysl = (h * A * h_T)[0,0]
display(Math("\\langle A\\cdot h,h = \\rangle = h*A*h^T = %s = %s "\
             %(sym.latex(vysl), sym.latex(vysl.expand()))))
```

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad h^T = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

**********

Najděte hodnotu: $hAh^T =?$

```
Pro pokračování stiskneme enter
```

$\langle A \cdot h, h = \rangle = h * A * h^T = h_1 (h_1 + 3h_2) + h_2 (2h_1 + 4h_2) = h_1^2 + 5h_1h_2 + 4h_2^2$

Předchozí matice je evidentně indefinitní maticí neboť např. volbou vektorů $h^T = (1,1)$ resp. $h^T = (-2,1)$ má součin $h * A * h^T$ různé znaménko.

[32]:
```
display(Math("h*A*h^T = %s" %sym.latex(vysl.subs({h1: -2, h2: 1}))))
```

$h * A * h^T = -2$

[26]:
```
def is_positive(A):
    if A[0,0] > 0 and det(A) > 0:
        return True
    else:
```

```
        return False
is_positive(A)
```

[26]: False

[80]:
```
# Vymazání řádku či sloupce z matice
C = Matrix([[1, 8, -5], [2, 0, 3], [1, 1 , 6]])
D = Matrix([[1, 8, -5], [2, 0, 3], [1, 1 , 6]])
C.row_del(2)
C.col_del(2)
C
```

[80]:
$$\begin{bmatrix} 1 & 8 \\ 2 & 0 \end{bmatrix}$$

[64]:
```
# Matici převedeme do redukované podoby pomocí metody rref()
A.rref()
```

[64]:
$$\left(\begin{bmatrix} 1 & 0 & -\frac{5}{3} \\ 0 & 1 & \frac{7}{3} \end{bmatrix}, \quad (0, \quad 1)\right)$$

[2]:
```
# Ukažme si metodu det() použitá na matici
from sympy import *
B = Matrix([[1, 0, 2], [4, 1, 0], [1, 1, 0]])
B.det()
det(B)
```

[2]: 6

### 2.1.2 Testování definitnosti pro matice 2x2

[4]:
```
# Testování pozitivní definitnosti matice typu (2,2)
# Zadejme matici
A = Matrix([[2,1], [0, 1]])
print('A = ', A)
print(A[0,0],',', A.det())
if A[0,0] > 0 and A.det() > 0:              # Rozhodovací blok
    print('Matice A je pozitivně definitní.')
else:
    print('Matice není pozitivně definitní.')
```

```
A =  Matrix([[2, 1], [0, 1]])
2 , 2
Matice A je pozitivně definitní.
```

[72]:
```
# Testování negativní definitnosti matice typu (2,2)
# Zadejme matici
A = Matrix([[2,1], [0, 1]])
```

```
print('A = ', A)
if A[0,0] < 0 and A.det() > 0:
    print('Matice A je negativně definitní.')
else:
    print('Matice není negativně definitní.')
```

```
A =  Matrix([[2, 1], [0, 1]])
Matice není negativně definitní.
```

[74]:
```
# Vytvořme program, který provede obojí test:
A = Matrix([[2,1], [0, 1]])
print('A = ', A)
if A[0,0] > 0 and A.det() > 0:
    print('Matice A je pozitivně definitní.')
elif A[0,0] < 0 and A.det() > 0:
    print('Matice A je negativně definitní.')
else:
    print('Matice A není ani pozitivně ani negativně definitní.')
```

```
A =  Matrix([[2, 1], [0, 1]])
Matice A je pozitivně definitní.
```

### 2.2 Cvičení

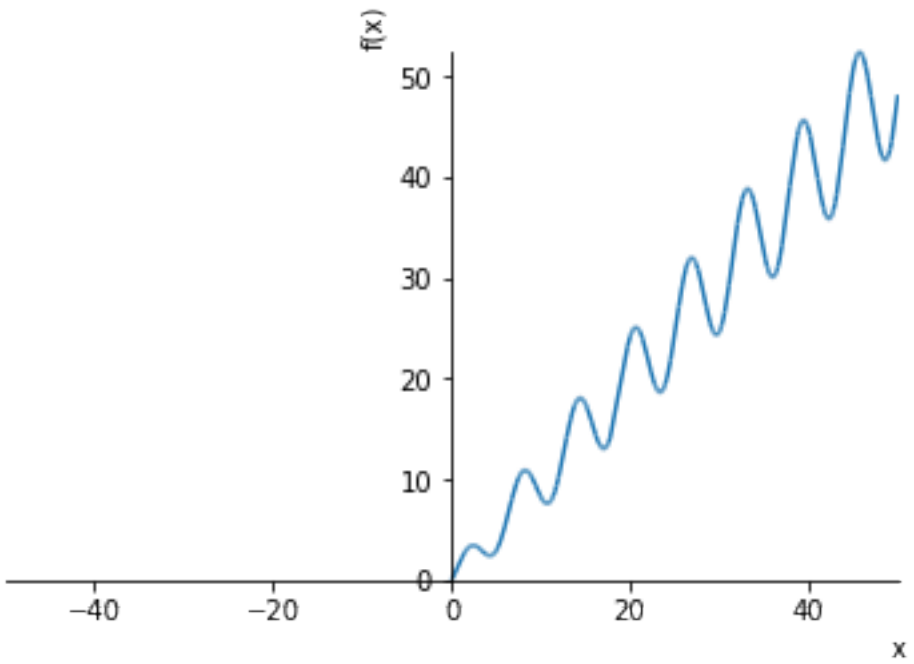
Napište program, který má zjistit, je-li daná matice $A$ typu $2 \times 2$ ivertibilní.

[ ]:
```
# Řešení:
A = Matrix([[1, 2], [3,4]])
if A.det() != 0:
    print('Matice A je regulární.')
else:
    print('Matice A je singulární.')
```

### 2.3 Grafy funkcí

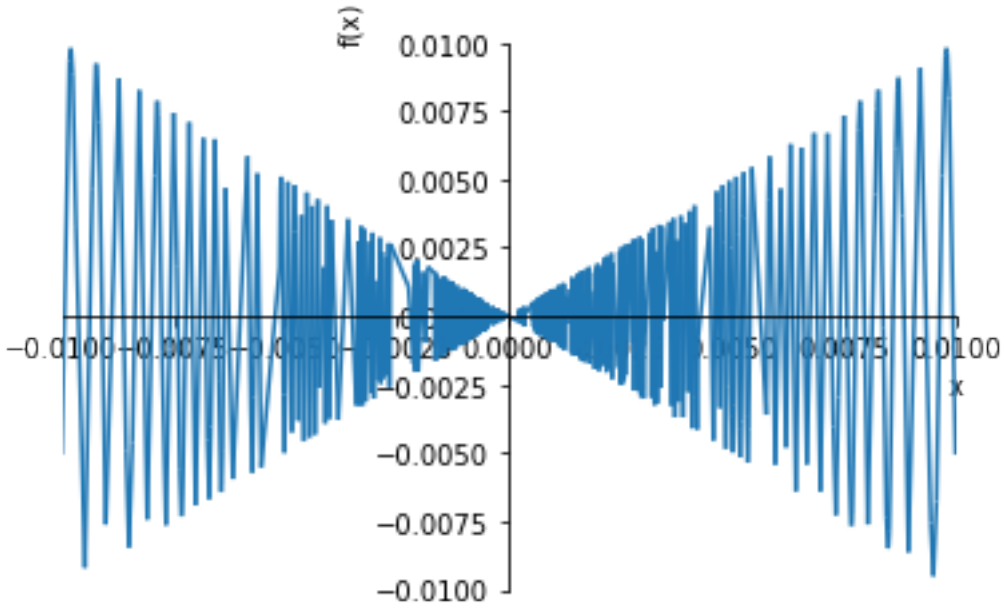
Uvažujme funkci $F(\lambda) = \lambda^2$.

[110]:
```
# volání modulu pro symbolické výpočty a použití funkce plot pro kreslení grafu␣
 ↪funkce jedné proměnné:
from sympy import *
x, y, z, t = symbols('x y z t')  # zde řekneme programu, že budeme používat␣
 ↪symboly pro proměnné x,y,z,t
plot(x+sqrt(x)*sin(x), (x, -50, 50))   # zde si voláme funkci plot(), rozsah␣
 ↪proměnné x je z intervalu <-50,50>.
```



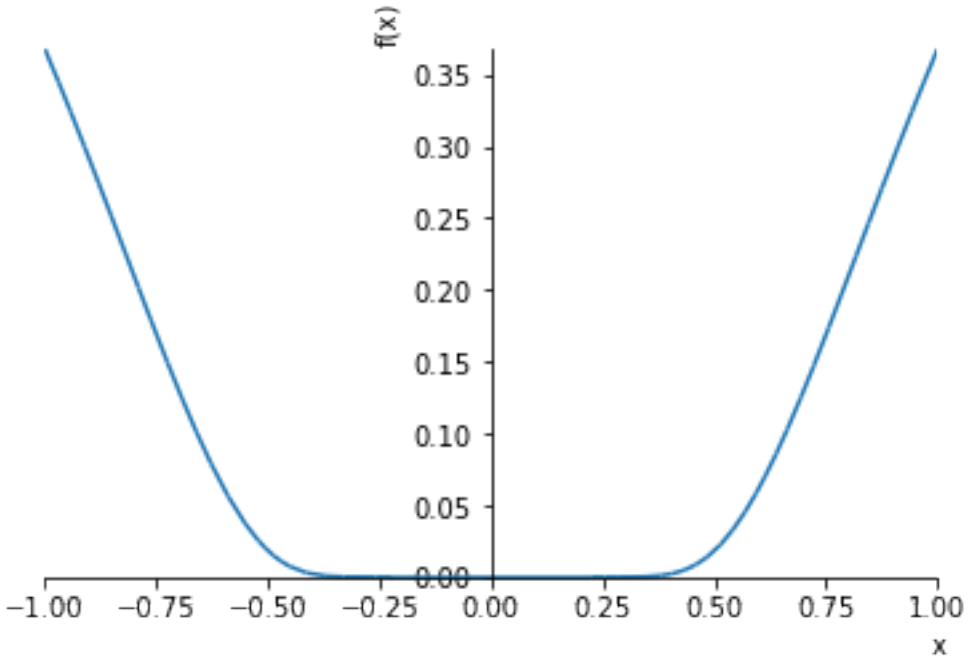

[110]: <sympy.plotting.plot.Plot at 0x27cfac06630>

[112]:
```
plot(x*sin(1/x),(x,-0.01,0.01))
```



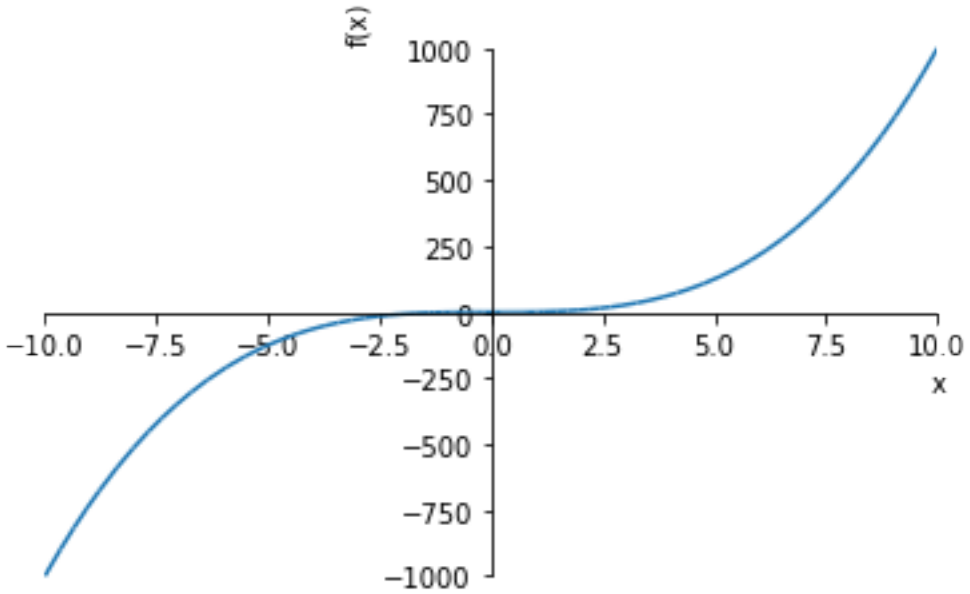

[112]: <sympy.plotting.plot.Plot at 0x27cfafc8828>

[113]:
```
plot(exp(-1/x**2), (x,-1,1))
```



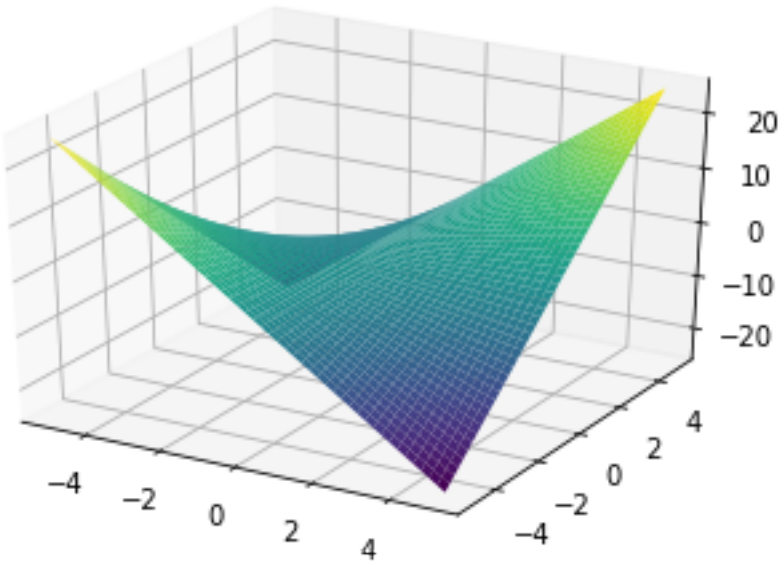

[113]:
```
<sympy.plotting.plot.Plot at 0x27cfafe96d8>
```

[114]:
```
plot(x**3,(x,-10,10))
```



[114]: <sympy.plotting.plot.Plot at 0x27cfafe9748>

[115]:
```
from sympy.plotting import plot3d
plot3d(x*y, (x, -5, 5), (y, -5, 5))
```




[115]: <sympy.plotting.plot.Plot at 0x27cfb139e10>

### 2.4 Derivování funkcí

[5]:
```
# Zde použijeme metodu diff() s parametrem x na objekt funkce sin(x**2)
# resp. použijeme funkci diff na symbolický výpočet derivace funkce cos(x)
from sympy import *
x,y = symbols('x, y')
print(sin(x**2).diff(x),',', diff(cos(x), x))  # použijeme příkaz print(<>,<>,<>)
```

