VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ŘEŠENÍ NĚKTERÝCH ÚLOH PRUŽNOSTI POMOCÍ PROGRAMOVACÍHO JAZYKA PYTHON

ELASTICITY PROBLEMS IN PYTHON LANGUAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Dušan Tichoň**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Tomáš Profant, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Dušan Tichoň**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**doc. Ing. Tomáš Profant, Ph.D.**
Akademický rok:	2017/18

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Řešení některých úloh pružnosti pomocí programovacího jazyka Python

Stručná charakteristika problematiky úkolu:

Současnou dobu lze charakterizovat slovem svoboda. Součástí společenských procesů související s tímto fenoménem je přístup k tzv. svobodnému softwaru. Součástí ohromného balíku tohoto typu softwaru je i velice výkonný dynamický programovací jazyk Python. Python není primárně určen k výpočtům, avšak jeho otevřenost a práce s knihovnami umožňuje jeho velice efektivní přeorientování se na výkonný nástroj v oblasti numerických i symbolických výpočtů. Uchazeč se seznámí s knihovnami Numpy, Sympy a Matplotlib a jejich solidní výpočetní potenciál využije k řešení některých úloh pružnosti.

Cíle bakalářské práce:

1. Nastudování a seznámení se s programovacím jazykem Python.
2. Aplikace programovacího jazyku Python na některé úlohy pružnosti a pevnosti I, resp. II.
3. Vytvoření prezentací úloh a výsledků ve webové aplikaci Jupyter.

Seznam doporučené literatury:

BRAND, L., Vector and Tensor Analysis, John Willey & Sons, London, 1947

JANÍČEK, P., ONDRÁČEK, E., VRBKA, J., Mechanika těles. Pružnost a pevnost I. VUT v Brně, 1992.

ONDRÁČEK, E., VRBKA, J., JANÍČEK, P., Mechanika těles. Pružnost a pevnost II. VUT v Brně, 1991.

FLORIAN, Z., ONDRÁČEK, E., PŘIKRYL, K., Mechanika těles. Statika. VUT v Brně, 1992.

JANÍČEK, P., FLORIAN, Z., Mechanika těles. Úlohy z pružnosti a pevnosti I. VUT v Brně, 1990.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2017/18

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Táto bakalárska práca sa zaoberá programovacím jazykom Python. Cieľom práce je oboznámenie sa a aplikácia nástrojov knižníc NumPy, SymPy a Matplotlib, ako aj využitie solídneho výpočtového potenciálu programovacieho jazyka k vytvoreniu názorných úloh, ktoré budú ďalej prezentované a využívané pre výuku.

Abstract

This bachelor thesis deals with the programming language Python. Its aim is acquaintance of NumPy, SymPy and Matplotlib libraries and their later application. Also it shows the great computing potential of programming language in creating exercises which will be used and presented for educational purposes.

Kľúčové slová

Python, Sympy, Numpy, Matplotlib, Jupyter, riešenie úloh pružnosti v jazyku Python, spustiteľné aplikácie

Keywords

Python, Sympy, Numpy, Matplotlib, Jupyter, elasticity problems in Python, executable applications

BIBLOGRAFICKÁ CITÁCIA

TICHOŇ, D. *Řešení některých úloh pružnosti pomocí programovacího jazyka Python.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 36 s. Vedoucí bakalářské práce doc. Ing. Tomáš Profant, Ph.D..

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že som bakalársku prácu na tému „ *Řešení některých úloh pružnosti pomocí programovacího jazyka Python“* vypracoval sám s použitím odbornej literatúry uvedenej v práci a pod odborným vedením vedúceho práce.

V Brne 25. mája 2018 .....

Dušan Tichoň

POĎAKOVANIE

Na úvod by som chcel predovšetkým poďakovať vedúcemu mojej bakalárskej práce pánovi doc. Ing.Tomášovi Profantovi, Ph.D. za ochotu, znalosti a pripomienky, ktoré mi predal a čas, ktorý mi počas riešenia tejto práce venoval. Ďalej by som rád poďakoval svojej rodine a blízkym, ktorý ma počas celého štúdia podporovali.

Obsah

1 Úvod

Dnešný priemysel neúprosne napreduje a vyžaduje neustálu tvorbu a inováciu najrôznejších aplikácií. Súčasné pomery technologickej úrovne a náročnosti daných procesov sa nezaobídu bez implementácie programovacích jazykov pre inžinierske potreby. V tejto dobe preto získavajú interpretované jazyky čím ďalej tým väčšiu popularitu.

Programovacie jazyky môžeme najjednoduchšie rozdeliť na objektovo orientované, procedurálne, logické a funkcionálne. Ďalším významným fenoménom je cenová relácia a dostupnosť ich jednotlivých knižníc a licencií. Keďže filozofia modernej doby je úzko spätá s minimalizáciou prevádzkových nákladov, do popredia sa dostávajú takzvané slobodné softvére. Súčasťou širokej disponibilnej škály týchto softvérov je aj vysokovýkonný viacúrovňový dynamický jazyk Python, ktorý nie je primárne určený pre výpočty, avšak vďaka jeho všestrannosti a prepracovanej, no jednoduchej koncepcii využívania knižníc sa stáva zaujímavým nástrojom v oblasti symbolických a numerických výpočtov. Je veľmi ľahko šíriteľný v rôznych verziách a podporovaný na rôznych platformách ako Windows či Linux. Najvariabilnejšími knižnicami využívanými v praxi sú NumPy, SymPy, SciPy a Matplotlib. Oproti ostatným jazykom je imúnnejší voči chybovým hláseniam spôsobených chýbajúcimi zátvorkami či bodkočiarkami. Vďaka menšiemu dôrazu na syntax je dnes mnohými považovaný za najjednoduchší jazyk ideálny pre začiatočníkov.

V nasledujúcich kapitolách budú rozobraté základné princípy a výhody využívania jednotlivých knižníc doložené praktickými výstupmi vytvorenými v prostredí Jupyter. Ďalej sa budeme zaoberať samotnou aplikáciou Jupyter a jej možnosťami efektívnej práce s príkazmi a vykresľovaním výsledkov, konštelációu základných teoretických pilierov nutných pre správny výpočet, prostredníctvom ktorých nakoniec budeme prezentovať samotné riešené úlohy pružnosti vytvorené pre výuku.

2 Python

2.1 Charakteristika Pythonu

Python je interpretovaný, objektovo orientovaný, vysokoúrovňový, dynamický programovací jazyk. Vďaka svojim vysokoúrovňovým dátovým štruktúram je veľmi atraktívnym pre RAD[1] (Rapid Application Development), ale aj ako skriptovací jazyk, či jazyk spájajúci rôzne komponenty vytvorené dokonca pomocou rôznych jazykov. [1]

Filozofia Pythonu kladie dôraz na čitateľnosť kódu a na syntax, ktorá umožňuje programátorom vyjadriť dané pojmy čo najjednoduchšou formou, a teda s použitím čo najmenej kódu v príkazových riadkoch. Python má dynamický typ systému a správu automatickej pamäte. Verzie Pythonu sú k dispozícii pre mnoho operačných systémov. CPython[2], referenčná implementácia Pythonu, je open - source softvér a komunitný model, rovnako ako takmer všetky jeho varianty. [2] CPython spravuje nezisková Python Software Foundation. [1]

Obrázok 1 Logo spoločnosti Python

2.2 História

Python bol koncipovaný v neskorých 80. rokoch 20. storočia a jeho implementácia začala v decembri roku 1989, o ktorú sa postaral Guido van Rossum v Centre Wiskunde & Informatica (CWI) v Holandsku, ako nástupca jazyka ABC. Van Rossum ostáva hlavným autorom Pythonu. Keďže je zarytým fanúšikom britského komediálneho seriálu *„Monty Python's Flying Circus"*, bez okolkov asocioval daný názov s názvom programovacieho jazyka. [1]

Python 2.0 bol vydaný 16. októbra 2000 s novými vlastnosťami ako napríklad podporu pre Unicode.

Python 3.0 bol vydaný 3. decembra 2008 po dlhšom skúšobnom období. Táto verzia nie je spätne kompatibilná s ostatnými, avšak mnohé z jeho hlavných funkcií boli implementované do starších verzií 2.6.x a 2.7.x. Toto vydanie Pythonu 3 obsahuje funkciu *2to3*, ktorá automaticky prekladá kódy z Pythonu 2 do Pythonu 3. [3]

[1] Filozofia moderného prístupu pre vývoj aplikácií, umožňujúca testovanie prototypov a následnú analýzu spätnej väzby už v ranných fázach vývoja.
[2] Referenčná implementácia programovacieho jazyka Python napísaná v jazyku C s rozhraním funkcií ďalších jazykov ako Java, Haskell atď.