```
2*x*cos(x**2) , -sin(x)
```

## 3 Řešení rovnic a jejich soustav

[43]:
```
from sympy import *
solveset(x**3-1, x)
```

[43]: {1, -1/2 - sqrt(3)*I/2, -1/2 + sqrt(3)*I/2}

[108]:
```
solveset(x**3-1, x, domain = S.Reals)
```

[108]:

$$\{1\}$$

[116]:
```
# Řešení sousta nelineárních rovnic:
f = x*y**3 + (x + y)**2
gradf = derive_by_array(f, [x,y])
gradf
```

[116]:

$$\begin{bmatrix} 2x + y^3 + 2y & 3xy^2 + 2x + 2y \end{bmatrix}$$

[134]:
```
soln = list(nonlinsolve(gradf, [x, y]))
soln
```

[134]:

$$\left[ (0, \quad 0), \quad \left( -\frac{2i}{9}\sqrt{6}, \quad -\frac{2i}{3}\sqrt{6} \right), \quad \left( \frac{2i}{9}\sqrt{6}, \quad \frac{2i}{3}\sqrt{6} \right) \right]$$

[133]:
```
list(soln[1])
```

[133]:

$$\left[ -\frac{2i}{9}\sqrt{6}, \quad -\frac{2i}{3}\sqrt{6} \right]$$

[139]:
```
for s in soln:
    for ss in list(s):
        print(ss)
       # print(check_assumptions(ss, Real = True))
```

```
0
0
-2*sqrt(6)*I/9
-2*sqrt(6)*I/3
2*sqrt(6)*I/9
2*sqrt(6)*I/3
```

# 4 Příklady - funkce jedné proměnné

## 4.1 Funkce jedné proměnné

[6]:
```
# Užití první a druhé derivace.
# Uvažujme funkci jedné proměnné f(x). Najděme lokální extrémy této funkce.
from sympy import *
x = symbols('x')   # deklarace symbolu 'x'
f = S(input('Zadejte předpis funkce: y=')) # zde vložíme předpis
df = diff(f, x)    # diff je funkce,která derivuje výraz 'f' podle proměnné 'x'
solns = list(solveset(df, domain = S.Reals))   # řešíme rovnici df = 0, funkce␣
 ↪'list(par)' konvertuje výstup funkce solveset na
                                               # typ 'seznam'
delka_solns = max(solns) - min(solns) + 0.01
print('Tady je seznam stacionárních bodů: ', solns)  # vypíše se seznam kořenů␣
 ↪[koren1, koren2, ...]
```

```
d2f = diff(df, x)                                    # derivujeme opět␣
 ↪symbolicky podle 'x' výraz 'df'
# Nyní budeme muset iterovat přes všechny stac. body funkce:
for koren in solns:        # cyklus 'for <podmínka>:'
    d2f_solns = d2f.subs({x:koren})  # proměnné d2f_solns se přiřadí hodnota␣
 ↪(číslo): d2f.subs({x:koren})
    if d2f_solns > 0:                 # rozhodovací blok 'if <podmínka>:'
        print('Funkce f nabývá v bodě ', koren,)
        print('své lokální minimum a jeho hodnota je rovna ', f.subs({x:koren}))
    elif d2f_solns < 0:   # čteme: "jinak jestliže..."
        print('Funkce f nabývá v bodě ', koren,)
        print('své lokální maximum a jeho hodnota je rovna ', f.subs({x:koren}))
    else:
        print('Touto metodou nelze určit, zdali funkce f nabývá v bodě', koren)
        print('lokální extrém')
print('Strana a = ', 2.5, 'a strana b = ', (1/2)*10 - 2.5)
plot(f, (x, min(solns) - delka_solns, max(solns) + delka_solns))
# Problémy mohou nastat, pokud budeme mít funkci, jejíž derivace například
# bude mít nekonečně mnoho stacionárních bodů. Tedy třeba funci f(x)=sin(x).
# Dále vylepšete předchozí program tak, aby využíval derivací řádu k vyššího než␣
 ↪2.
```

```
Zadejte předpis funkce: y=x
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-6-303306e7a5eb> in <module>()
          7 solns = list(solveset(df, domain = S.Reals))   # řešíme rovnici df =␣
 ↪0, funkce 'list(par)' konvertuje výstup funkce solveset na
          8                                                 # typ 'seznam'
    ----> 9 delka_solns = max(solns) - min(solns) + 0.01
         10 print('Tady je seznam stacionárních bodů: ', solns)  # vypíše se␣
 ↪seznam kořenů [koren1, koren2, ...]
         11 d2f = diff(df, x)                                  # derivujeme opět␣
 ↪symbolicky podle 'x' výraz 'df'


        ValueError: max() arg is an empty sequence
```

## 4.2 Úloha o maximálním obsahu obdélníka.

$a, b$ označují délky stran. Obsah obdelnika je roven

$$S = ab$$

.

Chceme nalézt mezi všemi obdélníky o daném obvodu $o$ ten, který má největší obsah.

$$o = 2(a+b).$$

Odtud lze vyjádřit proměnnou $b$ jako funkci proměnné $a$.

$$b = (1/2)o - a.$$

Potom platí:

$$S = ab = a[(1/2)o - a].$$

Zde se předpokládá, že $a, b > 0$. Zvolíme-li například $o = 10$, pak dostaneme, že

$$S = a[(1/2)\cdot 10 - a] = a(5-a),\ a \in (0,5).$$

Připomeňme si tzv. Weierstrassovu větu:

**Věta.** Nechť $S \subset \mathbb{R}^n$ je omezenou a uzavřenou podmnožinou. Dále nechť $f : S \to \mathbb{R}$ je spojitou funkcí na množině $S$. Pak existují body $x_1, x_2 \in S$ takové, že $x_1 \in absmin_S(f)$ a $x_2 \in absmax_S(f)$. □

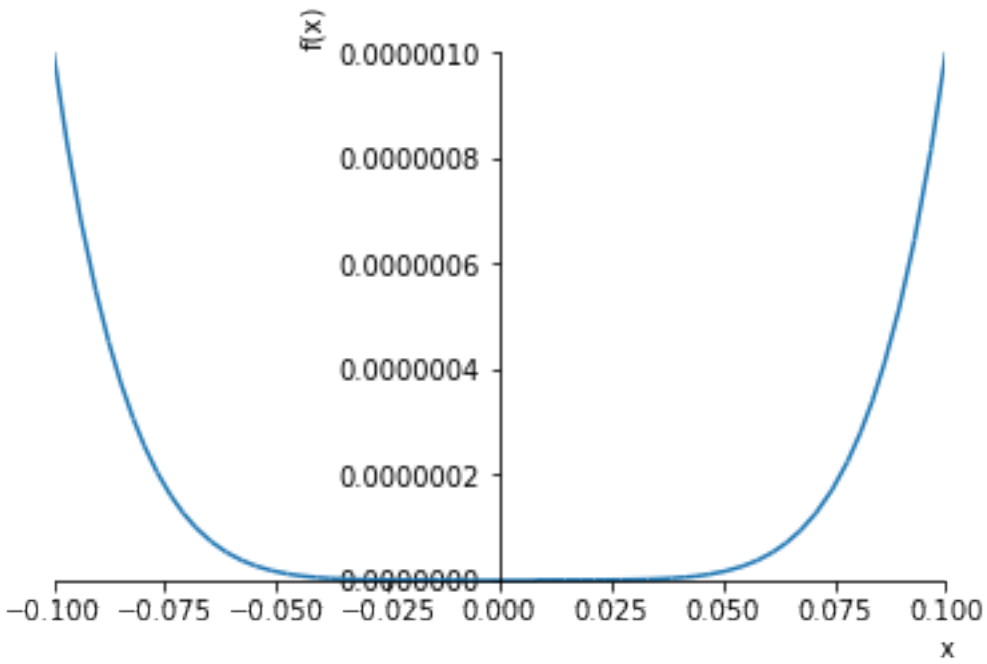
Například $S = [0,5]$ což je omezený a uzavřený interval v $\mathbb{R}$. Tedy spojitá funkce $f(x) = x(5-x)$ musí na tomto intervalu nabývat absolutní minimum a maximum. Je zřejmé, že $absmin(f)_S = 0$ a nabývá se na koncích intervalu $S$. Tedy absolutní maximum této funkce se musí nabývat uvnitř tohoto intervalu.

### 4.3 Užití derivací vyššího řádu v kódu.

[2]:
```
# Vylepšení předchozího programu, využívající derivace vyššího
# řádu než dva.
from sympy import *
x = symbols('x')
def main():
    f = S(input('Zadejte předpis funkce: y='))  # zadání předpisu funkce f
    solns = najdi_stac_body(f)
    delka_solns = max(solns) - min(solns) + 0.1
    print('Tady je seznam stacionárních bodů: ', solns)
    for stac_bod in solns:
        m = stupen_der_nonzero(f, stac_bod)
        if je_sude(m):
            if d(f, m).subs({x:stac_bod}) > 0:
                print('Funkce f nabývá v bodě ', stac_bod,)
                print('své lokální minimum a jeho hodnota je rovna ', f.subs({x:
 ↪stac_bod}))
            else:
                print('Funkce f nabývá v bodě ', stac_bod,)
                print('své lokální maximum a jeho hodnota je rovna ', f.subs({x:
 ↪stac_bod}))
        else:
            print('Funkce f nenabývá v bodě', stac_bod)
            print('lokální extrém')
```

```
    plot(f, (x, min(solns) - delka_solns, max(solns) + delka_solns))


def je_sude(k):
    if k % 2 == 0:
        return True
    else:
        return False
def najdi_stac_body(f):
    df = diff(f, x)
    return list(solveset(df, domain = S.Reals))
def stupen_der_nonzero(f, u):
    k = 2
    while d(f, k).subs({x:u}) == 0:
        k += 1
    return k
def pridej_koncove_body(a, b, seznam):
    seznam.append(a)
    seznam.append(b)
def d(f, k):
    if k == 0:
        return f
    if k > 0:
        return diff(d(f, k-1), x)
main()
```

```
Zadejte předpis funkce: y=x**6
Tady je seznam stacionárních bodů:  [0]
Funkce f nabývá v bodě  0
své lokální minimum a jeho hodnota je rovna  0
```

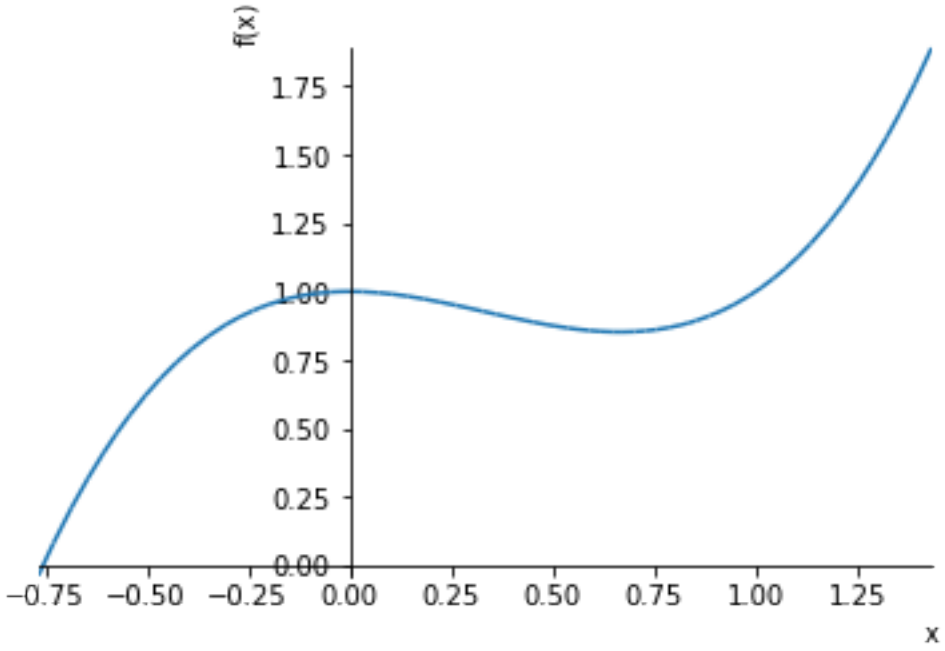



[4]:
```
# Program s využitím objektového přístupu.
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 15 15:20:06 2019

@author: Dušan Bednařík
"""

from math import pi, e
from sympy import *
x,y = symbols('x, y')

def d(f, k):
    if k == 0:
        return f
    if k > 0:
        return diff(d(f, k-1), x)

def je_sude(k):
    if k % 2 == 0:
        return True
    else:
        return False

class Function():
```

```
    def __init__(self, f):
        self.f = f
#       self.k_ta_derivace = self.f.der(k)
    def show_function(self):
        print('Tady je funkce y = ', self.f)
    def der(self, k):
        return d(self.f, k)
    def substitute(self, hodnota):
        return self.f.subs({x: hodnota})
    def update_function(self, new_f):
        self.f = new_f
    def null_points(self):
        return list(solveset(self.f, domain = S.Reals))
    def stupen_nonzero_der(self, hodnota):
        k = 2
        while d(self.f, k).subs({x:hodnota}) == 0:
            k += 1
        return k


def main():
    f = S(input('Define:  f(x)= '))
    function = Function(f)
    function.show_function()
    '''
    Nyní se zjistí stacionární body funkce f
    '''
    d1f = Function(function.der(1))
    solns = d1f.null_points()
    delka_solns = max(solns) - min(solns) + 0.1  # délka intervalu v němž leží
                                                  # stac body
    print('Tady je seznam stacionárních bodů: ', solns)
    for stac_point in solns:
        m = function.stupen_nonzero_der(stac_point)
        dmf = Function(function.der(m))
        if je_sude(m):
            if dmf.substitute(stac_point)>0:
                print('Funkce nabývá v bodě ', stac_point)
                print('lokální minimum a jeho hodnota je rovna: ', \
                      function.substitute(stac_point))
            else:
                print('Funkce nabývá v bodě ', stac_point)
                print('lokální maximum a jeho hodnota je rovna ', \
                      function.substitute(stac_point))
        else:
            print('Funkce nemá v bodě ', stac_point, 'lokální extrém')
    plot(f, (x, min(solns) - delka_solns, max(solns) + delka_solns))
```

```
main()
```

```
Define:  f(x)= x**3 - x**2 + 1
Tady je funkce y =  x**3 - x**2 + 1
Tady je seznam stacionárních bodů:  [0, 2/3]
Funkce nabývá v bodě  0
lokální maximum a jeho hodnota je rovna  1
Funkce nabývá v bodě  2/3
lokální minimum a jeho hodnota je rovna:  23/27
```




[7]:
```
# Vyzkoušejte si definici třídy Function(), vytvoření instance této třídy (řádek
 ↪č. 34)
# a zavolání metody null_points() na objekt s názvem function (řádek č. 35),
# který je právě "instancí" třídy Function()
from sympy import *
x,y = symbols('x, y')

def d(f, k):
    if k == 0:
        return f
    if k > 0:
        return diff(d(f, k-1), x)
```

```
class Function():
    def __init__(self, f):
        self.f = f
#       self.k_ta_derivace = self.f.der(k)
    def show_function(self):
        print('Tady je funkce y = ', self.f)
    def der(self, k):
        return d(self.f, k)
    def substitute(self, hodnota):
        return self.f.subs({x: hodnota})
    def update_function(self, new_f):
        self.f = new_f
    def null_points(self):
        return list(solveset(self.f, domain = S.Reals))
    def stupen_nonzero_der(self, hodnota):
        k = 2
        while d(self.f, k).subs({x:hodnota}) == 0:
            k += 1
        return k
f = S(input('Define:  f(x)= '))
function = Function(f)
function.null_points()
```

```
Define:  f(x)= x**6
```

[7]: `[0]`

### 4.4 Hledání absolutních extrémů

Uvažujme úlohu:

$$f(x) \longrightarrow min, \quad x \in (a,b).$$

Zde $-\infty \leq a < b \leq \infty$.

Dále přepokládejme, že $\lim_{x\to a+} f(x) = +\infty$ a $\lim_{x\to b-} f(x) = +\infty$. Je-li dále funkce $f$ spojitou funkcí na intervalu $(a,b)$, potom existuje $x_0 \in absmin(f)$.

**Důkaz.** V první řadě je zapotřebí si uvědomit, že za daných předpokladů je funkce $f$ zdola omezená na intervalu $(a,b)$. Dále položme $m = \inf\{f(x) : x \in (a,b)\}$. Vzhledem k tomu, že limity na koncích intervalu jsou rovny plus nekonečnu, existuje uzavřený podinterval $[c,d] \subset (a,b)$ takový, že $m = \inf\{f(x) : x \in [c,d]\}$. Nyní z Weierstrassovy věty plyne existence bodu $x_0 \in [c,d]$ tak, že $f(x_0) = m$. □

Jako **cvičení** si naformulujte analogické tvrzení pro úlohu nalezení absolutního maxima!

Jestliže například $\lim_{x\to a+} f(x) = -\infty$, potom funkce $f$ je zdola neomezená a tudíž nenabývá v žádném bodě absolutní minimum. Obdobně, jestliže je například $\lim_{x\to b-} f(x) = +\infty$, potom je funkce $f$ shora neomezenou funkcí a tudíž v žádném bodě nenabývá absolutní maximum.

## 5 Kalkulus

V následujícím příkladu budeme počítat parciální derivaci třetího řádu $\frac{\partial^3 f}{\partial x^3}(x,y)$. (někdy budeme používat označení $f_{xxx}$.)

Tato tetí derivace se počítá postupným parciálním derivováním:

$$\frac{\partial}{\partial x}(\frac{\partial}{\partial x}(\frac{\partial}{\partial x}f)).$$

[10]:
```
# Zde použijeme funkci diff tak, aby najednou spočítala třetí derivaci (třikrát␣
↪podle proměnné x)
diff(cos(x*y), y, x)
```

[10]:
```
-(x*y*cos(x*y) + sin(x*y))
```

[11]:
```
diff(cos(x*y), x, y)
```

[11]:
```
-(x*y*cos(x*y) + sin(x*y))
```

[82]:
```
# Nyní zkusme aplikovat funkci diff postupně:
f_x = diff(cos(x*y),x)
f_xx = diff(f_x,x)
f_xxx = diff(f_xx,x)
print(f_x,f_xx,f_xxx)
```

```
-y*sin(x*y) -y**2*cos(x*y) y**3*sin(x*y)
```

$g(x,y) = xe^{xy^2}$

[21]:
```
g = x*exp(x*y**2)
print(diff(g, x, x, y))
print(diff(g, y, x, x))
print(diff(g, x, y, x))
print(g.diff(x, x, y))
```

```
2*y*(x**2*y**4 + 4*x*y**2 + 2)*exp(x*y**2)
2*y*(x**2*y**4 + 4*x*y**2 + 2)*exp(x*y**2)
2*y*(x**2*y**4 + 4*x*y**2 + 2)*exp(x*y**2)
2*y*(x**2*y**4 + 4*x*y**2 + 2)*exp(x*y**2)
```

[ ]:
```
# Zkuste cvičně spočítat několik příkladů na parciální derivace!
# Například spočtěte parciální derivace druhého řádu funkce 1+x*y*z -y**2*z+z**3
```

Uvažujme funkci dvou proměnných $f : \mathbb{R}^n \to \mathbb{R}$. Dál předpokládejme, že existují parciální derivace prvního řádu, $f_x$ resp. $f_y$. Gradientem funkce $f$ budeme rozumět dvourozměrný vektor $gradf = (f_x, f_y)$.

[27]:
```
# Tady si zkusíme příklad na výpočet gradientu
from sympy import *
init_printing()
x ,y, z, t = symbols('x, y, z, t')
```

```
# u = [x,y]
gradf = derive_by_array(sin(x*y),[x, y])
pretty_print(gradf)
```

```
[y·cos(x·y)  x·cos(x·y)]
```

[33]:
```
gradf.subs({x:1, y:0})
```

[33]:
```
[0, 1]
```

Hessova matice funkce $f$ je matice:

$$A = f''(x_0) = \left( \frac{\partial f}{\partial x_i \partial x_j}(x_0) \right)_{i,j=1}^{n} = (a_{ij})_{i,j=1}^{n}.$$

[28]:
```
Hess = derive_by_array(gradf,[x, y])
Hess
```

[28]:
$$\begin{bmatrix} -y^2 \sin(xy) & -xy \sin(xy) + \cos(xy) \\ -xy \sin(xy) + \cos(xy) & -x^2 \sin(xy) \end{bmatrix}$$

[30]:
```
# Vyzkoušejte si výpočet tzv. Hessovy matice na příkladech funkcí:
init_printing()
h = x*y + x**2*y
u = [x, y]
print(derive_by_array(h, u))
gradh = derive_by_array(h, u)
Hess_matrix = derive_by_array(gradh, u)
Hess_matrix
```

```
[2*x*y + y, x**2 + x]
```

[30]:
$$\begin{bmatrix} 2y & 2x+1 \\ 2x+1 & 0 \end{bmatrix}$$

[31]:
```
Hess_matrix.subs({x:1, y:1})
```

[31]:
$$\begin{bmatrix} 2 & 3 \\ 3 & 0 \end{bmatrix}$$

[5]:
```
print(limit(sin(x)/x, x, 0))
print(limit(x**3 + x**2 -1, x, -oo))
print(limit(x**3 + x**2 -1, x, +oo))
```