2.3 Hlavné črty

Python je dynamický interpretovaný jazyk a často býva považovaný za jazyk skriptovací, avšak jeho využiteľnosť má oveľa širšie spektrum. K význačným vlastnostiam jazyka Python patrí jednoduchosť z hľadiska učenia sa. Býva považovaný za jeden z najvhodnejších jazykov pre začiatočníkov a to hlavne kvôli tomu, že bol skoncipovaný ako nástupca jazyka ABC, ktorý bol pre začiatočníkov priamo vytvorený. [2]

Zaujímavým ostáva fakt, že Python búra predstavu o tom, že programovací jazyk určený pre výuku nie je využiteľný v praxi. Veľkou mierou k tomu prispieva jednoduchosť syntaxe, ktorá zastáva významné miesto vo filozofii Pythonu. Na definíciu blokov sa používa iba odsadenie (na rozdiel od väčšiny jazykov). [4]

Ďalšou významnou vlastnosťou je produktívnosť z hľadiska písania textu. Pri jednoduchších programoch sa to prejavuje stručnosťou zápisu, pri zložitejších napríklad štandardne dodávané prostriedky pre písanie testov, ktoré slúžia na kontrolovanie funkčnosti menších častí kódu. Výraznou mierou k produktívnosti prispieva dostupnosť a ľahký prístup k použitiu knižných modulov, ktoré umožňujú ľahké riešenie úloh z rôznych oblastí.

2.4 Premenné

Do premenných ukladáme čísla, reťazce a ďalšie dátové štruktúry. Premenné sa pritom nedeklarujú ale typ premennej je určený automaticky, sú brané ako odkazy na objekt, vďaka čomu ich môžeme používať vo funkciách a aplikovať na ne napríklad rôzne výpočtové metódy. Premenná uchováva nejakú hodnotu dátového typu. Meno premennej môže obsahovať písmená, čísla a podčiarkovník. [2] Python rozlišuje medzi malými a veľkými písmenami a nám neostáva nič iné, ako to rešpektovať. Názvy premenných by tiež nemali byť zhodné s názvami, ktoré Python používa pre svoje účely (napríklad acces, and, break, if, elseif, return, while, for atď.).

2.5 Sekvenčné dátové typy

Sekvenčné dátové typy sú reťazce, tuple a polia. Reťazce a tuple sú nemenné, prvky polí sa zmeniť dajú. Ku sekvenčným dátovým typom máme prístup pomocou indexov. V Pythone nemusíme pristupovať k jednotlivým prvkom len jednotlivo ale ide to aj hromadne prostredníctvom slice notácie, a teda zápisu dvoch čísel oddelených dvojbodkou, pričom prvé číslo hovorí, u ktorého prvku zoznamu sa má začať a to druhé, pochopiteľne, skončiť. Číslovanie sa bez zadania štartu začína od nuly a je možné používať aj záporné indexy. [4]

Reťazce majú obsah premennej ohraničený úvodzovkami alebo apostrofami. Keď chceme vložiť do reťazca nejaké "divné" znaky, tak použijeme sekvenciu escape (\).

```
In [3]: 'Milan\'s car'
Out[3]: "Milan's car"
```

Polia vytvoríme priradením hodnôt v hranatých zátvorkách nejakej premennej. S jednotlivými prvkami poľa sa dá manipulovať, teda meniť ich a pridávať nové prvky. Ak priraďujeme menej hodnôt ako je prvkov, na ktoré ukazujeme, budú prebytočné prvky zrušené.

Tuple je kombinácia poľa a reťazca. Tuple je nemenné. Vytvoríme ho ako pole, ale jednotlivé prvky, ktoré ho tvoria môžu aj nemusia byť ohraničené v okrúhlych zátvorkách. [4]

3 Knižnica NumPy

3.1 Charakteristika knižnice NumPy

NumPy je základná knižnica pre vedecké výpočty v Pythone, primárne určená na prácu s numerickými dátami a to konkrétne s 1- až n- rozmernými maticami. Poskytuje pre operácie s nimi široký sortiment nástrojov, konkrétne matematické, logické a tvarové manipulácie, triedenie, diskrétna Fourierova transformácia, základná lineárna algebra, základné štatistické operácie, náhodné simulácie a oveľa viac. Jadrom balíka NumPy je objekt *ndarray*, čo reprezentuje spomínané n-rozmerné pole. [5]

Použiteľnosť knižnice najlepšie charakterizujú rozdiely medzi NumPy a základnými zoznamami z Pythonu :

- Polia NumPy majú pri vytváraní pevnú veľkosť, na rozdiel od zoznamov[3] Pythonu, ktoré môžu dynamicky rásť. Zmena veľkosti *ndarray* vytvorí nové pole a odstrániť pôvodné.
- Je požadované, aby prvky v poli NumPy mali rovnaký typ údajov, a teda boli rovnakej veľkosti v pamäti.
- Polia NumPy uľahčujú rozšírené matematické a iné typy operácií na veľké množstvo údajov. Tieto operácie sa vykonávajú efektívnejšie vďaka nutnosti použitia menšieho počtu kódu ako v prípade Pythonovských zoznamov.
- Rastúci počet dnešných balíkov a softvérov založených na Pythone funguje pomocou polí NumPy. Zvyčajne podporujú vstupy zo zoznamov Pythonu a prevádzajú ich na vstupy NumPy ešte pred spracovaním, a teda často aj vracajú výstupy vo formáte NumPy. Inými slovami, aby sa efektívne využívala veľká časť z dnešných vedecko-matematických softvérov na báze Pythonu nestačí vedieť používať Python, ale je dôležité vedieť používať aj NumPy. [6]

3.1.1 Importovanie NumPy

Pokiaľ chceme používať výhody knižnice NumPy, je samozrejme potrebné modul importovať. Pre interaktívne použitie je niekedy vhodné importovať všetko naraz pomocou:

```
In [1]: from numpy import *
```

Inak sa obvykle používa:

```
In [2]: import numpy as np
```

Rovnako postupujeme aj v prípade importovania ostatných knižníc potrebných k výpočtom.

[3] Interný dátový typ obsahujúci akékoľvek čísla či slovníky, vytvorený hranatými zátvorkami a indexmi.

3.2 Základné operácie

Základnou vlastnosťou polí NumPy je to, že aritmetické operácie so skalárnymi hodnotami ako sú napríklad čísla fungujú po prvkoch.

```
In [1]: import numpy as np

In [4]: matica = np.array([[8,34,73], [13,27,49], [82,61,18]])

In [7]: print (matica)

[[ 8 34 73]
 [13 27 49]
 [82 61 18]]

In [9]: matica-3

Out[9]: array([[ 5, 31, 70],
               [10, 24, 46],
               [79, 58, 15]])

In [10]: - ( matica / 4)

Out[10]: array([[ -2.  ,  -8.5 , -18.25],
                [ -3.25,  -6.75, -12.25],
                [-20.5 , -15.25,  -4.5 ]])

In [11]: matica > 20

Out[11]: array([[False,  True,  True],
                [False,  True,  True],
                [ True,  True, False]], dtype=bool)
```

Okrem aritmetických operácií takto funguje aj napríklad porovnávanie. Nasledujúcim výrazom dostaneme pravdivostnú tabuľku, kde pre *True* platí daná podmienka.

```
In [3]: (matica < 50) & (matica > 20)

Out[3]: array([[False,  True, False],
               [False,  True,  True],
               [False, False, False]], dtype=bool)
```

Keďže Python neumožňuje predefinovať chovanie operátorov and a or, logické spojenie operácií sa tradične vykonáva pomocou bitových operátorov & (a) a | (alebo). Kvôli ich neintuitívnej priorite sa hodí uzavierať výrazy do zátvoriek. [6]

Operácie s inými poľami pracujú taktiež po prvkoch. Obe však musia byť rovnako veľké.

```
In [4]: matica + matica

Out[4]: array([[ 16,  68, 146],
               [ 26,  54,  98],
               [164, 122,  36]])

In [5]: matica * np.array([[0,1,2], [1,0,2], [2,1,0]])

Out[5]: array([[  0,  34, 146],
               [ 13,   0,  98],
               [164,  61,   0]])
```

3.2.1 Tvorenie matíc

Často využívané typy matíc sa dajú vytvoriť pomocou vstavaných funkcií v knižnici NumPy. S výsledkami môžeme potom priamo pracovať, alebo môžeme tieto matice vyplniť pomocou vypočítaných dát. [7]

```
In [7]: np.zeros((4,4))

Out[7]: array([[ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.]])

In [9]: np.ones((4,4))

Out[9]: array([[ 1.,  1.,  1.,  1.],
               [ 1.,  1.,  1.,  1.],
               [ 1.,  1.,  1.,  1.],
               [ 1.,  1.,  1.,  1.]])

In [11]: np.full((4,4), 10.5)

Out[11]: array([[ 10.5,  10.5,  10.5,  10.5],
                [ 10.5,  10.5,  10.5,  10.5],
                [ 10.5,  10.5,  10.5,  10.5],
                [ 10.5,  10.5,  10.5,  10.5]])

In [12]: np.eye((4))

Out[12]: array([[ 1.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  1.]])

In [13]: np.diag((10,9,8,7))

Out[13]: array([[10,  0,  0,  0],
                [ 0,  9,  0,  0],
                [ 0,  0,  8,  0],
                [ 0,  0,  0,  7]])
```

4 Knižnica Matplotlib

4.1 Charakteristika knižnice Matplotib

Knižnica Matplotlib je určená k vytváraniu 2D grafov polí z Pythonu. Aj keď má svoj pôvod v emulácii príkazov v prostredí MATLAB®, je od tohto prostredia nezávislá a je používaná objektovo orientovaným spôsobom používajúc jazyk Python v plnom rozsahu. Aj keď je Matplotlib napísaný v čistom Pythone, vo veľkom využíva knižnicu NumPy a ďalšie rozšírenia pre čo najlepšiu interpretáciu aj veľmi rozsiahlych polí. Je navrhovaný s myšlienkou, aby bol schopný generovať a vytvárať grafy aj po zadaní jedného alebo veľmi malého počtu príkazov. [8]