```
1
-oo
oo
```

Vyšetřete absolutní extrémy funkcí na intervalu $\langle -1, 10 \rangle$ :

a) $f(x) = x^3 + x^2 - 1$.

b) $f(x) = (x-1)^4 + x^2$. (program nefunnguje a vrací chybu :....)

c) $f(x) = sin(x)$. (program nefunguje vlivem faktu, že funkce má na reálné ose nekonečně mnoho stac. bodů)

d) $f(x) = x * exp(x), x \in \langle -2, 10 \rangle$.

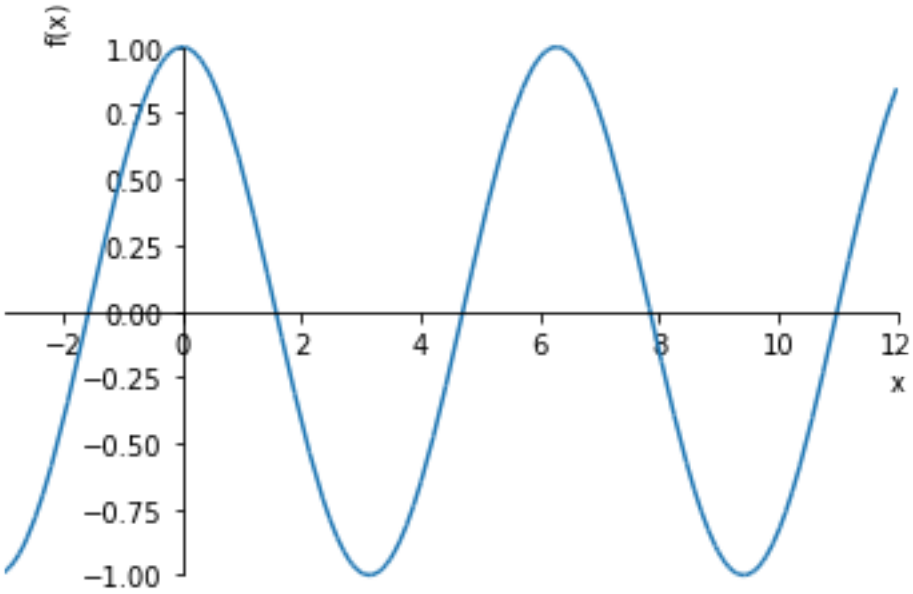
e) $f(x) = (x+1)/(2 * x * *3 - x * *2 + 1), x \in \langle -2, 10 \rangle$.
Chyba: —> 56 provizorniSeznam = list(solveset(df, x, domain = Interval.open(a,b))) TypeError: Not all constituent sets are iterable

[3]:
```
# Tento program má řešit úlohu nalezení
# maxima a minima spojité a diferencovatelné funkce na
# omezeném a uzavřeném intervalu
from sympy import *
x = symbols('x')
def main():
    #f = x**3 + x**2 - 1
    f = cos(x)
    #f = (x+1)/(2*x**3 - x**2 + 1)
    a = -2.0; b = 10.0
    #f = S(input('Zadej předpis funkce y = '))
    #a = float(input('Zadej levý koncový bod a = '))
    #b = float(input('Zadej pravý koncový bod b = '))
    print('----------------------------------------')
    while b <= a:
        print('Zadej body a,b tak, aby platilo: a < b !')
        print('----------------------------------------')
        a = float(input('Zadej levý koncový bod a = '))
        b = float(input('Zadej pravý koncový bod b = '))
    stacBody = najdi_stac_body_in(f, a, b)      # atd.
    print('Seznam stacionárních bodů: ', stacBody)
    seznam_podezrelych_bodu = pridej_koncove_body(a,b, stacBody)
    # metoda sort() uspořádá položky seznamu "seznam_podezrelych_bodu"
    # podle velikosti
    seznam_podezrelych_bodu.sort()
    print('Seznam podezřelých bodů: ', seznam_podezrelych_bodu)
    Max = max(seznam_fun_hodnot(f, a, b, seznam_podezrelych_bodu))
    Min = min(seznam_fun_hodnot(f, a, b, seznam_podezrelych_bodu))
    argMax = getKey(a,b,bodyGrafu(f, seznam_podezrelych_bodu), Max)
    argMin = getKey(a,b,bodyGrafu(f, seznam_podezrelych_bodu), Min)
    #print(argMax)
    #print(argMin)
    print('Maximum funkce f má hodnotu: ', Max.evalf(),\
          'a nabývá se v bodě: ', argMax)
    print('Minimum funkce f má hodnotu: ', \
```

```
        Min.evalf(), 'a nabývá se v bodě: ', argMin)
    #print(bodyGrafu(f, seznam_podezrelych_bodu))
    plot(f, (x, a - 1, b + 2))

def seznam_fun_hodnot(f, a, b, seznam):
    s = []
    for hodnota in seznam:
        if hodnota >= a and hodnota <= b:
            nova_polozka = f.subs({x: hodnota})
            s.append(nova_polozka)
    return s
def bodyGrafu(f, seznam):
    slovnik = {}
    for item in seznam:
        slovnik[item] = f.subs({x: item})
    return slovnik
# def najdi_stac_body(f):
#     df = diff(f, x)
#     return list(solveset(df, x, domain = S.Reals))

def najdi_stac_body_in(f, a, b):
    """ Tato funkce vrací stacionární body funkce f,
    které leží mezi hodnotami a, b."""
    df = diff(f, x)
    provizorniSeznam = list(solveset(df, x, domain = Interval.open(a,b)))
    s = []
    for bod in provizorniSeznam:
        if bod > a and bod < b:
            s.append(bod)
    return s

def pridej_koncove_body(a, b, seznam):
    """ Tato funkce má tři parametry a připojí
    k objektu seznam dvě položky jejichž hodnoty
    jsou obsaženy v proměnných
    a,b. Funkce nám pak vrací tento nový seznam.  """
    seznam.append(a)
    seznam.append(b)
    return seznam
def getKey(a, b, slovnik, value):
    list_of_keys = []
    klice = slovnik.keys()
    for klic in klice:
        if slovnik[klic] == value:
            if klic != a and klic != b:
                list_of_keys.append(klic.evalf())
            else:
```

```
                list_of_keys.append(klic)
    return list_of_keys

main()
```

```
----------------------------------------
Seznam stacionárních bodů:  [0, pi, 2*pi, 3*pi]
Seznam podezřelých bodů:  [-2.0, 0, pi, 2*pi, 3*pi, 10.0]
Maximum funkce f má hodnotu:  1.00000000000000 a nabývá se v bodě:  [0,
6.28318530717959]
Minimum funkce f má hodnotu:  -1.00000000000000 a nabývá se v bodě:
[3.14159265358979, 9.42477796076938]
```

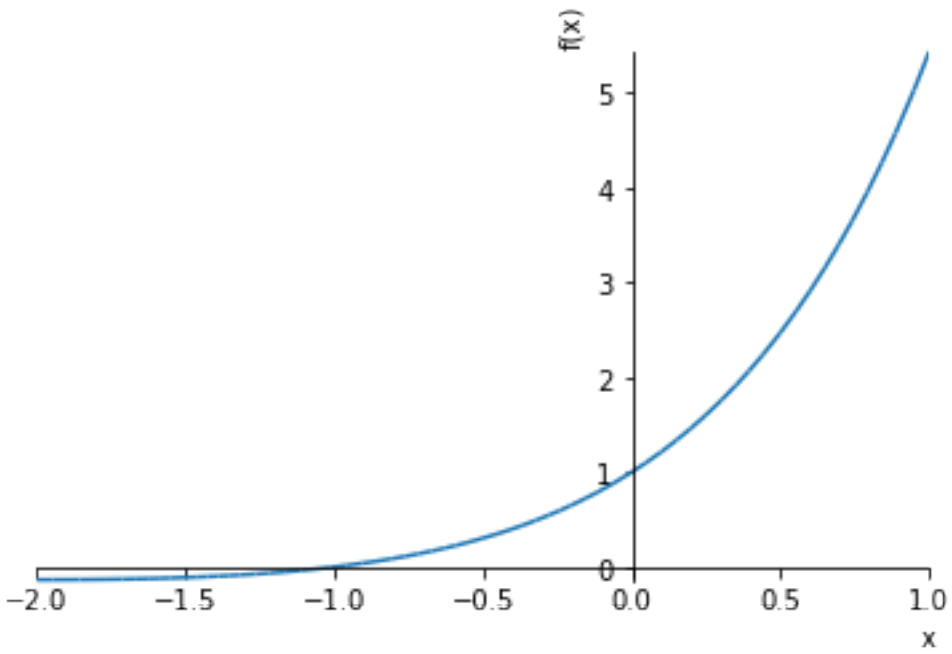



[20]:
```
from sympy import *
x = symbols('x')
def najdi_stac_body_in(f, a, b):
    """ Tato funkce vrací stacionární body funkce f,
    které leží mezi hodnotami a, b."""
    df = diff(f, x)
    provizorniSeznam = list(solveset(df, x, domain = Interval.open(a,b)))
    s = []
    for bod in provizorniSeznam:
        if bod > a and bod < b:
            s.append(bod)
    return s
najdi_stac_body_in((x+1)/(2*x**3 - x**2 + 1), -10, 5)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-20-74ed51a5402a> in <module>()
 11             s.append(bod)
 12     return s
---> 13 najdi_stac_body_in((x+1)/(2*x**3 - x**2 + 1), -10, 5)


<ipython-input-20-74ed51a5402a> in najdi_stac_body_in(f, a, b)
  5     které leží mezi hodnotami a, b."""
  6     df = diff(f, x)
----> 7     provizorniSeznam = list(solveset(df, x, domain = Interval.
↪open(a,b)))
  8     s = []
  9     for bod in provizorniSeznam:


C:\ProgramData\Anaconda3\lib\site-packages\sympy\sets\sets.py in␣
↪__iter__(self)
   1443             return roundrobin(*(iter(arg) for arg in self.args))
   1444         else:
-> 1445             raise TypeError("Not all constituent sets are iterable")
   1446
   1447 class Intersection(Set):


TypeError: Not all constituent sets are iterable
```

[25]:
```
df = diff(x*exp(x), x)
print(df)
print(df.subs({x: -1}))
plot(df, (x, -2, 1))
```

```
x*exp(x) + exp(x)
0
```




[25]:
```
<sympy.plotting.plot.Plot at 0x1134acbe0>
```

[4]:
```
def getKey(slovnik, value):
    list_of_keys = []
    klice = slovnik.keys()
    for klic in klice:
        if slovnik[klic] == value:
            list_of_keys.append(klic)
    return list_of_keys

slovnik = {'Honza': '333 556', 'Eva': '455 888', 'Petr': '411 777',\
           'Pavel': '566 777', 'Jana':'566 777'}
getKey(slovnik, '566 777')
```

[4]:
```
['Pavel', 'Jana']
```

[5]:
```
def bodyGrafu(f, seznam):
    slovnik = {}
    for item in seznam:
        slovnik[item] = f.subs({x: item})
    return slovnik
f = x**2; seznam = [0, 1, 5]
bodyGrafu(f, seznam)
```

[5]:
```
{0: 0, 1: 1, 5: 25}
```

[9]:
```
solveset(Eq(sin(x), 1), x, domain = Interval(0,2*pi))
```

[9]:
```
{pi/2}
```

[8]:
```
solveset(sin(x)-2, x, domain = Interval.open(0,2*pi))
```

[8]:
```
EmptySet()
```

[12]:
```
solveset(Eq(x**2, -1), x, domain = S.Reals)
```

[12]:
```
EmptySet()
```

[23]:
```
from math import pi, sin
a = 0; b = 2*pi
def muj_sin(x):
    if x >= a and x <= b:
        return sin(x)
    else:
        return 0
muj_sin(-pi/2)
```

[23]:
```
0
```

[24]:
```
def abs_hodnota(x):
    if x >= 0:
        return x
    else:
        return -x
solveset(Eq(abs_hodnota(x), 0), x)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-24-bd6876ddf9f1> in <module>()
  4     else:
  5         return -x
----> 6 solveset(Eq(abs_hodnota(x), 0), x)


<ipython-input-24-bd6876ddf9f1> in abs_hodnota(x)
  1 def abs_hodnota(x):
----> 2     if x >= 0:
  3         return x
  4     else:
  5         return -x


C:\ProgramData\Anaconda3\lib\site-packages\sympy\core\relational.py in
↪__nonzero__(self)
  193
  194     def __nonzero__(self):
--> 195         raise TypeError("cannot determine truth value of Relational")
```

```
196
197     __bool__ = __nonzero__


TypeError: cannot determine truth value of Relational
```

## 5.1 Funkce dvou proměnných

### 5.1.1 Cvičení - test definitnosti matice 2x2

Napište program, který vyštří, je-li matice 2x2 pozitivně definitní.

[1]:
```
from sympy import *
A = Matrix([[2, 5], [3, 0]])
A[0, 0]
A[1,1]
det(A)
A.det()
```

[1]:
```
-15
```

[2]:
```
# Řešení
def is_positive(A):
    if A[0,0] > 0 and det(A) > 0:
        return True
    else:
        return False
is_positive(A)

def is_negative(A):
    if A[0,0] < 0 and det(A) > 0:
        return True
    else:
        return False
is_negative(A)

if is_positive(A):
    print('Matice A je pozitivně definitní.')
elif is_negative(A):
    print('Matice A je negativně definitní.')
else:
    print('Matice A není ani pozitivně ani negativně definitní.')
```

```
Matice A není ani pozitivně ani negativně definitní.
```

Uvažujme funkci $Q(u) = u^T \cdot A \cdot u = \langle Au, u \rangle$

Zkoumejme minimum této kvadratické funkce. Platí, že pokud je matice A pozitivně definitní, pak funkce $Q$ nabývá v nějakém bodě $u \in R^n$ své absolutní minimum a jeho hodnota je rovna číslu $\lambda$ tak, že je splněna rovnice:

$$Au = \lambda u.$$

[ ]:
```
# Charakteristická rovnice matice A je rovnice: det(A -\lambda I)=0. Zde matice␣
↪I je
# tzv. jednotková matice.
```

[12]:
```
# Matice A má dvě vlastní čísla 5 a -3 ---> A není pozitivně definitní
slovnik = A.eigenvals()
print(slovnik)
klice = slovnik.keys()
for klic in klice:
    print(klic)
```

```
{5: 1, -3: 1}
5
-3
```

## 6 Vlastní čísla matice

Číslo $\lambda \in \mathbb{C}$ se nazývá vlastním číslem čtvercové matice $A$, existuje-li nenulový vektor $u$ takový, že

$$Au = \lambda u.$$

Viz Vlastní čísla a vektory

My budeme dále pracovat s tzv. symetrickými reálnými maticemi. To jsou takové čtvercové matice $(a_{ij})_{i,j=1}^{n}$, pro které platí:

$$\forall i, j \quad a_{ij} = a_{ji}.$$

Předpokládejme, že $A \in M_n(R)$ je symetrická matice kde $M_n(R)$ značí množinu všech čtvrcových matic rozměru $n \times n$ s reálnými prvky.

Nyní uvažujme úlohu maximalizovat funkci $q(x) = x^T Ax$ za předpokladu, že $x \in R^n$ a $x^T x = 1$.

Nyní lze dokázat, že tato úloha má řešení $\hat{x} \in R^n$ a pro nějaké reálné číslo $\lambda \in R^n$ pak musí nutně platit:

$$A\hat{x} = \lambda\hat{x}.$$

Odtud, je je zřejmé, že $\hat{x}$ je vlastním vektorem matice $A$ a $\lambda$ je vlastním číslem matice $A$. Navíc $\lambda$ je maximem funkce $q$.

Dále platí, že je-li $A \in M_n(R)$ symetrická matice, pak jsou všechny její vlastní čísla **reálnými čísly** a

$$\lambda_{min}|x|^2 \leq x^T Ax \leq \lambda_{max}|x|^2, \quad \forall x \in R^n,$$

kde $\lambda_{min}$, resp. $\lambda_{max}$ značí nejmenší resp. největší vlastní číslo matice $A$.

[5]:
```
# Konstrukce jednotkové matice:
from sympy import *
init_printing()
eye(3)
```

[5]:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[1]:
```
# Počítejme vlastní čísla matice:
from sympy import *
init_printing()
t, lamda = symbols('t, lamda')
A = Matrix([[0, -1, 4], [-1, 0, 1], [4, 1, 0]]) # matice A je symetrická!
E = eye(3)
D = det(A -lamda*E )
print("A - lamda*E = ")
pprint(A - lamda*E)
print("D = Det(A - lamda*E) = ", D, "=")
pprint(D)
```

```
A - lamda*E =
[U+23A1]-λ  -1  4 [U+23A4]
[U+23A2]          [U+23A5]
[U+23A2]-1  -λ  1 [U+23A5]
[U+23A2]          [U+23A5]
[U+23A3]4   1   -λ[U+23A6]
D = Det(A - lamda*E) =  -lamda**3 + 18*lamda - 8 =
   3
- λ  + 18⋅λ - 8
```

[4]:
```
print("Charakteristický mnohočlen p(lamda) = ")
simplify(D).expand(mul=True)
```

```
Charakteristický mnohočlen p(lamda) =
```

[4]:
$$-\lambda^3 + 18\lambda - 8$$

[8]:
```
# Ukazuje se, že ne každé vlastní číslo reálné matice
# musí být reálným číslem.
from sympy import *
A.eigenvals()
```

[8]:
$$\left\{4 : 1, \quad -2 + \sqrt{6} : 1, \quad -\sqrt{6} - 2 : 1\right\}$$

### 6.0.1 Cvičení - vlastní čísla

Napište program testující pozitivní resp. negativní definitnost matice A s využitím kritéria pracujícího s vlastními čísly.

[6]:
```
def main():
    A = Matrix([[1, 2], [3, 4]])
    slovnik = A.eigenvals()
    if len(pos_eigenvals(slovnik)) == 0:
        print('Matice A je negativně definitní.')
    elif len(neg_eigenvals(slovnik)) == 0:
        print('Matice A je pozitivně definitní.')
    else:
        print('Matice A není ani pozitivně ani negativně definitní.')




def pos_eigenvals(slovnik):
    klice = slovnik.keys()
    seznamKladnychKlicu = []
    for klic in klice:
        if klic > 0:
            seznamKladnychKlicu.append(klic)
    return seznamKladnychKlicu

def neg_eigenvals(slovnik):
    klice = slovnik.keys()
    seznamZapornychKlicu = []
    for klic in klice:
        if klic < 0:
            seznamZapornychKlicu.append(klic)
    return seznamZapornychKlicu

main()
```

```
Matice A není ani pozitivně ani negativně definitní.
```

# 7 Funkce dvou proměnných - příklady na lok. extrémy

Testovací funkce:

a) $f(x,y) = x**2 + y**2$.

b) $f(x,y) = x**4 + y**4 - (x+y)**2$

c) $f(x,y) = x**2 - y**2$

d) $f(x,y) = x*y + 50/x + 20/y$

e) $f(x,y) = 3*x + 2*y - x**3 - y**2$

[41]:
```
# Napišme program, vyšetřující lokální extrémy funkce dvou proměnných.
from sympy import *
from sympy.plotting import plot3d
x,y = symbols('x, y')
init_printing()
# Nyní definujme tzv.hlavní funkci, kde definujeme logiku celého programu. Tato␣
 ↪funkce pochopitelně může záviset
# na dalších speciálních funkcích, které jsou volány uvnitř hlavní funkce main()
def main():
    f = 3*x + 2*y - x**3 - y**2
    #f = S(input('Zadejte předpis funkce: z='))
    print('Gradientem funkce f v bodě (x,y)', 'je vektor: ')
    df = derive_by_array(f, [x,y])
    print(df)
    solns = najdi_stac_body(f)
    print('Celkový počet kořenů: ', len(solns))
    print('Seznam všech kořenů soustavy: ', solns )
    real_solns = get_real_solns(solns)
    xsouradnice = []
    for bod in real_solns:
        xsouradnice.append(bod[0])
    minx = min(xsouradnice)
    maxx = max(xsouradnice)
    ysouradnice = []
    for bod in real_solns:
        ysouradnice.append(bod[1])
    miny = min(ysouradnice)
    maxy = max(ysouradnice)
    print('Tady je seznam stacionárních bodů: ', real_solns)
    print('Hessova matice v obecném bodě (x,y) je rovna matici: ')
    pprint(derive_by_array(df, [x,y]))
    for stac_bod in real_solns:
        print('Tady je Hessova matice vyčíslená v bodě', stac_bod, ': ')
        pprint(d2f(f, stac_bod[0], stac_bod[1]))
        if is_positive(d2f(f, stac_bod[0], stac_bod[1])):  # podmínka testuje␣
 ↪poziivní definitnost Hessovy matice d2f(stac_bod)
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální minimum', f.subs({x: stac_bod[0], y:␣
 ↪stac_bod[1]}))
        elif is_negative(d2f(f, stac_bod[0], stac_bod[1])):
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální maximum', f.subs({x: stac_bod[0], y:␣
 ↪stac_bod[1]}))
        else:
            print('Postačující podmínky nejsou splněny a tudíž nelze')
            print('naši větu použít.')
            pprint(d2f(f, stac_bod[0], stac_bod[1]))
```

```
    if len(real_solns) > 1:
        plot3d(f, (x, minx, maxx), (y, miny, maxy))
    else:
        plot3d(f, (x, minx - 1, maxx + 1), (y, miny - 1, maxy + 1))


def get_real_solns(sezn_korenu):
    s = []
    for koren in sezn_korenu:
        if koren[0].is_real and koren[1].is_real:
            s.append(koren)
    return s

def d2f(f,a,b):
    u = [x,y]
    gradf = derive_by_array(f, u)
    Hess = derive_by_array(gradf, u)
    return Hess.subs({x:a, y:b})


def najdi_stac_body(f):
    gradf = derive_by_array(f, [x,y])
    return list(nonlinsolve(gradf, [x,y]))

def is_positive(A):
    if A[0,0] > 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:              #␣
 ↪Rozhodovací blok
        return True
    else:
        return False
def is_negative(A):
    if A[0,0] < 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:
        return True
    else:
        return False
main()
```

```
Gradientem funkce f v bodě (x,y) je vektor:
[3 - 3*x**2, 2 - 2*y]
Celkový počet kořenů:  2
Seznam všech kořenů soustavy:  [(-1, 1), (1, 1)]
Tady je seznam stacionárních bodů:  [(-1, 1), (1, 1)]
Hessova matice v obecném bodě (x,y) je rovna matici:
[U+23A1]-6·x  0 [U+23A4]
[U+23A2]        [U+23A5]
[U+23A3] 0    -2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (-1, 1) :
```

```
[U+23A1]6  0 [U+23A4]
[U+23A2]     [U+23A5]
[U+23A3]0  -2[U+23A6]
Postačující podmínky nejsou splněny a tudíž nelze
naši větu použít.
[U+23A1]6  0 [U+23A4]
[U+23A2]     [U+23A5]
[U+23A3]0  -2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (1, 1) :
[U+23A1]-6  0 [U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]0   -2[U+23A6]
Funkce nabývá v bodě (1, 1)
(ostré) lokální maximum 3

<Figure size 640x480 with 1 Axes>
```

[18]:
```
from sympy import *
from sympy.plotting import plot3d
x,y = symbols('x, y')
init_printing()
def d2f(f):
    u = [x,y]
    gradf = derive_by_array(f, u)
    Hess = derive_by_array(gradf, u)
    return Hess
d2f(x**3 -  y**2 + x)
```

[18]:
$$\begin{bmatrix} 6x & 0 \\ 0 & -2 \end{bmatrix}$$

[15]:
```
diff(x**2 + y**2, x, y)
```

[15]:
$$0$$

[13]:
```
diff(Out[11], y)
```

[13]:
$$0$$

## 8 Program pro 1,2 a 3 dimenzionální případ

V následujícím kódu budeme potřebovat testovat pozitivní definitnost matice 3x3. Lze použít tzv. Sylvestrovo kritérium. Viz. https://cs.wikipedia.org/wiki/Sylvesterovo_krit%C3%A9rium

[44]:
```
# Toto je pokus napsat kód, který bude vyšetřovat lokální extrémy
# funkce jedné, dvou nebo tří proměnných v úloze bez omezení.
```