Hlavné výhody používania Matplotlib:

- Riešenie Matplotlib je multiplatformné, obsah sa dá vytvárať a upravovať na Linuxe, Windowse alebo OS X
- Syntax je veľmi jednoduchá, takmer na úrovni kancelárskych balíkov (excel)
- Široký výber ukladania výstupov do požadovaného formátu (*.JPG, *.PNG, *.RAW, *.EPS, *.PDF, *.SVG)
- Rýchlosť, flexibilita, automatizácia, open- source [8]

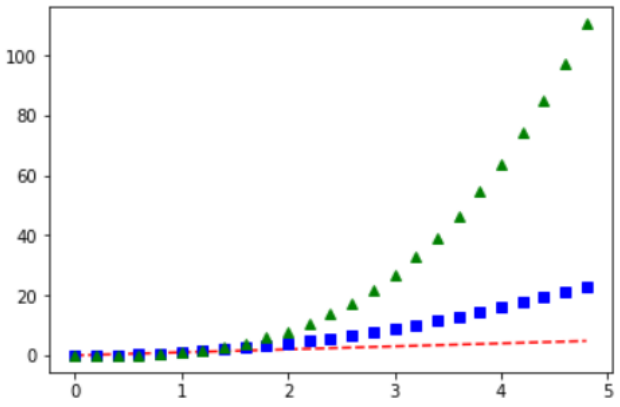
4.2 Grafy v Matplotlibe

Vynášať sa dajú body, stĺpce, kruhové výseky (kruhový graf), symboly reprezentujúce viac bodov (napr. krabicový graf), početnosť javov (histogram), krivky prechádzajúce danými bodmi, šípky zachycujúce priebeh siločiar... [9] Vymedzenie intenzity a kvality javu sa prevádza symbolom, veľkosťou symbolu, farbou, orientáciou šípky... Do jedného výstupu sa dajú naskladať rôzne grafy. V skratke Matplotlib sa vie vysporiadať s rôznymi problémami technickej praxe.

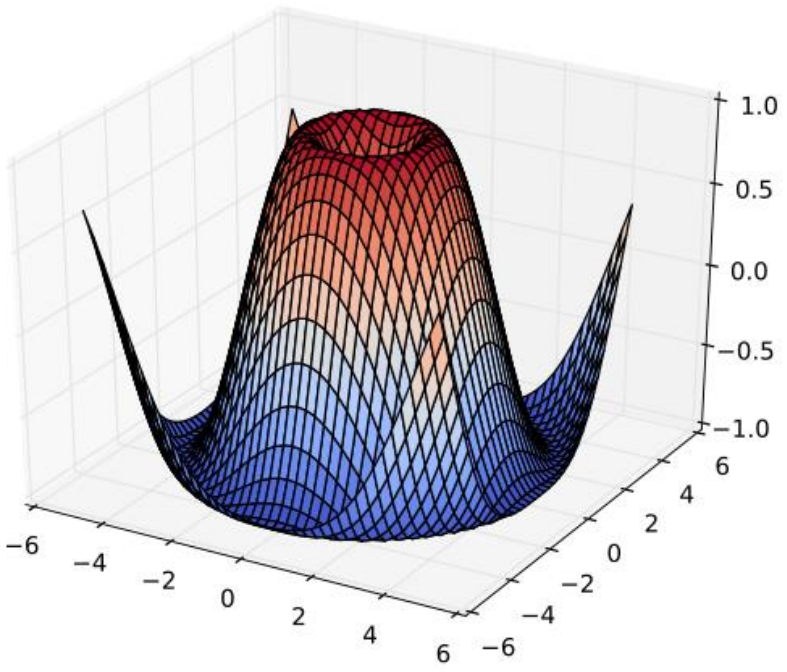
```
In [2]: import numpy as np
        import matplotlib.pyplot as plt

        t = np.arange(0., 5., 0.2)

        plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
        plt.show()
```



Rozšírenie Mplot3d prináša do sveta Matplotlibu tretí rozmer podporením osí, ktoré vytvárajú 2D projekciu 3D scény. Interaktívne backendy[4] umožňujú otáčanie (kliknutie ľavým tlačidlom a ťahaním scény) a približovanie - zoom (kliknutie pravým tlačidlom a pohybom nahor a nadol). [8]



Obrázok 2 3D graf vytvorený pomocou knižnice Matplotlib

[4] Skrytá časť aplikácie, ktorá spolupracuje so serverom a nezaťažuje sieť s primárnu úlohou zrýchliť a zjednodušiť prácu s výpočtami a výsledkami.

5 Knižnica SymPy

5.1 Charakteristika knižnice SymPy

SymPy je knižnica jazyka Python pre symbolickú matematiku. Jej cieľom je stať sa kompletným počítačovým algebrickým systémom[5] (CAS) a zároveň byť čo najjednoduchším systémom, aby bola SymPy jednoducho pochopiteľná a ľahko rozšíriteľná. SymPy je napísaná kompletne v jazyku Python a nepotrebuje žiadne ďalšie externé knižnice. [10]

Symbolické premenné, tzv. symboly, sa musia definovať a byť priradené do premenných v Pythone predtým, než môžu byť použité. To sa zvyčajne vykonáva prostredníctvom funkcie *symbols*, ktoré môžu vytvárať viaceré symboly za použitia jedného príkazu.

Nasledujúci príklad vytvára tri symboly predstavujúce premenné s názvom x, y a z.

```
In [1]: import sympy as sp

In [2]: x,y,z = sp.symbols('x y z')
```

V tomto konkrétnom prípade sú všetky symboly priradené premenným rovnakého mena. Avšak užívateľ má možnosť priradiť im rôzne premenné v Pythone, a zároveň by zastupovali rovnaký symbol, napríklad:

```
In [3]: a,b,c = sp.symbols ('x y z')
```

Takto sa dajú vytvárať výrazy zo syntaxe matematických symbolov Pythonu. Dané výrazy však zostávajú výrazmi a teda nie sú vyčísľované.

```
In [4]: (a**2-7*a+9)/b/c
```

Out[4]: $\frac{1}{yz}\left(x^2 - 7x + 9\right)$

5.2 Predpoklady systému

Predpoklady systému umožňujú používateľom určiť, že symboly majú určité spoločné matematické vlastnosti, ako pozitívne, imaginárne alebo celé čísla. SymPy nikdy nevytvára zjednodušenia, pokiaľ to nedovoľujú predpoklady. [10]

Napríklad zjednodušenie $\sqrt{t^2} = t$ je zadržané, pokiaľ nie je definované, že premenná t je nezáporná ($t \geq 0$). SymPy sa snaží zabrániť matematicky neplatným operáciám a vždy počíta s najviac generalizovaným predpokladom pri výpočtoch.

```
In [5]: t = sp.symbols('t')
        sp.sqrt(t**2)
```

Out[5]: $\sqrt{t^2}$

[5] Computer algebra system – systém pre počítačové spracovanie symbolických matematických výrazov.

V mnohých prípadoch je žiadané zjednodušiť výrazy obsahujúce pojmy ako $\sqrt{t^2}$. Predpoklady sú nastavené na symboly objektov po ich vytvorení. Napríklad symbol (*'t ', positive = True*) vytvorí symbol t, ktorý predpokladáme ako kladný.

```
In [6]: t = sp.symbols('t', positive = True)
        sp.sqrt(t**2)
```

Out[6]: t

Zjednodušenie výrazov sa dá docieliť aj pomocou funkcie *simplify*. Je dôležité si uvedomiť, že zjednodušenie nie je definovaná matematická operácia. [11]Aplikovaním tejto funkcie zjednodušíme výrazy pomocou viacerých elementárnejších zjednodušení. Niekedy je však vhodnejšie upriamenie sa na konkrétne funkcie pre docielenie efektívnejšieho zjednodušenia.

Najbežnejšie funkcie pre zjednodušenie sú [11]:

- *expand* - rozšírenie výrazu (roznásobenie, umocnenie...)

```
In [7]: y = sp.symbols('y')
        sp.expand((y+3)**2)
```

Out[7]: $y^2 + 6y + 9$

- *factor* - zoberie polynóm a pretransformuje ho na nedeliteľný výraz

```
In [8]: x = sp.symbols('x')
        sp.factor(x**3-x**2+x-1)
```

Out[8]: $(x - 1)\left(x^2 + 1\right)$

- *collect* - zhromažďuje spoločné premenné vo výraze, nutné upresnenie, ktorú premennú má vnímať

```
In [9]: p,r,s = sp.symbols('p r s')
        vyraz = r*p**2 + p - 3 + 2*p**3 - r*s*p**2 + p**4
        vyraz
```

Out[9]: $p^4 + 2p^3 - p^2rs + p^2r + p - 3$

```
In [10]: collect_vyraz = sp.collect(vyraz,p)
         collect_vyraz
```

Out[10]: $p^4 + 2p^3 + p^2\left(-rs + r\right) + p - 3$

- *cancel* – upraví výrazy na kanonickú formu, na spoločného menovateľa

```
In [11]: u = sp.symbols('u')
         vyraz2 = 5/u**2 + (3*u/2 - 2)/(u**2 - 4)
         vyraz2
```

Out[11]: $$\frac{\frac{3u}{2} - 2}{u^2 - 4} + \frac{5}{u^2}$$

```
In [12]: sp.cancel(vyraz2)
```

Out[12]: $$\frac{3u^3 + 6u^2 - 40}{2u^4 - 8u^2}$$

- *trigsimp* - zjednodušenie výrazov s trigonometrickými funkciami

```
In [13]: sp.trigsimp(sin(x)**4 - 2*cos(x)**2*sin(x)**2 + cos(x)**4)
```

Out[13]: $$\frac{1}{2}\cos(4x) + \frac{1}{2}$$

5.3 Kalkulus

SymPy poskytuje všetky základné výpočtové operácie ako počítanie limít, derivácií, integrálov a sumácií. Tieto funkcie budeme využívať pri riešení úloh pružnosti a pevnosti.