```
from sympy import *
from numpy.matlib import *
import numpy as np
from sympy.plotting import plot3d
x, y, z = symbols('x, y, z') # tady definujeme kolekci symbolických proměnných


class Function():
    def __init__(self, f):
        self.f = f
    def show_function(self):
        print('Tady je funkce y = ', self.f)
    def der(self, k):
        return d(self.f, k)
    def substitute(self, hodnota):
        return self.f.subs({x: hodnota})
    def update_function(self, new_f):
        self.f = new_f
    def null_points(self):
        return list(solveset(self.f, domain = S.Reals))
    def stupen_nonzero_der(self, hodnota):
        k = 2
        while d(self.f, k).subs({x:hodnota}) == 0:
            k += 1
        return k


class Funkce2:
    def __init__(self, f):
        self.f = f

class Funkce3():
    def __init__(self, f):
        self.f = f
    def fun_value(self, a, b, c):
        return self.f.subs({x: a, y: b, z: c})
    def najdi_stac_body(self):
        return nonlinsolve(derive_by_array(self.f, (x,y,z)),(x, y, z))
    def najdi_gradient(self):
        return derive_by_array(self.f, (x, y, z))
    def najdi_Hess_matrix(self):
        gradf = derive_by_array(self.f, (x, y, z))
        Hess = derive_by_array(gradf, (x, y, z))
        return Hess
    def Hess_subst(self, a, b, c):
        u = [x, y, z]
        gradf = derive_by_array(f, u)
```

```
        Hess = derive_by_array(gradf, u)
        H11 = Hess[0,0]; H12 = Hess[0,1]; H13 = Hess[0,2];\
        H21 = Hess[1,0]; H22 = Hess[1,1]; H23 = Hess[1,2];\
        H31 = Hess[2,0]; H32 = Hess[2,1]; H33 = Hess[2,2]
        Hess_matrix = Matrix([[H11, H12, H13], [H21, H22, H23], [H31, H32, H33]])
        return Hess_matrix.subs({x:a, y:b, z:c})




class Gradient:
    def __init__(self, f):
        self.f = f
    def get_gradient(self, u):
        return derive_by_array(f, u)

def get_real_solns(sezn_korenu):
    s = []
    for koren in sezn_korenu:
        if koren[0].is_real and koren[1].is_real:
            s.append(koren)
    return s

def d2f(f,a,b):
    u = [x,y]
    gradf = derive_by_array(f, u)
    Hess = derive_by_array(gradf, u)
    return Hess.subs({x:a, y:b})

def najdi_stac_body(f):
    gradf = derive_by_array(f, [x,y])
    return list(nonlinsolve(gradf, [x,y]))

def is_positive(A):
    if A[0,0] > 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:              #␣
 ↪Rozhodovací blok
        return True
    else:
        return False
def is_negative(A):
    if A[0,0] < 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:
        return True
    else:
        return False

def is_positive_3d(A):
```

```
    D3 = det(A)
    A.row_del(2)
    A.col_del(2)
    D2 = det(A)
    D1 = A[0, 0]
    if D1 > 0 and D2 > 0 and D3 > 0:
        return True
    else:
        return False

def is_negative_3d(A):
    D3 = A.det()
    A.row_del(2)
    A.col_del(2)
    D2 = A.det()
    D1 = A[0, 0]
    if D1 < 0 and D2 > 0 and D3 < 0:
        return True
    else:
        return False

# def is_nonnegative_3d(A):


def d(f, k):
    if k == 0:
        return f
    if k > 0:
        return diff(d(f, k-1), x)

def je_sude(k):
    if k % 2 == 0:
        return True
    else:
        return False


def optimize_loc_1():
    f = S(input('Define:  f(x)= '))
    function = Function(f)
    function.show_function()
    '''
    Nyní se zjistí stacionární body funkce f
    '''
    d1f = Function(function.der(1))
    solns = d1f.null_points()
    delka_solns = max(solns) - min(solns) + 0.1  # délka intervalu v němž leží
```

```
                                                    # stac body
    print('Tady je seznam stacionárních bodů: ', solns)
    for stac_point in solns:
        m = function.stupen_nonzero_der(stac_point)
        dmf = Function(function.der(m))
        if je_sude(m):
            if dmf.substitute(stac_point)>0:
                print('Funkce nabývá v bodě ', stac_point)
                print('lokální minimum a jeho hodnota je rovna: ', \
                      function.substitute(stac_point))
            else:
                print('Funkce nabývá v bodě ', stac_point)
                print('lokální maximum a jeho hodnota je rovna ', \
                      function.substitute(stac_point))
        else:
            print('Funkce nemá v bodě ', stac_point, 'lokální extrém')
    plot(f, (x, min(solns) - delka_solns, max(solns) + delka_solns))

def optimize_loc_2():
    f = S(input('Zadejte předpis funkce: z='))
    print('Gradientem funkce f v bodě (x,y)', 'je vektor: ')
    df = derive_by_array(f, [x,y])
    print(df)
    solns = najdi_stac_body(f)
    print('Celkový počet kořenů: ', len(solns))
    print('Seznam všech kořenů soustavy: ', solns )
    real_solns = get_real_solns(solns)
    xsouradnice = []
    for bod in real_solns:
        xsouradnice.append(bod[0])
    minx = min(xsouradnice)
    maxx = max(xsouradnice)
    ysouradnice = []
    for bod in real_solns:
        ysouradnice.append(bod[0])
    miny = min(ysouradnice)
    maxy = max(ysouradnice)
    print('Tady je seznam stacionárních bodů: ', real_solns)
    print('Hessova matice v obecném bodě (x,y) je rovna matici: ')
    pprint(derive_by_array(df, [x,y]))
    for stac_bod in real_solns:
        print('Tady je Hessova matice vyčíslená v bodě', stac_bod, ': ')
        pprint(d2f(f, stac_bod[0], stac_bod[1]))
        if is_positive(d2f(f, stac_bod[0], stac_bod[1])):  # podmínka testuje␣
↪pozitivní definitnost Hessovy matice d2f(stac_bod)
            print('Funkce nabývá v bodě', stac_bod)
```

```
            print('(ostré) lokální minimum', f.subs({x: stac_bod[0], y:␣
↪stac_bod[1]}))
        elif is_negative(d2f(f, stac_bod[0], stac_bod[1])):
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální minimum', f.subs({x: stac_bod[0], y:␣
↪stac_bod[1]}))
        else:
            print('Postačující podmínky nejsou splněny a tudíž nelze')
            print('naši větu použít.')
            pprint(d2f(f, stac_bod[0], stac_bod[1]))
    if len(real_solns) > 1:
        plot3d(f, (x, minx, maxx), (y, miny, maxy))
    else:
        plot3d(f, (x, minx - 1, maxx + 1), (y, miny - 1, maxy + 1))

def optimize_loc_3():
    f = S(input('Zadejte předpis funkce: w = '))
    funkce = Funkce3(f)
    gradf = Gradient(f)
    solns = funkce.najdi_stac_body()
    print('Celkový počet kořenů: ', len(solns))
    print('Seznam všech kořenů soustavy: ', solns )
    real_solns = get_real_solns(solns)
    xsouradnice = []
    for bod in real_solns:
        xsouradnice.append(bod[0])
    minx = min(xsouradnice)
    maxx = max(xsouradnice)
    ysouradnice = []
    for bod in real_solns:
        ysouradnice.append(bod[1])
    miny = min(ysouradnice)
    maxy = max(ysouradnice)
    zsouradnice = []
    for bod in real_solns:
        zsouradnice.append(bod[2])
    minz = min(zsouradnice)
    maxz = max(zsouradnice)
    print('Tady je seznam stacionárních bodů: ', real_solns)
    print('Hessova matice v obecném bodě (x,y) je rovna matici: ')
    print(funkce.najdi_Hess_matrix())
    for stac_bod in real_solns:
        print('Tady je Hessova matice vyčíslená v bodě', stac_bod, ': ')
        print(funkce.Hess_subst(stac_bod[0], stac_bod[1], stac_bod[2]))
        if is_positive_3d(funkce.Hess_subst(stac_bod[0], stac_bod[1],␣
↪stac_bod[2])):  # podmínka testuje poziivní definitnost Hessovy matice␣
↪d2f(stac_bod)
```

```
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální minimum', funkce.fun_value(stac_bod[0],␣
 ↪stac_bod[1], stac_bod[2]))
        elif is_negative3d(funkce.Hess_subst(stac_bod[0], stac_bod[1],␣
 ↪stac_bod[2])):
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální maximum', funkce.fun_value(stac_bod[0],␣
 ↪stac_bod[1], stac_bod[2]))
        else:
            print('Postačující podmínky nejsou splněny a tudíž nelze')
            print('naši větu použít.')




def main():
    dimenze = int(input('Zadejte počet proměnných (1,2, nebo 3) : ')) #␣
 ↪předpokládáme, že zadáte hodnotu 1,2 nebo 3
    if dimenze == 1:
        optimize_loc_1()
    elif dimenze == 2:
        optimize_loc_2()
    else:
        optimize_loc_3()

main()
```

```
Zadejte počet proměnných (1,2, nebo 3) : 3
Zadejte předpis funkce: w = x*y -z**2
Celkový počet kořenů:  1
Seznam všech kořenů soustavy:  {(0, 0, 0)}
Tady je seznam stacionárních bodů:  [(0, 0, 0)]
Hessova matice v obecném bodě (x,y) je rovna matici:
[[0, 1, 0], [1, 0, 0], [0, 0, -2]]
Tady je Hessova matice vyčíslená v bodě (0, 0, 0) :
Matrix([[0, 1, 0], [1, 0, 0], [0, 0, 0]])
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-44-6c14e02d1001> in <module>
        252         optimize_loc_3()
        253
    --> 254 main()
```

```
        <ipython-input-44-6c14e02d1001> in main()
        250         optimize_loc_2()
        251     else:
    --> 252         optimize_loc_3()
        253
        254 main()


        <ipython-input-44-6c14e02d1001> in optimize_loc_3()
        232             print('Funkce nabývá v bodě', stac_bod)
        233             print('(ostré) lokální minimum', funkce.
↪fun_value(stac_bod[0], stac_bod[1], stac_bod[2]))
    --> 234         elif is_negative3d(funkce.Hess_subst(stac_bod[0], stac_bod[1],␣
↪stac_bod[2])):
        235             print('Funkce nabývá v bodě', stac_bod)
        236             print('(ostré) lokální maximum', funkce.
↪fun_value(stac_bod[0], stac_bod[1], stac_bod[2]))


        NameError: name 'is_negative3d' is not defined
```

[35]:
```
from sympy import *
x, y, z = symbols("x, y, z")
f = x**2 + 2*y**2 + z**2 - 2*x*y + 2*z
```

[4]:
```
df_x = diff(f, x)
df_x
```

[4]:
```
2*x - 2*y - 1
```

[36]:
```
# Výpočet gradientu df funkce f:
gradf = derive_by_array(f, [x, y, z])    #  gradient funkce f
df = gradf
df[0]    # parciální derivace podle první proměnné x
df[1]     # parciální derivace podle první proměnné y
df[2]     # parciální derivace podle první proměnné z
df
```

[36]:
```
[2*x - 2*y, -2*x + 4*y, 2*z + 2]
```

Dále budeme řešit soustavu tří rovnic o třech neznámých:
gradf = 0

$$2 * x - 2 * y - 1 = 0 - 2 * x + 4 * y = 02 * z + 2 = 0$$

[37]:
```
reseni = nonlinsolve(gradf, [x, y, z])
reseni
```

[37]:
```
{(0, 0, -1)}
```

[40]:
```
HessMatrix = derive_by_array(gradf, [x, y, z])
pprint(HessMatrix)
HessMatrix
```

```
[U+23A1]2   -2  0[U+23A4]
[U+23A2]         [U+23A5]
[U+23A2]-2  4   0[U+23A5]
[U+23A2]         [U+23A5]
[U+23A3]0   0   2[U+23A6]
```

[40]:
```
[[2, -2, 0], [-2, 4, 0], [0, 0, 2]]
```

[41]:
```
HessMatrix = Matrix([[2, -2, 0], [-2, 4, 0], [0, 0, 2]])
HessMatrix
```

[41]:
```
Matrix([
[ 2, -2, 0],
[-2,  4, 0],
[ 0,  0, 2]])
```

[30]:
```
def is_positive_3d(A):
    D3 = det(A)
    A.row_del(2)
    A.col_del(2)
    D2 = det(A)
    D1 = A[0, 0]
    if D1 > 0 and D2 > 0 and D3 > 0:
        return True
    else:
        return False
```

[42]:
```
is_positive_3d(HessMatrix)
```

[42]:
```
True
```

Hessova matice je tedy ve stacionárním bodě (0, 0, -1) pozitivně definitní. Funkce nabývá v bodě (0, 0, -1) (ostré) lokální minimum.

[43]:
```
# Hodnota lokálního minima je rovna:
lokmin = f.subs({x: 1, y: 1/2, z: -1})
lokmin
```

[43]:
```
-0.500000000000000
```

[46]:
```
# Příklad č. 1
Zadejte počet proměnných (1,2, nebo 3) : 3
Zadejte předpis funkce: w = x**2 + 2*y**2 + z**2 - 2*x*y - x + 2*z
Celkový počet kořenů:  1
Seznam všech kořenů soustavy:  {(1, 1/2, -1)}
Tady je seznam stacionárních bodů:  [(1, 1/2, -1)]
Hessova matice v obecném bodě (x,y) je rovna matici:
[[2, -2, 0], [-2, 4, 0], [0, 0, 2]]
```

```
Tady je Hessova matice vyčíslená v bodě (1, 1/2, -1) :
Matrix([[2, 0, 0], [0, 2, 0], [0, 0, 2]])
Funkce nabývá v bodě (1, 1/2, -1)
(ostré) lokální minimum -3/2
--------------------------------------------------------------------------------
# Příklad č. 2
Zadejte předpis funkce: z=x**4 + y**4 - (x + y)**2
Gradientem funkce f v bodě (x,y) je vektor:
[4*x**3 - 2*x - 2*y, -2*x + 4*y**3 - 2*y]
Celkový počet kořenů:  7
Seznam všech kořenů soustavy:  [(-1, -1), (0, 0), (1, 1), ((-1 +␣
 ↪sqrt(3)*I)*sqrt(1 + sqrt(3)*I)/4, sqrt(1/4 + sqrt(3)*I/4)), (-sqrt(1 -␣
 ↪sqrt(3)*I)*(1 + sqrt(3)*I)/4, sqrt(1/4 - sqrt(3)*I/4)), (sqrt(1 -␣
 ↪sqrt(3)*I)*(1 + sqrt(3)*I)/4, -sqrt(1/4 - sqrt(3)*I/4)), ((1 -␣
 ↪sqrt(3)*I)*sqrt(1 + sqrt(3)*I)/4, -sqrt(1/4 + sqrt(3)*I/4))]
Tady je seznam stacionárních bodů:  [(-1, -1), (0, 0), (1, 1)]
Hessova matice v obecném bodě (x,y) je rovna matici:
[U+23A1]    2               [U+23A4]
[U+23A2]12·x  - 2     -2    [U+23A5]
[U+23A2]                    [U+23A5]
[U+23A2]                2   [U+23A5]
[U+23A3]   -2      12·y  - 2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (-1, -1) :
[U+23A1]10  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  10[U+23A6]
Funkce nabývá v bodě (-1, -1)
(ostré) lokální minimum -2
Tady je Hessova matice vyčíslená v bodě (0, 0) :
[U+23A1]-2  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  -2[U+23A6]
Postačující podmínky nejsou splněny a tudíž nelze
naši větu použít.
[U+23A1]-2  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  -2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (1, 1) :
[U+23A1]10  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  10[U+23A6]
Funkce nabývá v bodě (1, 1)
(ostré) lokální minimum -2
--------------------------------------------------------------------------------
```

```
# Příklad 3:
```

```
  File "<ipython-input-46-02b822a0bd99>", line 2
    Zadejte počet proměnných (1,2, nebo 3) : 3
                ^
SyntaxError: invalid syntax
```

# 9 Metoda nejm. čtverců, abs. extrémy

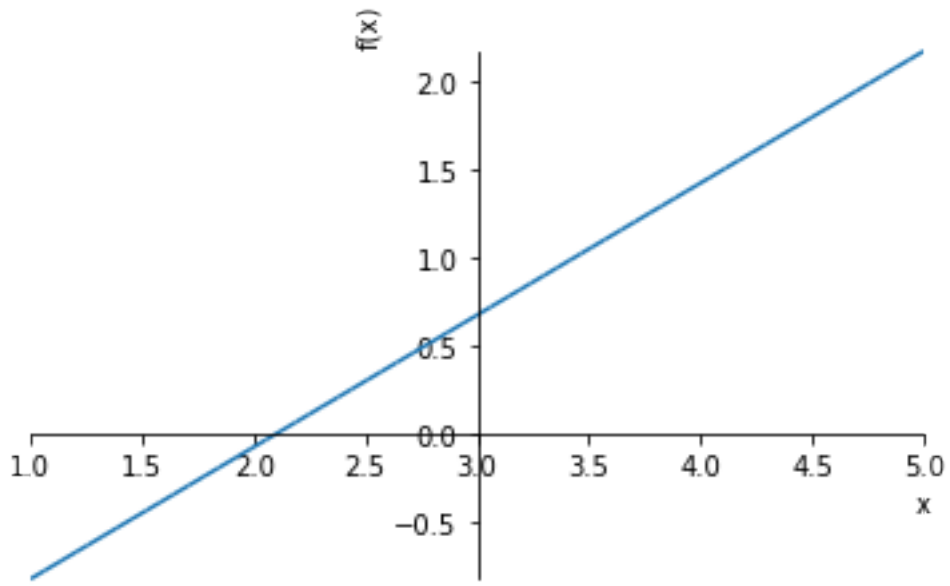
## 9.1 Metoda nejmenších čtverců

Představme si následující problém. Máme dánu množinu bodů v rovině mající souřadnice $[x_0, y_0], [x_1, y_1], \ldots, [x_n, y_n]$ Nyní hledejme lineární funkci $y = ax + b$ takovou, aby co nejlépe predikovala závislost mezi proměnnými $x$ a $y$. To představuje nalezení hodnot parametrů $a, b$. Použijme metodu tzv. nejmenších čtverců. Smysl metody spočívá v tom, že se snažíme minimalizovat účelovou funkci $f(a,b) = \sum_{i=0}^{n}[(ax_i + b) - y_i]^2$. Napišme program, který si načte souřadnice daných bodů a vrátí nám pokud možno optimální hodnoty paramerů $a, b$.

[3]:
```
# Řešení
import matplotlib.pyplot as plt
import numpy as np
from sympy import *
a,b,x,y,s,t = symbols('a, b, x, y, s, t')

def nacti_data():
    control = 'A'
    seznam_bodu = []  # zde jsme vytvořili dočasně tzv. prázdný seznam
    # Nyní bude následovat cyklus, kterým postupně vytvoříme seznam,
    # obsahující souřadnice daných bodů:
    while control == 'A' or control =='a':
        print('Je zapotřebí zadat alespoň dva body !')
        prvni_souradnice = float(input('Zadej první souřadnici: '))
        druha_souradnice = float(input('Zadej druhou souřadnici: '))
        seznam_bodu.append([prvni_souradnice, druha_souradnice])
        control = input('Chceš-li zadat souřadnice dalšího bodu, pak napiš A: ')
    return seznam_bodu

def najdi_stac_body(f):
    gradf = derive_by_array(f, [a,b])
    print('gradf = ', gradf)
    return list(nonlinsolve(gradf, [a,b]))



data = nacti_data()
```

```
print(data)
f = S(0)
for bod in data:
    f = f + ((a*bod[0] + b) - bod[1])**2
print('f(a,b) = ', f)
parametry = najdi_stac_body(f)
A = float(format(parametry[0][0], '.2f'))
B = float(format(parametry[0][1], '.2f'))
print('Seznam paramerů a,b : ')
print('a = ', A)
print('b = ', B)
L = S(A*x + B)
print('Optimální lineární funkcí je funkce: ')
print('y = ', L)
# nyní vytvoříme seznam x-ových souřadnic bodů:
xsouradnice = []
for bod in data:
    xsouradnice.append(bod[0])
minx = min(xsouradnice)
maxx = max(xsouradnice)
plot(L, (x, minx, maxx))
```

```
Je zapotřebí zadat alespoň dva body !
Zadej první souřadnici: 1
Zadej druhou souřadnici: 2
Chceš-li zadat souřadnice dalšího bodu, pak napiš A: A
Je zapotřebí zadat alespoň dva body !
Zadej první souřadnici: 3
Zadej druhou souřadnici: -5
Chceš-li zadat souřadnice dalšího bodu, pak napiš A: A
Je zapotřebí zadat alespoň dva body !
Zadej první souřadnici: 5
Zadej druhou souřadnici: 5
Chceš-li zadat souřadnice dalšího bodu, pak napiš A: N
[[1.0, 2.0], [3.0, -5.0], [5.0, 5.0]]
f(a,b) =  (1.0*a + b - 2.0)**2 + (3.0*a + b + 5.0)**2 + (5.0*a + b - 5.0)**2
gradf =  [70.0*a + 18.0*b - 24.0, 18.0*a + 6*b - 4.0]
Seznam paramerů a,b :
a =  0.75
b =  -1.58
Optimální lineární funkcí je funkce:
y =  0.75*x - 1.58
```




[3]:
```
<sympy.plotting.plot.Plot at 0x110f85b00>
```

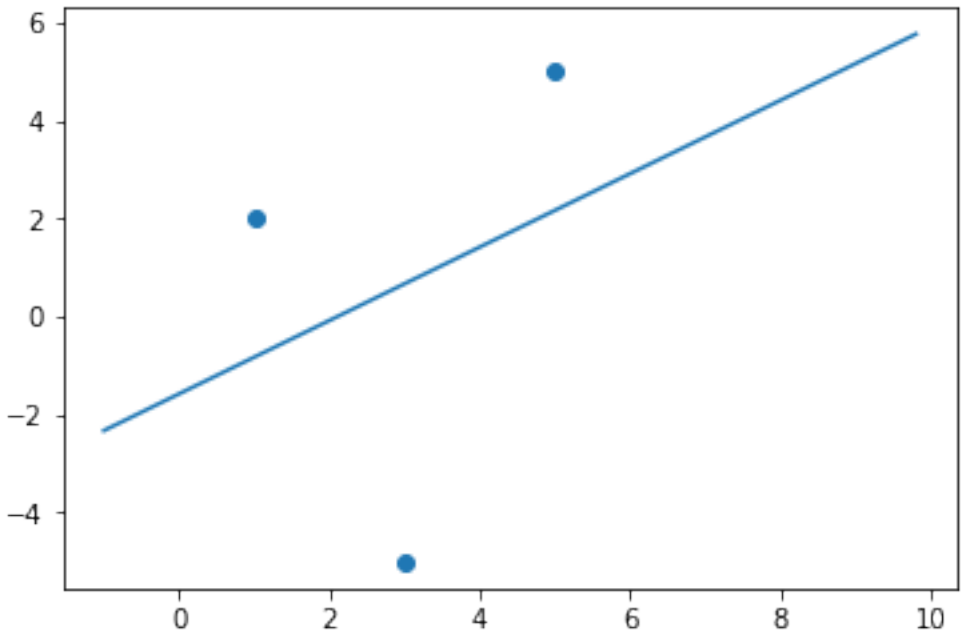
[10]:
```
f = (1.0*a + b - 2.0)**2 + (3.0*a + b + 5.0)**2 + (5.0*a + b - 5.0)**2
gradf = derive_by_array(f, [a,b])
hess_f = derive_by_array(gradf, [a,b])
hess_f
```

[10]:
```
[[70.0000000000000, 18.0000000000000], [18.0000000000000, 6]]
```

[15]:
```
A = Matrix([[70.0000000000000, 18.0000000000000], [18.0000000000000, 6]])
A.det()
```

[15]:
```
96.0000000000000
```

[4]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(-1, 10, 0.2)
y = 0.75*x - 1.58
# data = [[1, 2], [3, -1]]
xsouradniceBodu = [x[0] for x in data]
ysouradniceBodu = [y[1] for y in data]
fig, ax = plt.subplots()
plt.scatter(xsouradniceBodu, ysouradniceBodu)
ax.plot(x, y)
plt.show()
```