Výpočet limít sa realizuje pomocou príkazu *limit*, používajúc Gruntzov algoritmus (Gruntz,1996). Nekonečno v Pythone je značené dvoma znakmi "o". [10]

```
In [14]: sp.limit((9 - x**2)/(sp.sqrt(3*x) - 3), x , 3)
```

Out[14]: -12

Príkaz *diff* nám slúži na počítanie derivácií všeobecne n-tého rádu, ktorý treba špecifikovať. [10]

```
In [15]: sp.diff(cos(x)*x**8, x, x)
```

Out[15]: $$x^6\left(-x^2\cos(x) - 16x\sin(x) + 56\cos(x)\right)$$

Integrály sú počítané funkciou *integrate*, ktorá kombinuje hneď niekoľko algoritmov (Bronstein-2005,Roach-1996/7) pre umožnenie počítania čo najširšieho spektra určitých a neurčitých integrálov. [10]

```
In [16]: sp.integrate((x**2 + 5*x - 8*x**3)/(5 + x**2)**2, x)
```

Out[16]: $$-\frac{x + 45}{2x^2 + 10} - 4\log\left(x^2 + 5\right) + \frac{\sqrt{5}}{10}\operatorname{atan}\left(\frac{\sqrt{5}x}{5}\right)$$

6 Jupyter Notebook

6.1 Čo je to Jupyter Notebook?

Notebook rozširuje konzolovo založený prístup k interaktívnym výpočtom a dátovým analýzam v novom kvalitatívnom smere, poskytuje webovú aplikáciu vhodnú pre celý výpočet a proces snímania: rozvoj, dokumentáciu a vykonávanie kódu, ako aj jednoduché prezentovanie výsledkov. Jupyter Notebook kombinuje dve zložky:

- webová aplikácia: nástroj v podobe prehliadača pre interaktívne kompletizovanie dokumentov, ktoré spájajú vlastný popisujúci text, matematiku, výpočty a ich bohatý mediálny výstup.
- Notebook dokumenty: zobrazenie všetkého obsahu viditeľného vo webovej aplikácii, vrátane vstupných a výstupných výpočtov, popisujúceho textu, matematiky, obrázkov, a bohatej mediálnej prezentácie objektov. [12]

Obrázok 3 Logo spoločnosti Jupyter

Hlavné vlastnosti webovej aplikácie:

- Editácia kódu priamo v prehliadači, automatické zvýrazňovanie syntaxe, odsadzovanie...
- Možnosť spustiť kód z prehliadača s výsledkami výpočtov pripojených do kódu, ktorý ich vytvoril.
- Zobrazenie výsledkov výpočtu pomocou multimediálnych reprezentácií, ako HTML, LaTeX, PNG, SVG... Uverejnenie údajov poskytnutých pomocou knižnice matplotlib môže byť napríklad zahrnuté bez zdržania za sebou . [12]

6.2 Jupyter Notebook Application

Aplikácia Jupyter Notebook je klientova serverová aplikácia, ktorá umožňuje editáciu a spúšťanie dokumentov prostredníctvom webového prehliadača. Sú dva spôsoby ako sa dostať do aplikácie Jupyter. Jedna alternatíva je, že môže byť nainštalovaná a spustená na miestnej pracovnej ploche bez požiadavku internetového pripojenia alebo môže byť nainštalovaná na vzdialenom serveri a prístupná prostredníctvom internetu. [13]

Kým Jupyter dokáže spustiť kódy v mnohých jazykoch, požiadavkou pre inštaláciu Jupyter Notebooku je Python (verzia 3.3 a vyššia alebo 2.7). Odporúčané je používať distribúciu Anaconda a nainštalovať Python aj Jupyter spolu s ďalšími balíkmi vhodnými pre vedecké výpočty.

6.2.1 Notebook dokument

Notebook dokumenty sú súbory vytvorené aplikáciou Jupyter Notebook App, ktoré obsahujú počítačový kód (napr. Python) a textové prvky (rovnice, obrázky, odkazy, atď.). Notebook dokumenty sú buďto v podobe dokumentu na čítanie, obsahujúc popis analýzy a jej výsledky (obrázky, tabuľky, atď.), ako aj vo forme spustiteľného dokumentu, ktorý je možné spustiť v prípade potreby vykonania analýzy dát. Tieto dokumenty sú vnútorne JSON[6] súbory a sú uložené vo formáte *.ipynb*. Keďže JSON je vo forme prostého textu, je ľahko šíriteľný medzi kolegami. [12]

Súbory môžu byť exportované do rôznych formátov vrátane HTML, LaTex, PDF a prezentácií pomocou príkazu *nbconvert*.

Okrem toho akýkoľvek dokument vo formáte *.ipynb*, ktorý je dostupný z verejnej adresy URL, môže byť zdieľaný cez Jupyter Notebook Viewer (nbviewer). Táto služba dokáže daný dokument načítať z URL adresy a vykresliť ho ako statickú webovú stránku. Výsledky sa teda ľahko kolegiálne zdieľajú alebo sa zdieľajú ako verejný blog, bez toho, aby ostatní používatelia museli spúšťať alebo dokonca inštalovať aplikáciu Jupyter Notebook.

6.2.2 Nový poznámkový blok

Nový poznámkový blok je možné vytvoriť kedykoľvek buď z panela alebo pomocou nového súboru z možnosti ponuky v rámci aktívneho notebooku. Nový notebook je vytvorený v rovnakom adresári a otvorí sa na novej karte prehliadača. To sa tiež prejaví ako nová položka v zozname notebookov na palubnej doske (dashboard). [13]

6.3 Užitočné klávesové skratky

Všetky akcie môžu byť v notebooku vykonávané pomocou myši, ale niektoré klávesové skratky sú vhodné pre zapamätanie.

Shift-Enter skratka spustí aktuálnu bunku, zobrazí výstup bunky (ak existuje) a prejde na nasledujúcu pod ňou. Ak sa uplatní na poslednú bunku, bude vytvorená nová bunka pre zadanie kódu. Treba si uvedomiť, že stlačenie samotnej klávesy Enter nespustí bunku ale iba pridá nový riadok pre písanie kódu.

Ctrl-Enter skratka spustí kód aktuálnej bunky akoby v "terminálovom režime", zobrazí výstup bunky, ale kurzor zostáva aktívny pre danú bunku. Celý obsah sa po spustení vyberie, takže stačí začať písať a v bunke pribudnú iba ďalšie vstupy. Je to vhodný spôsob pre rýchle experimentovanie na mieste, bez nutnosti vytvárať ďalšie bunky, ktoré nechceme mať uložené v dokumente.

Alt-Enter skratka spustí aktuálnu bunku, zobrazí výstup a vloží novú bunku medzi aktuálnu a bunku dole (ak existuje).

[6] JavaScript Object Notation – spôsob zápisu dát, určený na ich prenos vo forme textu.

7 Riešenie úloh pružnosti

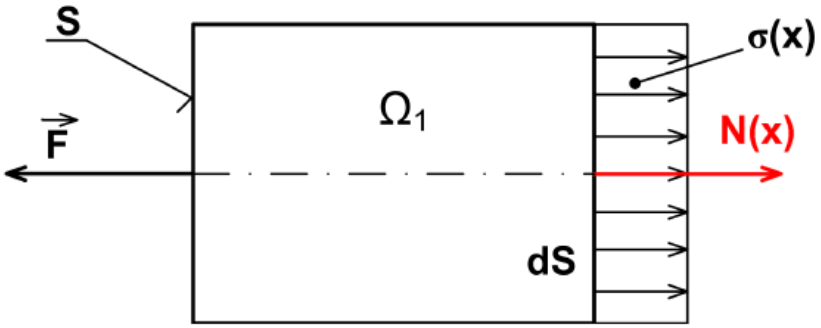
7.1 Namáhanie prostým ťahom/tlakom

Pod definíciou prostý ťah (tlak) rozumieme namáhanie priameho prizmatického prútu za splnenia istých podmienok [14]:

a) Sú splnené všeobecné prútové predpoklady

b) Priečne prierezy sa vzájomne odďaľujú a približujú, pričom zostávajú rovinnými a kolmými ku strednici. Strednica zostáva priamkou

c) Jedinou nenulovou zložkou VVÚ je normálová sila



Obrázok 4 Úplné uvoľnenie prvku zaťaženého prostým ťahom

Vzťahy v tejto kapitole sú preberané z literatúry [14]. Deformácia pri prostom ťahu (tlaku) je priestorová. Pre prostý ťah a prostý smyk platí Hookov zákon, G je modul pružnosti v šmyku a μ je Poissonové číslo

$$\sigma_x = E \cdot \varepsilon_x \ , \quad \tau = G \cdot \gamma \qquad\qquad G = \frac{E}{2(1+\mu)}$$

Napätosť pri prostom ťahu (tlaku) je jednoosová. Vzhľadom ku konštantnému priebehu pomerného pretvorenia ε_x je priebeh napätia σ_x taktiež v reze konštantný.

$$N(x) = \int_{\psi} \sigma_x \cdot dS = \sigma_x \cdot S \qquad\qquad \sigma(x) = \frac{N(x)}{S(x)} = \frac{F}{S}$$

Vzťah pre posuv u v mieste x :

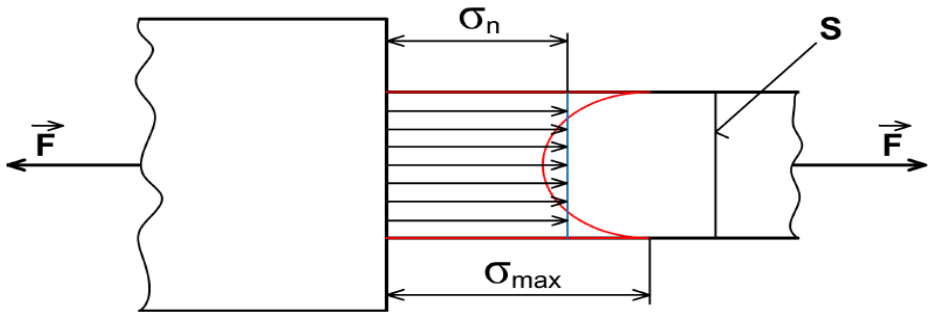
$$u(x) = \int_0^x \frac{N(x')}{ES} dx' = \frac{Nx}{ES}$$ a celkové predĺženie prútu $\Delta l = \frac{Nl}{ES}$

Energia napätosti celého prútu:

$$W = \frac{1}{2} \int_0^l \frac{N^2}{ES} dx = \frac{1}{2} \frac{N^2 l}{ES}$$

7.1.1 Vplyv konštrukčného vrubu

V mieste vrubu vzniká priestorová deformácia a dochádza tu ku koncentrácii napätia. Vplyv vrubu na napätosť vyjadruje zmluvne súčiniteľ koncentrácie napätia α. [14]



Obrázok 5 Priebeh skokovej zmeny ťahového napätia v okolí vrubu

$\sigma_n - nominálne\ napätie$ $\sigma_{max} = \sigma_n \cdot \alpha$

Bezpečnosť vzhľadom k medznému stavu pružnosti:

$$k_k = \frac{\sigma_k}{\sigma_{max}}$$

7.2 Namáhanie prostým ohybom

Prostý ohyb sa dá na základe súhrnnej definície prostých namáhaní konkrétne definovať ako

označenie pre namáhanie priameho prizmatického prútu, ak sú na danej rozlišovacej úrovni [14]:

a) Splnené prútové predpoklady
b) Priečne prierezy sa vzájomne natáčajú okolo osi ležiacej v priečnom priereze a následne sa deformujú
c) jedinou nenulovou zložkou VVÚ je ohybový moment

7.2.1 Schwedlerove vzťahy

$$\frac{dN(x_R)}{dx_R} = -q_N(x_R), \qquad \frac{dT(x_R)}{dx_R} = -q_T(x_R)$$

$$\frac{dM_o(x_R)}{dx_R} = -\int_0^{x_R} q_T(x_R)\,dx = T(x_R)$$

Ak derivujeme ešte posledný výraz podľa x_R dostaneme:

$$\frac{d^2M_o(x_R)}{dx_R^2} = \frac{dT(x_R)}{dx_R} = -q_T(x_R)$$

Pri materiáloch v tvárnom stave nerozlišujme ťahovú a tlakovú bezpečnosť a potom výraz pre výpočet ohybového napätia má tvar [14]:

$$\sigma_o = \frac{M_o}{W_o}$$

Pričom modul prierezu určíme ako:

$$W_y = \frac{J_y}{a_{ax}}$$

Energia napätosti pre celý prút:

$$W = \int_\gamma \frac{M_y^2(x)}{2EJ_y}dx + \int_\gamma \frac{M_z^2(x)}{2EJ_z}dx$$

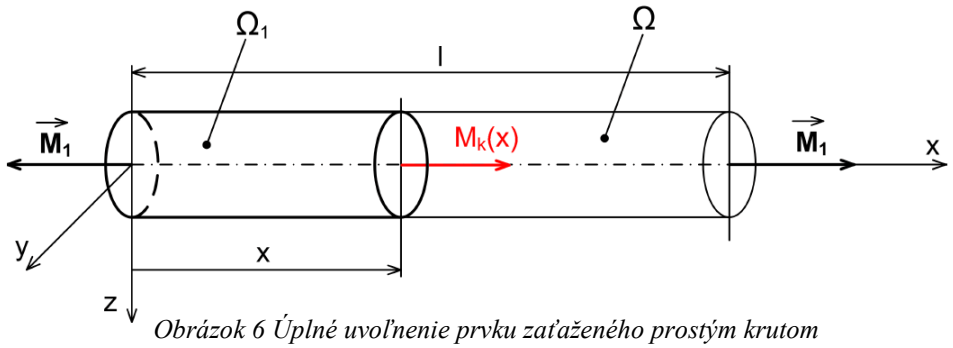
Deformácia prútu podľa Castiglianovej vety:

$$w_F = \frac{\partial W}{\partial F} = \frac{\partial}{\partial F}\int_\gamma \frac{M_y^2(x)}{2EJ_y}dx = \int_\gamma \frac{M_y}{EJ_y}\frac{\partial M_y}{\partial F}dx$$

7.3 Namáhanie prostým krutom

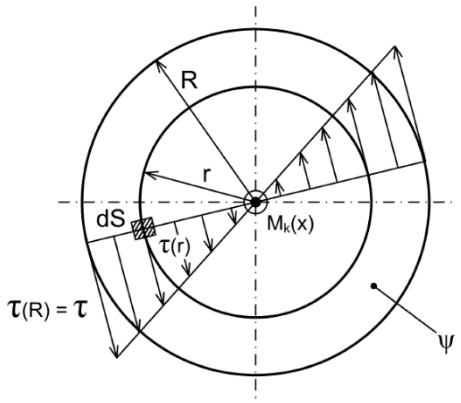
Prostým krutom rozumieme namáhanie priameho prizmatického prútu kruhového alebo medzikruhového prierezu, ak je splnené [14]:

a) platia všeobecné prútové predpoklady
b) priečne prierezy zostávajú v priebehu zaťažovania rovinnými a otáčajú sa okolo strednice, ktorá zostáva priama
c) jedinou nenulovou zložkou VVÚ je krútiaci moment $M_k(x)$, ktorý je konštantný po celej dĺžke prútu



Obrázok 6 Úplné uvoľnenie prvku zaťaženého prostým krutom

Momentová podmienka: $\sum M_x: \; M_k(x) = M_1$



Priebehy skosenia γ *(r)* a šmykového napätia τ *(r)* v priereze sú lineárne závislé na súradnici *r*. Priebeh napätia je uvedený na obrázku. [14]

Obrázok 7 Priebeh šmykového napätia v priereze

Natočenie Δφ (x) prierezu v mieste *x* vzhľadom k ľavému okraju prútu je:

$$\Delta\varphi = \int_{\gamma} \frac{M_k(x)}{GJ_p} dx$$

Pre natočenie celého prizmatického prútu zaťaženého silovou dvojicou *M* na oboch koncoch máme vzťah:

$$\Delta\varphi(l) = \frac{Ml}{GJ_p}$$

Pre maximálne šmykové napätie v krajnom vlákne pre $r = R$ platí :

$$\tau = \frac{M_k}{\frac{J_p}{R}} = \frac{M_k}{W_k}$$

Kde veličina W_k sa nazýva modul prierezu v krute.

V prípade, že v materiálovom liste nenájdeme medzu klzu v šmyku τ_k, dá sa stanoviť na základe podmienky medzného stavu pružnosti a to podmienka maximálnych šmykových napätí a podmienka HMH. [14]

$\tau_k = 0{,}5 \cdot \sigma_k$ – Trescova podmienka

$\tau_k = 0{,}577 \cdot \sigma_k$ – podmienka HMH

Energia napätosti prútu

$$dW = \int_{\gamma} \frac{M_k^2(x)}{2GJ_p} dx$$

Uhol natočenia v mieste pôsobenia silovej dvojice *M* je možné stanoviť pomocou Castiglianovej vety

$$\Delta\varphi_M = \frac{\partial W}{\partial M} = \int_{\gamma} \frac{M_k(x)}{2GJ_p} \frac{\partial M_k}{\partial M} dx$$

8. Riešené príklady

8.1 Prostý ohyb

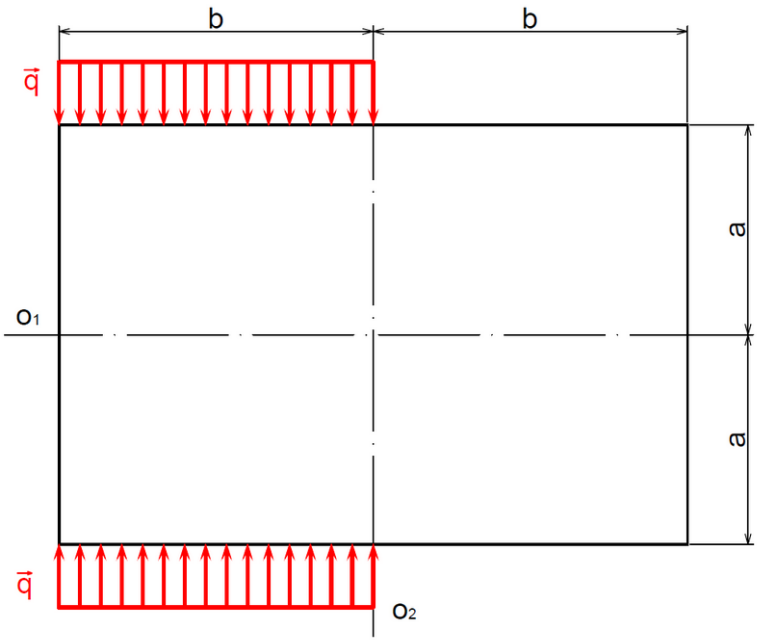
In [1]:
```
%matplotlib inline
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from IPython.core.display import Image
sp.init_printing()
```

Príklad 1

Pri ráme zaťaženom podľa obrázku určite bezpečnosť vzhľadom k medznému stavu pružnosti, ak dané parametre sú: rozmery prútov $\boldsymbol{a}$ a $\boldsymbol{b}$, líniové zaťaženie $\boldsymbol{q}$, rozmery obdĺžnikového priečneho prierezu $\boldsymbol{H}$ a $\boldsymbol{B}$, Youngov modul pružnosti $\boldsymbol{E}$ a medza klzu $\boldsymbol{\sigma_K}$.

In [2]:
```
Image(filename='Ohyb1a.PNG', width=700, height=300)
```

Out[2]:



Daný prút je symetrický podľa osí $\boldsymbol{o_1}$ a $\boldsymbol{o_2}$. Zaťaženie je rozložené symetricky vzhľadom k osi $\boldsymbol{o_1}$, avšak vzhľadom k osi $\boldsymbol{o_2}$ nie je rozložené ani symetricky ani antimetricky. Z toho vyplýva, že môžeme zaviesť symetriu a riešiť iba $\frac{1}{2}$ prútu a nie $\frac{1}{4}$.

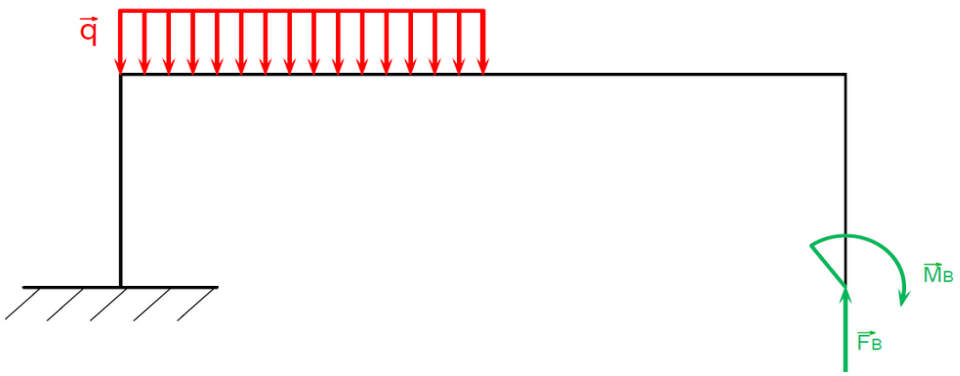
Podmienky symetrie sa môžu zapísať nasledovne

$$u_{F_B} = \frac{\partial W}{\partial F_B} = 0 \qquad \varphi_{M_B} = \frac{\partial W}{\partial M_B} = 0$$

a čiastočné uvoľnenie bude vyzerať týmto spôsobom:

In [3]:
```
Image(filename='Ohyb1b.PNG', width=800, height=400)
```

Out[3]:



Zavedenie potrebných symbolov:

In [4]:
```
a, b, q = sp.symbols('a b q')
H, B = sp.symbols('H B')
x1, x2, x3, x4 = sp.symbols('x_1 x_2 x_3 x_4')
FB, MB = sp.symbols('F_B M_B')
E, J, S = sp.symbols('E J S')
```

Výsledné vnútorné účinky sa vyjadria nasledovne, keďže v prípade zalomeného prútu uvoľňujeme od voľného konca:

VVÚ:

Úsek I:

$$\text{pre } x_1 \in (0;a)$$

$$N_1 = -F_B\,, \qquad T_1 = 0\,, \qquad M_1 = -M_B$$

Úsek II:

$$\text{pre } x_2 \in (0;b)$$

$$N_1 = 0\,, \qquad T_1 = -F_B\,, \qquad M_1 = F_B x_2 - M_B$$

Úsek III:

$$\text{pre } x_3 \in (0;b)$$

$$N_1 = 0\,, \qquad T_1 = q x_3 - F_B\,, \qquad M_1 = F_B(x_3 + b) - M_B - q\frac{x_3^2}{2}$$

Úsek IV:

$$\text{pre } x_4 \in (0;a)$$

$$N_1 = F_B - qb\,, \qquad T_1 = 0\,, \qquad M_1 = 2F_B b - M_B - q\frac{b^2}{2}$$

Zapísané do Pythonu:

In [5]:
```
N1 = -FB
M1 = -MB
N1, M1
```

Out[5]: $(-F_B, \quad -M_B)$

In [6]:
```
T2 = - FB
M2 = FB * x2 - MB
T2, M2
```

Out[6]: $(-F_B, \quad F_B x_2 - M_B)$

In [7]:
```
T3 = q*x3 - FB
M3 = FB*(x3 + b) - MB - q*x3**2/2
T3, M3
```

Out[7]: $\left(-F_B + q x_3, \quad F_B\,(b + x_3) - M_B - \frac{q x_3^2}{2}\right)$

In [8]:
```
N4 = FB - q*b
M4 = FB*2*b - MB - q*b**2/2
N4, M4
```

Out[8]: $\left(F_B - bq, \quad 2F_B b - M_B - \frac{b^2 q}{2}\right)$

Derivácie vnútorných účinkov podľa M_B a F_B:

In [9]:
```
dM1dMB = M1.diff(MB)
dM2dMB = M2.diff(MB)
dM3dMB = M3.diff(MB)
dM4dMB = M4.diff(MB)
dM1dFB = M1.diff(FB)
dM2dFB = M2.diff(FB)
dM3dFB = M3.diff(FB)
dM4dFB = M4.diff(FB)
```

In [10]:
```
dM1dMB, dM2dMB, dM3dMB, dM4dMB
```

Out[10]: $(-1, \quad -1, \quad -1, \quad -1)$

In [11]:
```
dM1dFB, dM2dFB, dM3dFB, dM4dFB
```

Out[11]: $(0, \quad x_2, \quad b + x_3, \quad 2b)$

Podmienky symetrie môžu byť zapísané nasledovne,

$$\frac{1}{EJ_P}\left[\int_0^a M_1 \frac{\partial M_1}{\partial M_B}dx_1 + \int_0^b M_2 \frac{\partial M_2}{\partial M_B}dx_2 + \int_0^b M_3 \frac{\partial M_3}{\partial M_B}dx_3 + \int_0^a M_4 \frac{\partial M_4}{\partial M_B}dx_4\right] = 0$$

$$\frac{1}{EJ}\left[\int_0^a M_1 \frac{\partial M_1}{\partial F_B}dx_1 + \int_0^b M_2 \frac{\partial M_2}{\partial F_B}dx_2 + \int_0^b M_3 \frac{\partial M_3}{\partial F_B}dx_3 + \int_0^a M_4 \frac{\partial M_4}{\partial F_B}dx_4\right] = 0$$

Keďže obe rovnice sú homogénne, môžeme ich zjednodušiť prenásobením a zapísať do tvaru:

In [12]:
```
rov1 =sp.Integral(M1*dM1dMB,[x1,0,a]) + \
      sp.Integral(M2*dM2dMB,[x2,0,b]) + \
      sp.Integral(M3*dM3dMB,[x3,0,b]) + \
      sp.Integral(M4*dM4dMB,[x4,0,a])
rov2 =sp.Integral(M1*dM1dFB,[x1,0,a]) + \
      sp.Integral(M2*dM2dFB,[x2,0,b]) + \
      sp.Integral(M3*dM3dFB,[x3,0,b]) + \
      sp.Integral(M4*dM4dFB,[x4,0,a])
rov1, rov2
```

Out[12]:

$$\left(\int_0^a M_B\,dx_1 + \int_0^b -F_B x_2 + M_B\,dx_2 + \int_0^a -2F_B b + M_B + \frac{b^2 q}{2}\,dx_4 + \int_0^b -F_B\,(b + x_3) + M_B + \frac{q x_3^2}{2}\,dx_3,\quad \int_0^a 0\,dx_1 + \int_0^a 2b\left(2F_B b - M_B - \frac{b^2 q}{2}\right)dx_4 + \int_0^b x_2\,(F_B x_2 - M_B)\,dx_2 + \int_0^b (b + x_3)\left(F_B\,(b + x_3) - M_B - \frac{q x_3^2}{2}\right)dx_3\right)$$

Integráciou dostaneme sústavu rovníc o dvoch neznámych M_B a F_B a ich riešenie vyzerá nasledovne:

In [13]:
```
rov1_, rov2_ = rov1.doit(), rov2.doit()
rov1_ = sp.expand(rov1_)
rov2_ = sp.expand(rov2_)
vysl1 = sp.linsolve([rov1_,rov2_],[MB, FB])
MB_vysl, FB_vysl = next(iter(vysl1))
MB_vysl, FB_vysl
```

Out[13]:

$$\left(\frac{b^3 q\,(21a + 5b)}{144a^2 + 192ab + 48b^2},\quad \frac{3bq\,(4a + b)}{48a + 16b}\right)$$

Dosadenie čísel pre numerický výpočet

$$a = 500\ \text{mm};\ b = 750\ \text{mm};\ q = 15\ \text{Nmm}^{-1};\ B = 20\ \text{mm};\ H = 50mm$$

In [14]:
```
a_, b_ = 500., 750.
q_= 15.
B_, H_ = 20., 50.
```

Dosadenie numerických hodnôt do vyjadrených neznámych M_B a F_B:

In [15]:
```
MB_ = MB_vysl.subs({a:a_,b:b_,q:q_})
FB_ = FB_vysl.subs({a:a_,b:b_,q:q_})
MB_, FB_
```

Out[15]: $(667968.75,\quad 2578.125)$

Dosadenie číselných hodnôt a hodnôt $\boldsymbol{M_B}$ a $\boldsymbol{F_B}$ do výsledných vnútorných účinkov:

In [16]:
```
N1_ = N1.subs(FB,FB_)
M1_ = M1.subs(MB,MB_)
N1_, M1_
```

Out[16]: $(-2578.125,\quad -667968.75)$

In [17]:
```
T2_ = T2.subs(FB,FB_)
M2_ = M2.subs({FB:FB_,MB:MB_})
T2_, M2_
```

Out[17]: $(-2578.125,\quad 2578.125x_2 - 667968.75)$

In [18]:
```
T3_ = T3.subs({FB:FB_,q:q_})
M3_ = M3.subs({FB:FB_,MB:MB_,q:q_,b:b_})
T3_, M3_
```

Out[18]: $\left(15.0x_3 - 2578.125,\quad -7.5x_3^2 + 2578.125x_3 + 1265625.0\right)$

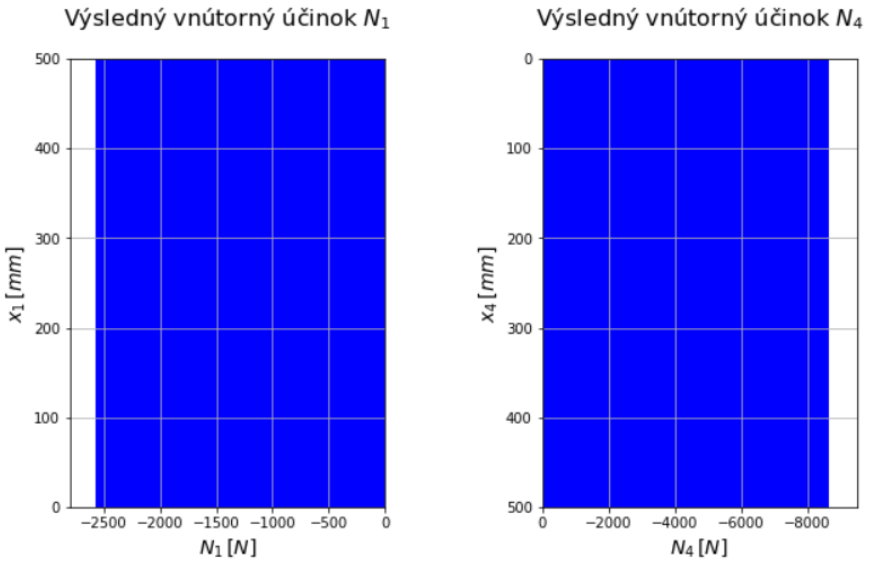
In [19]:
```
N4_ = N4.subs({FB:FB_,q:q_,b:b_})
M4_ = M4.subs({FB:FB_,MB:MB_,q:q_,b:b_})
N4_, M4_
```

Out[19]: $(-8671.875,\quad -1019531.25)$

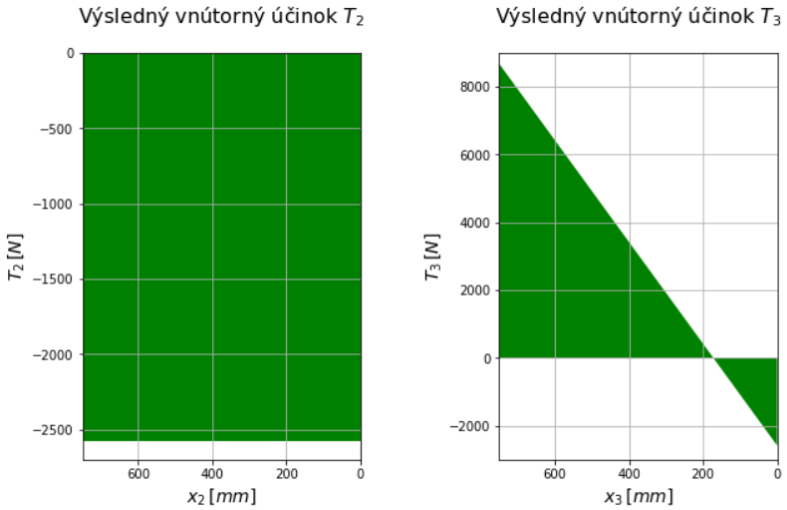
Vykreslenie výsledných vnútorných účinkov

Vyjadrenie VVÚ rovnomerne pozdĺž strednice: Kód programu pre vykreslenie grafov viz príloha [P1/01].

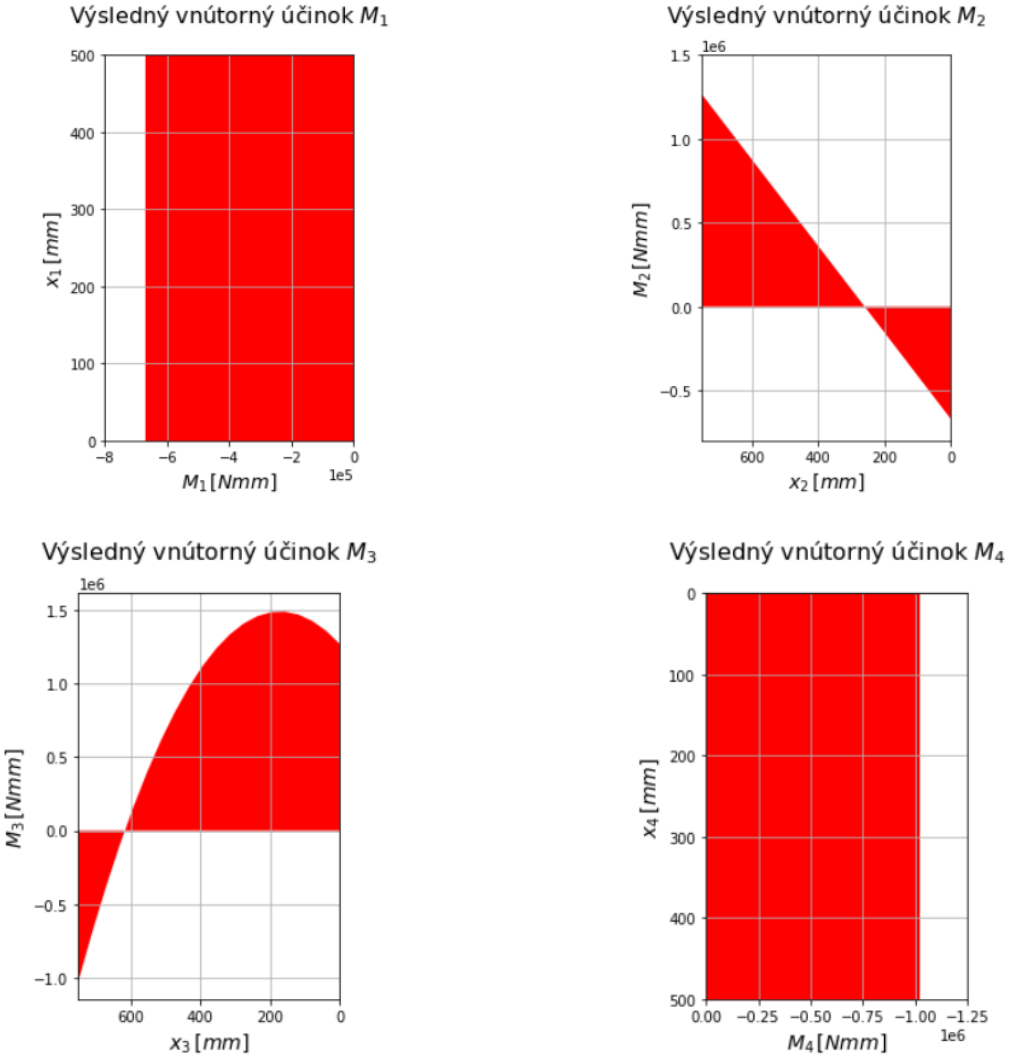
Vykreslenie vnútorných normálových účinkov $\boldsymbol{N(x)}$:



Vykreslenie šmykových vnútorných účinkov $\boldsymbol{T(x)}$:



Vykreslenie momentových vnútorných účinkov $\boldsymbol{M(x)}$:



Extrémne napätie vyjadríme iba z momentových vnútorných účinkov $\boldsymbol{M(x)}$. Z uvedených grafov je zrejmé, že globálnu extrémnu hodnotu nadobúda moment $\boldsymbol{M_3(x_3)}$ v bode $\boldsymbol{x_{3max}}$, ktorý nájdeme riešením rovnice

$$\frac{\mathbf{d}}{\mathbf{dx}}\boldsymbol{M_3(x)} = \boldsymbol{T_3(x)} = \mathbf{0}$$

In [24]:
```
vysl2 = sp.solve(T3_,x3)
x3_max = float(vysl2[0])
x3_max
```

Out[24]: 171.875

Hodnota vnútorného momentu $\boldsymbol{M_3}$ v tomto bode má hodnotu $\boldsymbol{(Nmm)}$:

In [25]:
```
M3_max = abs(float(M3_.subs(x3,x3_max)))
M3_max
```

Out[25]: 1487182.6171875

Príslušné ohybové napätie pre zaťaženie momentom $\boldsymbol{M_{3max}}$ sa spočíta podľa vzťahu:

$$\sigma_o = \frac{M_3(x_{3max})}{W_o}, \quad W_o = \frac{1}{6}BH^2$$

In [26]:
```
Wo = B_*H_**2/6
sigma_max = M3_max/Wo
o = 'Maximálny ohybový moment je ' +repr(round(M3_max,3))+
' Nmm a maximálne ohybové napätie je '
+repr(round(sigma_max,3))+ ' MPa.'
print(o)
```

```
Maximálny ohybový moment je 1487182.617 Nmm a maximálne
ohybové napätie je 178.462 MPa.
```

Pokiaľ je daná $\boldsymbol{\sigma_K = 350 MPa}$ bezpečnosť vzhľadom k medznému stavu pružnosti je:

In [27]:
```
sigma_K = 350.
kk=sigma_K/sigma_max
kk
```

Out[27]: 1.9612027688856055

Ďalšie názorné úlohy vytvorené pomocou jazyka Python a prezentované skrz aplikáciu Jupyter budú obsiahnuté v prílohách a takisto umiestnené na webovej stránke a využívané pre výuku. Odkaz na stránku: http://www.old.umt.fme.vutbr.cz/~tprofant/

9 Záver

V predchádzajúcom texte bol najskôr predstavený Python ako interpretovaný, interaktívny open-source programovací jazyk ponúkajúci široké spektrum možností využitia v technickej praxi, ako aj niektoré z jeho knižníc, ktoré našli najväčšie uplatnenie pri dosahovaní daných aspektov, a aplikácia Jupyter, ktorá zefektívňuje a dynamizuje prácu a prezentáciu vytvorených kódov.

Hlavným cieľom tejto práce bolo oboznámenie sa a naštudovanie si multi-pradigmového programovacieho jazyka Python spolu s jeho modulmi NumPy, SymPy a Matplotlib, ktorých fundamentálne funkcie boli aplikované na výpočty niektorých úloh pružnosti a pevnosti I.

K samotnému spracovávaniu úloh bolo využité spojenie základných charakteristík jazyka Python podporené výpočtovou univerzálnosťou spomínaných knižníc. Výpočtom prilieha teoretické okienko obsahujúce základné poznatky z oblasti pružnosti a pevnosti I nutné k náležitým výpočtom. Finálne spustiteľné aplikácie vznikli uplatnením kombinácie lineárnej algebry a prácou s maticami či vektormi obsiahnutých v knižnici NumPy, o ktorej pojednáva kap. 3, spolu s možnosťou symbolických výpočtov vedúcich ku kompletizácii výsledných analytických vzťahov prostredníctvom knižnice SymPy (kap. 5) doplnených o možnosť 2D vykresľovania grafov a diagramov vďaka knižnici Matplotlib (kap.4).

Každá jednotlivá úloha začala načítaním potrebných knižníc a modulov nutných k adekvátnej úrovni ich prezentovania prostredníctvom webovej aplikácie Jupyter. Koncepcia vychádzala z priblíženia daného problému z hľadiska základnej idey postupu riešenia staticky určitých alebo neurčitých úloh a ich analytickým riešením, nasledované implementáciou numerickej časti výpočtu, prípadne vykreslením výsledných vnútorných účinkov. Všetky tieto fenomény boli umožnené vďaka využitiu príkazov rozobratých v teoretickej časti práce.

Takto bolo pojednaných 6 exemplárnych úloh konkretizujúcich základy prostého ťahu/tlaku, ohybu a krútenia. Výsledky a výpočty z oblasti pružnosti a pevnosti budú ďalej aktívne využívané a prezentované ako podklady k výučbe.

10 Zoznam použitých zdrojov

[1] JIRASEK. *Python* [online]. [cit. 2018-01-09]. Dostupné z: http://ics.upjs.sk/~jirasek/sps/python/python.html

[2] Python (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2018-01-06]. Dostupné z: https://sk.wikipedia.org/wiki/Python_(programovac%C3%AD_jazyk)

[3] BLAHO, A. *Jazyk Python* [online]. 2017 [cit. 2018-01-12]. Dostupné z: http://python.input.sk/01.html#ako-ho-ziskat

[4] BLAHO, A. *Programovanie v Pythone 1*. Bratislava: Knižničné a edičné centrum FMFI UK, 2016, 322 s. ISBN 978-80-8147-067-7.

[5] VIKTORIN, P., HRONČOK, M. *NumPy* [online]. 2016 [cit. 2018-01-19]. Dostupné z: http://naucse.python.cz/lessons/intro/numpy/

[6] URBAN, J., PIPEK, J. *Základy Numpy* [online]. 2013 [cit. 2018-01-24]. Dostupné z: http://pythonic.eu/fjfi/posts/zaklady-numpy.html

[7] JOHNSON, J. *Python Numpy Tutorial* [online]. [cit. 2018-01-22]. Dostupné z: http://cs231n.github.io/python-numpy-tutorial/#numpy

[8] HUNTER, J. *Matplotlib* [online]. [cit. 2018-01-26]. Dostupné z: http://matplotlib.org

[9] WILLEMS, K. *Matplotlib Tutorial: Python Plotting* [online]. 2017 [cit. 2018-01-26]. Dostupné z: http://www.datacamp.com/community/tutorials/matplotlib-tutorial-python

[10] MEURER, A., ČERTÍK, O. et. al. *SymPy: symbolic computing in Python* [online]. 2017 [cit. 2018-01-28]. Dostupné z: http://peerj.com/articles/cs-103/

[11] SymPy Development Team. *SymPy Tutorial* [online]. 2017 [cit. 2018-02-04]. Dostupné z: http://docs.sympy.org/latest/tutorial/index.html

[12] Jupyter Team. Jupyter Notebook. *Project Jupyter* [online]. 2018 [cit. 2018-02-14]. Dostupné z: http://jupyter.org/

[13] INGARGIOLA, Antonino. *What is the Jupyter Notebook?* [online]. 2015 [cit. 2018-02-11]. Dostupné z: http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

[14] JANÍČEK, P., ONDRÁČEK, E. *Mechanika těles: pružnost a pevnost I*. Brno: Akademické nakladatelství CERM, 2004. ISBN 80-214-2592-X.

[15] FLORIAN, Z., ONDRÁČEK, E., PŘIKRYL, K. *Mechanika těles: statika*. Vyd. 7., V Akademickém nakladatelství CERM 2. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-214-3440-0.

Zoznam príloh

Spustiteľné súbory *.ipynb:*

Prostý ohyb (príklad 1)	P1/01
Prostý ohyb (príklad 2)	P1/02
Prostý ohyb (príklad 3)	P1/03
Prostý ťah (príklad 1)	P1/04
Prostý ťah (príklad 2)	P1/05
Prostý krut (príklad 3)	P1/06

Obrázky:

Ohyb1a.png	P2/01
Ohyb1b.png	P2/02
Ohyb2a.png	P2/03
Ohyb2b.png	P2/04
Ohyb3a.png	P2/05
Ohyb3b.png	P2/06
Ohyb3c.png	P2/07
Tah1a.png	P2/08
Tah1b.png	P2/09
Tah2a.png	P2/10
Tah2b.png	P2/11
Krut1a.png	P2/12
Krut1b.png	P2/13

HTML súbory:

Prostý ohyb (príklad 1)	P3/01
Prostý ohyb (príklad 2)	P3/02
Prostý ohyb (príklad 3)	P3/03
Prostý ťah (príklad 1)	P3/04
Prostý ťah (príklad 2)	P3/05
Prostý krut (príklad 3)	P3/06