## 9.2 Nalezení absolutních extrémů

Připomeňme následující větu:

**Věta.** Mějme dánu spojitou funkci $f : \mathbb{R}^n \to \mathbb{R}$.

a) Jestliže $\lim_{\|x\|\to\infty} f(x) = +\infty$, potom pro libovolnou omezenou, uzavřenou množinu $S \subset \mathbb{R}^n$ existuje bod $x_0 \in absmin_S(f)$.\

b) Jestliže $\lim_{\|x\|\to\infty} f(x) = -\infty$, potom pro libovolnou omezenou, uzavřenou množinu $S \subset \mathbb{R}^n$ existuje $x_0' \in absmax_S(f)$. □

Zadání: vyšetřeme absolutní extrémy funkcí:

a) $x**4 + y**4 - (x+y)**2$;

b) $x**2 - y**2 - 4*x + 6*y$; (1.2.)

c) $3*x**2 + y**2 + 4*x*y - 8*x - 12*y$ (1.3.)

d) $x*y + 50/x + 20/y$

[9]:
```
from sympy import *
x, y, R, t = symbols('x, y, R, t')
# příklad a)
f = x**4 + y**4 - (x+y)**2
# příklad d)
#f = x*y + 50/x + 20/y
```

```
V = f.subs({x: R*cos(t), y: R*sin(t)})
print(V)
print(limit(V, R, oo ))
```

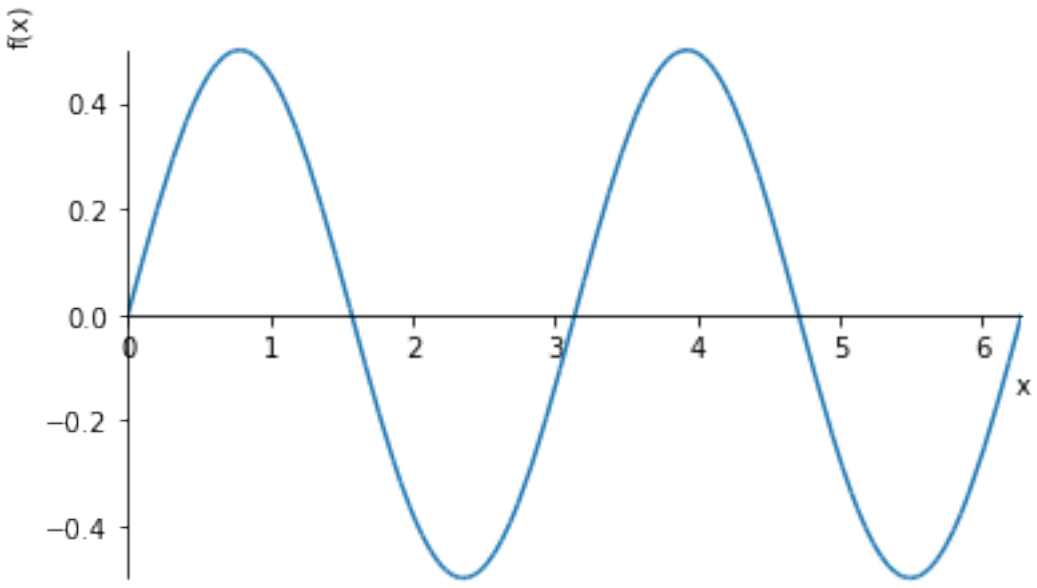
```
R**4*sin(t)**4 + R**4*cos(t)**4 - (R*sin(t) + R*cos(t))**2
oo*sign(sin(t)**4 + cos(t)**4)
```

[10]:
```
solve_univariate_inequality(sin(t)**4 + cos(t)**4 > 0, t, relational=False)
```

[10]: $\mathbb{R}$

[7]:
```
from sympy import *
t = symbols('x')
vyraz = sin(t)*cos(t)
print(vyraz.subs({t: 2.1}))
print(vyraz.subs({t: 2.1}) >= 0)
plot(vyraz, (t, 0, 2*pi))
```

```
-0.435787886206794
False
```




[7]: `<sympy.plotting.plot.Plot at 0x113151d30>`

a) Daná funkce má v nekonečnu limitu rovnu $+\infty$. Tedy $S_{max} = +\infty$. Na druhé straně funkce nabývá v jistém bodě své absolutní minimum.

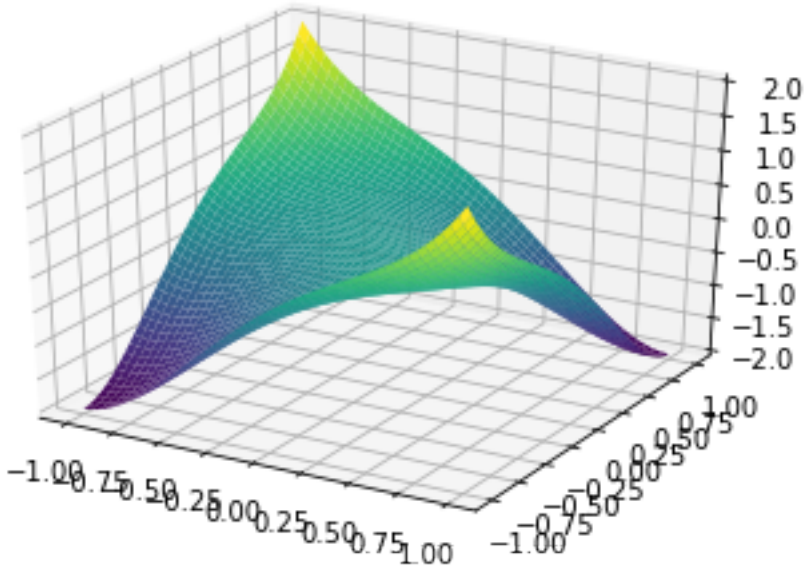
b) Daná funkce nemá v nekonečnu limitu a dále $S_{min} = -\infty$ a $S_{max} = \infty$.

[10]:
```
# Nyní určíme lokální extrémy:
# Napišme program, vyšetřující lokální extrémy funkce dvou proměnných.
from sympy import *
from sympy.plotting import  plot3d
x,y = symbols('x, y')
init_printing()
# Nyní definujme tzv.hlavní funkci, kde definujeme logiku celého programu. Tato␣
 ↪funkce pochopitelně může záviset
# na dalších speciálních funkcích, které jsou volány uvnitř hlavní funkce main()
def main():
    f = S(input('Zadejte předpis funkce: z='))
    print('Gradientem funkce f v bodě (x,y)', 'je vektor: ')
    df = derive_by_array(f, [x,y])
    print(df)
    solns = najdi_stac_body(f)
    print('Celkový počet kořenů: ', len(solns))
    print('Seznam všech kořenů soustavy: ', solns )
    real_solns = get_real_solns(solns)
    xsouradnice = []
    for bod in real_solns:
        xsouradnice.append(bod[0])
    minx = min(xsouradnice)
    maxx = max(xsouradnice)
    ysouradnice = []
    for bod in real_solns:
        ysouradnice.append(bod[1])
    miny = min(ysouradnice)
    maxy = max(ysouradnice)
    print('Tady je seznam stacionárních bodů: ', real_solns)
    print('Hessova matice v obecném bodě (x,y) je rovna matici: ')
    pprint(derive_by_array(df, [x,y]))
    for stac_bod in real_solns:
        print('Tady je Hessova matice vyčíslená v bodě', stac_bod, ': ')
        pprint(d2f(f, stac_bod[0], stac_bod[1]))
        if is_positive(d2f(f, stac_bod[0], stac_bod[1])):  # podmínka testuje␣
 ↪poziivní definitnost Hessovy matice d2f(stac_bod)
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální minimum', f.subs({x: stac_bod[0], y:␣
 ↪stac_bod[1]}))
        elif is_negative(d2f(f, stac_bod[0], stac_bod[1])):
            print('Funkce nabývá v bodě', stac_bod)
            print('(ostré) lokální minimum', f.subs({x: stac_bod[0], y:␣
 ↪stac_bod[1]}))
        else:
            print('Postačující podmínky nejsou splněny a tudíž nelze')
            print('naši větu použít.')
            pprint(d2f(f, stac_bod[0], stac_bod[1]))
```

```
    if len(real_solns) > 1:
        plot3d(f, (x, minx, maxx), (y, miny, maxy))
    else:
        plot3d(f, (x, minx - 1, maxx + 1), (y, miny - 1, maxy + 1))


def get_real_solns(sezn_korenu):
    s = []
    for koren in sezn_korenu:
        if koren[0].is_real and koren[1].is_real:
            s.append(koren)
    return s

def d2f(f,a,b):
    u = [x,y]
    gradf = derive_by_array(f, u)
    Hess = derive_by_array(gradf, u)
    return Hess.subs({x:a, y:b})


def najdi_stac_body(f):
    gradf = derive_by_array(f, [x,y])
    return list(nonlinsolve(gradf, [x,y]))

def is_positive(A):
    # Rozhodovací blok:
    if A[0,0] > 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:
        return True
    else:
        return False
def is_negative(A):
    if A[0,0] < 0 and A[0,0]*A[1,1] - A[0,1]*A[1,0] > 0:
        return True
    else:
        return False
main()
```

```
Zadejte předpis funkce: z=x**4 + y**4 - (x+y)**2
Gradientem funkce f v bodě (x,y) je vektor:
[4*x**3 - 2*x - 2*y, -2*x + 4*y**3 - 2*y]
Celkový počet kořenů:  7
Seznam všech kořenů soustavy:  [(-1, -1), (0, 0), (1, 1), ((-1 +
sqrt(3)*I)*sqrt(1 + sqrt(3)*I)/4, sqrt(1/4 + sqrt(3)*I/4)), (-sqrt(1 -
sqrt(3)*I)*(1 + sqrt(3)*I)/4, sqrt(1/4 - sqrt(3)*I/4)), (sqrt(1 - sqrt(3)*I)*(1
+ sqrt(3)*I)/4, -sqrt(1/4 - sqrt(3)*I/4)), ((1 - sqrt(3)*I)*sqrt(1 +
sqrt(3)*I)/4, -sqrt(1/4 + sqrt(3)*I/4))]
Tady je seznam stacionárních bodů:  [(-1, -1), (0, 0), (1, 1)]
```

```
Hessova matice v obecném bodě (x,y) je rovna matici:
[U+23A1]    2                [U+23A4]
[U+23A2]12·x  - 2     -2     [U+23A5]
[U+23A2]                     [U+23A5]
[U+23A2]                2    [U+23A5]
[U+23A3]   -2       12·y  - 2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (-1, -1) :
[U+23A1]10  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  10[U+23A6]
Funkce nabývá v bodě (-1, -1)
(ostré) lokální minimum -2
Tady je Hessova matice vyčíslená v bodě (0, 0) :
[U+23A1]-2  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  -2[U+23A6]
Postačující podmínky nejsou splněny a tudíž nelze
naši větu použít.
[U+23A1]-2  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  -2[U+23A6]
Tady je Hessova matice vyčíslená v bodě (1, 1) :
[U+23A1]10  -2[U+23A4]
[U+23A2]      [U+23A5]
[U+23A3]-2  10[U+23A6]
Funkce nabývá v bodě (1, 1)
(ostré) lokální minimum -2
```



Závěr :

a) funkce $z = x**4 + y**4 - (x+y)**2$ nabývá v bodech $(-1,-1)$ a $(1,1)$ své absolutní minimum $S_{min} = -2$. Absolutní maximum funkce nenabývá vzhledem k tomu, že limita v nekonečnu je rovna $\infty$. Tedy $S_{max} = \infty$.

b) funkce $z = x**2 - y**2 - 4*x + 6*y$ nemá v bodě $(2,3)$ lokální extrém a absolutní extrémy též nenabývá, $S_{min} = -\infty$ a $S_{max} = \infty$.

c) $(8,-10) \notin locextr(f)$ a $S_{min} = -\infty$ a $S_{max} = \infty$.

d) Funkce nabývá v bodě (5, 2) (ostré) lokální minimum 30 a $S_{min} = -\infty$ a $S_{max} = \infty$.

[ ]: