

Diplomová práce



České  
vysoké  
učení technické  
v Praze

**F1**

Fakulta stavební  
Katedra betonových a zděných konstrukcí

## Výpočetní nástroje pro analýzu keramobetonových stropních a střešních systémů z trámů a vložek s využitím pokročilých numerických metod

**Bc. Daniel Beránek**

Vedoucí práce: Ing. Radek Štefan, Ph.D.  
Studijní program: Stavební inženýrství  
Specializace: Konstrukce pozemních staveb  
Květen 2024

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Beránek** Jméno: **Daniel** Osobní číslo: **484656**  
Fakulta/ústav: **Fakulta stavební**  
Zadávací katedra/ústav: **Katedra betonových a zděných konstrukcí**  
Studijní program: **Stavební inženýrství**  
Studijní obor: **Konstrukce pozemních staveb**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Výpočetní nástroje pro analýzu keramobetonových stropních a střešních systémů z trámů a vložek s využitím pokročilých numerických metod**

Název diplomové práce anglicky:

**Computational tools for analysis of clay beam-and-block floor and roof systems using advanced numerical methods**

Pokyny pro vypracování:

Rešerše literatury  
Popis výpočetních modelů a metod  
Popis výpočetních nástrojů  
Verifikace a validace výpočetních nástrojů  
Vzorové příklady  
Závěr

Seznam doporučené literatury:

ČSN EN 1990 Zásady navrhování konstrukcí  
ČSN EN 1991 Zatížení konstrukcí  
ČSN EN 1992 Navrhování betonových konstrukcí  
ČSN EN 1994 Navrhování spřažených ocelobetonových konstrukcí  
ČSN EN 1995 Navrhování dřevěných konstrukcí  
ČSN EN 15037-1 Betonové prefabrikáty - Stropní systémy z trámů a vložek - Část 1: Trámy

Jméno a pracoviště vedoucí(ho) diplomové práce:

**Ing. Radek Štefan, Ph.D. katedra betonových a zděných konstrukcí FSV**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **22.02.2024**

Termín odevzdání diplomové práce: **20.05.2024**

Platnost zadání diplomové práce: \_\_\_\_\_

Ing. Radek Štefan, Ph.D.  
podpis vedoucí(ho) práce

doc. Ing. Lukáš Vráblík, Ph.D.  
podpis vedoucí(ho) ústavu/katedry

prof. Ing. Jiří Máca, CSc.  
podpis děkana(ky)

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

\_\_\_\_\_  
Datum převzetí zadání

\_\_\_\_\_  
Podpis studenta

## Poděkování

Práce vznikla za podpory poskytnuté Ministrstvem průmyslu a obchodu ČR v rámci programu OP PIK, Aplikace (Výzva IX), č. projektu CZ.01.1.02/0.0/0.0/21\_374/0026789, Vývoj komplexního softwaru pro optimalizaci návrhu a posouzení střešních a stropních konstrukcí.

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o do-  
držování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20.5.2024

---

Bc. Daniel Beránek

## Abstrakt

Diplomová práce se zaměřuje na vývoj a implementaci nástrojů pro statické výpočty prutových konstrukcí v programovacím jazyce Python. Cílem této práce je vytvoření výpočetních nástrojů a jejich verifikace. V první části jsou představeny základy teorie pružnosti a odvození matic tuhostí a vektorů zatížení pomocí knihovny pro symbolické výpočty. Dále je představena objektově orientovaná knihovna pro výpočty prutových konstrukcí. Další kapitola se věnuje knihovně pro navrhování konstrukcí podle Eurokódů. Závěrem je popsána integrace těchto nástrojů do webové aplikace, která poskytuje uživatelsky přívětivé rozhraní pro návrh a posouzení konstrukcí.

**Klíčová slova:** výpočetní nástroj, program, prutová konstrukce, Python, deformační metoda

**Vedoucí práce:** Ing. Radek Štefan, Ph.D.

## Abstract

This thesis is focused on the development and implementation of software tools for structural analysis of frame members in the Python programming language. The aim of this thesis is to develop and verify software tools. First part introduces the basics of elasticity theory and the derivation of stiffness matrices and load vectors using computer algebra system for symbolic calculations. Furthermore, an object-oriented library for the computation of member structures is presented. The next chapter is devoted to the library for designing structures according to Eurocodes. Finally, the integration of these tools into a web application that provides a user-friendly interface for the design and assessment of structures is described.

**Keywords:** computational tool, program, frame structure, Python, direct stiffness method

**Title translation:** Computational tools for analysis of clay beam-and-block floor and roof systems using advanced numerical methods

# Obsah

<b>Seznam použitých symbolů</b>	<b>1</b>	<b>3.2 Aplikace STROP</b>	<b>81</b>
<b>Úvod</b>	<b>3</b>	3.2.1 Prostý nosník	81
Motivace	3	<b>Závěr a diskuze</b>	<b>87</b>
Cíle	3	<b>Bibliografie</b>	<b>89</b>
Rozsah	4		
<b>1 Knihovna pro statické výpočty prutových konstrukcí</b>	<b>5</b>		
1.1 Základní rovnice teorie pružnosti	5		
1.1.1 Přehled základních veličin	5		
1.1.2 Přehled základních rovnic	6		
1.1.3 Okrajové podmínky	9		
1.1.4 Shrnutí	10		
1.2 Princip virtuálních prací	11		
1.2.1 Princip virtuálních posunutí	13		
1.2.2 Přibližné řešení	14		
1.3 Prutové prvky	15		
1.3.1 Tažený-tlačený prut	17		
1.3.2 Ohýbaný prvek bez vlivu smyku	20		
1.3.3 Ohýbaný prvek s vlivem smyku	26		
1.4 Statická kondenzace	33		
1.5 Transformační matice	35		
1.6 Implementace	37		
1.6.1 Použití externího open-source softwaru	37		
1.6.2 Instalace	38		
1.6.3 Architektura knihovny	39		
1.6.4 Předpoklady	43		
1.6.5 Algoritmizace	45		
1.6.6 Řídké matice	45		
1.6.7 Testování	47		
1.6.8 Příklad	48		
<b>2 Knihovna pro navrhování konstrukcí podle Eurokódů</b>	<b>61</b>		
2.1 Instalace	62		
2.2 Architektura knihovny	62		
2.2.1 Struktura knihovny	62		
2.2.2 Integrace s knihovnou pro výpočty prutových konstrukcí	63		
2.3 Příklad použití	66		
<b>3 Webová aplikace</b>	<b>72</b>		
3.1 Aplikace STŘECHA	72		
3.1.1 Původní verze aplikace	72		
3.1.2 Nová verze	74		
3.1.3 Příklad	78		

## Obrázky

1.1 Napětí v materiálovém bodě ....	6	2.4 Diagram třídy	
1.2 Deformace elementárního kvádru	7	DesignLoadCaseGroup .....	65
1.3 Napětí na diferenciálním výseku tělesa .....	8	3.1 STŘECHA – Modul pro finální posouzení 1/2 .....	73
1.4 Schéma vztahů základních rovnic teorie pružnosti a silného řešení...	10	3.2 STŘECHA – Modul pro finální posouzení 2/2 .....	74
1.5 Schéma vztahů základních rovnic teorie pružnosti a slabého řešení ..	15	3.3 STŘECHA – Nové uživatelské prostředí .....	75
1.6 Deformovaná konfigurace taženého-tlačeného prutu .....	17	3.4 STŘECHA – Záložka s uzly ....	76
1.7 Bázové funkce taženého-tlačeného prutu .....	18	3.5 STŘECHA – Záložka s prvky ..	76
1.8 Zatížení taženého-tlačeného prutu	19	3.6 STŘECHA – Záložka se zatěžovacími stavy .....	76
1.9 Deformovaná konfigurace Euler-Bernoulliho nosníku .....	20	3.7 STŘECHA – Záložka se zatížením	77
1.10 Bázové funkce ohýbaného nosníku .....	24	3.8 STŘECHA – Záložka s kombinacemi zatížení .....	77
1.11 Zatížení Euler-Bernoulliho ohýbaného nosníku .....	25	3.9 STŘECHA – Okno pro přidání nového uzlu .....	77
1.12 Deformovaná konfigurace Timoshenkova nosníku .....	26	3.10 STŘECHA – Okno pro přidání nového prvku .....	78
1.13 Zatížení Timoshenkova ohýbaného nosníku .....	33	3.11 STŘECHA – Okno pro přidání nového zatěžovacího stavu .....	78
1.14 Styčník s kloubově připojenými pruty .....	33	3.12 Zatížení .....	79
1.15 Prut typu vetknutí-kloub ....	34	3.13 Reakce .....	79
1.16 Transformace souřadnic .....	36	3.14 Normálová síla .....	80
1.17 Typy spojitého zatížení .....	40	3.15 Posouvající síla .....	80
1.18 Diagram modulu <b>pre</b> .....	41	3.16 Ohybové momenty .....	81
1.19 Diagram modulu <b>fea</b> .....	42	3.17 Zadání příkladu .....	82
1.20 Diagram modulu <b>solvers</b> ....	43	3.18 Schéma stropu .....	82
1.21 Diagram modulu <b>post</b> .....	43	3.19 Pole stropu - rozpětí, POT nosníky a kari sítě .....	82
1.22 Stupně volnosti v globálním systému .....	44	3.20 Podpory - šířky, kari sítě a příločky .....	82
1.23 Stupně volnosti v lokálním systému prvku .....	44	3.21 Rozměry stropu .....	83
1.24 Kladná orientace vnitřních sil .	45	3.22 Materiálové vlastnosti .....	83
1.25 Statické schéma konstrukce ...	48	3.23 Přetížení stropu (zatížení bez vlastní tíhy zmonolitněné konstrukce) .....	83
1.26 Reakce v podporách .....	55	3.24 Bodová zatížení na nosníku ...	84
1.27 Vnitřní síly na konstrukci .....	58	3.25 Výsledky posouzení .....	84
1.28 Porovnání souřadných systémů	60	3.26 Označení průřezů po délce nosníku .....	84
2.1 Diagram modulu <b>loads</b> .....	63	3.27 Průřez stropní konstrukce .....	85
2.2 Diagram třídy <b>DesignLoadCase</b>	64	3.28 Posouvající síla .....	85
2.3 Diagram třídy <b>DesignLoadCaseComtination</b> ....	65	3.29 Ohybový moment .....	85
		3.30 Využití průřezů .....	86


## Tabulky

1.1 Porovnání výsledků .....	59
------------------------------	----

## Seznam použitých symbolů

$A$	plocha průřezu
$E$	modul pružnosti
$G$	modul pružnosti ve smyku
$I_y$	moment setrvačnosti k ose $y$
$L$	délka prvku
$M_y$	ohybový moment
$N_i$	bázová funkce
$N$	normálová síla
$V_z$	posouvající síla
$\Gamma_p$	hranice tělesa s předepsaným zatížením
$\Gamma_u$	hranice tělesa s předepsanými posuny
$\Gamma$	hranice tělesa
$\Omega$	smyková štiřlost
$\Omega$	objem tělesa
$\sigma$	vektor napětí
$\varepsilon$	vektor poměrné deformace
$\delta$	virtuální veličina
$\gamma$	poměrná deformace (zkosení)
$\kappa$	křivost
$X$	globální osa $X$
$Y$	globální osa $y$
$Z$	globální osa $Z$
$B$	deformačně-posuvová matice
$D$	matice materiálové tuhosti
$K_{e,g}$	matice tuhosti elementu
$K_{e,l}$	matice tuhosti elementu
$K$	Matice tuhosti
$N$	matice bázových funkcí
$T$	transformační matice
$\bar{X}$	předepsaný vektor objemových sil
$\bar{p}$	předepsané zatížení na hranici tělesa
$\bar{u}$	předepsaný posun na hranici tělesa
$d$	vektor zobecněných posunů
$f$	vektor zatížení





$\mathbf{n}$	matice složek jednotkové normály
$\mathbf{u}$	vektor koncových posunutí a pootočení prvku
$\mathbf{u}$	vektor posunutí
$\mathbf{x}$	vektor polohy
$\nu$	Poissonův součinitel
$\sigma$	normálové napětí
$\tau$	smykové napětí
$\varepsilon$	poměrná deformace (prodloužení nebo zkrácení)
$\varphi_i$	pootočení okolo osy $y$
$b$	šířka průřezu
$h$	výška průřezu
$n$	počet stupňů volnosti
$u$	posun ve směru osy $x$
$v$	posun ve směru osy $y$
$w$	posun ve směru osy $z$
$x$	lokální osa $x$
$y$	lokální osa $y$
$z$	lokální osa $z$

## Úvod

## Motivace

V rámci bakalářské práce [4] byl vyvinut výpočetní nástroj zaměřený na analýzu průhybu stropních systémů Porotherm z trámů a vložek, tento nástroj byl integrován do webové aplikace vyvinuté v rámci projektu *Vývoj komplexního softwaru pro optimalizaci návrhu a posouzení střešních a stropních konstrukcí* [32]. Aplikace se však potýká s řadou omezení, statické schéma je omezené na spojitý nosník o maximálně pěti polích s konzolami na obou stranách, lze zadat pouze omezený počet působících sil, spojitě zatížení lze zadat pouze konstantní hodnotou působící na celé délce pole. Vizualizace výsledků je také omezena, neboť výstupy jsou zpracovávány pomocí knihovny matplotlib [14] na straně serveru, což omezuje možnost interakce a značně zvyšuje výpočetní čas.

Vzhledem k těmto omezením je cílem této diplomové práce vytvořit nástroje, které výrazně rozšíří funkčnost a použitelnost webové aplikace, především se jedná o podporu pro více typů zatížení a libovolná statická schémata, ale také o vylepšení grafického rozhraní a interakce s uživatelem. Toho bylo částečně dosaženo přechodem zpracování výsledků z strany serveru na stranu klienta s využitím dynamických a interaktivních vizualizací pomocí JavaScript knihovny THREE.js.

Diplomová práce navazuje na bakalářskou práci a představuje sadu pokročilých softwarových nástrojů, jejichž cílem je nejen rozšířit možnosti analýzy stropních a střešních konstrukcí, ale také zjednodušit a zefektivnit proces navrhování podle nejnovějších technických norem. Potřeba dynamičtějších, přístupnějších a komplexnějších nástrojů ve stavebním inženýrství je zřejmá z rostoucí složitosti architektonických návrhů a požadavků moderních stavebních předpisů.

## Cíle

Hlavními cíli této práce jsou:

- **Vývoj knihovny pro statickou analýzu prutových konstrukcí:**  
Tato knihovna bude určena pro výpočet prutových konstrukcí deformační metodou. Knihovna umožní definování různých typů zatížení a jejich sloučení do zatěžovacích stavů.
- **Vývoj knihovny pro posuzování konstrukcí podle Eurokódů:**

Vyvinutí nástroje, který umožní komplexní posouzení nosných prvků v souladu s Eurokódy.

- **Verifikace výsledků získaných vyvinutými výpočetními knihovnamí:** Provedení srovnání výsledků získaných aplikacemi s manuálními výpočty a existujícími softwarovými řešeními, za účelem ověření přesnosti a spolehlivosti nově vyvinutých knihoven.
- **Demonstrace praktického použití knihoven:** Prezentace aplikace knihoven na reálných konstrukcích a případových studiích, ilustrace jejich užitečnosti a efektivity.
- **Integrace knihoven do uživatelsky přívětivé webové aplikace:** Vývoj webové platformy s intuitivním grafickým uživatelským rozhraním, která zlepší přístupnost a interaktivitu výpočetních nástrojů.

## Rozsah

V této práci bude popsán vývoj a implementace jednotlivých softwarových nástrojů, bude diskutována integrace těchto nástrojů do jedné webové aplikace a jejich použití bude demonstrováno na jednoduchých příkladech. K dosažení cílů jsou využity programovací jazyky Python a JavaScript.

# Kapitola 1

## Knihovna pro statické výpočty prutových konstrukcí

V první části jsou představeny základy teorie pružnosti, které jsou dále zjednodušeny se zaměřením se na prutové prvky. Pomocí knihovny pro symbolické výpočty SymPy[18] bude předvedeno odvození matic tuhostí a vektorů zatížení nejběžněji používaných prutových prvků.

Druhá část této kapitoly se zaměřuje na implementaci prutových prvků v programovacím jazyce Python. Knihovna je navržena objektově orientovaně, což zajišťuje přehlednost a modularitu kódu. Tento přístup usnadňuje údržbu a rozšiřování funkcionality knihovny a zajišťuje její flexibilitu při řešení různých inženýrských úloh.

### 1.1 Základní rovnice teorie pružnosti

V této části budou představeny základní veličiny a rovnice nezbytné pro popis chování pružných těles. Zaměříme se na vektor posunutí, vektor napětí a vektor deformací. Dále popíšeme geometrické, fyzikální a statické rovnice, které jsou klíčové pro pochopení základních principů pružnosti. Na závěr uvedeme okrajové podmínky.

#### 1.1.1 Přehled základních veličin

##### Vektor posunutí

Posunutí libovolného bodu pružného tělesa v prostoru můžeme rozložit do tří vzájemně kolmých složek, které je možné zapsat ve formě vektoru posunutí [7, s. 2]

$$\mathbf{u} = \begin{Bmatrix} u & v & w \end{Bmatrix}^T. \quad (1.1)$$

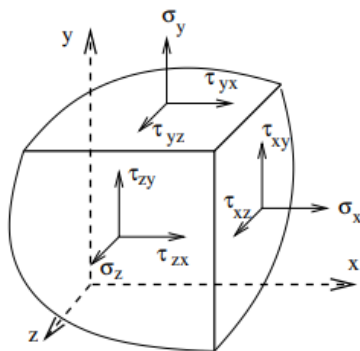
##### Vektor deformací

Každé z uvedených napětí pracuje na odpovídající poměrné deformaci, normálovému napětí  $\sigma_i$  odpovídá poměrná deformace  $\varepsilon_i$  a smykovému napětí  $\tau_{ij}$  odpovídá zkosení  $\gamma_{ij}$ . Poměrné deformace je možné zapsat v podobě vektoru poměrných deformací [7, s. 3]

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{yz} & \gamma_{zx} & \gamma_{xy} \end{Bmatrix}^T. \quad (1.2)$$

## Vektor napětí

Na obrázku 1.1 jsou zobrazena napětí v materiálovém bodě tělesa. Ve směru jednotlivých os systému souřadnic působí tři normálová napětí  $\sigma_x$ ,  $\sigma_y$  a  $\sigma_z$ . Rovnoběžně s osami systému působí šest smykových napětí v rovinách  $xy$ ,  $yz$ ,  $zx$  [7, s. 2].



**Obr. 1.1:** Napětí v materiálovém bodě, převzato z [7, s. 3]

Uplatněním předpokladu o vzájemnosti smykových napětí, je možné považovat jen tři smyková napětí za nezávislá [7, s. 3].

$$\begin{aligned}\tau_{xy} &= \tau_{yx}, \\ \tau_{yz} &= \tau_{zy}, \\ \tau_{xz} &= \tau_{zx}.\end{aligned}\tag{1.3}$$

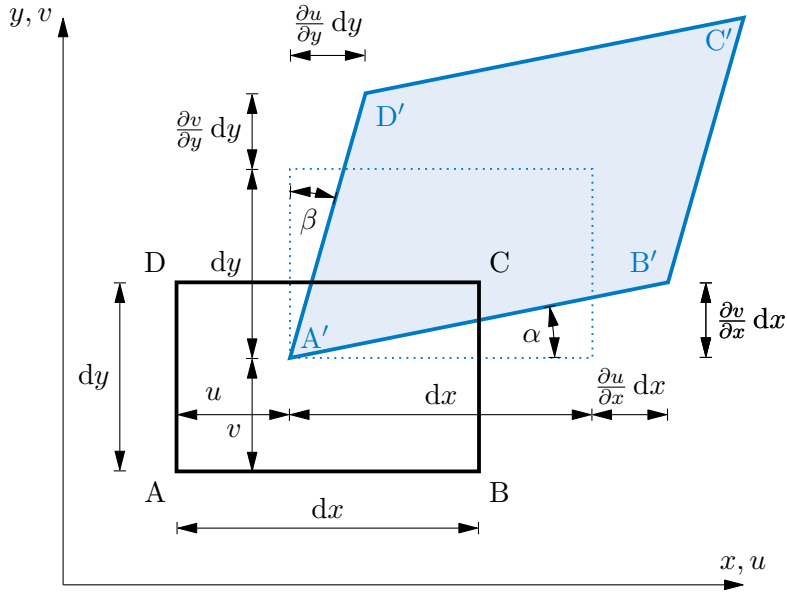
Složky lze zapsat v podobě vektoru napětí

$$\boldsymbol{\sigma} = \left\{ \sigma_x \quad \sigma_y \quad \sigma_z \quad \tau_{yz} \quad \tau_{zx} \quad \tau_{xy} \right\}^T.\tag{1.4}$$

## 1.1.2 Přehled základních rovnic

### Geometrické rovnice

Tělesa mění z nejrůznějších příčin svůj tvar a objem – deformují se. Pro deformaci elementárního kvádru jsou typické dva základní geometricko-deformační modely. První předpokládá protažení hran kvádru ve směrech souřadnicových os při zachování pravých úhlů mezi stěnami a druhý model se vyznačuje změnami pravých úhlů mezi stěnami kvádru při zachování délek hran [30, s. 9]. Na obr. 1.2 je pro názornost nakreslen pouze průmět kvádru do roviny  $xy$ .



**Obr. 1.2:** Deformace elementárního kvádru, podle [7, obr. 1.2]

Prodloužení kvádru ve směru  $x$  můžeme vyjádřit následovně

$$\varepsilon_x = \frac{|A'B'| - |AB|}{|AB|} = \frac{(dx + \frac{\partial u}{\partial x} dx) - dx}{dx} = \frac{\partial u}{\partial x}. \quad (1.5)$$

Analogicky lze získat vztahy pro  $\varepsilon_y$  a  $\varepsilon_z$ . Shrnutí vztahů pro poměrné prodloužení ve směru souřadnicových os je uvedeno v 1.6.

$$\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x}, \\ \varepsilon_y &= \frac{\partial v}{\partial y}, \\ \varepsilon_z &= \frac{\partial w}{\partial z}. \end{aligned} \quad (1.6)$$

Smykové zkosení je možné stanovit na základě určení velikostí úhlů  $\alpha$  a  $\beta$  na obr. 1.2, zavedeme předpoklad, že  $\tan \alpha = \alpha$ ,  $\tan \beta = \beta$ ,  $\frac{\partial v}{\partial x} dx = 0$  a  $\frac{\partial v}{\partial y} dy = 0$ .

$$\gamma_{xy} = \alpha + \beta \approx \frac{\partial v}{\partial x} dx + \frac{\partial u}{\partial y} dy = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}. \quad (1.7)$$

Analogicky lze zapsat vztahy pro zbývající smyková zkosení

$$\begin{aligned} \gamma_{xy} &= \gamma_{yx} = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}, \\ \gamma_{yz} &= \gamma_{zy} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}, \\ \gamma_{zx} &= \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}. \end{aligned} \quad (1.8)$$

## ■ Fyzikální rovnice

Jako fyzikální vztahy se označují vztahy mezi napětím a poměrnými deformacemi. Jak je známo z pružnosti, poměr mezi podélnou a příčnou změnou

délky tělesa je konstantní a je popsán Poissonovým součinitelem  $\nu$ . Potom je na místě předpokládat, že velikost poměrného prodloužení  $\varepsilon_x$  bude ovlivněna nejen napětím  $\sigma_x$ , ale v závislosti na hodnotě  $\nu$  také napětím ve směrech  $y$  a  $z$  [7, s. 7]

$$\varepsilon_x = \frac{1}{E} [\sigma_x - \nu(\sigma_y + \sigma_z)]. \quad (1.9)$$

U smyku lze předpokládat, že vztah mezi smykovým napětím  $\tau_{ij}$  a zkosením  $\gamma_{ij}$  bude lineární

$$\gamma_{ij} = \frac{\tau_{ij}}{2G}. \quad (1.10)$$

Fyzikální vztahy pro pružné těleso v prostoru můžeme zapsat ve tvaru

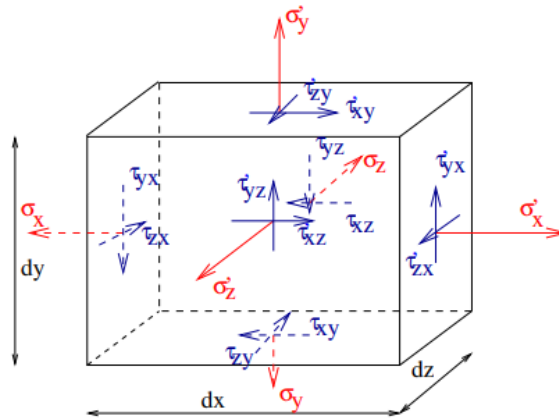
$$\begin{aligned} \varepsilon_x &= \frac{1}{E} [\sigma_x - \nu(\sigma_y + \sigma_z)], & \gamma_{yz} &= \frac{\tau_{yz}}{2G}, \\ \varepsilon_y &= \frac{1}{E} [\sigma_y - \nu(\sigma_x + \sigma_z)], & \gamma_{xz} &= \frac{\tau_{xz}}{2G}, \\ \varepsilon_z &= \frac{1}{E} [\sigma_z - \nu(\sigma_y + \sigma_x)], & \gamma_{xy} &= \frac{\tau_{xy}}{2G}. \end{aligned} \quad (1.11)$$

### ■ Statické rovnice

V případě pružného tělesa je vhodné napsat silové podmínky rovnováhy na vyjmutém diferenciálním objemu o rozměrech  $dx$ ,  $dy$ ,  $dz$ , který je zobrazen na obrázku 1.3 [7, s. 6].

Pro lepší přehlednost jsou napětí označena jako  $\sigma_i'$  a  $\tau_{ij}'$  napětí změněná o přírůstek na diferenciálním rozměru objemu, tedy například

$$\sigma_x' = \sigma_x + \frac{\partial \sigma_x}{\partial x} dx, \quad \tau_{xy}' = \tau_{xy} + \frac{\partial \tau_{xy}}{\partial y} dy, \dots \quad (1.12)$$



**Obr. 1.3:** Napětí na diferenciálním výseku tělesa, převzato z [7, s. 5]

Výslednice napětí  $\sigma_x$  se získá vynásobením napětí a plochy na které působí,

$$F_{\sigma,x} = \sigma_x dy dz. \quad (1.13)$$

Silovou podmínku rovnováhy ve směru osy  $x$  je možné zapsat jako

$$\sum F_{i,x} = (\sigma_x' - \sigma_x) dy dz + (\tau_{xy}' - \tau_{xy}) dx dz + (\tau_{xz}' - \tau_{xz}) dx dy = 0. \quad (1.14)$$

Rozepsáním rovnice 1.14 pomocí vztahů 1.12 dostaneme výraz

$$\begin{aligned} & \left( \sigma_x - \sigma_x - \frac{\partial \sigma_x}{\partial x} dx \right) dy dz \\ & + \left( \tau_{xy} - \tau_{xy} - \frac{\partial \tau_{xy}}{\partial y} dy \right) dx dz \\ & + \left( \tau_{xz} - \tau_{xz} - \frac{\partial \tau_{xz}}{\partial z} dz \right) dx dy = 0, \end{aligned} \quad (1.15)$$

který lze dále upravit na

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} = 0. \quad (1.16)$$

Obdobně lze napsat podmínky rovnováhy o pro směry  $y$  a  $z$ . Takto sestavené rovnice doplníme o objemové síly  $X$ ,  $Y$  a  $Z$ , působící ve směrech jednotlivých souřadnicových os, získáme výsledný tvar podmínek rovnováhy,

$$\begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + X &= 0, \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + Y &= 0, \\ \frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + Z &= 0. \end{aligned} \quad (1.17)$$

### 1.1.3 Okrajové podmínky

Složky napětí  $\sigma$  i složky posunutí  $\varepsilon$  musí vyhovovat okrajovým podmínkám předepsaným na hranici tělesa  $\Gamma = \Gamma_p + \Gamma_u$ .

#### Statické okrajové podmínky

Statické okrajové podmínky tvoří soustavu tří lineárních algebraických rovnic a vyžadují požadavek rovnováhy pole napětí  $\sigma$  s předepsaným zatížením  $\bar{\mathbf{p}}$  na části hranice  $\Gamma_p$  [30, s. 35]. V maticovém tvaru je lze zapsat

$$\begin{bmatrix} n_x & 0 & 0 & 0 & n_z & n_y \\ 0 & n_y & 0 & n_z & 0 & n_x \\ 0 & 0 & n_z & n_y & n_z & 0 \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{zx} \\ \tau_{xy} \end{Bmatrix} = \begin{Bmatrix} \bar{p}_x \\ \bar{p}_y \\ \bar{p}_z \end{Bmatrix}, \quad (1.18)$$

neboli

$$\mathbf{n}\sigma = \bar{\mathbf{p}} \quad \text{na } \Gamma_p. \quad (1.19)$$



### Kinematické okrajové podmínky

Kinematické (geometrické) okrajové podmínky se vztahují na část hranice  $\Gamma_u$ , kde jsou předepsány posuny  $\bar{\mathbf{u}} = \{\bar{u} \ \bar{v} \ \bar{w}\}^T$  a mají tvar

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{na } \Gamma_u. \quad (1.20)$$

### 1.1.4 Shrnutí

Pro úplný popis chování pružného tělesa je v každém jeho bodě potřeba získat hodnoty 15 neznámých veličin, 3 složky posunutí  $\mathbf{u}$ , 6 složek deformací  $\boldsymbol{\varepsilon}$  a 6 složek napětí  $\boldsymbol{\sigma}$ . K jejich vypočtení máme k dispozici 15 rovnic, 6 geometrických rovnic, 6 fyzikálních rovnic a 3 statické rovnice (podmínky rovnováhy) [7, s. 8].

Zmíněné rovnice lze zapsat v kompaktním tvaru:

$$\text{statické rovnice} \quad \partial \boldsymbol{\sigma}^T + \bar{\mathbf{X}} = \mathbf{0}, \quad (1.21)$$

$$\text{fyzikální rovnice} \quad \boldsymbol{\sigma} = \mathbf{D} \boldsymbol{\varepsilon}, \quad (1.22)$$

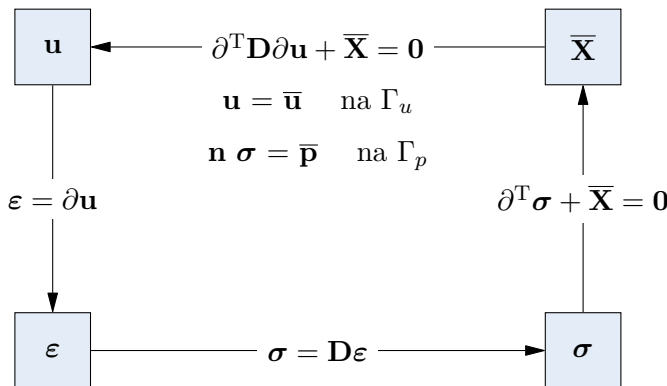
$$\text{geometrické rovnice} \quad \boldsymbol{\varepsilon} = \partial \mathbf{u}, \quad (1.23)$$

kde

$$\begin{aligned} \text{vektor posunutí} \quad & \mathbf{u} = \{u \ v \ w\}^T, \\ \text{vektor poměrné deformace} \quad & \boldsymbol{\varepsilon} = \{\varepsilon_x \ \varepsilon_y \ \varepsilon_z \ \gamma_{xy} \ \gamma_{yz} \ \gamma_{zx}\}^T, \\ \text{vektor napětí} \quad & \boldsymbol{\sigma} = \{\sigma_x \ \sigma_y \ \sigma_z \ \tau_{xy} \ \tau_{yz} \ \tau_{zx}\}^T, \\ \text{předepsaný vektor objemových sil} \quad & \bar{\mathbf{X}} = \{X \ Y \ Z\}^T, \\ \text{operátorová matice} \quad & \partial = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \end{bmatrix}^T, \\ \text{nulový vektor} \quad & \mathbf{0} = \{0 \ 0 \ 0\}^T. \end{aligned}$$

Postupným dosazením 1.23 a 1.22 do 1.21 získáme silné řešení ve tvaru

$$\partial^T \mathbf{D} \partial \mathbf{u} + \bar{\mathbf{X}} = \mathbf{0}. \quad (1.24)$$



**Obr. 1.4:** Schéma vztahů základních rovnic teorie pružnosti a silného řešení, podle [17, obr. 1.2]

„S přesným řešením systému podmíněčných rovnic se v inženýrské praxi setkáváme poměrně zřídka. S výjimkou několika speciálních typů konstrukcí, jako jsou prutové soustavy nebo rotačně symetrické a symetricky zatížené prostorové konstrukce (např. válcové, kulové ap.), je zpravidla třeba použít řešení přibližných.“

[30, s. 36]

„According to solid mechanics, the solution must satisfy this set of differential equations with additional constraints (leading to the so called Boundary Value Problem). Closed-form solutions, such as  $\mathbf{u}(\mathbf{x}) = [u(\mathbf{x}) \ v(\mathbf{x}) \ w(\mathbf{x})]^T$  defined over the entire problem domain, are possible only when simple geometries and loadings are considered.“

[17, kap. 1.1]

## 1.2 Princip virtuálních prací

Princip virtuálních prací a variační principy mechaniky jsou základem většiny přibližných metod mechaniky. Princip virtuálních prací lze dělit na princip virtuálních posunutí a princip virtuálních sil.

Publikace [31, s. 53] uvádí následující definice:

**Definice 1.1** (Virtuální posun). Libovolný možný posun elementu mechanické soustavy, který je v souladu s jejími pohybovými možnostmi. Elementem mechanické soustavy rozumíme jakoukoliv její část, jež je s ostatními částmi spojena vazbami.

**Definice 1.2** (Virtuální deformace). Jsou odvozeny z virtuálních posunů pomocí geometrických rovnic. Virtuální posuny i deformace nenarušují vazby soustavy. Jsou fiktivní, myšlené a uděleme je elementům soustavy bez ohledu na síly a napětí, které na ně působí.

**Definice 1.3** (Virtuální síla). Duální protějšek virtuálního posunu. Je to síla, kterou na soustavu umísťujeme nezávisle na skutečných posunech.

**Definice 1.4** (Virtuální napětí). Fiktivní, myšlené veličiny. Jsou stanoveny tak, aby v každém bodě tělesa, které je zatíženo virtuálními silami, byla zajištěna rovnováha.

Předpokládejme, že se těleso nachází v jisté rovnovážné konfiguraci, která je jednoznačně popsána vektorovým polem posunutí

$$\mathbf{u} = \begin{Bmatrix} u & v & w \end{Bmatrix}^T,$$

tenzorovým polem deformace

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{yz} & \gamma_{zx} & \gamma_{xy} \end{Bmatrix}^T,$$

a tenzorovým polem napětí

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_x & \sigma_y & \sigma_z & \tau_{yz} & \tau_{zx} & \tau_{xy} \end{Bmatrix}^T.$$

Dále předpokládejme, že každému bodu tělesa je udělen malý virtuální posun

$$\delta \mathbf{u} = \begin{Bmatrix} \delta u & \delta v & \delta w \end{Bmatrix}^T,$$

kteřý jej vychýlí z původní polohy v rovnovážné konfiguraci. Funkce  $\delta u$ ,  $\delta v$  a  $\delta w$  jsou ve smyslu definice 1.1 spojité a mají spojité první parciální derivace podle proměnných  $x$ ,  $y$  a  $z$ . Výsledné posuny  $(\mathbf{u} + \delta \mathbf{u})$  a odvozené deformace  $(\boldsymbol{\varepsilon} + \delta \boldsymbol{\varepsilon})$  se nazývají kinematicky přípustné.

Připojením virtuálních objemových sil

$$\delta \bar{\mathbf{X}} = \left\{ \delta \bar{X} \quad \delta \bar{Y} \quad \delta \bar{Z} \right\}^T$$

a virtuálních povrchových sil

$$\delta \bar{\mathbf{p}} = \left\{ \delta p_x \quad \delta p_y \quad \delta p_z \right\}^T,$$

změníme stav zatížení tělesa a připojením virtuálních napětí

$$\delta \boldsymbol{\sigma} = \left\{ \delta \sigma_x \quad \delta \sigma_y \quad \delta \sigma_z \quad \delta \tau_{yz} \quad \delta \tau_{zx} \quad \delta \tau_{xy} \right\}^T,$$

jeho napjatost.

Nemá-li být narušena rovnováha tělesa, musí v každém bodě platit rovnice 1.21

$$\boldsymbol{\sigma} \delta \boldsymbol{\varepsilon} + \delta \bar{\mathbf{X}} = \mathbf{0}, \quad (1.25)$$

a na hranici tělesa  $\Gamma$

$$\mathbf{n} \delta \boldsymbol{\sigma} - \delta \bar{\mathbf{p}} = \mathbf{0}. \quad (1.26)$$

Výsledné síly  $(\bar{\mathbf{X}} + \delta \bar{\mathbf{X}})$ ,  $(\bar{\mathbf{p}} + \delta \bar{\mathbf{p}})$  a výsledná napětí  $(\boldsymbol{\sigma} + \delta \boldsymbol{\sigma})$  se nazývají **staticky přípustné**.

Princip virtuálních prací lze matematicky zapsat následovně

$$\underbrace{\int_{\Omega} (\boldsymbol{\sigma} + \delta \boldsymbol{\sigma})^T (\boldsymbol{\varepsilon} + \delta \boldsymbol{\varepsilon}) d\Omega}_{\text{virtuální práce vnitřních sil}} = \underbrace{\int_{\Omega} (\bar{\mathbf{X}} + \delta \bar{\mathbf{X}})^T (\mathbf{u} + \delta \mathbf{u}) d\Omega + \int_{\Gamma} (\bar{\mathbf{p}} + \delta \bar{\mathbf{p}})^T (\mathbf{u} + \delta \mathbf{u}) d\Gamma}_{\text{virtuální práce vnějších sil}}. \quad (1.27)$$

Vhodným upravením rovnice 1.27, při uvážení vzájemné nezávislosti zavedených virtuálních polí, obdržíme čtyři rovnice, které musí být splněny nezávisle na sobě,

■ Clapeyronův (divergenční) teorém

$$\int_{\Omega} \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \mathbf{u} d\Omega + \int_{\Gamma} \bar{\mathbf{p}}^T \mathbf{u} d\Gamma, \quad (1.28)$$

■ princip virtuálních posunutí

$$\int_{\Omega} \boldsymbol{\sigma}^T \delta \boldsymbol{\varepsilon} d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \delta \mathbf{u} d\Omega + \int_{\Gamma} \bar{\mathbf{p}}^T \delta \mathbf{u} d\Gamma, \quad (1.29)$$

■ princip virtuálních sil

$$\int_{\Omega} \delta \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} d\Omega = \int_{\Omega} \delta \bar{\mathbf{X}}^T \mathbf{u} d\Omega + \int_{\Gamma} \delta \bar{\mathbf{p}}^T \mathbf{u} d\Gamma, \quad (1.30)$$

■ čtvrtá rovnice, která nemá bezprostřední využití, ve tvaru

$$\int_{\Omega} \delta \boldsymbol{\sigma}^T \delta \boldsymbol{\varepsilon} d\Omega = \int_{\Omega} \delta \bar{\mathbf{X}}^T \delta \mathbf{u} d\Omega + \int_{\Gamma} \delta \bar{\mathbf{p}}^T \delta \mathbf{u} d\Gamma. \quad (1.31)$$

### 1.2.1 Princip virtuálních posunutí

Dále se budeme zabývat principem virtuálních posunutí, ze kterého odvodíme algoritmus deformační metody.

Uvažujme takové virtuální posuny  $\delta \mathbf{u}$ , které nenarušují kinematické okrajové podmínky

$$\delta \mathbf{u} = \mathbf{0}, \quad \text{na hranici } \Gamma_u, \quad (1.32)$$

a zároveň splňují geometrické rovnice uvnitř tělesa

$$\delta \boldsymbol{\varepsilon} = \boldsymbol{\partial} \delta \mathbf{u}, \quad \text{uvnitř tělesa } \Omega. \quad (1.33)$$

Rovnici 1.29 zapíšeme s přihlédnutím k 1.32,

$$\int_{\Omega} \boldsymbol{\sigma}^T \delta \boldsymbol{\varepsilon} \, d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \delta \mathbf{u} \, d\Omega + \int_{\Gamma_p} \bar{\mathbf{p}}^T \delta \mathbf{u} \, d\Gamma_p. \quad (1.34)$$

Levou stranu rovnice 1.34 upravíme dosazením z rovnice 1.33,

$$\int_{\Omega} \boldsymbol{\sigma}^T \delta \boldsymbol{\varepsilon} \, d\Omega = \int_{\Omega} \boldsymbol{\sigma}^T \boldsymbol{\partial} \delta \mathbf{u} \, d\Omega, \quad (1.35)$$

a zintegrujeme per partes, pomocí věty o integraci tenzorového pole [31, s. 55],

$$\int_{\Omega} \boldsymbol{\sigma}^T \boldsymbol{\partial} \delta \mathbf{u} \, d\Omega = \int_{\Gamma} \delta \mathbf{u}^T \mathbf{n} \boldsymbol{\sigma} \, d\Gamma - \int_{\Omega} \delta \mathbf{u}^T \boldsymbol{\partial} \boldsymbol{\sigma} \, d\Omega. \quad (1.36)$$

Dosazením z rovnice 1.36 do původního vztahu 1.34 dostaneme rovnici

$$\int_{\Gamma} \delta \mathbf{u}^T \mathbf{n} \boldsymbol{\sigma} \, d\Gamma - \int_{\Omega} \delta \mathbf{u}^T \boldsymbol{\partial} \boldsymbol{\sigma} \, d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \delta \mathbf{u} \, d\Omega + \int_{\Gamma_p} \bar{\mathbf{p}}^T \delta \mathbf{u} \, d\Gamma_p, \quad (1.37)$$

rozepsáním prvního integrálu na část s předepsaným zatížením  $\Gamma_p$  a s předepsanými posuny  $\Gamma_u$  dostaneme tvar

$$\int_{\Gamma_p} \delta \mathbf{u}^T \mathbf{n} \boldsymbol{\sigma} \, d\Gamma_p + \int_{\Gamma_u} \delta \mathbf{u}^T \mathbf{n} \boldsymbol{\sigma} \, d\Gamma_u - \int_{\Omega} \delta \mathbf{u}^T \boldsymbol{\partial} \boldsymbol{\sigma} \, d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \delta \mathbf{u} \, d\Omega + \int_{\Gamma_p} \bar{\mathbf{p}}^T \delta \mathbf{u} \, d\Gamma_p, \quad (1.38)$$

kde podle kinematické okrajové podmínky 1.32

$$\int_{\Gamma_u} \delta \mathbf{u}^T \mathbf{n} \boldsymbol{\sigma} \, d\Gamma_u = 0.$$

Po úpravě dostaneme

$$\int_{\Omega} \delta \mathbf{u}^T (\boldsymbol{\partial} \boldsymbol{\sigma} + \bar{\mathbf{X}}) \, d\Omega + \int_{\Gamma_p} \delta \mathbf{u}^T (-\mathbf{n} \boldsymbol{\sigma} + \bar{\mathbf{p}}) \, d\Gamma_p = 0. \quad (1.39)$$

Protože virtuální posuny  $\delta \mathbf{u}$  jsou libovolné, rovnice 1.39 bude splněna právě tehdy, pokud budou zároveň platit statické rovnice 1.21 a statické okrajové podmínky 1.19.

### 1.2.2 Přibližné řešení

Rovnice 1.34 může být použita k přibližnému řešení úlohy teorie pružnosti. Předpokládejme, že neznámé posuny  $\mathbf{u}(\mathbf{x})$  lze aproximovat pomocí

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{N}(\mathbf{x})\mathbf{d}, \quad (1.40)$$

kde

$\mathbf{N}(\mathbf{x})$  matice bazových funkcí,  
 $\mathbf{d}$  vektor zobecněných posunů.

Pomocí geometrických rovnic 1.23 určíme pole deformace,

$$\begin{aligned} \boldsymbol{\varepsilon}(\mathbf{x}) &= \partial \mathbf{N}(\mathbf{x})\mathbf{d} \\ &= \mathbf{B}(\mathbf{x})\mathbf{d}. \end{aligned} \quad (1.41)$$

Pole napětí vypočítáme z pole deformace pomocí fyzikálních rovnic 1.22,

$$\begin{aligned} \boldsymbol{\sigma}(\mathbf{x}) &= \mathbf{D}\boldsymbol{\varepsilon}(\mathbf{x}) \\ &= \mathbf{D}\mathbf{B}(\mathbf{x})\mathbf{d}. \end{aligned} \quad (1.42)$$

Virtuální pole posunutí je aproximováno stejnými bazovými funkcemi,

$$\delta \mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\delta \mathbf{d}, \quad (1.43)$$

pole virtuálních deformací je s virtuálními posuny svázáno stejnou maticí  $\mathbf{B}$ ,

$$\delta \boldsymbol{\varepsilon} = \mathbf{B}(\mathbf{x})\delta \mathbf{d}. \quad (1.44)$$

Pro zjednodušení v dalším textu budeme matici bazových funkcí  $\mathbf{N}(\mathbf{x})$  a matici  $\mathbf{B}(\mathbf{x})$  značit pouze jako  $\mathbf{N}$  a  $\mathbf{B}$ , přičemž je třeba mít na paměti, že jsou závislé na  $\mathbf{x}$ .

Dosazením vztahů 1.40 až 1.44 do rovnice 1.34 obdržíme,

$$\int_{\Omega} (\mathbf{D}\mathbf{B}\mathbf{d})^T \mathbf{B}\delta \mathbf{d} \, d\Omega = \int_{\Omega} \bar{\mathbf{X}}^T \mathbf{N}\delta \mathbf{d} \, d\Omega + \int_{\Gamma_p} \bar{\mathbf{p}}^T \mathbf{N}\delta \mathbf{d} \, d\Gamma_p. \quad (1.45)$$

Úpravou dostaneme tvar,

$$\mathbf{d}^T \underbrace{\int_{\Omega} \mathbf{B}^T \mathbf{D}^T \mathbf{B} \, d\Omega}_{\mathbf{K}^T} \delta \mathbf{d} = \underbrace{\left( \int_{\Omega} \bar{\mathbf{X}}^T \mathbf{N} \, d\Omega + \int_{\Gamma_p} \bar{\mathbf{p}}^T \mathbf{N} \, d\Gamma_p \right)}_{\mathbf{f}^T} \delta \mathbf{d}. \quad (1.46)$$

Rovnici dále prepíšeme pomocí matice tuhosti  $\mathbf{K}$  a vektoru zatížení  $\mathbf{f}$  a upravíme\*,

$$\begin{aligned} \mathbf{d}^T \mathbf{K}^T \delta \mathbf{d} &= \mathbf{f}^T \delta \mathbf{d} \\ \delta \mathbf{d}^T \left( \mathbf{d}^T \mathbf{K}^T \right)^T &= \delta \mathbf{d}^T \mathbf{f} \\ \delta \mathbf{d}^T (\mathbf{K}\mathbf{d} - \mathbf{f}) &= 0. \end{aligned}$$

---

\* $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$

Protože rovnice musí platit pro libovolné virtuální posunutí  $\delta \mathbf{d}$ , musí být výraz v závorce nulovým vektorem. Odvodili jsme tedy zobecněné podmínky rovnováhy ve tvaru

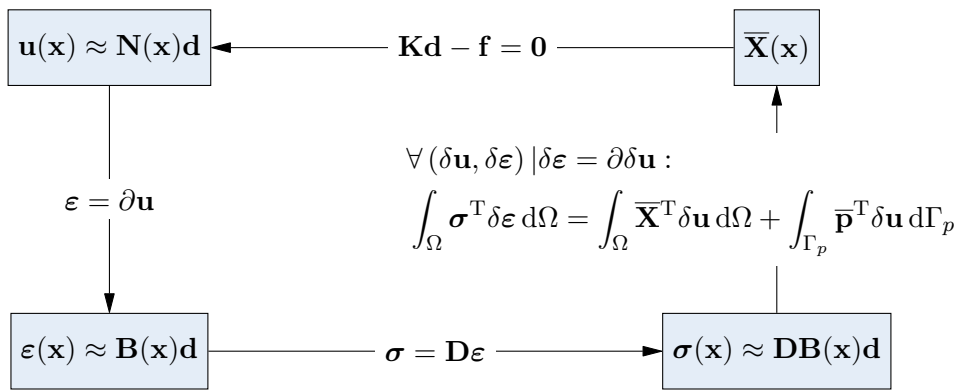
$$\mathbf{K}\mathbf{d} - \mathbf{f} = \mathbf{0}, \quad (1.47)$$

kde  $\mathbf{K}$  je matice tuhosti,

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} \, d\Omega, \quad (1.48)$$

a  $\mathbf{f}$  je vektor zatížení,

$$\mathbf{f} = \int_{\Omega} \mathbf{N}^T \bar{\mathbf{X}} \, d\Omega + \int_{\Gamma_p} \mathbf{N}^T \bar{\mathbf{p}} \, d\Gamma_p. \quad (1.49)$$



**Obr. 1.5:** Schéma vztahů základních rovnic teorie pružnosti a slabého řešení, podle [17, obr. 1.3]

## 1.3 Prutové prvky

V této části se zaměříme na řešení prutových konstrukcí. Prut je těleso s výrazně převládající délkou nad rozměry průřezu. Dimenzi z pohledu napjatosti budeme redukovat na jednorozměrný problém.

Přijmeme následující předpoklady,

- zatížení působí pouze v rovině XZ, která je i rovinou symetrie, řešení není funkcí souřadnice  $y$ ,
- průhyb ve směru osy  $z$  je po výšce prvku konstantní,
- výrazně převládajícím rozměrem je délka prvku  $L$ ,
- uvažujeme prizmatický prut (s neměnným průřezem po délce prvku),
- uvažujeme teorii malých přetvoření,
- materiál prutu se chová pružně a je homogenní.

### Vektor posunutí

Vzhledem k přijatým předpokladům je posun ve směru osy  $Y$  nulový, zároveň uvažujeme po výšce nestlačitelný prut, to znamená, že posun ve směru osy  $z$  je závislý pouze na souřadnici  $x$ . V dalších kapitolách si vystačíme s vektorem posunutí ve tvaru

$$\mathbf{u}(\mathbf{x}) = \begin{Bmatrix} u(x, z) & w(x) \end{Bmatrix}^T. \quad (1.50)$$

### Vektor deformací

Vektor deformací odpovídající předpokladům má dvě nenulové složky,

$$\boldsymbol{\varepsilon}(x, z) = \begin{Bmatrix} \varepsilon_x(x, z) & \gamma_{zx}(x, z) \end{Bmatrix}^T = \begin{Bmatrix} \frac{\partial u(x, z)}{\partial x} & \frac{\partial u(x, z)}{\partial z} + \frac{\partial w(x)}{\partial x} \end{Bmatrix}^T. \quad (1.51)$$

### Vektor napětí

Při analýze prutových konstrukcí je běžné pracovat s vnitřními silami místo s napětím. Vnitřní síly jsou s napětím svázány podmínkami ekvivalence

$$N(x) = \int_A \sigma_x \, dA, \quad (1.52)$$

$$V_z(x) = \int_A \tau_{xz} \, dA, \quad (1.53)$$

$$M_y(x) = \int_A \sigma_x z \, dA, \quad (1.54)$$

kde  $dA = dy \, dz$ .

### Podmínky rovnováhy

Podmínky rovnováhy na prvku lze zapsat jako

$$\frac{dN(x)}{dx} + \bar{f}_x(x) = 0, \quad (1.55)$$

$$\frac{dV_z(x)}{dx} + \bar{f}_z(x) = 0, \quad (1.56)$$

$$\frac{dM_y(x)}{dx} - V_z(x) = 0, \quad (1.57)$$

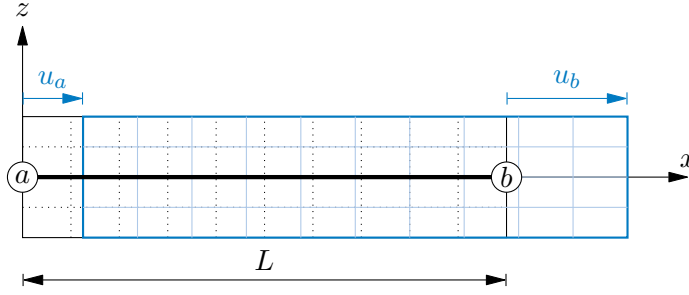
kde  $\bar{f}_x(x)$  a  $\bar{f}_z(x)$  je zatížení působící po délce prutu.

### Kinematické podmínky

Kinematické podmínky jsou dány typem podpory, např. pro vetknutí platí

$$u = 0, \quad w = 0, \quad \varphi = 0. \quad (1.58)$$

### 1.3.1 Tažený-tlačený prut



**Obr. 1.6:** Deformovaná konfigurace taženého-tlačeného prutu

Tažený-tlačený prut je v rovině dán pomocí dvou uzlů,  $a$  a  $b$ . V každém bodě zavedeme koncové posunutí ve směru lokální osy  $x$ . Vektor zobecněných posunutí má tvar

$$\mathbf{d} = \begin{Bmatrix} u_a \\ u_b \end{Bmatrix}. \quad (1.59)$$

Pomocí vektoru koncových posunutí  $\mathbf{d}$  a matice báзовých funkcí  $\mathbf{N}$  budeme aproximovat posunutí  $u(x)$  libovolného průřezu prvku,

$$u(x) \approx \mathbf{N}(x)\mathbf{d} = \begin{bmatrix} N_1(x) & N_2(x) \end{bmatrix} \begin{Bmatrix} u_a \\ u_b \end{Bmatrix}. \quad (1.60)$$

#### Bázové funkce

Bázové funkce  $N_i(x)$  budeme hledat ve tvaru polynomu prvního stupně,

$$N_i(x) = a_i x + b_i, \quad (1.61)$$

kde  $a_i$  a  $b_i$  jsou zatím neznámé koeficienty. Rozepsáním vztahu 1.60 do maticové podoby dostaneme výraz

$$u(x) = \begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{Bmatrix} u_a \\ u_b \end{Bmatrix}. \quad (1.62)$$

Okrajové podmínky, patrné z obrázku 1.6 jsou

$$u(0) = u_a, \quad (1.63a)$$

$$u(L) = u_b. \quad (1.63b)$$

Dosazením okrajových podmínek 1.63a a 1.63b do vztahu 1.62 dostaneme soustavu rovnic

$$\underbrace{\begin{bmatrix} 0 & 1 \\ L & 1 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{Bmatrix} u_a \\ u_b \end{Bmatrix} = \begin{Bmatrix} u_a \\ u_b \end{Bmatrix}. \quad (1.64)$$

Přénásobením rovnice zleva maticí  $\mathbf{A}^{-1}$  dostaneme

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{Bmatrix} u_a \\ u_b \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ L & 1 \end{bmatrix}^{-1} \begin{Bmatrix} u_a \\ u_b \end{Bmatrix}, \quad (1.65)$$



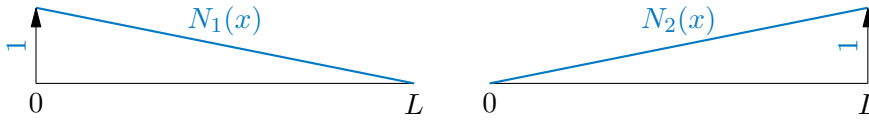
ze které je patrné, že matice koeficientů báзовých funkcí se rovná inverzní matici k matici  $\mathbf{A}$ ,

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \\ 1 & 0 \end{bmatrix}. \quad (1.66)$$

Bázové funkce, aproximující posunutí  $u(x)$  po délce prvku, tedy jsou

$$N_1(x) = 1 - \frac{x}{L}, \quad (1.67a)$$

$$N_2(x) = \frac{x}{L}. \quad (1.67b)$$



**Obr. 1.7:** Bázové funkce taženého-tlačeného prutu

Dále vypočítáme matici  $\mathbf{B}$ , která popisuje vztah mezi posuny a poměrným přetvořením. Pro přetvoření taženého-tlačeného prvku platí vztah

$$\varepsilon_x(x) = \frac{du(x)}{dx} \approx \frac{d\mathbf{N}(x)\mathbf{d}}{dx} = \mathbf{B}(x)\mathbf{d}. \quad (1.68)$$

V matici  $\mathbf{B}$  se tedy vyskytují první derivace báзовých funkcí  $N_i$  podle  $x$ ,

$$\mathbf{B}(x) = \frac{d\mathbf{N}(x)}{dx} = \frac{d}{dx} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}. \quad (1.69)$$

### Matice tuhosti

Pro výpočet matice tuhosti využijeme vztah 1.48

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega,$$

integrál přes oblast  $\Omega$  rozdělíme na integrál přes průřez a integrál po délce prutu,

$$\mathbf{K} = \int_0^L \int_A \mathbf{B}^T \mathbf{D} \mathbf{B} dA dx,$$

výraz  $\mathbf{B}^T \mathbf{D} \mathbf{B}$  nezávisí na průřezové ploše, můžeme jej tedy vytknout před vnitřní integrál,

$$\mathbf{K} = \int_0^L \mathbf{B}^T \mathbf{D} \mathbf{B} \int_A dA dx,$$

uvažujeme prizmatický prut (s konstantním průřezem), což znamená, že integrál  $\int_A dA = A$ ,

$$\mathbf{K} = \int_0^L \mathbf{B}^T \mathbf{D} \mathbf{B} A dx,$$

matice materiálové tuhosti  $\mathbf{D}$  při jednoosém namáhání odpovídá modulu pružnosti  $E$ ,

$$\mathbf{K} = \int_0^L \mathbf{B}^T E \mathbf{B} A \, dx,$$

modul pružnosti  $E$  a průřezová plocha  $A$  nejsou funkcí proměnné  $x$ , můžeme je tedy vytknout před integrál,

$$\mathbf{K} = EA \int_0^L \mathbf{B}^T \mathbf{B} \, dx,$$

dosazením za  $\mathbf{B}$  z rovnice 1.69 dostaneme

$$\mathbf{K} = EA \int_0^L \frac{1}{L^2} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} dx,$$

délka prvku  $L$  také není funkcí proměnné  $x$ , opět ji můžeme vytknout před integrál. Zároveň roznásobíme matice,

$$\mathbf{K} = \frac{EA}{L^2} \int_0^L \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx,$$

integrací dostaneme,

$$\mathbf{K} = \frac{EA}{L^2} \left[ \begin{bmatrix} x & -x \\ -x & x \end{bmatrix} \right]_0^L,$$

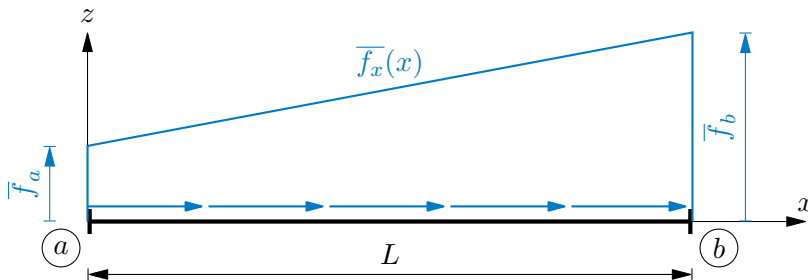
a nakonec dosazením horní a spodní meze obdržíme známý výraz pro matici tuhosti taženého-tlačeného prvku,

$$\mathbf{K} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (1.70)$$

### ■ Vektor zatížení

Předpokládejme, že tažený-tlačený prvek je zatížený lineárně se měnícím spojitým zatížením působícím v těžištové ose,

$$\bar{f}_x(x) = \frac{\bar{f}_b - \bar{f}_a}{L} x + \bar{f}_a, \quad (1.71)$$



Obr. 1.8: Zatížení taženého-tlačeného prutu

Pro vyjádření vektoru zatížení použijeme dříve odvozený vztah 1.49,

$$\mathbf{f} = \int_{\Omega} \mathbf{N}^T \bar{\mathbf{X}} d\Omega + \int_{\Gamma_p} \mathbf{N}^T \bar{\mathbf{p}} d\Gamma_p,$$

kde  $\bar{\mathbf{X}} = \mathbf{0}$  a  $\bar{\mathbf{p}} = \left\{ \bar{f}_x \quad 0 \quad 0 \right\}^T$ . Vztah 1.49 se zjednoduší na

$$\mathbf{f} = \int_0^L \mathbf{N}^T \bar{f}_x dx. \quad (1.72)$$

Dosazením za  $\mathbf{N}$  a  $\bar{f}_x$  dostaneme,

$$\mathbf{f} = \int_0^L \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \left( \frac{\bar{f}_b - \bar{f}_a}{L} x + \bar{f}_a \right) dx, \quad (1.73)$$

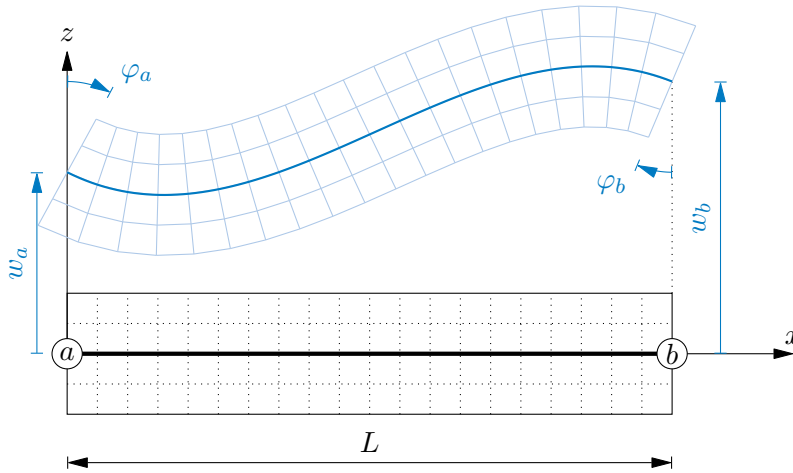
integrací dostaneme,

$$\mathbf{f} = \left[ \begin{bmatrix} \bar{f}_a x + \frac{x^2(-2\bar{f}_a + \bar{f}_b)}{2L} + \frac{x^3(\bar{f}_a - \bar{f}_b)}{3L^2} \\ \frac{\bar{f}_a x^2}{2L} + \frac{x^3(-\bar{f}_a + \bar{f}_b)}{3L^2} \end{bmatrix} \right]_0^L. \quad (1.74)$$

Po dosazení mezí a úpravě se výraz zjednoduší a obdržíme vztah pro vektor zatížení  $\mathbf{f}$ ,

$$\mathbf{f} = \begin{bmatrix} \frac{L(2\bar{f}_a + \bar{f}_b)}{6} \\ \frac{L(\bar{f}_a + 2\bar{f}_b)}{6} \end{bmatrix}. \quad (1.75)$$

### 1.3.2 Ohýbaný prvek bez vlivu smyku



**Obr. 1.9:** Deformovaná konfigurace Euler-Bernoulliho nosníku

Dle Euler-Bernoulliho hypotézy o zachování kolmosti příčných řezů k deformované střednici prutu lze posunutí  $u$  ve směru osy  $x$ , vyjádřit jako funkci vzdálenosti od osy  $z$  a úhlu pootočení příčného řezu  $\varphi(x)$  [16, s. 58]. Vektor posunutí  $\mathbf{u}$  tedy lze zapsat jako

$$\mathbf{u} = \begin{Bmatrix} u(x, z) \\ w(x) \end{Bmatrix} = \begin{Bmatrix} -z \frac{\partial w(x)}{\partial x} \\ w(x) \end{Bmatrix} = \begin{Bmatrix} z\varphi(x) \\ w(x) \end{Bmatrix} \quad (1.76)$$

Vektor poměrných deformací  $\varepsilon$  se díky přijatým předpokladům výrazně zjednoduší,

$$\varepsilon = \begin{Bmatrix} \varepsilon_x \\ \gamma_{zx} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial}{\partial x} \left( -z \frac{\partial w}{\partial x} \right) \\ \frac{\partial}{\partial z} \left( -z \frac{\partial w}{\partial x} \right) + \frac{\partial w}{\partial x} \end{Bmatrix}. \quad (1.77)$$

Nejprve upravíme člen  $\gamma_{zx}$ ,

$$\gamma_{zx} = \frac{\partial}{\partial z} \left( -z \frac{\partial w}{\partial x} \right) + \frac{\partial w}{\partial x} = -\frac{\partial w}{\partial x} + \frac{\partial w}{\partial x} = 0, \quad (1.78)$$

který je důsledkem Euler-Bernoulliho teorie nulový. Jedinou nenulovou složkou je díky přijatým předpokladům člen  $\varepsilon_x$ ,

$$\varepsilon_x(x, z) = \frac{d}{dx} \left( -z \frac{dw(x)}{dx} \right) = -z \frac{d^2 w(x)}{dx^2} = z\kappa(x), \quad (1.79)$$

kde záporně vzatou druhou derivaci průhybu  $w$  podle  $x$  označíme jako křivost  $\kappa$ ,

$$\kappa = \frac{d\varphi(x)}{dx} = -\frac{d^2 w(x)}{dx^2}. \quad (1.80)$$

Vektor napětí má také pouze jednu nenulovou složku,

$$\sigma_x(x, z) = E\varepsilon_x = E\kappa(x)z. \quad (1.81)$$

Funkci ohybového momentu lze získat integrací po průřezu podle vztahu,

$$M_y(x) = \int_A z\sigma_x(x, z) dA = E \int_A z^2 dA \kappa(x) = EI_y \kappa(x). \quad (1.82)$$

## ■ Bázové funkce

Ohýbaný prut je v rovině dán pomocí dvou uzlů,  $a$  a  $b$ . V každém koncovém uzlu zavedeme koncové posunutí ve směru lokální osy  $z$  a pootočení okolo lokální osy  $y$ , viz obr. 1.9. Vektor zobecněných posunutí má tvar

$$\mathbf{d} = \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}. \quad (1.83)$$

Posunutí  $w(x)$  budeme opět aproximovat pomocí vektoru koncových posunutí  $\mathbf{d}$  a matice bázových funkcí  $\mathbf{N}$ ,

$$w(x) \approx \mathbf{N}(x)\mathbf{d} = \begin{bmatrix} N_1(x) & N_2(x) & N_3(x) & N_4(x) \end{bmatrix} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}. \quad (1.84)$$

Pootočení průřezu  $\varphi(x)$  lze určit ze vztahu

$$\varphi(x) = -\frac{dw(x)}{dx} \approx -\frac{d\mathbf{N}(x)}{dx}\mathbf{d}. \quad (1.85)$$

Bázové funkce  $N_i$  budeme hledat ve tvaru polynomu třetího stupně,

$$N_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \quad (1.86)$$

kde  $a_i, b_i, c_i$  a  $d_i$  jsou zatím neznámé koeficienty.

Derivace bázové funkce podle  $x$  je rovna

$$\frac{dN_i(x)}{dx} = 3a_i x^2 + 2b_i x + c_i. \quad (1.87)$$

Funkce 1.86 a 1.87 lze přepsat do maticové podoby,

$$N_i(x) = \begin{bmatrix} x^3 & x^2 & x & 1 \end{bmatrix} \begin{Bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{Bmatrix}, \quad (1.88)$$

$$\frac{dN_i(x)}{dx} = \begin{bmatrix} 3x^2 & 2x & 1 & 0 \end{bmatrix} \begin{Bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{Bmatrix}, \quad (1.89)$$

Dosazením vztahu 1.88 do 1.84 získáme

$$w(x) = \begin{bmatrix} x^3 & x^2 & x & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}, \quad (1.90)$$

a dosazením 1.89 do 1.85,

$$\varphi(x) = \begin{bmatrix} -3x^2 & -2x & -1 & 0 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}. \quad (1.91)$$

Okrajové podmínky, patrné z obr. 1.9 na prvku jsou,

$$w(0) = w_a, \quad (1.92a)$$

$$\varphi(0) = \varphi_a, \quad (1.92b)$$

$$w(L) = w_b, \quad (1.92c)$$

$$\varphi(L) = \varphi_b. \quad (1.92d)$$

Dosazením okrajových podmínek do rovnic 1.90 a 1.91 dostaneme soustavu rovnic,

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ L^3 & L^2 & L & 1 \\ -3L^2 & -2L & -1 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix} = \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}, \quad (1.93)$$

přenásobením zleva maticí  $\mathbf{A}^{-1}$  dostaneme,

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ L^3 & L^2 & L & 1 \\ -3L^2 & -2L & -1 & 0 \end{bmatrix}^{-1} \begin{Bmatrix} w_a \\ \varphi_a \\ w_b \\ \varphi_b \end{Bmatrix}. \quad (1.94)$$

Pro další výpočty použijeme knihovnu SymPy<sup>†</sup> v prostředí JupyterLab<sup>‡</sup>.

Nejprve importujeme knihovnu SymPy do prostředí JupyterLab.

```
[1]: import sympy as smp
```

V dalším kroku definujeme symbolické proměnné, které budeme potřebovat pro další výpočet.

```
[2]: L = smp.symbols('L')
w_a, phi_a, w_b, phi_b = smp.symbols(
    'w_a varphi_a w_b varphi_b'
)
x = smp.symbols('x')
EI = smp.symbols('EI_y')
```

Do proměnné A uložíme matici  $\mathbf{A}$  z rovnice 1.93 a vypíšeme ji do výstupní buňky.

```
[3]: A = smp.Matrix(
    [
        [0, 0, 0, 1],
        [0, 0, -1, 0],
        [L**3, L**2, L, 1],
        [-3*L**2, -2*L, -1, 0]
    ]
)

A
```

<sup>†</sup>SymPy je open source knihovna pro symbolické výpočty v jazyce Python. Umožňuje manipulaci a řešení matematických výrazů v symbolické podobě, což zahrnuje algebraické operace, diferenciální a integrální počty, řešení rovnic, práci s maticemi a další matematické úlohy [18].

<sup>‡</sup>JupyterLab je interaktivní vývojové prostředí, které umožňuje vytváření a sdílení dokumentů obsahujících živý kód, rovnice, vizualizace a text. Je navrženo jako nástupce klasických Jupyter Notebooků a nabízí rozšířené možnosti pro práci s daty a vývoj aplikací. JupyterLab podporuje různé programovací jazyky, přičemž nejpoužívanější je Python. Uživatelé mohou využívat JupyterLab k analýze dat, strojovému učení, vizualizaci dat a vědeckému výzkumu, díky jeho flexibilitě a širokému spektru dostupných rozšíření a integrací.

[3]: 
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ L^3 & L^2 & L & 1 \\ -3L^2 & -2L & -1 & 0 \end{bmatrix}$$

Matici báзовých funkcí  $\mathbf{N}$  určíme podle vztahu 1.88.

[4]: 

```
N = smp.Matrix([[x**3, x**2, x, 1]]) @ A.inv()
N
```

[4]: 
$$\begin{bmatrix} 1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3} & -x + \frac{2x^2}{L} - \frac{x^3}{L^2} & \frac{3x^2}{L^2} - \frac{2x^3}{L^3} & \frac{x^2}{L} - \frac{x^3}{L^2} \end{bmatrix}$$

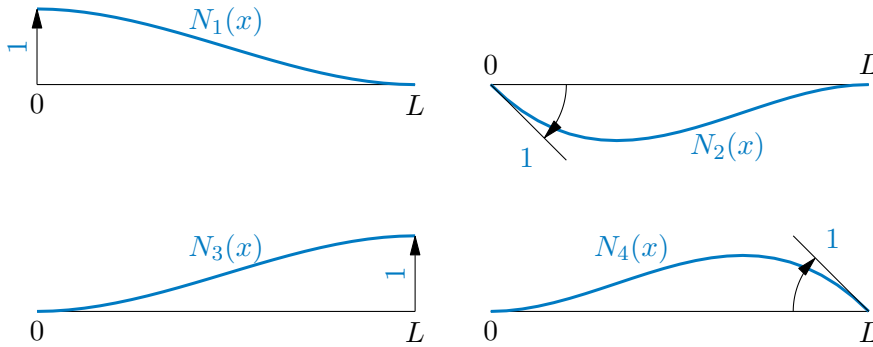
Po zjednodušení vyjde,

$$N_1(x) = 2 \left( \frac{x}{L} \right)^3 - 3 \left( \frac{x}{L} \right)^2 + 1, \quad (1.95)$$

$$N_2(x) = -x \left( \frac{x}{L} - 1 \right)^2, \quad (1.96)$$

$$N_3(x) = -2 \left( \frac{x}{L} \right)^3 + 3 \left( \frac{x}{L} \right)^2, \quad (1.97)$$

$$N_4(x) = \frac{x}{L} \left( 1 - \frac{x}{L} \right). \quad (1.98)$$



**Obr. 1.10:** Báзовé funkce ohýbaného nosníku

Dále určíme matici  $\mathbf{B}$ . Vyjdeme ze vztahu mezi křivostí a průhybem,

$$\kappa = -\frac{d^2 w(x)}{dx^2} \approx -\frac{d^2 \mathbf{N}(x)}{dx^2} \mathbf{d} = \mathbf{B} \mathbf{d}. \quad (1.99)$$

Matici  $\mathbf{B}$ , popisující vztah mezi průhybem a přetvořením, vypočítáme následovně,

$$\mathbf{B} = -\frac{d^2 \mathbf{N}(x)}{dx^2}. \quad (1.100)$$

[5]: 

```
B = -N.diff(x,x)
B.applyfunc(smp.simplify)
```

[5]: 
$$\begin{bmatrix} \frac{6(L-2x)}{L^3} & \frac{2(-2L+3x)}{L^2} & \frac{6(-L+2x)}{L^3} & \frac{2(-L+3x)}{L^2} \end{bmatrix}$$

### Matice tuhosti

Matici tuhosti vypočítáme opět podle vztahu 1.48, uvažujeme konstantní ohybovou tuhost průřezu  $EI_y$  na celém intervalu.

```
[6]: K = EI * smp.integrate(
      B.transpose() @ B,
      (x, 0, L)
    )
    K
```

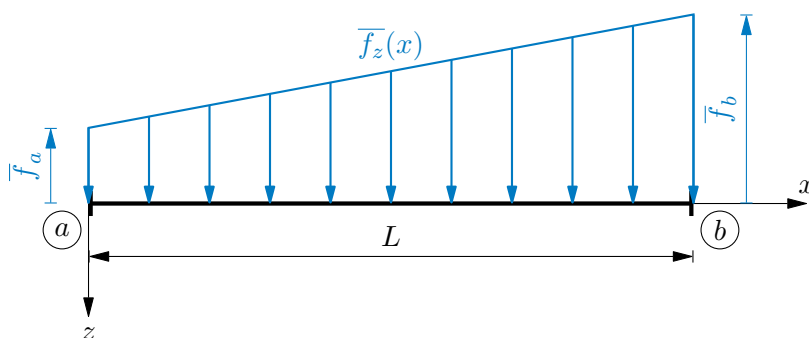
```
[7]: 
$$\begin{bmatrix} \frac{12EI_y}{L^3} & -\frac{6EI_y}{L^2} & -\frac{12EI_y}{L^3} & -\frac{6EI_y}{L^2} \\ -\frac{6EI_y}{L^2} & \frac{4EI_y}{L} & \frac{6EI_y}{L^2} & \frac{2EI_y}{L} \\ -\frac{12EI_y}{L^3} & \frac{6EI_y}{L^2} & \frac{12EI_y}{L^3} & \frac{6EI_y}{L^2} \\ -\frac{6EI_y}{L^2} & \frac{2EI_y}{L} & \frac{6EI_y}{L^2} & \frac{4EI_y}{L} \end{bmatrix}$$

```

### Vektor zatížení

Stejně, jako v případě taženého-tlačeného prutu, uvažujeme lineární spojité zatížení působící po celé délce prvku,

$$\bar{f}_z(x) = \frac{\bar{f}_b - \bar{f}_a}{L}x + \bar{f}_a. \quad (1.101)$$



**Obr. 1.11:** Zatížení Euler-Bernoulliho ohýbaného nosníku

Integrací výrazu  $\mathbf{N}^T \bar{f}_z$  po délce prutu obdržíme výraz pro vektor zatížení.

```
[7]: f_a, f_b = smp.symbols('f_a f_b')

f_z = x * (f_b - f_a) / L + f_a

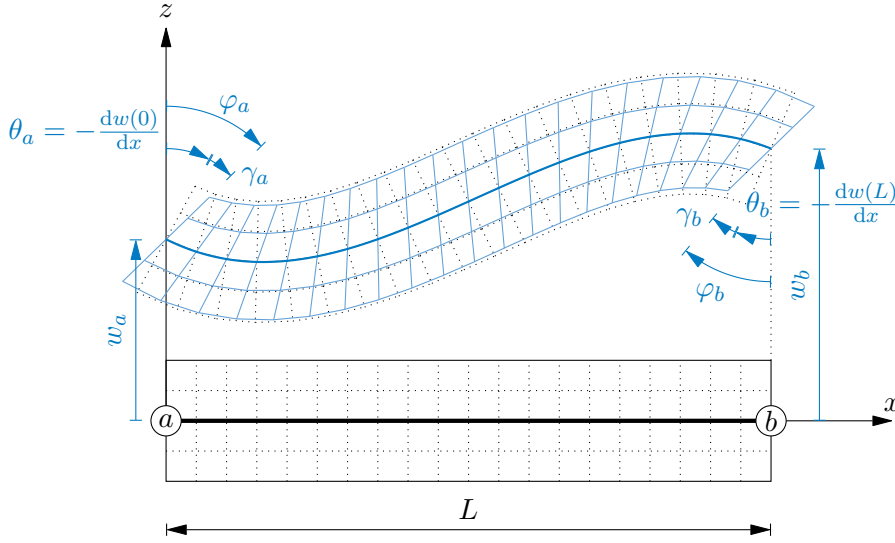
f = smp.integrate(
      N.transpose() * f_z,
      (x, 0, L)
    )
f.simplify()
```

```
[7]: 
$$\begin{bmatrix} \frac{L(7f_a+3f_b)}{20} \\ L^2 \left( -\frac{f_a}{20} - \frac{f_b}{30} \right) \\ \frac{L(3f_a+7f_b)}{20} \\ L^2 \left( \frac{f_a}{30} + \frac{f_b}{20} \right) \end{bmatrix}$$

```



### 1.3.3 Ohýbaný prvek s vlivem smyku



Obr. 1.12: Deformovaná konfigurace Timoshenkova nosníku

Základním předpokladem je, že průřez zůstává rovinný, ale ne nutně kolmý na deformovanou střednici prutu. Pole posunutí lze zapsat jako

$$\mathbf{u} = \begin{Bmatrix} u(x, z) \\ w(x, z) \end{Bmatrix} = \begin{Bmatrix} \varphi(x)z \\ w(x) \end{Bmatrix}, \quad (1.102)$$

kde  $\varphi$  je celkové zkosení, které se rovná součtu úhlu  $\theta = -\frac{dw}{dx}$  (dle Euler-Bernoulliho hypotézy, viz kap. 1.3.2) a úhlu  $\gamma$  způsobeného posouvající silou  $V_z(x)$ ,

$$\varphi(x) = \theta(x) + \gamma(x) = -\frac{dw(x)}{dx} + \gamma(x). \quad (1.103)$$

Pole poměrných deformací odpovídá

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \gamma_{xz} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial}{\partial w} + \frac{\partial u}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \frac{d\varphi(x)}{dx}z \\ \frac{dw(x)}{dx} + \varphi(x) \end{Bmatrix} = \begin{Bmatrix} \kappa(x)z \\ \frac{dw(x)}{dx} + \varphi(x) \end{Bmatrix}, \quad (1.104)$$

kde  $\kappa$  je označována jako pseudokřivost.

Pro nenulové složky napětí platí,

$$\sigma_x(x, z) = E\varepsilon_x(x, z) = E\kappa(x)z, \quad (1.105)$$

$$\tau_{xz}(x) = k G\gamma_{xz} = k G \left( \frac{dw(x)}{dx} + \varphi(x) \right), \quad (1.106)$$

kde  $k$  je korekční součinitel rozložení smykového napětí.

Místo v napětích, budeme pracovat s integrálními veličinami  $M_y, V_z$ ,

$$\begin{aligned} V_z(x) &= \int_A \tau_{xz} dA = k G \left( \frac{dw(x)}{dx} + \varphi(x) \right) \int_A dA = \\ &= k GA \left( \frac{dw(x)}{dx} + \varphi(x) \right), \end{aligned} \quad (1.107)$$

$$M_y(x) = \int_A \sigma_x z dA = E \int_A \kappa z^2 dA = EI_y \kappa. \quad (1.108)$$

Dosažením do podmínek rovnováhy 1.53 a 1.54 obdržíme dvě diferenciální rovnice,

$$\frac{d}{dx} \left( k GA \left( \frac{dw(x)}{dx} + \varphi(x) \right) \right) + \bar{f}_z = 0, \quad (1.109)$$

$$\frac{d}{dx} \left( EI_y \frac{d\varphi}{dx} \right) - k GA \left( \frac{dw(x)}{dx} + \varphi(x) \right) = 0, \quad (1.110)$$

ze kterých lze odvodit slabé řešení podle kap. 1.2.2.

## ■ Bázové funkce

Výběru bázových funkcí musí být věnována velká pozornost. Při použití polynomu stejného stupně pro aproximaci průhybu  $w(x)$  i pootočení  $\varphi(x)$  a plné integraci dochází v prvku k tzv. smykovému zamknutí (prvek se chová příliš tuze).

Například při použití polynomu prvního stupně pro aproximaci průhybu i pootočení je posouvající síla lineární funkcí,

$$V_z(x) = k GA \left( \underbrace{\frac{dw(x)}{dx}}_{\text{konstantní}} + \underbrace{\varphi(x)}_{\text{lineární}} \right),$$

a ohybový moment konstantní,

$$M_y(x) = EI_y \underbrace{\frac{d\varphi(x)}{dx}}_{\text{konstantní}},$$

což je v přímém rozporu se Schwedlerovou větou,

$$\frac{dM_y(x)}{dx} = V_z(x).$$

Řešením může být použití redukované integrace nebo zvolení polynomu vyššího stupně pro aproximaci průhybu.

Pro určení bázových funkcí použijeme postup, který byl použit v [23, kap. 5.6], v české odborné literatuře například v [5, kap. 2.2.2].

Vyjdeme z přetvoření nezátíženého nosníku, způsobeného pouze přemístěním a pootočením koncových průřezů, což odpovídá bázovým funkcím, které byly odvozeny v kap. 1.3.2.

Stejně jako v případě Hermitovského prvku, založeného na Euler-Bernoulliho teorii, použijeme knihovnu SymPy v prostředí JupyterLab.

[1]: `import sympy as smp`

Definujeme proměnné, které jsou potřeba pro výpočet, jedná se proměnnou  $x$ , délku nosníku  $L$ , posunutí koncových průřezů  $w_{a,b}$ , pootočení tečny k ose prutu  $\theta_{a,b}$  a pootočení průřezu  $\varphi_{a,b}$ , znázorněných na obr. 1.12, dále budeme potřebovat modul pružnosti  $E$ , moment setrvačnosti k těžišťové ose  $I_y$ , korekční součinitel rozložení smykového napětí  $k$ , modul pružnosti ve smyku  $G$ , plochu průřezu  $A$  a smykovou štiřlost  $\Omega$ .

[2]: `xi, x, L = smp.symbols('xi x L')`  
  
`w_a, phi_a, w_b, phi_b, theta_a, theta_b = smp.symbols(`

```

    'w_a varphi_a w_b varphi_b theta_a theta_b'
)

E, I, k, G, A = smp.symbols('E I~k~G A')

omega = smp.symbols('Omega')
omega_val = 12 * E * I / (k * G * A * L**2)

```

Definujeme matici báзовých funkcí aproximující průhyb nosníku bez vlivu smyku.

```

[3]: N_eb = smp.Matrix([
    [
        2 * (x/L)**3 - 3 * (x/L)**2 + 1,
        -x * (x/L - 1)**2,
        -2 * (x/L)**3 + 3*(x/L)**2,
        x**2/L * (1 - x/L)
    ]
])
N_eb

```

```

[3]: [1 - 3x^2/L^2 + 2x^3/L^3  -x(-1 + x/L)^2  3x^2/L^2 - 2x^3/L^3  x^2*(1-x/L)/L]

```

Průhyb nezátíženého nosníku lze aproximovat následující funkcí, kde  $w_{a,b}$  jsou posunutí průřezu ve směru osy  $z$  a  $\theta_{a,b}$  je záporně vzatá derivace průhybu podle  $x$ , viz 1.103.

```

[4]: w = (N_eb @ smp.Matrix([w_a, theta_a, w_b, theta_b]))[0]
w

```

```

[4]: -theta_a*x*(-1 + x/L)^2 + w_a*(1 - 3x^2/L^2 + 2x^3/L^3) + w_b*(3x^2/L^2 - 2x^3/L^3) +
theta_b*x^2*(1 - x/L)/L

```

Cílem je najít aproximaci funkce  $\varphi(x)$ . Při zatížení  $\overline{f_z} = 0$  podle vztahu 1.56 platí,

$$\frac{dV_z}{dx} = 0,$$

což znamená, že funkce posouvající síly musí být konstantní. Ze vztahu 1.107 vyplývá, že pro prizmatický pružný prut je zkosení  $\gamma$  také konstantní,

$$-\theta(x) + \varphi(x) = \frac{dw(x)}{dx} + \varphi(x) = \gamma = \text{konstantní}, \quad (1.111)$$

Parametr  $\gamma$  vyloučíme z požadavku, aby aproximace  $w$  a  $\varphi$  splnily momentovou podmínku rovnováhy 1.57,

$$\begin{aligned} \frac{dM_y(x)}{dx} &= V_z, \\ EI_y \frac{d^2\varphi(x)}{dx^2} &= kGA\gamma, \\ \gamma &= -\frac{EI_y}{kGA} \frac{d^3w(x)}{dx^3}. \end{aligned} \quad (1.112)$$

```
[5]: gamma = - E * I * w.diff(x, x, x) / (k * A * G)
      gamma
```

$$[5]: -\frac{6EI \left(-\theta_a - \theta_b + \frac{2w_a}{L} - \frac{2w_b}{L}\right)}{AGL^2k}$$

Vztah upravíme tak, aby obsahoval smykovou štiřlost  $\Omega$ ,

$$\Omega = \frac{12EI_y}{kGAL^2}. \quad (1.113)$$

```
[6]: gamma = (gamma / (omega_val)).expand() * omega
      gamma
```

$$[6]: \Omega \left( \frac{\theta_a}{2} + \frac{\theta_b}{2} - \frac{w_a}{L} + \frac{w_b}{L} \right)$$

Nyní můžeme pomocí vztahu 1.111 vyjádřit úhly  $\theta_a$  a  $\theta_b$  a vyřešit soustavu dvou rovnic,

$$\theta_a - \varphi_a + \gamma = 0, \quad (1.114)$$

$$\theta_b - \varphi_b + \gamma = 0. \quad (1.115)$$

```
[7]: solution = smy.solve(
      [theta_a - phi_a + gamma, theta_b - phi_b + gamma],
      (theta_a, theta_b)
    )
```

```
[8]: theta_a_solved = solution[theta_a]
      theta_a_solved.collect([w_a, w_b, phi_a, phi_b])
```

$$[8]: \frac{-L\Omega\varphi_b + 2\Omega w_a - 2\Omega w_b + \varphi_a(L\Omega + 2L)}{2L\Omega + 2L}$$

```
[9]: theta_b_solved = solution[theta_b]
      theta_b_solved.collect([w_a, w_b, phi_a, phi_b])
```

$$[9]: \frac{-L\Omega\varphi_a + 2\Omega w_a - 2\Omega w_b + \varphi_b(L\Omega + 2L)}{2L\Omega + 2L}$$

Podle vztahu 1.111 vypočítáme pootočení obecného průřezu,  $\varphi(x)$ ,

```
[10]: phi_y = gamma - w.diff(x)
      phi_y
```

$$[10]: \Omega \left( \frac{\theta_a}{2} + \frac{\theta_b}{2} - \frac{w_a}{L} + \frac{w_b}{L} \right) + \theta_a \left( -1 + \frac{x}{L} \right)^2 - w_a \left( -\frac{6x}{L^2} + \frac{6x^2}{L^3} \right) - w_b \left( \frac{6x}{L^2} - \frac{6x^2}{L^3} \right) + \frac{2\theta_a x \left( -1 + \frac{x}{L} \right)}{L} - \frac{2\theta_b x \left( 1 - \frac{x}{L} \right)}{L} + \frac{\theta_b x^2}{L^2}$$

Dosazením vypočítaných úhlů  $\theta_a$  a  $\theta_b$  do funkce pro průhyb,  $w(x)$ , obdržíme aproximaci vyjádřenou v závislosti na posunutí  $w_a, w_b$  a pootočení  $\varphi_a, \varphi_b$  koncových průřezu prutu.

```
[11]: w = w.subs(
      [
        (theta_a, theta_a_solved),
```

```
(theta_b, theta_b_solved)
],
)
w
```

$$[11]: \quad w_a \left( 1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3} \right) + w_b \left( \frac{3x^2}{L^2} - \frac{2x^3}{L^3} \right) - \frac{x \left( -1 + \frac{x}{L} \right)^2 (L\Omega\varphi_a - L\Omega\varphi_b + 2L\varphi_a + 2\Omega w_a - 2\Omega w_b)}{2L\Omega + 2L} + \frac{x^2 \cdot \left( 1 - \frac{x}{L} \right) (-L\Omega\varphi_a + L\Omega\varphi_b + 2L\varphi_b + 2\Omega w_a - 2\Omega w_b)}{L(2L\Omega + 2L)}$$

Dosažením vypočítaných úhlů  $\theta_a$  a  $\theta_b$  do odvozené funkce pro pootočení,  $\varphi(x)$ , obdržíme aproximaci vyjádřenou v závislosti na posunutí  $w_a, w_b$  a pootočení  $\varphi_a, \varphi_b$  koncových průřezu prutu.

```
[12]: phi_y = phi_y.subs(
[
(theta_a, theta_a_solved),
(theta_b, theta_b_solved)
])
phi_y
```

$$[12]: \quad \Omega \left( \frac{-L\Omega\varphi_a + L\Omega\varphi_b + 2L\varphi_b + 2\Omega w_a - 2\Omega w_b}{2 \cdot (2L\Omega + 2L)} + \frac{L\Omega\varphi_a - L\Omega\varphi_b + 2L\varphi_a + 2\Omega w_a - 2\Omega w_b}{2 \cdot (2L\Omega + 2L)} - \frac{w_a}{L} + \frac{w_b}{L} \right) - w_a \left( -\frac{6x}{L^2} + \frac{6x^2}{L^3} \right) - w_b \left( \frac{6x}{L^2} - \frac{6x^2}{L^3} \right) + \frac{\left( -1 + \frac{x}{L} \right)^2 (L\Omega\varphi_a - L\Omega\varphi_b + 2L\varphi_a + 2\Omega w_a - 2\Omega w_b)}{2L\Omega + 2L} + \frac{2x \left( -1 + \frac{x}{L} \right) (L\Omega\varphi_a - L\Omega\varphi_b + 2L\varphi_a + 2\Omega w_a - 2\Omega w_b)}{L(2L\Omega + 2L)} - \frac{2x \left( 1 - \frac{x}{L} \right) (-L\Omega\varphi_a + L\Omega\varphi_b + 2L\varphi_b + 2\Omega w_a - 2\Omega w_b)}{L(2L\Omega + 2L)} + \frac{x^2 (-L\Omega\varphi_a + L\Omega\varphi_b + 2L\varphi_b + 2\Omega w_a - 2\Omega w_b)}{L^2 \cdot (2L\Omega + 2L)}$$

Bázové funkce, aproximující průběh pootočení průřezu, získáme jako koeficienty jednotlivých zobecněných posunů z rovnice pro pootočení průřezu.

```
[13]: N_phi = smp.Matrix(
[
phi_y.expand().coeff(w_a),
phi_y.expand().coeff(phi_a),
phi_y.expand().coeff(w_b),
phi_y.expand().coeff(phi_b)
])
.applyfunc(smp.simplify).applyfunc(smp.factor)

N_phi
```

$$[13]: \quad \begin{bmatrix} -\frac{6x(-L+x)}{L^3(\Omega+1)} \\ \frac{(-L+x)(-L\Omega-L+3x)}{L^2(\Omega+1)} \\ \frac{6x(-L+x)}{L^3(\Omega+1)} \\ \frac{x(L\Omega-2L+3x)}{L^2(\Omega+1)} \end{bmatrix}$$

Bázové funkce, aproximující průběh posunutí, získáme jako koeficienty jednotlivých zobecněných posunů z rovnice pro průhyb.

```
[14]: N_w = smp.Matrix([
    [
        w.expand().coeff(w_a),
        w.expand().coeff(phi_a),
        w.expand().coeff(w_b),
        w.expand().coeff(phi_b)
    ]
]).applyfunc(smp.simplify).applyfunc(smp.factor)

N_w
```

$$[14]: \begin{bmatrix} \frac{(-L+x)(-L^2\Omega-L^2-Lx+2x^2)}{L^3(\Omega+1)} \\ -\frac{x(-L+x)(-L\Omega-2L+2x)}{2L^2(\Omega+1)} \\ -\frac{x(-L^2\Omega-3Lx+2x^2)}{L^3(\Omega+1)} \\ -\frac{x(-L+x)(L\Omega+2x)}{2L^2(\Omega+1)} \end{bmatrix}$$

Dále báze funkce uložíme do matice báze funkcí  $\mathbf{N}$ .

```
[15]: N = smp.Matrix([
    N_phi.transpose(),
    N_w.transpose()
]).applyfunc(smp.simplify).applyfunc(smp.factor)

N
```

$$[15]: \begin{bmatrix} -\frac{6x(-L+x)}{L^3(\Omega+1)} & \frac{(-L+x)(-L\Omega-L+3x)}{L^2(\Omega+1)} & \frac{6x(-L+x)}{L^3(\Omega+1)} & \frac{x(L\Omega-2L+3x)}{L^2(\Omega+1)} \\ \frac{(-L+x)(-L^2\Omega-L^2-Lx+2x^2)}{L^3(\Omega+1)} & -\frac{x(-L+x)(-L\Omega-2L+2x)}{2L^2(\Omega+1)} & -\frac{x(-L^2\Omega-3Lx+2x^2)}{L^3(\Omega+1)} & -\frac{x(-L+x)(L\Omega+2x)}{2L^2(\Omega+1)} \end{bmatrix}$$

Po určení báze funkcí můžeme přistoupit k výpočtu matice  $\mathbf{B}$ , která popisuje vztah mezi zobecněnými posuny a deformací. Ze vztahu pro pole deformace,

$$\varepsilon = \left\{ \frac{\frac{d\varphi(x)}{dx} z}{\frac{dw(x)}{dx} + \varphi(x)} \right\} \approx \mathbf{B} \mathbf{d}. \quad (1.116)$$

je zřejmé, že matice  $\mathbf{B}$  bude mít tvar

$$\mathbf{B} = \left\{ \frac{\frac{d\mathbf{N}_\varphi}{dx}}{\frac{d\mathbf{N}_w}{dx} + \mathbf{N}_\varphi} \right\}. \quad (1.117)$$

```
[16]: B = smp.Matrix([
    N_phi.transpose().diff(x),
    N_phi.transpose() + (N_w.diff(x)).transpose()
]).applyfunc(smp.simplify).applyfunc(smp.factor)

B
```

$$[16]: \begin{bmatrix} -\frac{6(-L+2x)}{L^3(\Omega+1)} & \frac{-L\Omega-4L+6x}{L^2(\Omega+1)} & \frac{6(-L+2x)}{L^3(\Omega+1)} & \frac{L\Omega-2L+6x}{L^2(\Omega+1)} \\ -\frac{\Omega}{L(\Omega+1)} & \frac{\Omega}{2(\Omega+1)} & \frac{\Omega}{L(\Omega+1)} & \frac{\Omega}{2(\Omega+1)} \end{bmatrix}$$

### Matice tuhosti

Matice materiálové tuhosti je

$$\mathbf{D} = \begin{bmatrix} EI_y & 0 \\ 0 & kGA \end{bmatrix}, \quad (1.118)$$

kde výraz  $kGA$  nahradíme podle vztahu 1.113 ekvivalentním výrazem  $\frac{12EI_y}{\Omega L^2}$ . Tato úprava je nezbytná pro zajištění správného zjednodušení při výpočtu matice tuhosti.

```
[17]: D = smp.Matrix([
      [E * I, 0],
      [0, 12*E*I/ (omega * L**2)],
    ])

      D
```

```
[17]:
```

$$\begin{bmatrix} EI & 0 \\ 0 & \frac{12EI}{L^2\Omega} \end{bmatrix}$$

Konečně můžeme přejít k výpočtu matice tuhosti podle známého vztahu,

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega.$$

```
[18]: K = smp.integrate(
      B.transpose() @ D @ B,
      (x, 0, L)
    ).applyfunc(smp.expand).applyfunc(smp.factor)

      K
```

```
[18]:
```

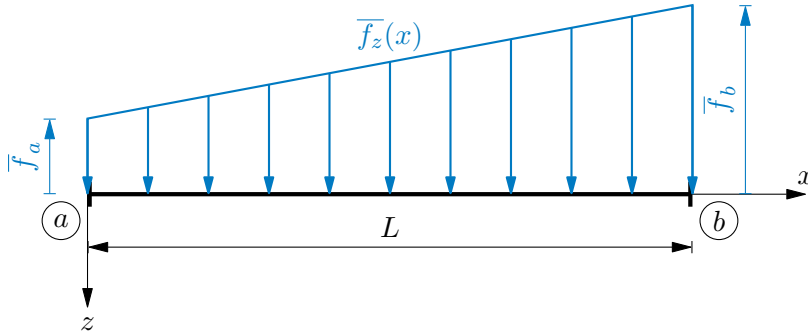
$$\begin{bmatrix} \frac{12EI}{L^3(\Omega+1)} & -\frac{6EI}{L^2(\Omega+1)} & -\frac{12EI}{L^3(\Omega+1)} & -\frac{6EI}{L^2(\Omega+1)} \\ -\frac{6EI}{L^2(\Omega+1)} & \frac{EI(\Omega+4)}{L(\Omega+1)} & \frac{6EI}{L^2(\Omega+1)} & -\frac{EI(\Omega-2)}{L(\Omega+1)} \\ -\frac{12EI}{L^3(\Omega+1)} & \frac{6EI}{L^2(\Omega+1)} & \frac{12EI}{L^3(\Omega+1)} & \frac{6EI}{L^2(\Omega+1)} \\ -\frac{6EI}{L^2(\Omega+1)} & -\frac{EI(\Omega-2)}{L(\Omega+1)} & \frac{6EI}{L^2(\Omega+1)} & \frac{EI(\Omega+4)}{L(\Omega+1)} \end{bmatrix}$$

Lze si všimnout, že pro  $\Omega = 0$  obdržíme matici tuhosti Hermitovského nosníku s kubickou aproximací pro Euler-Bernoulliho model, odvozeného v kap. 1.3.2.

### Vektor zatížení

Pro na prvku platí stejné předpoklady jako u taženého-tlačeného prvku i Euler-Bernoulliho nosníku, uvažujeme lineární spojité zatížení působící na celé délce prvku,

$$\overline{f_z}(x) = \frac{\overline{f_b} - \overline{f_a}}{L}x + \overline{f_a}. \quad (1.119)$$



Obr. 1.13: Zatížení Timoshenkova ohýbaného nosníku

Integrací výrazu  $\mathbf{N}^T \bar{f}_z$  po délce prutu obdržíme výraz pro vektor zatížení.

[19]:

```
f_a, f_b = smp.symbols('f_a f_b')

f_z = x * (f_b - f_a) / L + f_a

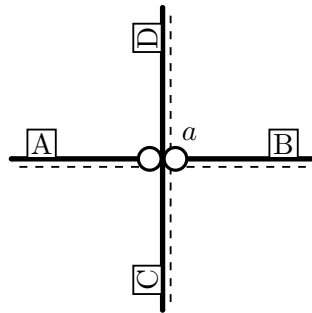
f = smp.integrate(
    N.transpose() * f_z,
    (x, 0, L)
).applyfunc(smp.expand).applyfunc(smp.factor)
f
```

[19]:

$$\begin{bmatrix} \frac{f_a + f_b}{2(\Omega + 1)} & \frac{L(20\Omega f_a + 10\Omega f_b + 21f_a + 9f_b)}{60(\Omega + 1)} \\ \frac{L(4\Omega f_a + 2\Omega f_b + f_a - f_b)}{12(\Omega + 1)} & -\frac{L^2 \cdot (5\Omega f_a + 5\Omega f_b + 6f_a + 4f_b)}{120(\Omega + 1)} \\ -\frac{f_a + f_b}{2(\Omega + 1)} & \frac{L(10\Omega f_a + 20\Omega f_b + 9f_a + 21f_b)}{60(\Omega + 1)} \\ \frac{L(2\Omega f_a + 4\Omega f_b - f_a + f_b)}{12(\Omega + 1)} & \frac{L^2 \cdot (5\Omega f_a + 5\Omega f_b + 4f_a + 6f_b)}{120(\Omega + 1)} \end{bmatrix}$$

## 1.4 Statická kondenzace

Při výpočtu prutových konstrukcí se lze často setkat s kloubovým připojením prutů. V případě na obr. 1.14 se ve styčnicku stýkají 4 pruty, posunutí  $u_a$  a  $w_a$  jsou jednoznačně popsány, ale pootočení koncových průřezů levého a pravého prutu jsou různá od pootočení  $\varphi_a$ . Ve styčnicku tedy vzniká celkem 5 neznámých, 2 posunutí a 3 pootočení. Podmínky rovnováhy pro pootočení kloubově připojených prutů požadují, aby ohybový moment ve styčnicku byl roven nule. Podmínku nulového koncového momentu lze uplatnit přímo v matici tuhosti prutového prvku a do globálních podmínek rovnováhy přičíst síly z prutu, který je typu vetknutí-kloub.



Obr. 1.14: Styčník s kloubově připojenými pruty



Matici tuhosti prvku, vektor uzlových posunutí a vektor zatížení rozdělíme na submatice odpovídající ponechaným (index  $a$ ) a kondenzovaným (index  $b$ ) stupňům volnosti,

$$\begin{bmatrix} \mathbf{K}_{aa} & \mathbf{K}_{ab} \\ \mathbf{K}_{ba} & \mathbf{K}_{bb} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_a \\ \mathbf{d}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_a \\ \mathbf{f}_b \end{Bmatrix}. \quad (1.120)$$

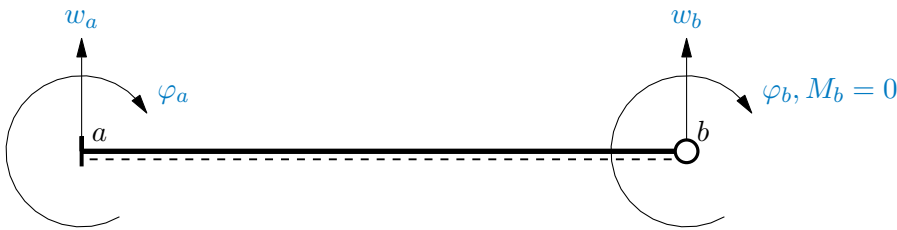
Z druhé rovnice vyjádříme  $\mathbf{d}_b$ ,

$$\begin{aligned} \mathbf{K}_{ba}\mathbf{d}_a + \mathbf{K}_{bb}\mathbf{d}_b &= \mathbf{f}_b \\ \mathbf{d}_b &= \mathbf{K}_{bb}^{-1}(\mathbf{f}_b - \mathbf{K}_{ba}\mathbf{d}_a). \end{aligned} \quad (1.121)$$

Po dosazení do první rovnice obdržíme

$$\begin{aligned} \mathbf{K}_{aa}\mathbf{d}_a + \mathbf{K}_{ab}(\mathbf{K}_{bb}^{-1}(\mathbf{f}_b - \mathbf{K}_{ba}\mathbf{d}_a)) &= \mathbf{f}_a \\ (\mathbf{K}_{aa} - \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{K}_{ba})\mathbf{d}_a &= \mathbf{f}_a - \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{f}_b \\ \mathbf{K}^*\mathbf{d}_a &= \mathbf{f}^*. \end{aligned} \quad (1.122)$$

Uvedeme příklad kondenzace matice tuhosti pro prut typu vetknutí-klobuk z matice tuhosti odvozené v kap. 1.3.2.



**Obr. 1.15:** Prut typu vetknutí-klobuk

Importujeme knihovnu SymPy.

```
[1]: import sympy as smp
```

Definujeme proměnné.

```
[2]: E, I, L = smp.symbols('E I~L')
w_a, phi_a, w_b, phi_b = smp.symbols(
    'w_a varphi_a w_b varphi_b'
)
```

Definujeme matici tuhosti  $\mathbf{K}$ .

```
[3]: K = E * I / L * smp.Matrix(
    [
        [12/L**2, -6/L, -12/L**2, -6/L],
        [-6/L, 4, 6/L, 2],
        [-12/L**2, 6/L, 12/L**2, 6/L],
        [-6/L, 2, 6/L, 4]
    ]
)

K
```

```
[3]:
```

$$\begin{bmatrix} \frac{12EI}{L^3} & -\frac{6EI}{L^2} & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{4EI}{L} & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{12EI}{L^3} & \frac{6EI}{L^2} & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{2EI}{L} & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}$$

Matici  $\mathbf{K}$  rozdělíme na submatice.

```
[4]: K_aa = K[0:3, 0:3]
      K_aa
```

```
[4]:  $\begin{bmatrix} \frac{12EI}{L^3} & -\frac{6EI}{L^2} & -\frac{12EI}{L^3} \\ -\frac{6EI}{L^2} & \frac{4EI}{L} & \frac{6EI}{L^2} \\ -\frac{12EI}{L^3} & \frac{6EI}{L^2} & \frac{12EI}{L^3} \end{bmatrix}$ 
```

```
[5]: K_ab = K[0:3, 3:]
      K_ab
```

```
[5]:  $\begin{bmatrix} -\frac{6EI}{L^2} \\ \frac{2EI}{L} \\ \frac{6EI}{L^2} \end{bmatrix}$ 
```

```
[6]: K_ba = K[3:, 0:3]
      K_ba
```

```
[6]:  $\begin{bmatrix} -\frac{6EI}{L^2} & \frac{2EI}{L} & \frac{6EI}{L^2} \end{bmatrix}$ 
```

```
[7]: K_bb = K[3:, 3:]
      K_bb
```

```
[7]:  $\begin{bmatrix} \frac{4EI}{L} \end{bmatrix}$ 
```

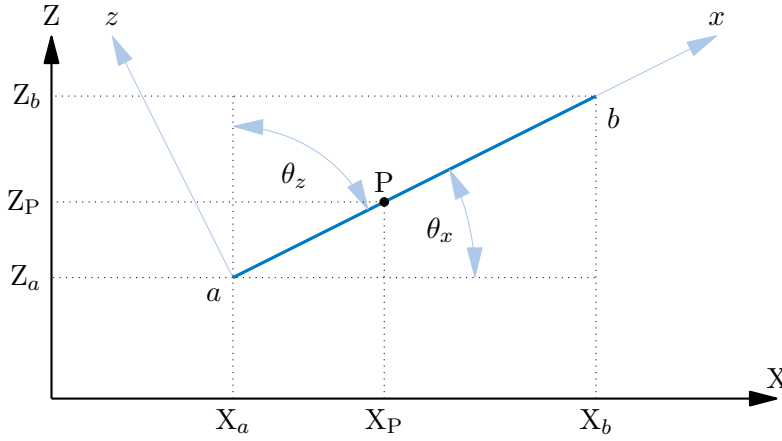
Provedeme statickou kondenzaci podle vztahu 1.122.

```
[8]: K_condensed = K_aa - K_ab @ K_bb.inv() @ K_ba
      K_condensed
```

```
[8]:  $\begin{bmatrix} \frac{3EI}{L^3} & -\frac{3EI}{L^2} & -\frac{3EI}{L^3} \\ -\frac{3EI}{L^2} & \frac{3EI}{L} & \frac{3EI}{L^2} \\ -\frac{3EI}{L^3} & \frac{3EI}{L^2} & \frac{3EI}{L^3} \end{bmatrix}$ 
```

## 1.5 Transformační matice

Matice tuhosti a vektory zatížení jsme v předešlých kapitolách odvodili za předpokladu, že lokální souřadnicový systém je shodně orientovaný s globálním souřadnicovým systémem. Aby bylo možné sestavit podmínky rovnováhy na celé konstrukci, je nutné pracovat v globálním systému souřadnic.



Obr. 1.16: Transformace souřadnic

Polohu bodu P na obr. 1.16 v lokálním souřadném systému lze vyjádřit pomocí vztahu

$$\mathbf{x}_{P,\ell} = \mathbf{T} \mathbf{x}_{P,g}, \quad (1.123)$$

kde  $\mathbf{T}$  je transformační matice,  $\mathbf{x}_{P,\ell}$  je polohový vektor v lokálním souřadném systému  $xyz$  a  $\mathbf{x}_{P,g}$  je polohový vektor v globálním souřadném systému  $XYZ$ .

Rovnici 1.123 lze pro lepší představu rozepsat po složkách,

$$\begin{Bmatrix} x_P \\ y_P \\ z_P \end{Bmatrix} = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{bmatrix}^T \begin{Bmatrix} X_P \\ Y_P \\ Z_P \end{Bmatrix}, \quad (1.124)$$

kde  $\mathbf{e}_i$  označuje jednotkové vektory ve směru lokálních souřadnicových os v globálním systému souřadnic.

Vektor  $\mathbf{e}_x$  je ve 3D kartézské soustavě souřadnic jednoznačně dán polohou bodů  $a$  a  $b$ ,

$$\mathbf{e}_x = \left\{ \cos(\theta_x) \quad \cos(\theta_y) \quad \cos(\theta_z) \right\}^T, \quad (1.125)$$

kde

$$\begin{aligned} \cos(\theta_x) &= \frac{X_b - X_a}{L} = \frac{\Delta X}{L} \\ \cos(\theta_y) &= \frac{Y_b - Y_a}{L} = \frac{\Delta Y}{L} \\ \cos(\theta_z) &= \frac{Z_b - Z_a}{L} = \frac{\Delta Z}{L} \\ L &= \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2} \end{aligned} \quad (1.126)$$

Směrové vektory  $\mathbf{e}_y$  a  $\mathbf{e}_z$  se určí pomocí dalšího bodu, případně vektoru, ležícího v lokální rovině  $xy$  nebo  $xz$ . Při výpočtu prutového prvku v rovině  $xz$  je pomocný vektor  $\mathbf{v}$  rovnoběžný s globální osou  $Y$ , v tom případě platí

$$\begin{aligned} \mathbf{e}_x &= \left\{ \frac{\Delta X}{L} \quad 0 \quad \frac{\Delta Z}{L} \right\}^T, \\ \mathbf{e}_y &= \left\{ 0 \quad 1 \quad 0 \right\}^T, \\ \mathbf{e}_z &= \left\{ -\frac{\Delta Z}{L} \quad 0 \quad \frac{\Delta X}{L} \right\}^T, \end{aligned} \quad (1.127)$$

a transformační matice má tvar

$$\mathbf{T} = \begin{bmatrix} \frac{\Delta X}{L} & 0 & \frac{\Delta Z}{L} \\ 0 & 1 & 0 \\ -\frac{\Delta Z}{L} & 0 & \frac{\Delta X}{L} \end{bmatrix}. \quad (1.128)$$

Při ortogonální transformaci souřadnic je transformační matice  $\mathbf{T}$  ortogonální, to znamená, že

$$\mathbf{T}^{-1} = \mathbf{T}^T \quad (1.129)$$

Pro vektor koncových posunutí a vektor zatížení lze podle vztahu 1.123 psát

$$\begin{aligned} \mathbf{d}_\ell &= \mathbf{T} \mathbf{d}_g, \\ \mathbf{f}_\ell &= \mathbf{T} \mathbf{f}_g. \end{aligned} \quad (1.130)$$

Dosazením vztahu 1.130 do podmínek rovnováhy odvodíme vztah pro transformaci matice tuhosti z lokálního do souřadného systému,

$$\begin{aligned} \mathbf{K}_\ell \mathbf{d}_\ell - \mathbf{f}_\ell &= \mathbf{0} \\ \mathbf{K}_\ell \mathbf{T} \mathbf{d}_g - \mathbf{T} \mathbf{f}_g &= \mathbf{0} \\ \underbrace{\mathbf{T}^T \mathbf{K}_\ell \mathbf{T}}_{\mathbf{K}_g} \mathbf{d}_g - \mathbf{f}_g &= \mathbf{0}, \end{aligned} \quad (1.131)$$

globální matici tuhosti vypočítáme podle vztahu

$$\mathbf{K}_g = \mathbf{T}^T \mathbf{K}_\ell \mathbf{T}. \quad (1.132)$$

## 1.6 Implementace

Knihovna `framesss` je open-source nástroj vyvinutý v rámci této diplomové práce určená pro výpočty prutových konstrukcí metodou konečných prvků. cílem této knihovny je nabídnout bezplatný a uživatelsky přívětivý nástroj, který lze používat jak v akademické sféře, tak ve stavební praxi.

Knihovna je plně objektově orientovaná a napsaná v jazyce Python, což zajišťuje přehlednost a modularitu kódu. Tento přístup usnadňuje údržbu a rozšiřování funkcionality knihovny. Implementace knihovny je založena na využití klíčových vědeckých a technických knihoven, jako jsou NumPy [13] a SciPy [33], které jsou nezbytné pro efektivní provádění numerických výpočtů.

Zdrojový kód `framesss` je veřejně dostupný na platformě GitHub na adrese <https://github.com/DanBeranek/framesss>. Dokumentace knihovny, která je k dispozici na adrese <https://danberanek.github.io/framesss/index.html>, poskytuje podrobné informace o funkcionalitě, implementaci a použití knihovny.

### 1.6.1 Použití externího open-source softwaru

Při vývoji knihovny `framesss` bylo využito komponent z existujících open-source softwarů. Tento krok byl proveden s úmyslem stavět na robustních základech poskytnutých open-source komunitou.

Základní koncept architektury byl převzat ze softwaru LESM [25], který byl původně vyvinut v prostředí MATLAB a licencován pod MIT licenci.

Kromě toho byly při vývoji knihovny použity následující open-source knihovny a nástroje:

- **NumPy** [13]: Knihovna pro numerické výpočty v Pythonu, která poskytuje podporu pro práci s více-dimenzionálními poli a matice. NumPy je základním stavebním kamenem pro efektivní provádění numerických operací.
- **SciPy** [33]: Nadstavba nad NumPy, která rozšiřuje jeho funkčnost o pokročilé matematické, vědecké a technické nástroje, včetně řešení diferenciálních rovnic a optimalizace.
- **SymPy** [18]: Knihovna pro symbolické výpočty, která byla využita pro odvození matic tuhostí a vektorů zatížení.

Pro usnadnění vývoje a zajištění kvality kódu byly použity následující nástroje:

- **Git** [11]: Distribuovaný verzovací systém, který umožňuje správu verzí kódu.
- **GitHub** [12]: Platforma pro hostování a správu verzí kódu, která poskytuje nástroje pro spolupráci a integraci s dalšími nástroji.
- **pre-commit** [22]: Nástroj pro automatizaci úloh před commitováním kódu, který zajišťuje, že všechny změny projdou definovanými kontrolami.
- **Nox** [20]: Automatizační nástroj pro správu testovacích prostředí a spouštění úloh.
- **Ruff** [26]: Rychlý linter pro Python, který pomáhá udržovat konzistenci kódu a detekovat chyby.
- **Poetry** [21]: Nástroj pro správu závislostí a balíčkování, který zjednodušuje správu projektů v Pythonu.
- **Black** [6]: Formátovač kódu, který automaticky formátuje kód podle stanovených pravidel.
- **mypy** [19]: Statický typový kontroler pro Python, který pomáhá detekovat potenciální chyby v kódu.
- **pytest** [24]: Framework pro psaní a spouštění testů.
- **Sphinx** [29]: Nástroj pro generování dokumentace, který umožňuje snadnou tvorbu a správu dokumentace kódu.
- **SonarCloud** [28]: Platforma pro analýzu kvality kódu, která poskytuje nástroje pro kontinuální integraci a hodnocení kódu.

## 1.6.2 Instalace

Knihovna framesss je navržena tak, aby byla snadno dostupná a uživatelsky přívětivá nejen z hlediska jejího používání, ale i instalace. Pro instalaci této knihovny můžete využít pip, což je standardní správce balíčků pro Python, který zjednodušuje správu softwarových závislostí a zaručuje rychlé a snadné nasazení knihovny framesss.

Pro instalaci knihovny framesss stačí spustit následující příkaz v terminálu:

```
pip install framesss
```

Tento příkaz stáhne nejnovější verzi knihovny framesss z Python Package Index (PyPI) a nainstaluje ji spolu s nezbytnými závislostmi. Před instalací se ujistěte, že máte na svém systému již nainstalovaný Python a pip. Pokud narazíte během instalace na jakékoli problémy, zkuste nejprve aktualizovat pip pomocí příkazu `pip install --upgrade pip` a poté pokračujte v reinstalaci balíčku.

Pro uživatele, kteří potřebují integrovat knihovnu framesss do větších projektů nebo virtuálních prostředí, doporučujeme instalaci v rámci Python virtuálního prostředí. Tento postup pomáhá lépe spravovat závislosti a udržuje vaše Python projekty organizované a bez konfliktů.

Pro instalaci knihovny framesss v Python virtuálním prostředí, postupujte podle následujících kroků pro váš operační systém.

#### ■ Pro Unixové systémy (Linux/Mac):

```
python -m venv framesss-env
source framesss-env/bin/activate
pip install framesss
```

#### ■ Pro Windows:

```
python -m venv framesss-env
framesss-env\Scripts\activate
pip install framesss
```

Po instalaci můžete knihovnu framesss importovat do svých Python skriptů následovně:

```
import framesss
```

### ■ 1.6.3 Architektura knihovny

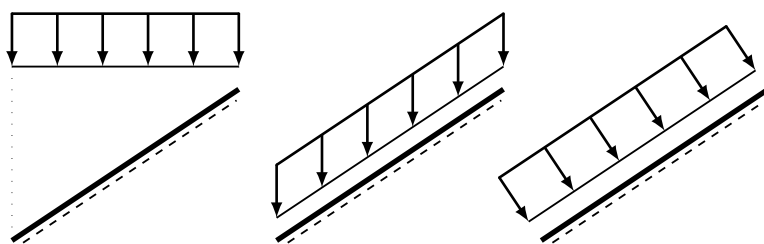
Filozofie architektury za framesss spočívá v tvorbě robustního a flexibilního systému schopného přizpůsobit se aktuálním i budoucím potřebám. Tento přístup byl klíčový při formování modulárního designu softwaru, který umožňuje aktualizaci nebo výměnu jednotlivých komponent bez dopadu na celkový systém. Objektově orientovaný design podporuje tuto modularitu, což zvyšuje opětovnou použitelnost kódu a udržitelnost.

#### ■ Funkce programu

Knihovna framesss je vybavena následující funkčností:

- **Typy modelů:** V současné době knihovna podporuje pouze 2D statický výpočet rámu v rovině XZ, plánuje se rozšíření na 3D modely.
- **Typ výpočtu:** Lineárně pružná statická analýza.
- **Podpory:**
  - **Pevné:** Kompletně zabraňují posunu nebo pootočení v daném směru.

- **Pružné:** Podpora je v daném směru pružná, uživatel musí zadat tuhost podepření.
- **Typy prutů:**
  - **Navier (Euler-Bernoulli):** Průřezy po deformaci zůstávají kolmé na deformovanou střednici prutu,
  - **Timoshenko:** Průřez po deformaci zůstává rovinný, ale ne nutně kolmý na deformovanou střednici prutu.
- **Zatížení:**
  - **Silová zatížení:** Uzlové síly a momenty, osamělé síly a momenty na prvcích, rovnoměrná a lineární spojitá zatížení působící na celé délce nebo jen části prutu. Zatížení na prvcích mohou působit v globálním souřadném systému nebo v lokálním systému prutu. Spojitá zatížení v globálním souřadném systému lze zadávat jako působící na délku nebo na průmět prvku, viz obr. 1.17.
  - **Předepsané deformace uzlů:** Tento typ zatížení lze zadat pouze v podepřených uzlech.
  - **Zatížení teplotou:** Lze zadat zatížení rovnoměrnou nebo nerovnoměrnou změnou teplot na celém prvku.
- **Zatěžovací stavy:** Jednotlivá zatížení lze sloučit do zatěžovacích stavů.
- **Kombinace zatížení:** Více zatěžovacích stavů může být kombinováno s uplatněním kombinačních součinitelů.
- **Obálka:** Kombinuje různé kombinace zatížení pro získání obálky výsledků s maximální a minimální odezvou.



Obr. 1.17: Typy spojitého zatížení

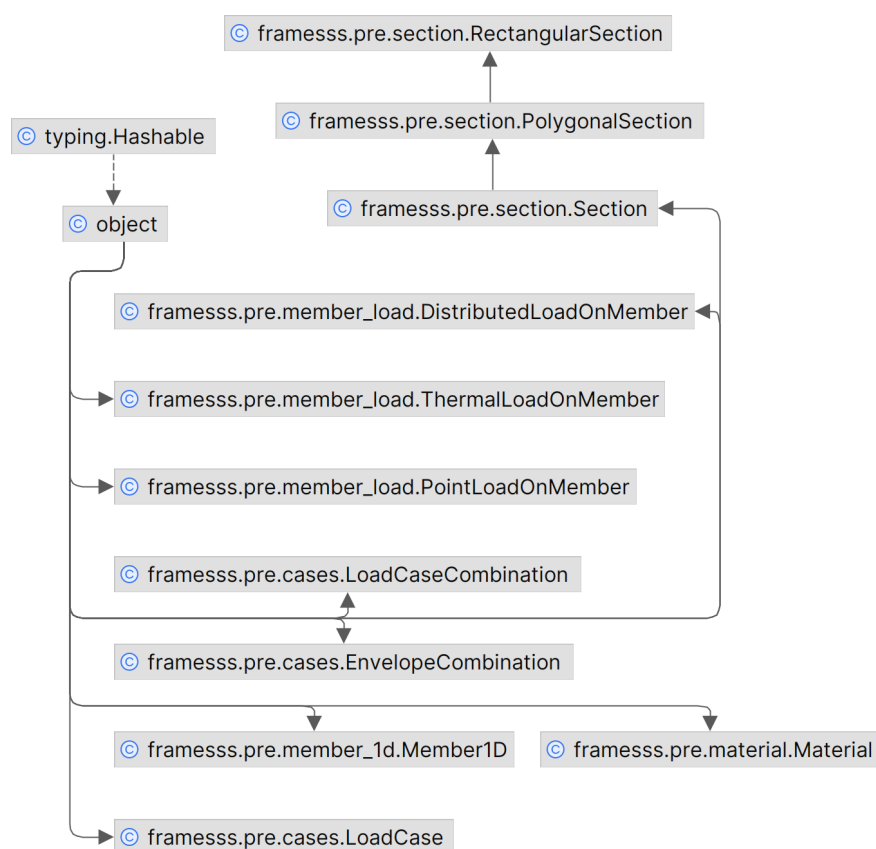
framesss je strukturován do čtyř základních modulů, které společně zajišťují celý proces statického výpočtu. Tyto moduly jsou:

- pre,
- fea,
- solvers,
- post.

## Modul pre

Modul zodpovídá za přípravu a definici vstupních dat modelu. Uživatelé zde mohou definovat geometrii, materiály, zatížení a podpory. Jedná se o klíčový modul, který umožňuje definici modelu před spuštěním vlastního výpočtu.

Na obr. 1.18 je diagram hierarchie tříd v modulu `pre`. Třídy v tomto modulu se starají o přípravu a definici vstupních dat modelu. V diagramu vidíme třídu pro průřezy `Section` a její konkrétní implementace `RectangularSection` a `PolygonalSection`. Dále jsou zde třídy pro zatížení, jako `DistributedLoadOnMember`, `ThermalLoadOnMember` a `PointLoadOnMember`. Modul také obsahuje třídu pro zatěžovací stavy `LoadCase`, jejich kombinace `LoadCaseCombination` a obálku výsledků `EnvelopeCombination`. Také je zde třída pro prutový prvek `Member1D` a třída pro materiál `Material`.



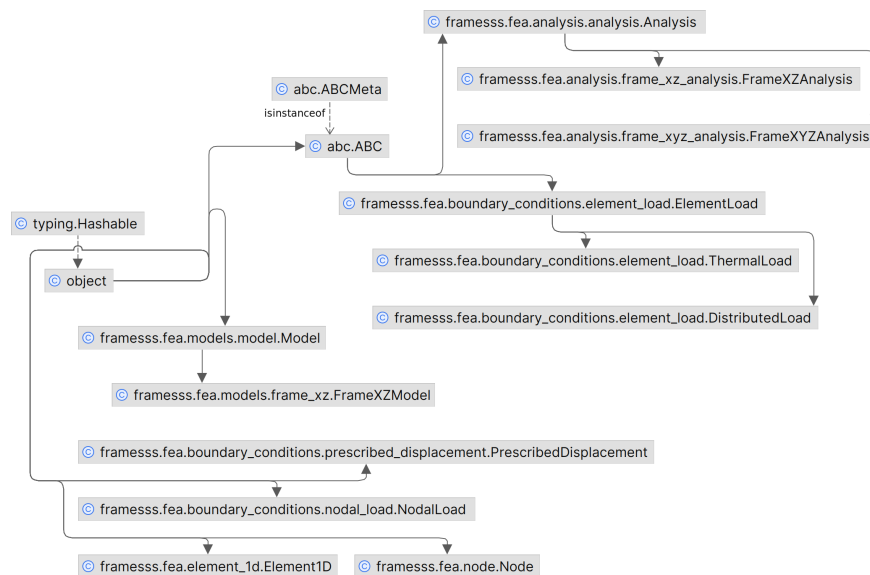
Obr. 1.18: Diagram modulu `pre`

## Modul fea

Na obr. 1.19 je diagram zobrazující hierarchii tříd v modulu `fea`, který je srdcem knihovny `framesss`. Klíčovou třídou je `Model`, která je zodpovědná za přidávání prvků (uzly, prvky, zatěžovací stavy, zatížení atd.) a drží v sobě všechny instance, které se následně předávají do řešiče (solveru). Pod ní jsou specifické modely, jako `FrameXZModel`. Modul také obsahuje třídy pro definici elementů `Element1D` a uzlů `Node`. Pro zatížení elementů zde máme abstraktní třídu `ElementLoad`, ze které vychází třída pro zatížení spojitým zatížením `DistributedLoad` a třída pro zatížení teplotou `ThermalLoad`, dále je zde třída pro uzlové zatížení `NodalLoad`, a pro předepsané přemístění podpor



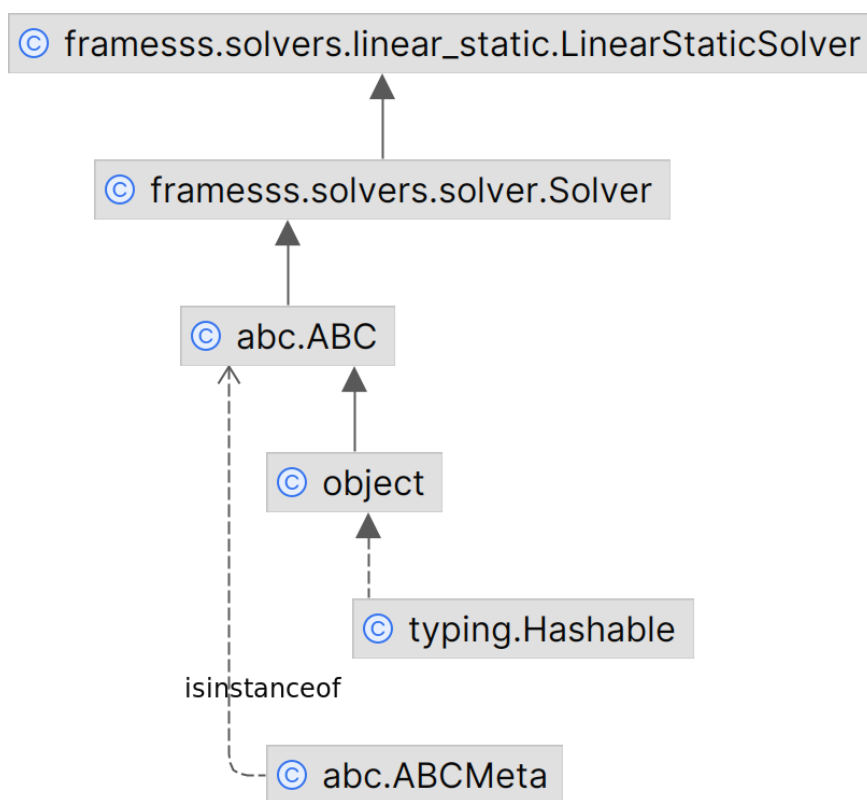
**PrescribedDisplacement.** Modul také zahrnuje třídy pro specifikování dimenze problému, které vychází z abstraktní třídy **Analysis** a její konkrétní implementace pro různé typy konstrukcí, např. **FrameXZAnalysis**.



Obr. 1.19: Diagram modulu fea

## Modul solvers

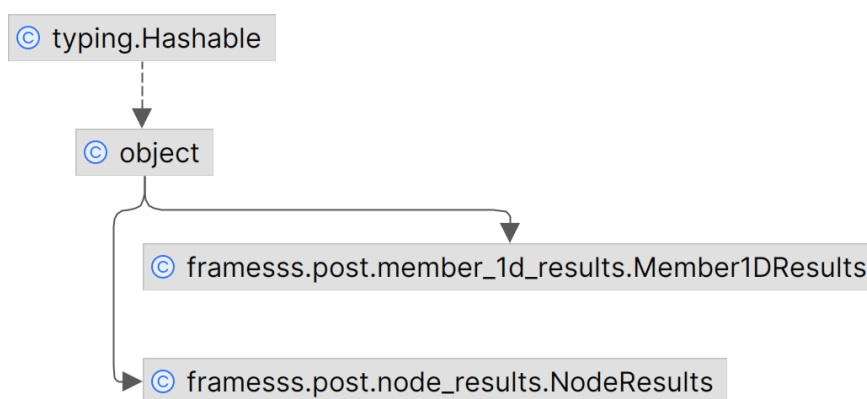
Modul obsahuje řešiče používané pro výpočty v modulu **fea**. Na obr. 1.20 je znázorněný diagram hierarchie tříd v modulu **solvers**. Základem této hierarchie je třída **abc.ABC**, což je základní abstraktní třída v Pythonu. Na jejím základě je definována třída **Solver**, která slouží jako základ pro konkrétní implementace řešičů. Jediným implementovaným řešičem je v současné době třída **LinearStaticSolver**, která implementuje lineární statické výpočty.



Obr. 1.20: Diagram modulu solvers

## Modul post

Zpracovává výsledky, jako jsou posuny a vnitřní síly, které jsou získány z výpočtu provedených modulem `fea`. Modul nezahrnuje vizualizaci, ale umožňuje uživatelům extrahovat a prohlížet vypočítané hodnoty, které mohou být dále použity. Diagram na obr. 1.21 ukazuje hierarchii tříd v modulu `post`, který je zodpovědný za zpracování výsledků. Třída `Member1DResults` zpracovává výsledky pro jednotlivé prutové prvky, třída `NodeResults` zpracovává výsledky pro jednotlivé uzly konstrukce.



Obr. 1.21: Diagram modulu post

## 1.6.4 Předpoklady

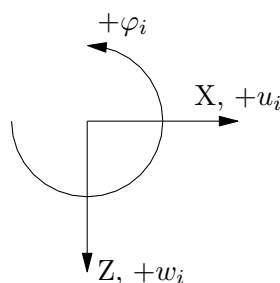
V této sekci jsou uvedeny základní předpoklady, které byly přijaty při vývoji knihovny `framessss`,

- konstrukce je umístěna v pravotočivé souřadné soustavě,
- pro kladný směr pootočení platí pravidlo pravé ruky<sup>§</sup>,
- výpočet probíhá podle teorie malých deformací,
- materiál je uvažován jako pružný, platí Hookův zákon.

Tyto předpoklady umožňují uplatnění principu superpozice.

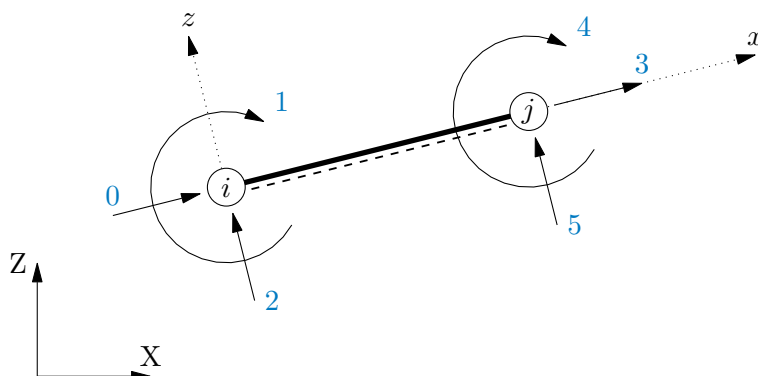
### ■ Předpoklady pro model v rovině XZ

- zatížení působí pouze v rovině XZ, která je zároveň rovinou symetrie,
- každý uzel má přiřazené tři stupně volnosti, posun ve směru globální osy X, pootočení okolo globální osy Y a posun ve směru globální osy Z. Kladný směr je naznačený na [obr. 1.22](#),



**Obr. 1.22:** Stupně volnosti v globálním systému

- stupně volnosti a jejich kódová čísla<sup>¶</sup> v lokálním systému prutového prvku jsou označena na [obr. 1.23](#),

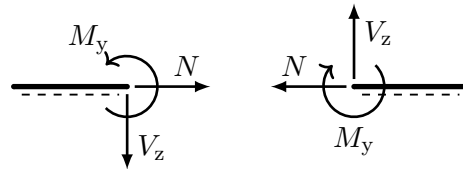


**Obr. 1.23:** Stupně volnosti v lokálním systému prvku

- vnitřní síly v jakémkoliv průřezu prvku jsou: normálová síla  $N$ , posouvající síla  $V_z$  a ohybový moment  $M_y$ . Kladná orientace vnitřních sil je naznačená na [obr. 1.24](#).

<sup>§</sup>Palec natočíme ve směru kladné části osy a zahnuté prsty pravé ruky značí kladný směr otáčení.

<sup>¶</sup>Indexování v jazyce Python začíná od 0, což znamená, že první stupeň volnosti má přiřazené kódové číslo 0, druhý prvek má kódové číslo 1 atd.



Obr. 1.24: Kladná orientace vnitřních sil

### 1.6.5 Algoritmizace

V této sekci je představen algoritmus, který je jádrem metody konečných prvků implementované v knihovně `framesss`. Tento algoritmus systematicky zpracovává model od jeho diskretizace až po výpočet odezvy konstrukce na zatížení jednotlivými zatěžovacími stavy a jejich kombinacemi.

---

#### Pseudokód 1: Proces výpočtu modelu

---

##### Metoda `solve()`:

```

Diskretizace prutů na jednotlivé elementy
Očíslování stupňů volnosti
Lokalizace globální matice tuhosti
for každý zatěžovací stav do
    Inicializace vektoru koncových posunů
    Lokalizace předepsaných deformací
    Inicializace vektoru zatížení
    Lokalizace sil v uzlech
    Lokalizace koncových sil na prvcích
    Výpočet vektoru neznámých koncových posunů
    Výpočet reakcí
    Výpočet koeficientů vnitřních sil
    Výpočet extrémů vnitřních sil
    Uložení vypočítaných hodnot
for každá kombinace zatížení do
    Inicializace výsledků
    for každý zatěžovací stav v kombinaci do
        Přičtení hodnot k výsledkům
    Uložení výsledků
for každá obálka do
    Inicializace
    Určení extrémních hodnot v každé kombinaci
    Uložení hodnot
return
```

---

### 1.6.6 Řídké matice

V rámci knihovny `framesss` je zásadní efektivní manipulace s řídkými maticemi, které se často vyskytují v důsledku velkého množství nulových prvků v maticích tuhosti. Pro ukládání a zpracování těchto matic využívá `framesss` knihovnu `SciPy` [33], specificky její modul pro řídké matice.

## ■ Formáty ukládání řídkých matic

V rámci knihovny SciPy jsou k dispozici různé formáty pro ukládání řídkých matic, z nichž každý je optimalizován pro specifické typy operací.

- **CSR (Compressed Sparse Row)**: Ukládá data kompresí řádkových indexů. Tento formát je efektivní pro řádkové operace.
- **CSC (Compressed Sparse Column)**: Podobné jako CSR, ale komprimuje sloupcové indexy. Je efektivní pro sloupcové operace.
- **BSR (Block Sparse Row)**: Velmi podobný formátu CSR. Je vhodný pro řídké matice s hustými submaticemi.
- **COO (COOrdinate format)**: Ukládá seznam trojic (řádkový index, sloupcový index a hodnotu), což umožňuje efektivní postupné sestavování řídkých matic.
- **DIA (DIAGonal storage)**: Ukládá hodnoty na diagonálách v samostatných polích a jejich offset od diagonály, což je užitečné pro matice s dominantními diagonálami.
- **DOK (Dictionary Of Keys)**: Ukládá prvky matice ve slovníku, kde klíče jsou dvojice (řádek, sloupec).
- **LIL (LIst of Lists)**: Ukládá matice pomocí dvou seznamů, jeden pro řádky, ve kterých jsou ukládány sloupcové indexy, a druhý pro hodnoty.

## ■ Využití ve framesss

Proces vytváření a manipulace s řídkými maticemi v knihovně framesss zahrnuje několik klíčových kroků, které využívají různé formáty matic pro optimalizaci různých fází výpočtu:

1. **Vytvoření matice jako COO**: Ve fázi lokalizace je globální matice tuhosti vytvářena jako COO, což je formát efektivní pro konstrukci řídkých matic
2. **Převod do LIL**: Po sestavení základní struktury matice je převedena do formátu LIL pro rychlé řezání (slicing), indexování a manipulaci. LIL umožňuje efektivní změny řádků a sloupců matice, což je vyžadováno během dělení soustavy rovnic na dvě soustavy rovnic.
3. **Převod do CSR**: Konečný převod matice do formátu CSR je proveden pro rychlé aritmetické operace, které jsou nezbytné pro řešení systému rovnic. CSR formát je vysoce optimalizovaný pro rychlé maticové operace.

Každý z těchto formátů má specifické výhody pro různé typy operací a jejich výběr je motivován potřebou maximalizovat výpočetní efektivitu a minimalizovat dobu provádění v různých fázích výpočetního procesu. Použití COO pro počáteční sestavení, LIL pro manipulaci s řádky a sloupci a CSR pro finální výpočty umožňuje dosáhnout optimálního výkonu.

### 1.6.7 Testování

Pro zajištění spolehlivosti a přesnosti knihovny `framesss` byla implementována řada automatizovaných testů. Automatizované testování umožňuje včasné odhalení a opravu chyb, což přispívá k celkové stabilitě a kvalitě knihovny.

Testování knihovny `framesss` probíhá v různých verzích Pythonu (3.9, 3.10, 3.11, 3.12), aby byla odhalena případná kolize v závislostech mezi verzemi. Lokálně jsou testy spouštěny pomocí nástroje `nox` [20], který vytváří virtuální prostředí, instaluje potřebné knihovny a spouští všechny testy, včetně verifikačních, `pre-commit` [22] a `mypy` [19] testů.

#### Unit testy

Unit test je základní typ testu používaný v softwarovém vývoji, jehož cílem je ověřit správnou funkci jednotlivých částí kódu. Unit testy se zaměřují na testování nejmenších, izolovaných částí aplikace, jako jsou jednotlivé funkce nebo metody tříd. Hlavní charakteristiky unit testů jsou

- **Izolace:** Testy jsou navrženy tak, aby byly nezávislé na ostatních částech kódu. Každý test kontroluje pouze jednu konkrétní část funkčnosti.
- **Automatizace:** Testy jsou spouštěny automaticky, což umožňuje rychlou a efektivní kontrolu správnosti
- **Opakovatelnost:** Testy mohou být opakovaně spouštěny při každé změně kódu, což zajišťuje, že nové změny neovlivní existující funkčnost.

#### Verifikační testy

Kromě unit testů se správnost knihovny testuje na jednoduchých konstrukcích (prostý nosník, šikmé nosníky, konzoly, oboustranně vetknutý nosník apd.), kde je známé analytické řešení.

Pro verifikační testy byly použity následující zdroje:

- *DESIGN AID No. 6 — BEAM DESIGN FORMULAR WITH SHEAR AND MOMENT DIAGRAMS* od American Wood Council [1],
- *Sbírka příkladů stavební mechaniky: princip virtuálních sil, silová metoda, deformační metoda* od Jíra, A. a kol. [15].

#### Kontinuální integrace a spuštění testů na GitHubu

Kromě lokálního testování je nezbytné zajistit, aby všechny změny v kódu byly automaticky testovány i při jejich nahrání do centrálního repozitáře. Tento proces, známý jako kontinuální integrace (Continuous Integration, CI), je implementován pomocí GitHub Actions.

Při každém pushnutí změn do GitHub repozitáře se automaticky spustí CI pipeline, která zahrnuje následující kroky:

- **Instalace závislostí:** Nezbytné knihovny a závislosti jsou nainstalovány podle specifikací v konfiguračním souboru.
- **Statická analýza kódu:** Kód je zkontrolován nástrojem `mypy` [19], který provádí statickou typovou kontrolu, aby se zajistila konzistence typů a odhalily potenciální typové chyby.

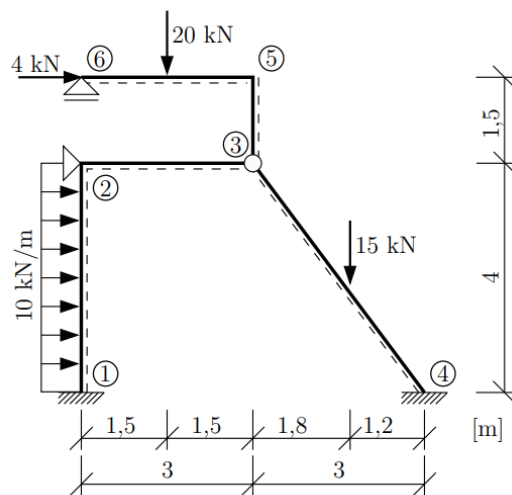
- **Spuštění testů:** Pomocí nástroje pytest [24] se spustí všechny unit testy a verifikační testy, aby se ověřila funkčnost všech částí kódu.

Jednou z výhod používání GitHub Actions je možnost testovat kód na různých operačních systémech, knihovna framesss je testován na následujících platformách

- Windows,
- Linux,
- MacOS.

### 1.6.8 Příklad

Následující příklad je převzat ze *Sbírký příkladů stavební mechaniky* [15, Příklad 5.2].



Obr. 1.25: Statické schéma konstrukce, převzato z [15, Příklad 5.2]

Všechny pruty mají Youngův modul pružnosti  $E = 30 \text{ GPa}$ , moment setrvačnosti k ose, kolem které jsou pruty namáhané ohybem  $I_y = 1 \times 10^{-3} \text{ m}^4$  a plochu  $A = 4 \times 10^{-3} \text{ m}^2$ .

Na příkladu bude ilustrováno použití knihovny framesss v prostředí JupyterLab.

V prvním kroku je nutné importovat knihovny, se kterými budeme pracovat.

```
[1]: import numpy as np

from tabulate import tabulate

from framesss.fea.models.frame_xz import FrameXZModel
from framesss.pre.material import Material
from framesss.pre.section import Section
from framesss.solvers.linear_static import LinearStaticSolver
```

Začneme definicí materiálu, knihovna nepracuje s převody jednotek, musíme být konzistentní. Zvolíme systém jednotek kN, kPa a m. Poissonův součinitel

$\nu$ , teplotní součinitel roztažnosti, ani objemovou hmotnost materiálu bychom nemuseli definovat, protože mají výchozí hodnoty, ale uvádíme je zde pro přehlednost.

```
[2]: material = Material(
    label="foo",
    elastic_modulus=30.e6,
    poissons_ratio=0.2,
    thermal_expansion_coefficient=1.0e-3,
    density=0.0,
)
```

Jako další definujeme průřez prvku pomocí třídy `Section`, proměnná `area_x` slouží pro výpočet normálové tuhosti, `area_y` a `area_z` se ve výpočtu použijí pouze v případě, že prutový prvek, kterému tento průřez přiřadíme bude typu `'timoshenko'`. Plocha zadaná těmito atributům má odpovídat hodnotě  $kA$ , kde  $k$  je korekční součinitel rozložení smykového napětí. Pro získání správného výsledku je dále potřeba přiřadit správnou hodnotu momentu setrvačnosti  $I_y$  do atributu `inertia_y` a přiřadit materiál definovaný v předchozí buňce, ostatní hodnoty při tomto výpočtu nehrají roli, uvádíme je pouze pro přehlednost.

```
[3]: section = Section(
    label="bar",
    area_x=4.0e-3,
    area_y=1.0,
    area_z=1.0,
    inertia_x=1.0,
    inertia_y=1.0e-3,
    inertia_z=1.0,
    height_z=1.0,
    height_y=1.0,
    material=material
)
```

V dalším kroku vytvoříme instanci modelu `FrameXZModel`. Pomocí metod této třídy budeme modelovat celou konstrukci i se zatížením.

```
[4]: model = FrameXZModel()
```

Pomocí metody `add_node()` definujeme uzly konstrukce. Atributy, které musíme zadat jsou označení uzlu `label` a souřadnice uzlu `coords` ve formátu  $[X, Y, Z]$ . Nepovinným atributem metody je `fixity`, kterou definujeme okrajové podmínky pro každý stupeň volnosti v uzlu, zadáváme je pomocí stringů `'fixed'` nebo `'free'` a řadíme je v pořadí  $[u, v, w, \varphi_x, \varphi_y, \varphi_z]$ .

```
[5]: N1 = model.add_node(
    label="N1",
    coords=[0.0, 0.0, 0.0],
    fixity=['fixed', 'free', 'fixed', 'free', 'fixed', '
→'free'],
)

N2 = model.add_node(
```



```

        label="N2",
        coords=[0.0, 0.0, 4.0],
        fixity=["fixed", "free", "fixed", "free", "free",
→"free"],
    )

    N3 = model.add_node(
        label="N3",
        coords=[3.0, 0.0, 4.0],
    )

    N4 = model.add_node(
        label="N4",
        coords=[6.0, 0.0, 0.0],
        fixity=["fixed", "free", "fixed", "free", "fixed",
→"free"]
    )

    N5 = model.add_node(
        label="N5",
        coords=[3.0, 0.0, 5.5],
    )

    N6 = model.add_node(
        label="N6",
        coords=[0.0, 0.0, 5.5],
        fixity=['free', 'free', 'fixed', 'free', 'free',
→'free']
    )

```

Pomocí metody `add_member()` přidáme jednotlivé pruty. Atributy jsou označení `label`, typ použitého elementu `element_type`, který může být buď `'navier'` pro prut bez vlivu smyku nebo `'timoshenko'` pro prut s vlivem smyku. Dále zadáme počáteční a koncový uzel, průřez prvku a můžeme zadat nepovinný atribut `hinges`, který udává, zda je v odpovídajícím uzlu prut připojen kloubově nebo pevně.

```

[6]: B12 = model.add_member(
        label="1-2",
        element_type="navier",
        nodes=[N1, N2],
        section=section,
    )

    B23 = model.add_member(
        label="2-3",
        element_type="navier",
        nodes=[N2, N3],
        section=section,
        hinges=["fixed", "hinged"],
    )

```

```

B34 = model.add_member(
    label="3-4",
    element_type="navier",
    nodes=[N3, N4],
    section=section,
    hinges=["hinged", "fixed"],
)

B35 = model.add_member(
    label="3-5",
    element_type="navier",
    nodes=[N3, N5],
    section=section,
    hinges=["hinged", "fixed"],
)

B65 = model.add_member(
    label="6-5",
    element_type="navier",
    nodes=[N6, N5],
    section=section,
)

```

V dalším kroku definujeme zatížení. Zatěžovací stav do modelu přidáme pomocí metody `add_load_case()`. Jednotlivé síly a spojitá zatížení přidáváme přímo na dříve definované instance prutů a uzlů pomocí metod `add_distributed_load()` a `add_point_load()` pro pruty, případně `add_nodal_load()` pro uzly. Výchozí směr zatížení je v globálním souřadném systému. Polohu spojitěho zatížení zadáváme pomocí atributů `x_start` a `x_end`, hodnoty zadáváme hodnotou na začátku a na konci intervalu pro směry  $x$ ,  $y$  a  $z$ . Tímto přístupem je možné modelovat lichoběžníkové zatížení kdekoliv na nosníku. Pomocí metody `add_point_load()` lze přidat osamělou sílu nebo moment, hodnoty musí být v pořadí  $[F_x, F_y, F_z, M_x, M_y, M_z]$ .

```

[7]: LC1 = model.add_load_case(
    label='LC1'
)

B12.add_distributed_load(
    load_components=[10., 0., 0., 10., 0., 0.],
    load_case=LC1,
    x_start=0.0,
    x_end=1.0,
    coordinate_system="global",
    location="length",
    coordinate_definition="relative",
)

B34.add_point_load(
    load_components=[0., 0., -15., 0., 0., 0.],
    load_case=LC1,
)

```

```

        x=1.8/3,
        coordinate_system="global",
        coordinate_definition="relative",
    )

    B65.add_point_load(
        load_components=[0., 0., -20., 0., 0., 0.],
        load_case=LC1,
        x=0.5,
    )

    N6.add_nodal_load(
        load_components=[4., 0., 0., 0., 0., 0.],
        load_case=LC1,
    )

```

To samé zatížení přidáme do modelu ještě jednou, tentokrát však každé zatížení přiřadíme jinému zatěžovacímu stavu, poté z jednotlivých stavů vytvoříme novou kombinaci zatížení. Hodnoty zatížení vydělíme součiniteli `LC<i>_factor` a při vytváření nové kombinace tento součinitel použijeme jako součinitel kombinace.

```

[8]: LC2_factor = 2
      LC3_factor = 18
      LC4_factor = 5.5
      LC5_factor = -12

      LC2 = model.add_load_case(
          label="LC2"
      )
      B12.add_distributed_load(
          load_components=np.array(
              [10., 0., 0., 10., 0., 0.]
          ) / LC2_factor,
          load_case=LC2,
      )

      LC3 = model.add_load_case(
          label="LC3"
      )
      B34.add_point_load(
          load_components=np.array(
              [0., 0., -15., 0., 0., 0.]
          ) / LC3_factor,
          load_case=LC3,
          x=1.8/3,
      )

      LC4 = model.add_load_case(
          label="LC4"
      )
      B65.add_point_load(

```

```

        load_components=np.array(
            [0., 0., -20., 0., 0., 0.]
        ) / LC4_factor,
        load_case=LC4,
        x=0.5,
    )

    LC5 = model.add_load_case(
        label="LC5"
    )
    N6.add_nodal_load(
        load_components=np.array(
            [4., 0., 0., 0., 0., 0.]
        ) / LC5_factor,
        load_case=LC5,
    )

    C01 = model.add_load_case_combination(
        label="C01",
        combination={
            LC2: LC2_factor,
            LC3: LC3_factor,
            LC4: LC4_factor,
            LC5: LC5_factor,
        }
    )

```

Vytvoříme instanci řešiče a propojíme ho s instancí `model`.

```
[9]: solver = LinearStaticSolver(
        model=model
    )

```

Výpočet provedeme pomocí metody `solve()`.

```
[10]: solver.solve()

```

Výsledky jsou uloženy v instancích tříd `Node` a `Member1D` v objektu `results`. Pomocí metody `get()`, dostaneme výsledky pro zatěžovací stavy nebo kombinace. Hodnoty nakonec vypíšeme do tabulky. V tabulce si můžeme všimnout, že program vytvořil vlastní uzly v místech působících osamělých sil.

```
[11]: displacements= []
        reactions = []
        for node in model.nodes:
            for case in [LC1, C01]:
                displacements.append([
                    case.label,
                    node.label,
                    node.results.translation_x.get(case),
                    node.results.translation_z.get(case),
                    node.results.rotation_y.get(case),
                ])
            r_x = node.results.reaction_force_x.get(case)

```

```

r_z = node.results.reaction_force_z.get(case)
r_my = node.results.reaction_moment_y.get(case)

if r_x or r_z or r_my:
    reactions.append([
        case.label,
        node.label,
        r_x,
        r_z,
        r_my
    ])

displacements.append([None] * 5)
if r_x or r_z or r_my:
    reactions.append([None] * 5)

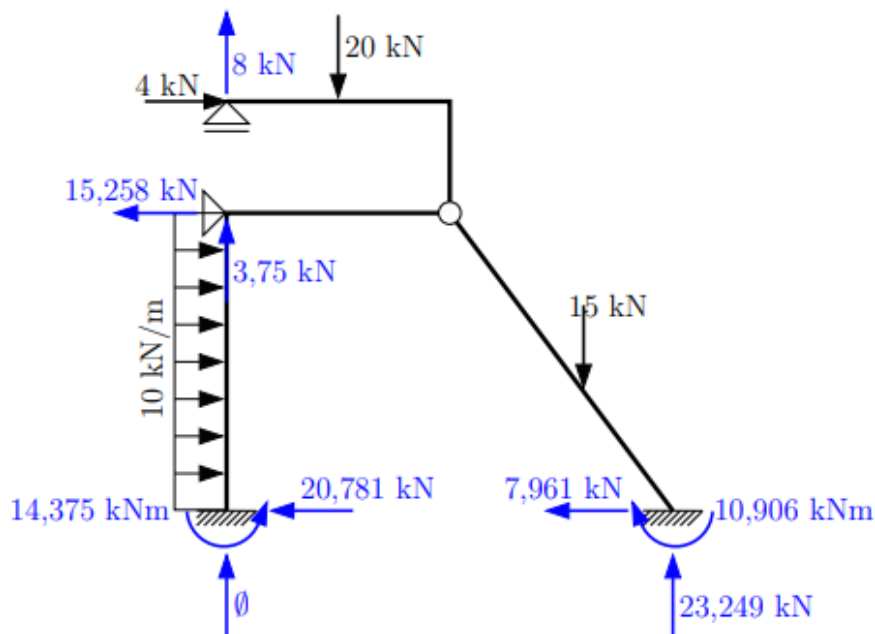
print(tabulate(
    displacements,
    headers=["case", "node", "u_x", "u_z", "phi_y"],
    tablefmt="rst",
    floatfmt=".3e"
))
print(tabulate(
    reactions,
    headers=["case", "node", "R_x", "R_z", "R_My"],
    tablefmt="rst",
    floatfmt=".3f"
))

```

case	node	u_x	u_z	phi_y
LC1	N1	0.000e+00	0.000e+00	0.000e+00
C01	N1	0.000e+00	0.000e+00	0.000e+00
LC1	N2	0.000e+00	0.000e+00	-6.942e-05
C01	N2	0.000e+00	0.000e+00	-6.942e-05
LC1	N3	-9.902e-05	-9.168e-04	0.000e+00
C01	N3	-9.902e-05	-9.168e-04	0.000e+00
LC1	N4	0.000e+00	0.000e+00	0.000e+00
C01	N4	0.000e+00	0.000e+00	0.000e+00
LC1	N5	3.219e-04	-1.067e-03	1.806e-04
C01	N5	3.219e-04	-1.067e-03	1.806e-04
LC1	N6	4.219e-04	0.000e+00	6.306e-04
C01	N6	4.219e-04	0.000e+00	6.306e-04
LC1	3-4(1)	-7.831e-05	-5.457e-04	-2.216e-04

C01	3-4(1)	-7.831e-05	-5.457e-04	-2.216e-04
LC1	6-5(1)	3.719e-04	-7.959e-04	3.306e-04
C01	6-5(1)	3.719e-04	-7.959e-04	3.306e-04
=====				
case	node	R_x	R_z	R_My
=====				
LC1	N1	-20.781	0.000	-14.375
C01	N1	-20.781	0.000	-14.375
LC1	N2	-15.258	3.750	
C01	N2	-15.258	3.750	
LC1	N4	-7.961	23.250	10.905
C01	N4	-7.961	23.250	10.905
LC1	N6		8.000	
C01	N6		8.000	
=====				

Reakce:



Obr. 1.26: Reakce v podporách, převzato z [15, Příklad 5.2]

Stejně jako v případě uzlů, výsledky se ukládají do objektu `results`, do tabulky vypíšeme extrémy vnitřních sil pro jednotlivé prvky.

```
[12]: headers = ["case", "member", "N_min", "N_max", "V_min", "V_max", "M_min", "M_max"]
axial = []
shear = []
```

```

moment = []
for member in model.members:
    for case in [LC1, C01]:
        axial.append([
            case.label,
            member.label,
            *member.results.min_max_axial_forces.
→get(case).tolist()
        ])
        shear.append([
            case.label,
            member.label,
            *member.results.min_max_shear_forces_z.
→get(case).tolist()
        ])
        moment.append([
            case.label,
            member.label,
            *member.results.min_max_bending_moments_y.
→get(case).tolist()
        ])
        axial.append([None] * 2)
        shear.append([None] * 2)
        moment.append([None] * 2)

for i, data in enumerate([axial, shear, moment]):
    print(tabulate(
        data,
        headers=headers[0:2] + headers[2+i*2:4+i*2],
        tablefmt="rst",
        floatfmt=".3f"
    ))

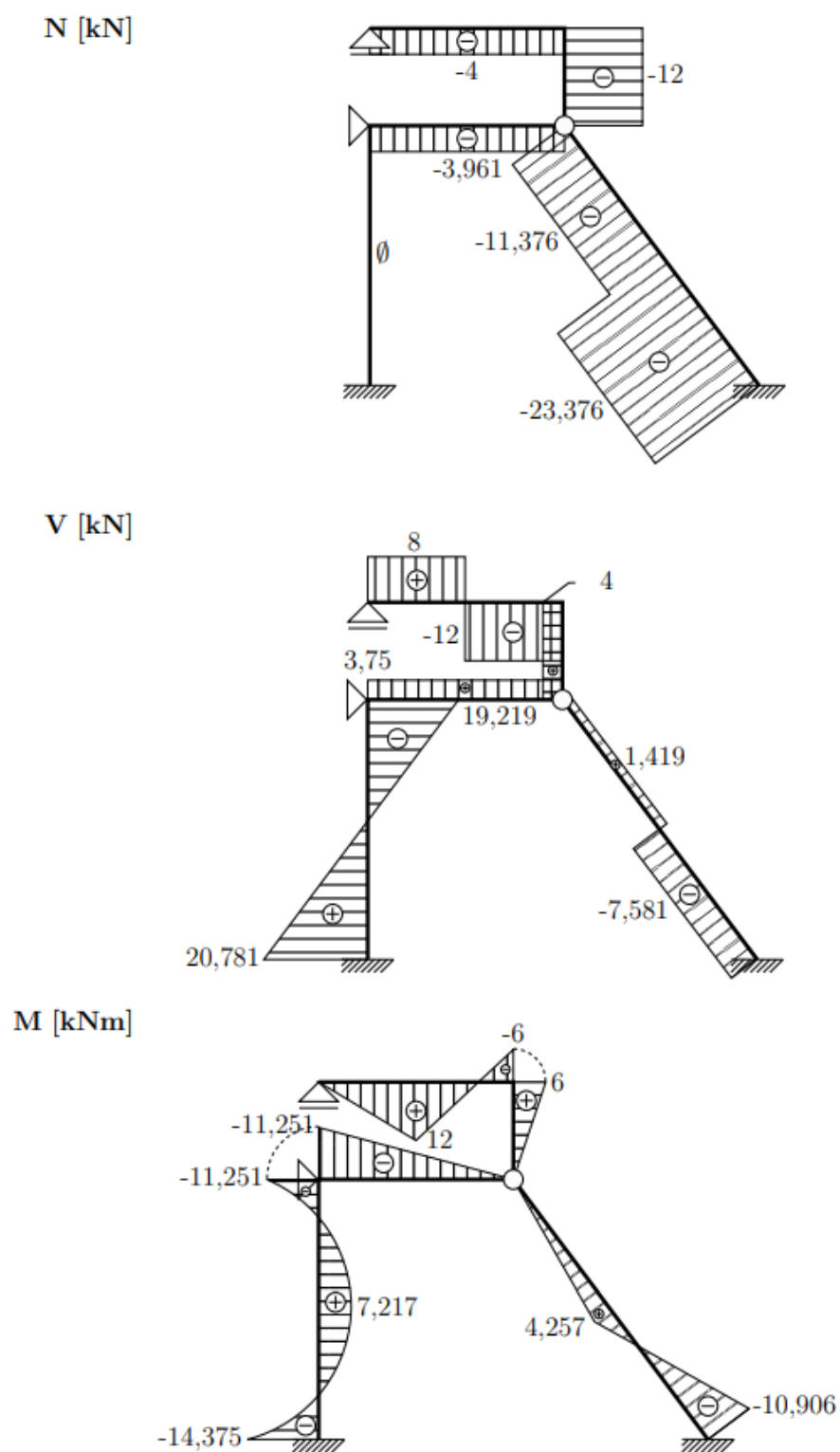
```

=====	=====	=====	=====
case	member	N_min	N_max
=====	=====	=====	=====
LC1	1-2	0.000	0.000
C01	1-2	0.000	0.000
LC1	2-3	-3.961	-3.961
C01	2-3	-3.961	-3.961
LC1	3-4	-23.376	-11.376
C01	3-4	-23.376	-11.376
LC1	3-5	-12.000	-12.000
C01	3-5	-12.000	-12.000
LC1	6-5	-4.000	-4.000
C01	6-5	-4.000	-4.000
=====	=====	=====	=====

=====	=====	=====	=====
case	member	V_min	V_max
=====	=====	=====	=====
LC1	1-2	-19.219	20.781
C01	1-2	-19.219	20.781
LC1	2-3	3.750	3.750
C01	2-3	3.750	3.750
LC1	3-4	-7.581	1.419
C01	3-4	-7.581	1.419
LC1	3-5	4.000	4.000
C01	3-5	4.000	4.000
LC1	6-5	-12.000	8.000
C01	6-5	-12.000	8.000
=====	=====	=====	=====

=====	=====	=====	=====
case	member	M_min	M_max
=====	=====	=====	=====
LC1	1-2	-14.375	7.218
C01	1-2	-14.375	7.218
LC1	2-3	-11.251	0.000
C01	2-3	-11.251	0.000
LC1	3-4	-10.905	4.257
C01	3-4	-10.905	4.257
LC1	3-5	0.000	6.000
C01	3-5	0.000	6.000
LC1	6-5	-6.000	12.000
C01	6-5	-6.000	12.000
=====	=====	=====	=====





**Obr. 1.27:** Vnitřní síly na konstrukci, převzato z [15, Příklad 5.2]

veličina	rozměr	LC1	CO1	Sbírka
$R_{1,x}$	kN	-20.781	-20.781	-20.781
$R_{1,z}$	kN	0.000	0.000	0.000
$M_{1,y}$	kN m	-14.375	-14.375	14.375
$R_{2,x}$	kN	-15.258	-15.258	-15.258
$R_{2,z}$	kN	3.750	3.750	-3.750
$R_{4,x}$	kN	-7.961	-7.961	-7.961
$R_{4,z}$	kN	23.250	23.250	-23.249
$M_{4,y}$	kN	10.905	10.905	-10.906
$R_{6,z}$	kN	8.000	8.000	-8.000
$\varphi_2$	rad · 10 <sup>-5</sup>	-6.942	-6.942	6.942
$u_3$	m · 10 <sup>-5</sup>	-9.902	-9.902	-9.903
$w_3$	m · 10 <sup>-4</sup>	-9.168	-9.168	9.168
$N_{1,2}$	kN	0.000	0.000	0.000
$N_{2,3}$	kN	-3.961	-3.961	-3.961
$\min(N_{3,4})$	kN	-23.376	-23.376	-23.376
$\max(N_{3,4})$	kN	-11.376	-11.376	-11.376
$N_{3,5}$	kN	-12.000	-12.000	-12.000
$N_{6,5}$	kN	-4.000	-4.000	-4.000
$\min(V_{z1,2})$	kN	-19.219	-19.219	-19.219
$\max(V_{z1,2})$	kN	20.781	20.781	20.781
$V_{z2,3}$	kN	3.750	3.750	3.750
$\min(V_{z3,4})$	kN	-7.581	-7.581	-7.581
$\max(V_{z3,4})$	kN	1.419	1.419	1.419
$V_{z3,5}$	kN	4.000	4.000	4.000
$\min(V_{z6,5})$	kN	-12.000	-12.000	-12.000
$\max(V_{z6,5})$	kN	8.000	8.000	8.000
$\min(M_{y1,2})$	kN m	-14.375	-14.375	-14.375
$\max(M_{y1,2})$	kN m	7.218	7.218	7.217
$\min(M_{y2,3})$	kN m	-11.251	-11.251	-11.251
$\max(M_{y2,3})$	kN m	0.000	0.000	0.000
$\min(M_{y3,4})$	kN m	-10.905	-10.905	-10.906
$\max(M_{y3,4})$	kN m	4.257	4.257	4.257
$\min(M_{y3,5})$	kN m	0.000	0.000	0.000
$\max(M_{y3,5})$	kN m	6.000	6.000	6.000
$\min(M_{y6,5})$	kN m	-6.000	-6.000	-6.000
$\max(M_{y6,5})$	kN m	12.000	12.000	12.000

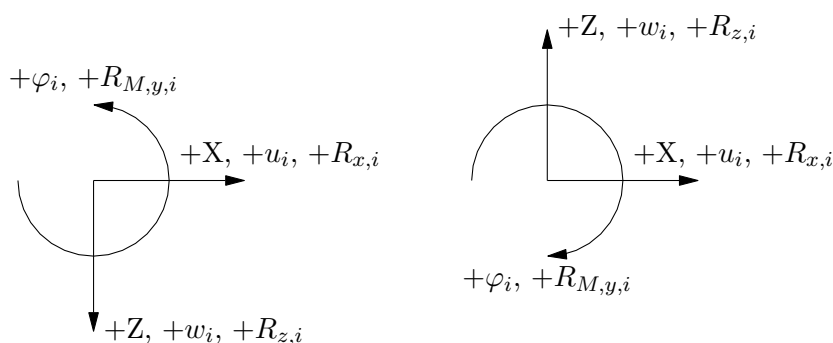
**Tab. 1.1:** Porovnání výsledků

V [tab. 1.1](#) jsou uvedeny hodnoty vypočítané knihovnou *framesss* a srovnány s referenčními hodnotami ze *Sbírky příkladů stavební mechaniky* [15]. Testování probíhalo ve dvou různých scénářích: zatěžovací stav (LC1) a kombinace zatěžovacích stavů (CO1). LC1 představuje stav, kde je veškeré zatížení apliko-

váno najednou. CO1 je kombinace jednotlivých zatěžovacích stavů, kde každé zatížení bylo odděleně definováno a hodnota zatížení byla vydělena předem definovaným součinitel. Následně byly tyto zatěžovací stavy zkombinovány do kombinace zatěžovacích stavů s těmito součiniteli.

Porovnání výsledků ukazuje, že hodnoty vypočítané knihovnou framesss jsou téměř stejné jako referenční hodnoty, přičemž rozdíly (vyznačené červenou barvou) se nacházejí v rámci zaokrouhlovací chyby.

Hodnoty posunutí a reakcí ve směru osy Z a hodnoty pootočení a momentů okolo osy Y vykazují opačné znaménko (vyznačeno žlutou barvou) ve srovnání s referenčními hodnotami, což je způsobeno rozdílným natočením globálního souřadného systému, viz obr. 1.28. Systém ze sbírky je vlevo, uvažovaný systém při zadávání konstrukce v této práci vpravo. Při stanovení kladného směru pootočení platí pravidlo pravé ruky.



**Obr. 1.28:** Porovnání souřadných systémů

## Kapitola 2

### Knihovna pro navrhování konstrukcí podle Eurokódů

Knihovna `desssign` byla vyvinuta s cílem usnadnit klasifikaci, třídění a generování zatěžovacích stavů podle požadavků Eurokódu. Eurokódy představují soubor evropských norem pro navrhování konstrukcí, které obsahují specifické požadavky na kombinace zatížení a jejich součinitele. Tato knihovna umožňuje automatizaci tohoto procesu.

Hlavním důvodem vzniku knihovny bylo zajistit, aby bylo možné automaticky generovat kombinace zatěžovacích stavů podle jejich charakteru a přiřazovat jednotlivým zatěžovacím stavům odpovídající součinitele, jak to vyžadují Eurokódy. Tento přístup umožňuje snadno a rychle vytvářet správné kombinace zatížení, které mohou být následně použity v knihovně `framesss` pro statické výpočty prutových konstrukcí.

Při návrhu knihovny pro analýzu konstrukcí `framesss` jsem chtěl zachovat její obecnost a flexibilitu, aniž by byla zatížena specifickými normami nebo jinými specifickými závislostmi. Cílem bylo vytvořit obecný program pro analýzu prutových konstrukcí, který by nebyl příliš komplikovaný a nebyl závislý na konkrétní normě. Knihovna `desssign` tedy vznikla jako doplněk k `framesss`, aby umožnila specifickou implementaci podle Eurokódů, aniž by zatěžovala základní knihovnu zbytečnými závislostmi.

Knihovna `desssign` se skládá z několika modulů, z nichž nejvýznamnější je modul `loads`. Tento modul obsahuje třídy a funkce potřebné pro definici a správu zatěžovacích stavů a jejich kombinací. Mezi hlavní třídy patří `DesignLoadCase`, `DesignLoadCaseCombination` a `DesignLoadCaseGroup`, které umožňují definici vztahů mezi jednotlivými zatěžovacími stavy. Dále je zde třída `CombinationsGenerator`, která generuje kombinace zatížení na základě zadaného mezního stavu, typu kombinace a klasifikaci zatěžovacích stavů.

Knihovna také obsahuje submodul pro generování zatížení sněhem a větrem pro jednoduché tvary střech.

V následujících částech této kapitoly se podrobněji podíváme na architekturu knihovny `desssign`, jednotlivé moduly a jejich funkce, a také na praktické použití této knihovny v kontextu navrhování konstrukcí podle Eurokódů.

Zdrojový kód `desssign` je veřejně dostupný na platformě GitHub na adrese <https://github.com/DanBeranek/desssign>. Dokumentace knihovny, která je k dispozici na adrese <https://danberanek.github.io/desssign/index.html>, poskytuje podrobné informace o funkcionalitě, implementaci a použití knihovny.

## 2.1 Instalace

Knihovna `desssign` je snadno instalovatelná pomocí `pip`. Pro instalaci knihovny `desssign` stačí spustit následující příkaz v terminálu:

```
pip install desssign
```

Podrobný návod na instalaci knihovny v Python virtuálním prostředí lze nalézt v [kapitola 1.6.2](#).

## 2.2 Architektura knihovny

Knihovna `desssign` je navržena jako modulární a rozšiřitelný systém, který umožňuje snadnou integraci nových funkcionalit. Hlavní moduly knihovny zahrnují modul `loads` a modul `wood`. Tato část kapitoly se zaměří na popis struktury knihovny a jednotlivých modulů.

### 2.2.1 Struktura knihovny

Knihovna `desssign` je organizována do následujících hlavních modulů

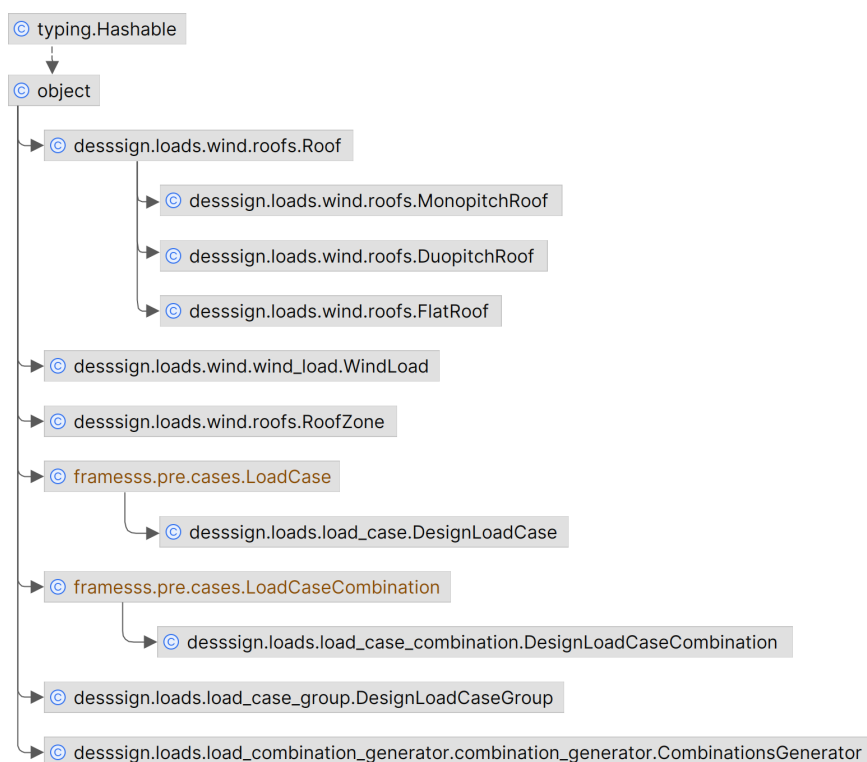
- **loads:** Tento modul obsahuje třídy a funkce pro definici, správu a generování kombinací zatěžovacích stavů. Hlavní třídy zahrnují `DesignLoadCase`, `DesignLoadCaseCombination`, `DesignLoadCaseGroup` a `CombinationsGenerator`. Tento modul také obsahuje submodul pro generování zatížení sněhem a větrem pro jednoduché tvary střech.
- **wood:** Tento modul obsahuje funkce pro posouzení dřevěných konstrukcí podle *ČSN EN 1995-1-1*. Přestože se jedná o důležitou součást knihovny, v této diplomové práci se zaměříme především na modul `loads`.

### Modul `loads`

Modul `loads` je klíčovou součástí knihovny `desssign` a je zodpovědný za

- **Definici zatěžovacích stavů:** Pomocí tříd `DesignLoadCase` a `DesignLoadCaseCombination`, které dědí vlastnosti tříd z knihovny `framesss`, lze definovat jednotlivé zatěžovací stavy a jejich kombinace.
- **Kategorizaci zatěžovacích stavů:** Třída `DesignLoadCaseGroup` umožňuje definovat vztahy mezi jednotlivými zatěžovacími stavy.
- **Generování kombinací zatížení:** Třída `CombinationsGenerator` umožňuje automatické generování kombinací zatížení na základě zadaných parametrů a kategorizaci zatěžovacích stavů.

Tento modul také zahrnuje submodul pro klimatická zatížení, který umožňuje generování hodnot zatížení sněhem na střeše podle kapitoly 5 v *ČSN EN 1991-1-3* [10] a zatížení větrem pro obdélníkový tvar střech podle kapitoly 5 v *ČSN EN 1991-1-4* [8].



Obr. 2.1: Diagram modulu loads

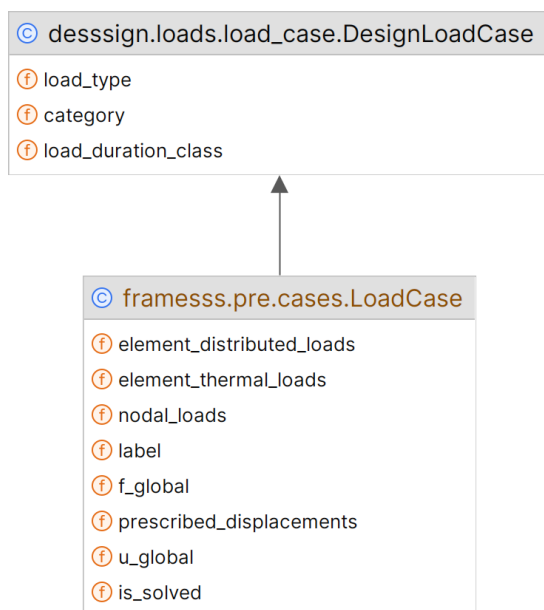
## 2.2.2 Integrace s knihovnou pro výpočty prutových konstrukcí

Jednou z klíčových vlastností knihovny `desssign` je její schopnost generovat kombinace zatěžovacích stavů, které lze následně použít v knihovně `framesss` pro statické výpočty. Tímto způsobem knihovna `desssign` výrazně usnadňuje a automatizuje proces návrhu konstrukcí podle Eurokódů.

### Třída `DesignLoadCase`

Tato třída rozšiřuje třídu `LoadCase` z knihovny `framesss` a přidává specifické atributy,

- `load_type`: typ zatížení, lze volit mezi stálým a užitným zatížením,
- `category`: kategorie zatížení podle ČSN EN 1991-1-1, kapitola 6.3.1.1 [2], specifikuje se pouze pro užitné zatížení,
- `load_duration_class`: třída trvání zatížení podle ČSN EN 1995-1-1, tab. 2.1 [3].



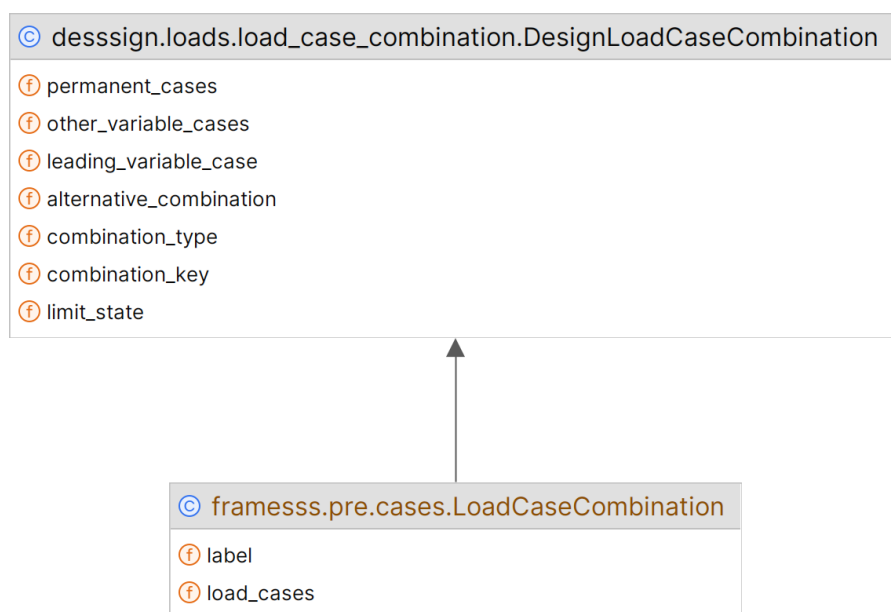
Obr. 2.2: Diagram třídy DesignLoadCase

### ■ Třída DesignLoadCaseCombination

Tato třída rozšiřuje třídu `LoadCaseCombination` z knihovny `framesss` a zahrnuje atributy

- `permanent_cases`: zatěžovací stavy pro stálé zatížení,
- `leading_variable_case`: zatěžovací stav pro hlavní proměnné zatížení,
- `other_variable_cases`: vedlejší proměnné zatěžovací stavy,
- `limit_state`: mezní stav,
- `combination_type`: typ kombinace,
- `alternative_combination`: specifikátor použité rovnice, vyžadovaný pouze pro alternativní kombinace MSÚ, buď '6.10a', nebo '6.10b',
- `combination_key`: automaticky vygenerovaný kombinační klíč.

Podle zadaného mezního stavu a kombinace se jednotlivým zatěžovacím automaticky přiřadí kombinační součinitelé podle pravidel v ČSN EN 1990 [9].

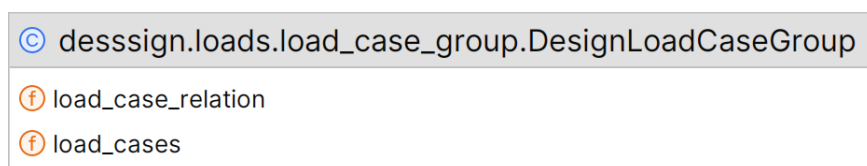
Obr. 2.3: Diagram třídy `DesignLoadCaseCombination`

### ■ Třída `DesignLoadCaseGroup`

Třída `DesignLoadCaseGroup` umožňuje definovat vztahy mezi jednotlivými zatěžovacími stavy. Tato třída je klíčová pro správné generování kombinací zatěžovacích stavů.

Atributy:

- `load_case_relation`: definuje vztah mezi zatěžovacími stavy,
  - zatěžovací stavy působí vždy společně (např. stálá zátěž),
  - zatěžovací stavy můžou a nemusí působit společně (např. užitná zátěž),
  - zatěžovací stavy nikdy nepůsobí společně.
- `load_cases`: zatěžovací stavy.

Obr. 2.4: Diagram třídy `DesignLoadCaseGroup`

### ■ Třída `CombinationsGenerator`

Třída `CombinationsGenerator` je zodpovědná za generování kombinací zatížení na základě zadaného mezního stavu a typu kombinace.

Atributy:

- `limit_state`: typ mezního stavu (mezní stav únosnosti, mezní stav použitelnosti),
- `combination_type`: typ kombinace zatížení (základní, alternativní, charakteristická, častá, kvazistálá)



Metody:

- `generate_combinations(load_case_groups)`: generuje kombinace zatížení na základě listu instancí `DesignLoadCaseGroup`.

Kombinace zatížení se generují podle vztahů uvedených v kapitole 6 ČSN EN 1990 [9],

- mezní stavy únosnosti, kombinace pro trvalé a dočasné návrhové situace:

- základní kombinace zatížení podle vztahu 6.10,

$$\sum_{j \geq 1} \gamma_{G,j} G_{k,j} + \gamma_P P + \gamma_{Q,1} Q_{k,1} + \sum_{i > 1} \gamma_{Q,i} \psi_{0,i} Q_{k,i}, \quad (2.1)$$

- alternativní kombinace zatížení podle vztahu 6.10a a 6.10b,

$$\sum_{j \geq 1} \gamma_{G,j} G_{k,j} + \gamma_P P + \gamma_{Q,1} \psi_{0,1} Q_{k,1} + \sum_{i > 1} \gamma_{Q,i} \psi_{0,i} Q_{k,i}, \quad (2.2a)$$

$$\sum_{j \geq 1} \xi_j \gamma_{G,j} G_{k,j} + \gamma_P P + \gamma_{Q,1} Q_{k,1} + \sum_{i > 1} \gamma_{Q,i} \psi_{0,i} Q_{k,i}, \quad (2.2b)$$

- mezní stavy použitelnosti,

- charakteristická kombinace zatížení podle vztahu 6.14b,

$$\sum_{j \geq 1} G_{k,j} + P + Q_{k,1} + \sum_{i > 1} \psi_{0,i} Q_{k,i}, \quad (2.3)$$

- častá kombinace zatížení podle vztahu 6.15b,

$$\sum_{j \geq 1} G_{k,j} + P + \psi_{1,i} Q_{k,1} + \sum_{i > 1} \psi_{2,i} Q_{k,i}, \quad (2.4)$$

- kvazistálou kombinaci zatížení podle vztahu 6.16b,

$$\sum_{j \geq 1} G_{k,j} + \sum_{i \geq 1} \psi_{2,i} Q_{k,i}, \quad (2.5)$$

kde  $G$  označuje stálé zatížení,  $Q$  proměnné zatížení,  $Q_1$  označuje hlavní proměnné zatížení,  $\gamma$  jsou dílčí součinitelé zatížení podle ČSN EN 1990, příloha A [9],  $\psi$  jsou součinitele pro proměnná zatížení podle ČSN EN 1990, příloha A [9] a  $\xi$  je redukční součinitel pro stálé zatížení.

## 2.3 Příklad použití

Pro klasifikaci zatěžovacích jsou v knihovně vytvořeny `Enum*` třídy, které jasné stanovují způsob, jakým způsobem se mají zadávat parametry při vytváření nových instancí. Tyto `Enum` třídy zajišťují, že parametry jsou zadávány konzistentně a správně, což minimalizuje riziko chyb při definování zatěžovacích stavů.

Nejprve musíme do prostředí JupyterLab importovat všechny potřebné třídy a balíčky.

\*`Enum` je speciální datový typ, který umožňuje definovat sadu pojmenovaných konstant.

```
[1]: from desssign.loads.enums import LoadType
      from desssign.loads.enums import LimitState
      from desssign.loads.enums import VariableCategory as cat
      from desssign.loads.enums import ULSCombination as uls
      from desssign.loads.enums import SLSCombination as sls
      from desssign.loads.enums import
↳ULSAlternativeCombination as alt
      from desssign.loads.enums import LoadCaseRelation as rel

      from desssign.loads.load_case import DesignLoadCase
      from desssign.loads.load_case_group import
↳DesignLoadCaseGroup
      from desssign.loads.load_combination_generator.
↳combination_generator import CombinationsGenerator
```

Vytvoříme 5 instancí typu `DesignLoadCase`, dva zatěžovací stavy pro stálé zatížení (G1, G2), jeden zatěžovací stav pro užitné zatížení kategorie A (Q3) a dva zatěžovací stavy pro zatížení sněhem (S4, S5).

```
[2]: G1 = DesignLoadCase(
      label="G1",
      load_type=LoadType.PERMANENT,
  )

      G2 = DesignLoadCase(
          label="G2",
          load_type=LoadType.PERMANENT,
      )

      Q3 = DesignLoadCase(
          label="Q3",
          load_type=LoadType.VARIABLE,
          category=cat.A,
      )

      S4 = DesignLoadCase(
          label="S4",
          load_type=LoadType.VARIABLE,
          category=cat.SNOW_BELLOW_1000_M,
      )

      S5 = DesignLoadCase(
          label="S5",
          load_type=LoadType.VARIABLE,
          category=cat.SNOW_BELLOW_1000_M,
      )
```

Vytvoříme skupiny zatížení, zatěžovací stavy ve skupině pro stálé zatížení, `LG_permanent`, působí v každé kombinaci dohromady. Skupina zatížení pro užitné zatížení, `LG_imposed`, je typu `STANDARD`. Zatěžovací stavy ve skupině pro zatížení sněhem, `LG_snow`, nepůsobí v žádné kombinaci společně.

```
[3]: LG_permanent = DesignLoadCaseGroup(
    load_cases=[G1, G2],
    load_case_relation=rel.TOGETHER,
)

LG_imposed = DesignLoadCaseGroup(
    load_cases=[Q3],
    load_case_relation=rel.STANDARD,
)

LG_snow = DesignLoadCaseGroup(
    load_cases=[S4, S5],
    load_case_relation=rel.EXCLUSIVE,
)
```

Inicializujeme generátor pro generování základních kombinací pro mezní stav únosnosti.

```
[4]: ULS_basic_generator = CombinationsGenerator(
    limit_state=LimitState.ULS,
    combination_type=uls.BASIC
)
```

Pomocí metody `generate_combinations()` vygenerujeme kombinace a uložíme je do proměnné `combinations`.

```
[5]: combinations = ULS_basic_generator.generate_combinations(
    [LG_permanent, LG_imposed, LG_snow]
)
```

Vypíšeme automaticky vygenerovaný název a klíč kombinace, který obsahuje jednotlivé součinitele určené podle ČSN EN 1990.

```
[6]: for combination in combinations:
    print(f"Kombinace: {combination.label}")
    print(f"Klíč: {combination.combination_key}\n")
```

Kombinace: ULS-basic(1)

Klíč: 1.35\*G1+1.35\*G2

Kombinace: ULS-basic(2)

Klíč: 1.35\*G1+1.35\*G2+1.5\*S4

Kombinace: ULS-basic(3)

Klíč: 1.35\*G1+1.35\*G2+1.5\*S5

Kombinace: ULS-basic(4)

Klíč: 1.35\*G1+1.35\*G2+1.5\*Q3

Kombinace: ULS-basic(5)

Klíč: 1.35\*G1+1.35\*G2+1.5\*Q3+1.5\*0.5\*S4

Kombinace: ULS-basic(6)

Klíč: 1.35\*G1+1.35\*G2+1.5\*S4+1.5\*0.7\*Q3

Kombinace: ULS-basic(7)  
 Klíč:  $1.35 \cdot G1 + 1.35 \cdot G2 + 1.5 \cdot Q3 + 1.5 \cdot 0.5 \cdot S5$

Kombinace: ULS-basic(8)  
 Klíč:  $1.35 \cdot G1 + 1.35 \cdot G2 + 1.5 \cdot S5 + 1.5 \cdot 0.7 \cdot Q3$

Pro ilustraci dále vygenerujeme charakteristické kombinace zatížení pro mezní stav použitelnosti.

```
[7]: SLS_char_generator = CombinationsGenerator(
      limit_state=LimitState.SLS,
      combination_type=sls.CHARACTERISTIC
    )

[8]: combinations = SLS_char_generator.generate_combinations(
      [LG_permanent, LG_imposed, LG_snow]
    )

[9]: for combination in combinations:
      print(f"Kombinace: {combination.label}")
      print(f"Klíč: {combination.combination_key}\n")
```

Kombinace: SLS-characteristic(1)  
 Klíč:  $G1 + G2$

Kombinace: SLS-characteristic(2)  
 Klíč:  $G1 + G2 + S4$

Kombinace: SLS-characteristic(3)  
 Klíč:  $G1 + G2 + S5$

Kombinace: SLS-characteristic(4)  
 Klíč:  $G1 + G2 + Q3$

Kombinace: SLS-characteristic(5)  
 Klíč:  $G1 + G2 + Q3 + 0.5 \cdot S4$

Kombinace: SLS-characteristic(6)  
 Klíč:  $G1 + G2 + S4 + 0.7 \cdot Q3$

Kombinace: SLS-characteristic(7)  
 Klíč:  $G1 + G2 + Q3 + 0.5 \cdot S5$

Kombinace: SLS-characteristic(8)  
 Klíč:  $G1 + G2 + S5 + 0.7 \cdot Q3$

Při generování alternativních kombinací pro mezní stav únosnosti se vygeneruje dvojnásobný počet kombinací, vždy jedna kombinace pro redukované hlavní proměnné zatížení a druhá kombinace pro redukované stálé zatížení.

```
[10]: ULS_alt_generator = CombinationsGenerator(
        limit_state=LimitState.ULS,
        combination_type=uls.ALTERNATIVE
    )

[11]: combinations = ULS_alt_generator.generate_combinations(
        [LG_permanent, LG_imposed, LG_snow]
    )

[12]: for combination in combinations:
        print(f"Kombinace: {combination.label}")
        print(f"Klíč: {combination.combination_key}\n")
```

Kombinace: ULS-alternative(1a)  
Klíč: 1.35\*G1+1.35\*G2

Kombinace: ULS-alternative(1b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2

Kombinace: ULS-alternative(2a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.5\*S4

Kombinace: ULS-alternative(2b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2+1.5\*S4

Kombinace: ULS-alternative(3a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.5\*S5

Kombinace: ULS-alternative(3b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2+1.5\*S5

Kombinace: ULS-alternative(4a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.7\*Q3

Kombinace: ULS-alternative(4b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2+1.5\*Q3

Kombinace: ULS-alternative(5a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.7\*Q3+1.5\*0.5\*S4

Kombinace: ULS-alternative(5b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2+1.5\*Q3+1.5\*0.5\*S4

Kombinace: ULS-alternative(6a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.5\*S4+1.5\*0.7\*Q3

Kombinace: ULS-alternative(6b)  
Klíč: 0.85\*1.35\*G1+0.85\*1.35\*G2+1.5\*S4+1.5\*0.7\*Q3

Kombinace: ULS-alternative(7a)  
Klíč: 1.35\*G1+1.35\*G2+1.5\*0.7\*Q3+1.5\*0.5\*S5

Kombinace: ULS-alternative(7b)

Klíč:  $0.85*1.35*G1+0.85*1.35*G2+1.5*Q3+1.5*0.5*S5$

Kombinace: ULS-alternative(8a)

Klíč:  $1.35*G1+1.35*G2+1.5*0.5*S5+1.5*0.7*Q3$

Kombinace: ULS-alternative(8b)

Klíč:  $0.85*1.35*G1+0.85*1.35*G2+1.5*S5+1.5*0.7*Q3$

Takto vygenerované kombinace je možné použít v kombinaci s knihovnou framesss.

## Kapitola 3

### Webová aplikace

V této kapitole budou představeny jednotlivé aplikace, jejich funkce a způsob integrace s vyvinutými nástroji framesss a desssign.

Webová aplikace, vyvinutá v rámci projektu *Vývoj komplexního softwaru pro optimalizaci návrhu a posouzení střešních a stropních konstrukcí*, zahrnuje dvě hlavní aplikace: STŘECHA a STROP. Tato kapitola popisuje jednotlivé aplikace, jejich funkce a integraci s vyvinutými knihovnami framesss a desssign.

Aplikace slouží pro předběžné ověření dimenzí konstrukčních prvků s využitím řešení a výrobků společnosti Wienerberger.

Kromě integrace knihoven představených v předchozích kapitolách bylo v rámci této diplomové práce vyvinuté nové uživatelské prostředí. Z časových důvodů bylo implementováno zatím pouze do aplikace STŘECHA. Aplikaci STROP tento přechod čeká v nejbližší době.

Webová aplikace zatím nemá stanovenou doménu, proto je přístupná přes přesměrování z adresy <https://people.fsv.cvut.cz/www/holanjak/software/wienerberger>.

### 3.1 Aplikace STŘECHA

Aplikace STŘECHA je určena pro předběžný návrh konstrukčních prvků krovu se střešními krytinami a skladbami Tondach. V rámci této diplomové práce došlo k výraznému zlepšení grafického uživatelského rozhraní a interakce s uživatelem.

#### 3.1.1 Původní verze aplikace

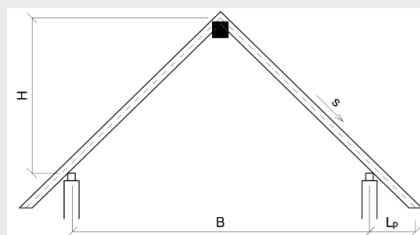
Původní verze aplikace STŘECHA poskytovala základní nástroje pro posouzení krovů. Uživatel procházel jednotlivými kroky, od zadání geometrie objektu, přes zadání lokality pro výpočet klimatických zatížení, zvolení krytiny, skladby střešního pláště, až po definování průřezů a materiálů jednotlivých prvků. Následně byla takto zadaná konstrukce posouzena na několik předem definovaných kombinací zatížení, které uživatel nemohl upravit.

Původní uživatelské prostředí bylo tvořeno několika velkými statickými formuláři, který byly sice přehledné, ale velmi dlouhé. Uživatelé zároveň neměli žádnou vizuální kontrolu nad tím, co zadávají, což mohlo vést k chybám při zadávání dat. Výsledky byly prezentovány ve formě textového výstupu, ve velmi stručné formě, což omezovalo možnosti interaktivní zpětné vazby.

## Zesílení krovu

Tento modul posoudí Vámi zadané **dimenze** nosných prvků a případně navrhne jejich **zesílení**.

## Rozměry krovu



Typ krovu:*	(C) Krov s vrcholovou vaznicí
Výška střechy od pozednice $H$ [m]:*	2.0
Šířka krovu $B$ [m]:*	6.0
Vodorovný přesah střechy $L_p$ [m]:*	0.5
Sklon střechy $s$ [°]:*	33.7
Vzdálenost kroků v podélném směru: [m]:*	1.0
Šířka průřezu krokve [mm]:*	120.0
Výška průřezu krokve [mm]:*	160.0
Svislá vzd.hambalku / kleštiny / stř. vaznice od pozednice [m]:*	0
Šířka průřezu hambalku [mm]:	0
Výška průřezu hambalku [mm]:	0
Vzdálenost sloupků pod vaznicemi v podélném směru: [m]:*	4.5
Šířka průřezu vaznice [mm]:	300.0
Výška průřezu vaznice [mm]:	500.0
Šířka průřezu sloupku [mm]:	180.0
Výška průřezu sloupku [mm]:	180.0
Šířka průřezu kleštin [mm]:	0
Výška průřezu kleštin [mm]:	0

**Obr. 3.1:** STŘECHA – Modul pro finální posouzení 1/2



**Zatížení krovu**

Char. zatížení stálé  $f_k$  [kN/m<sup>2</sup>]:\*

Char. zatížení sněhem  $s_k$  [kN/m<sup>2</sup>]:\*

Char. zatížení větrem (tlak)  $w_{t,k}$  [kN/m<sup>2</sup>]:\*

Char. zatížení větrem (sání)  $w_{s,k}$  [kN/m<sup>2</sup>]:\*

**Materiálové vlastnosti**

Třída dřeva:\*

**Výsledek posouzení**

Krokev: \*

Hambalek: \*

Vaznice: \*

Sloupek: \*

Kleštiny: \*

Obr. 3.2: STŘECHA – Modul pro finální posouzení 2/2

### 3.1.2 Nová verze

Nová verze aplikace STŘECHA přináší výrazné zlepšení v uživatelském rozhraní a interaktivitě. Používá moderní technologie pro dynamickou vizualizaci a umožňuje uživatelům interaktivně modelovat střešní konstrukce.

#### Funkčnost

Nová verze aplikace STŘECHA umožňuje uživatelům projít několika kroky k vygenerování statického schématu se zatíženími a zatěžovacími stavy, podobně jako tomu bylo v původní verzi. Nově však uživatel může měnit geometrii, přidávat a ubírat zatížení, definovat zatěžovací stavy, kombinovat je do kombinací a zobrazovat si výsledky pro jednotlivé stavy zvlášť.

Pro všechny tyto funkce aplikace využívá knihovny framesss a desssign, které byly představeny v předchozích kapitolách.

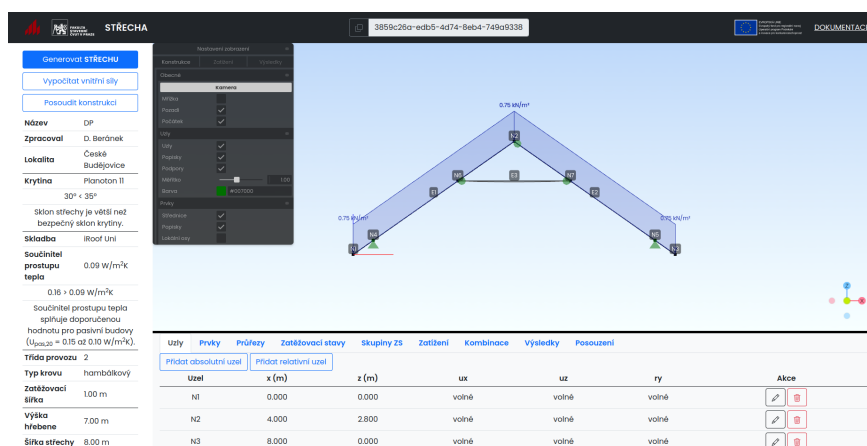
#### Uživatelské prostředí

Nové uživatelské prostředí aplikace STŘECHA nabízí intuitivní a interaktivní rozhraní, které uživatelům umožňuje snadno zadávat data a mít nad nimi vizuální kontrolu.

V hlavní části obrazovky se nachází vizualizace modelu střechy, kde jsou zobrazeny uzly, prvky a aplikovaná zatížení. Uživatel může interaktivně přidávat uzly, prvky a zatížení pomocí ovládacích prvků v dolní části obrazovky. Tento přístup zajišťuje, že uživatelé mají přehled o všech zadaných datech a mohou je snadno upravovat a kontrolovat.

Uživatel může v prostředí aplikace:

- generovat střešní konstrukci z předdefinovaných typů střešních konstrukcí,
- přidávat, editovat a mazat entity,
- vypočítat vnitřní síly,
- posoudit konstrukci.



Obr. 3.3: STŘECHA – Nové uživatelské prostředí

Na obrázku [obr. 3.3](#) je rozvržení nového uživatelského prostředí, na levé straně obrazovky se nachází panel s informacemi o projektu, jako je název, zpracovatel, lokalita, krytina, sklon střechy, skladba, součinitel prostupu tepla, typ krovu a další. V hlavní části obrazovky se nachází vizualizace statického modelu, kde jsou zobrazeny uzly, prvky a aplikovaná zatížení. Uživatel může interaktivně přidávat, měnit a mazat uzly, prvky, zatížení a veškeré další objekty pomocí ovládacích prvků v záložkách v dolní části obrazovky. Tento přístup zajišťuje, že uživatelé mají přehled o všech zadaných datech a mohou je snadno upravovat a kontrolovat.

### Spodní panel

V dolní části obrazovky se nachází karty s ovládacími prvky a přehlednými tabulkami, které zobrazují veškeré informace o modelované konstrukci. Tyto karty zahrnují:

- **Uzly:** Seznam všech uzlů v modelu s jejich souřadnicemi a definicí podepření v jednotlivých směrech.
- **Prvky:** Seznam všech prvků, pro každý prvek se zde zobrazují jeho výchozí a koncový uzel, relativně zadané uzly, koncové klouby, délka a průřez.
- **Průřezy:** Seznam všech prvků s definicí materiálu a základních geometrických veličin.
- **Zatěžovací stavy:** Seznam všech zatěžovacích stavů, jejich typ, kategorie, třída trvání zatížení a odpovídající součinitele podle ČSN EN 1990.

- **Skupiny ZS:** Skupiny zatěžovacích stavů, ze kterých se generují kombinace.
- **Zatížení:** Seznam spojitých a bodových zatížení na prvcích a v uzlech. Na této kartě lze přepínat mezi zatěžovacími stavy. Zatížení pro aktivní zatěžovací stav se zobrazuje ve vizualizaci.
- **Kombinace:** Seznam všech kombinací, jejich mezní stav, typ a kombinací klíč.
- **Výsledky:** Seznam vnitřních sil ve všech průřezech, kde se může vyskytovat extrém vnitřních sil. Na této kartě lze přepínat mezi zatěžovacími stavy a kombinacemi zatížení. Pro aktivní výběr se vykreslují výsledky v modelu.
- **Posouzení:** Přehled využití jednotlivých prutů.

Na každé kartě je tlačítko pro přidání nové entity do modelu. Zároveň lze každou entitu, kromě zatížení vlastní tíhou, vymazat nebo upravit.

Uzly	Prvky	Průřezy	Zatěžovací stavy	Skupiny ZS	Zatížení	Kombinace	Výsledky	Posouzení
Přidat absolutní uzel		Přidat relativní uzel						
Uzel	x (m)	z (m)	ux	uz	ry	Akce		
N1	0.000	0.000	volné	volné	volné			
N2	4.000	2.800	volné	volné	volné			
N3	8.000	0.000	volné	volné	volné			

Obr. 3.4: STŘECHA – Záložka s uzly

Uzly	Prvky	Průřezy	Zatěžovací stavy	Skupiny ZS	Zatížení	Kombinace	Výsledky	Posouzení
Přidat prvek								
Prvek	Výchozí uzel	Koncový uzel	Relativní uzly	Koncové klouby	Délka (m)	Průřez	Akce	
E1	N1	N2	N4, N6	False, False	4.883	Krokev		
E2	N2	N3	N5, N7	True, False	4.883	Krokev		
E3	N6	N7		True, True	2.800	Hambálek		

Obr. 3.5: STŘECHA – Záložka s prvky





Uzly	Prvky	Průřezy	Zatěžovací stavy	Skupiny ZS	Zatížení	Kombinace	Výsledky	Posouzení		
Přidat zatěžovací stav										
Zatěžovací stav	Popis	Typ	Kategorie	Třída trvání zatížení	$\gamma_{f,sup}$	$\xi$	$\psi_0$	$\psi_1$	$\psi_2$	Akce
G0	Vlastní tíha	stálé		stálé	1.35	0.85				
G1	Ostatní stálé	stálé		stálé	1.35	0.85				
Q2	Užitné (I)	proměnné	Kategorie H: Střechy nepřístupné s...	krátkodobé	1.5		0.0	0.0	0.0	
S3	Sníh, stav (I)	proměnné	Zatížení sněhem pro stavby umístěn...	střednědobé	1.5		0.5	0.2	0.0	

Obr. 3.6: STŘECHA – Záložka se zatěžovacími stavy

Uzly Prvky Průřezy Zatěžovací stavy Skupiny ZS **Zatížení** Kombinace Výsledky Posouzení

Aktivní zatěžovací stav S3 - Sníh, stav (i)

Přidat spojitě zatížení na prvek

Prvek	Souřadný systém	Směr	X <sub>1,lok</sub>	X <sub>2,lok</sub>	f <sub>x,1</sub>	f <sub>x,2</sub>	f <sub>z,1</sub>	f <sub>z,2</sub>	Akce
E1	GSS	Průmět	0 %	100 %	0.00 kN/m <sup>2</sup>	0.00 kN/m <sup>2</sup>	-0.67 kN/m <sup>2</sup>	-0.67 kN/m <sup>2</sup>	 
E2	GSS	Průmět	0 %	100 %	0.00 kN/m <sup>2</sup>	0.00 kN/m <sup>2</sup>	-0.67 kN/m <sup>2</sup>	-0.67 kN/m <sup>2</sup>	 

Přidat bodové zatížení na prvek

Prvek	X <sub>lok</sub>	Souřadný systém	F <sub>x</sub>	F <sub>z</sub>	M <sub>y</sub>	Akce
Nebylo nalezeno žádné bodové zatížení na prvcích.						

Přidat uzlové zatížení

Uzel	F <sub>x</sub>	F <sub>z</sub>	M <sub>y</sub>	Akce
Nebylo nalezeno žádné uzlové zatížení.				

Obr. 3.7: STŘECHA – Záložka se zatížením

Uzly Prvky Průřezy Zatěžovací stavy Skupiny ZS **Zatížení** Kombinace Výsledky Posouzení

Přidat kombinaci

Vymazat vše

Kombinace	Mezní stav	Typ	Kombinační klíč	Akce
CO1	MSÚ	základní	1.35*G0+1.35*G1	 
CO2	MSÚ	základní	1.35*G0+1.35*G1+1.5*Q2	 
CO3	MSÚ	základní	1.35*G0+1.35*G1+1.5*W6	 
CO4	MSÚ	základní	1.35*G0+1.35*G1+1.5*W7	 

Obr. 3.8: STŘECHA – Záložka s kombinacemi zatížení

## Definování entit

Téměř každou entitu lze editovat. Po kliknutí na tlačítko **Přidat** v horní části karty nebo tlačítko **Editovat** na řádku s entitou se zobrazí okno umožňující zadání potřebných údajů.

Absolutní uzel

Umístění uzlu

x  m z  m

Podpěření

Bráněno posunu ve směru osy

x volné  K  MN/m

z volné  K  MN/m

Bráněno pootočení okolo osy

y volné  K  MNm/rad

Zrušit Uložit

Obr. 3.9: STŘECHA – Okno pro přidání nového uzlu

Obr. 3.10: STŘECHA – Okno pro přidání nového prvku

Obr. 3.11: STŘECHA – Okno pro přidání nového zatěžovacího stavu

### 3.1.3 Příklad

Pro demonstraci schopností webové aplikace provedeme opět výpočet příkladu 5.2 ze *Sbírky příkladů stavební mechaniky* [15, Příklad 5.2], který byl již vypočítán v kapitole 1.6.8.

Protože v aplikaci STŘECHA můžeme zadávat pouze dřevěné obdélníkové průřezy, musíme určit rozměry průřezu tak, abychom dosáhli stejné ohybové a normálové tuhosti průřezu. Použijeme dřevo třídy C24, pro které platí  $E_{m,0,k} = 7.4 \text{ GPa}$ .

V zadání příkladu je uvedeno, že Youngův modul pružnosti je  $E = 30 \text{ GPa}$ , moment setrvačnosti k ose, kolem které jsou pruty namáhané ohybem  $I_y = 1 \times 10^{-3} \text{ m}^4$  a plochu  $A = 4 \times 10^{-3} \text{ m}^2$ .

Vyřešíme dvě rovnice pro dvě neznámé,

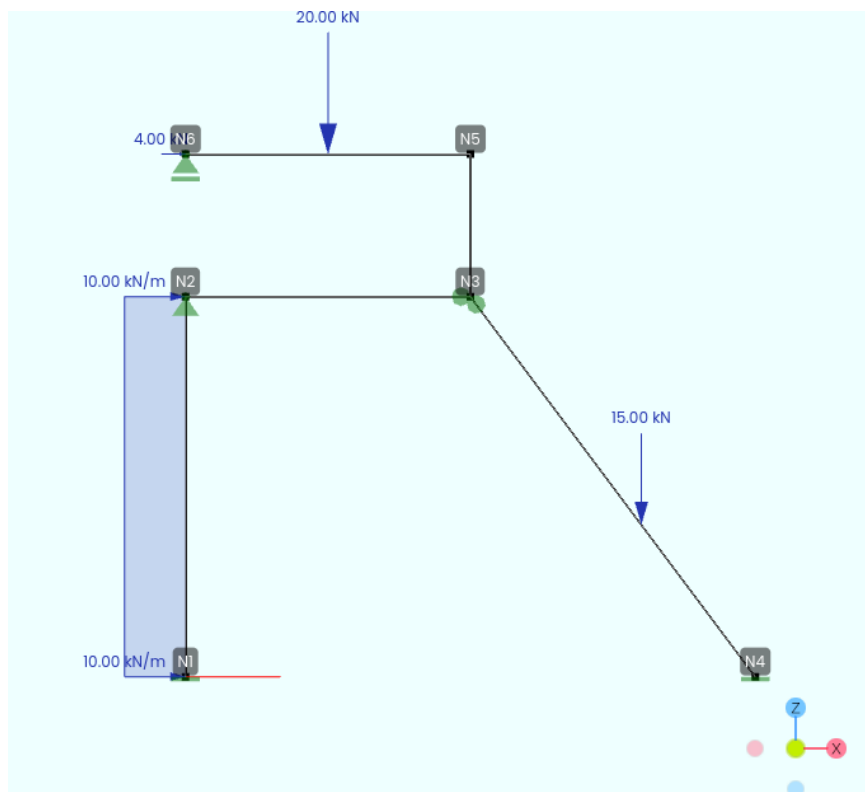
$$EI_y = E_{m,0,k} \frac{bh^3}{12}, \quad (3.1)$$

$$EA = E_{m,0,k}bh. \quad (3.2)$$

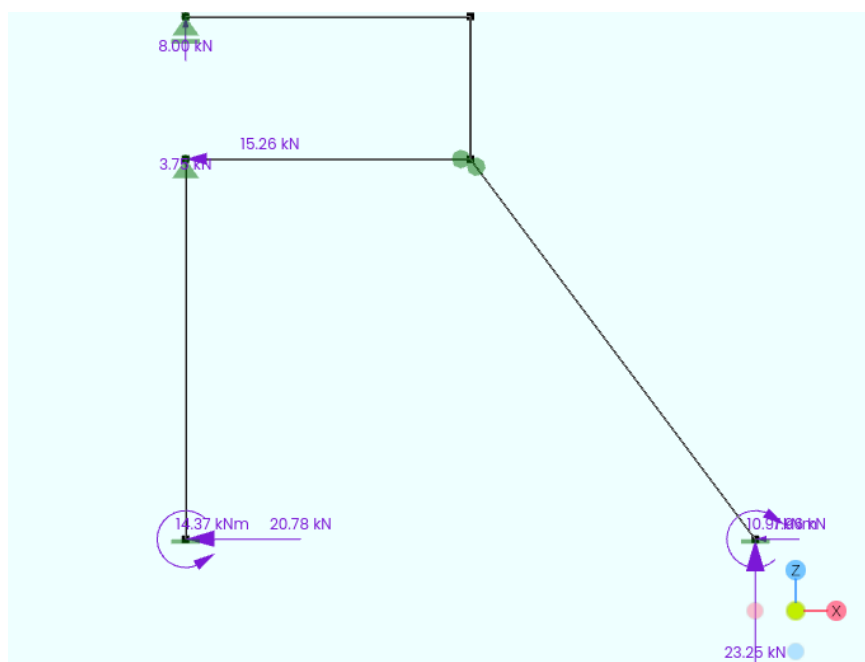
Po vyřešení vychází  $b \approx 9.362 \text{ mm}$  a  $h \approx 1732 \text{ mm}$ .

Z následujících obrázků je patrné, že velikosti reakcí a průběhy vnitřních sil se přesně shodují s průběhy uvedenými v kapitole 1.6.8.

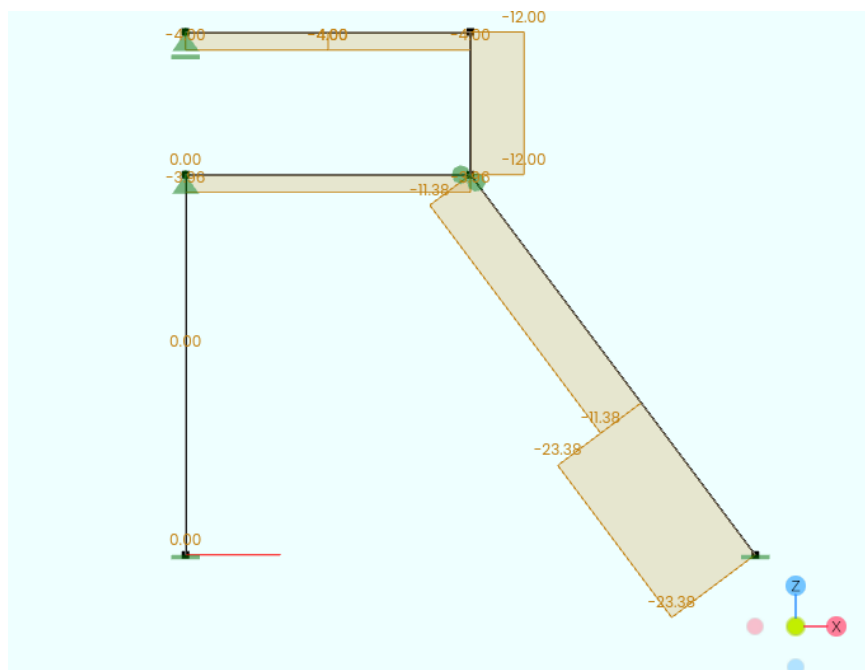
Tento příklad lze otevřít zadáním unikátního identifikačního kódu 0023d0bc-bc20-46b5-bfe7-dadc9bd10dfc.



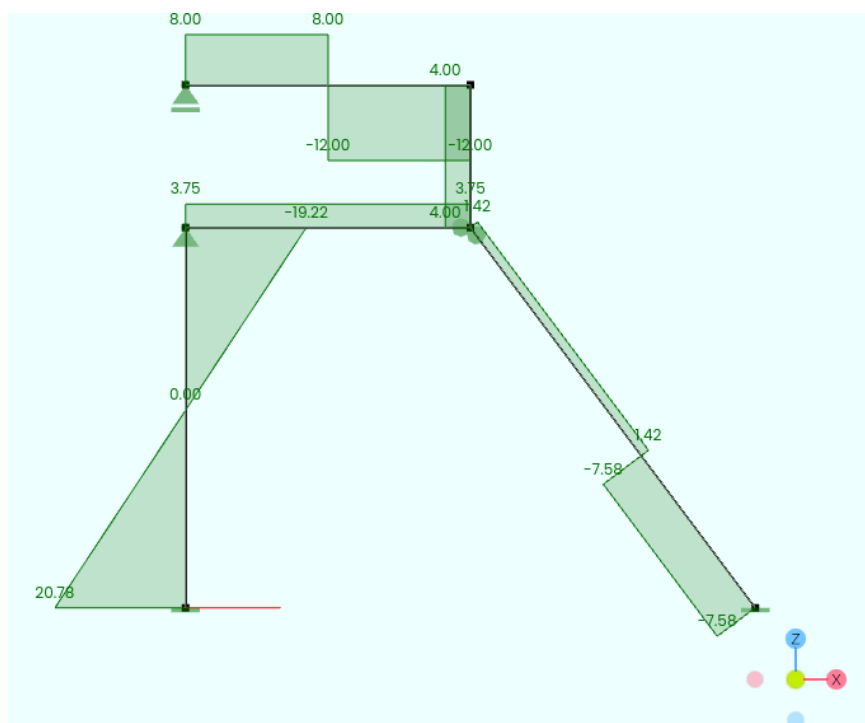
Obr. 3.12: Zatížení



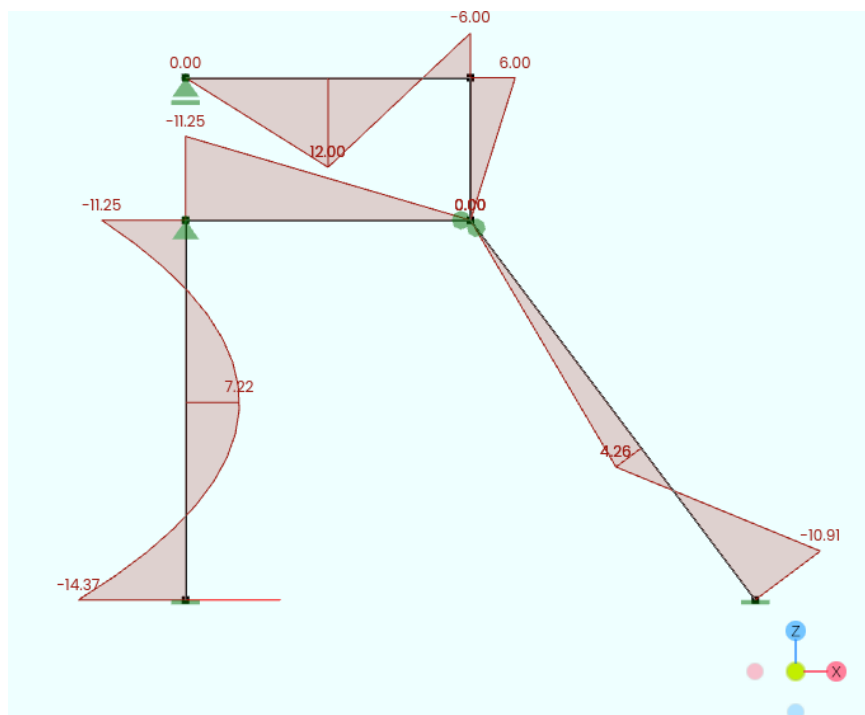
Obr. 3.13: Reakce



Obr. 3.14: Normálová síla



Obr. 3.15: Posouvající síla



Obr. 3.16: Ohybové momenty

## 3.2 Aplikace STROP

Aplikace STROP je určená pro návrh a posouzení stropních konstrukcí Porotherm MIAKO, které jsou tvořené cihelnými vložkami MIAKO a keramobetonovými stropními trámy.

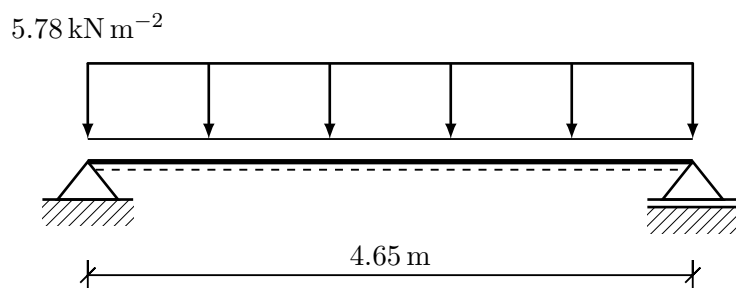
Pomocí aplikace lze posoudit stropní konstrukci, případně optimalizovat konstrukci na základě uživatelem definovaných parametrů.

V rámci předchozí kapitoly bylo představeno integrování knihoven framesss a desssign do uživatelského rozhraní aplikace STŘECHA. V rámci pokračující práce na představené webové aplikaci budou tyto knihovny v budoucnu integrovány i do aplikace STROP. Na následujících příkladech bude představen aktuální schopnosti aplikace a možnosti budoucího rozšíření a vylepšení.

### 3.2.1 Prostý nosník

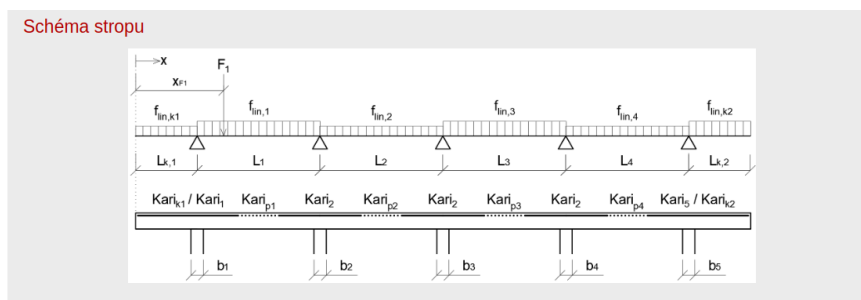
Podle [27, s. 261] je maximální hodnota návrhového spojitého rovnoměrného zatížení (bez vlastní tíhy zmonolitněné konstrukce), kterou je možno na zmonolitněný strop přiložit, aby byla zachována požadovaná únosnost konstrukce pro beton třídy C20/25 a jednoduchý stropní trám POT 475 při tloušťce stropní konstrukce 250 mm a osově vzdálenosti trámů 625 mm rovna hodnotě  $5.78 \text{ kN m}^{-2}$ .





Obr. 3.17: Zadání příkladu

Na obr. 3.18 jsou zobrazeny hodnoty, které musíme vyplnit ve formuláři před odesláním dat k výpočtu. Jedná se pouze o statický obrázek.



Obr. 3.18: Schéma stropu

Nejprve je nutné definovat jednotlivá pole stropu. Z obrázku 3.19 je patrné, že jediné nenulové pole je pole na druhém řádku.

Pole stropu - rozpětí, POT nosníky a kari sítě					
Osová rozpětí [m]		Počty POT nosníků		Typy POT nosníků	
L <sub>k1</sub> *	0	n <sub>k1</sub> *	1	POT <sub>k1</sub> *	475/902
L <sub>1</sub> *	4.65	n <sub>1</sub> *	1	POT <sub>1</sub> *	475/902
L <sub>2</sub> *	0	n <sub>2</sub> *	1	POT <sub>2</sub> *	175/902
L <sub>3</sub> *	0	n <sub>3</sub> *	1	POT <sub>3</sub> *	175/902
L <sub>4</sub> *	0	n <sub>4</sub> *	1	POT <sub>4</sub> *	175/902
L <sub>k2</sub> *	0	n <sub>k2</sub> *	1	POT <sub>k2</sub> *	475/902
				Kari sítě v polích	
				Kari <sub>k1</sub> *	5/100
				Kari <sub>p1</sub> *	5/100
				Kari <sub>p2</sub> *	5/100
				Kari <sub>p3</sub> *	5/100
				Kari <sub>p4</sub> *	5/100
				Kari <sub>k2</sub> *	5/100

Obr. 3.19: Pole stropu - rozpětí, POT nosníky a kari sítě

Tato část je určena především k zadání nadpodporové výztuže. Protože nyní řešíme prostý nosník, nemusíme tuto část řešit.

Podpory - šířky, kari sítě a příložky					
Šířka podpory [mm]		Typ kari sítě		Průměr příložek	
b <sub>1</sub> *	250	Kari <sub>1</sub> *	5/100	ø <sub>1</sub> *	0
b <sub>2</sub> *	250	Kari <sub>2</sub> *	5/100	ø <sub>2</sub> *	0
b <sub>3</sub> *	0	Kari <sub>3</sub> *	5/100	ø <sub>3</sub> *	0
b <sub>4</sub> *	0	Kari <sub>4</sub> *	5/100	ø <sub>4</sub> *	0
b <sub>5</sub> *	0	Kari <sub>5</sub> *	5/100	ø <sub>5</sub> *	0
				Počet příložek	
				n <sub>1</sub> *	0
				n <sub>2</sub> *	0
				n <sub>3</sub> *	0
				n <sub>4</sub> *	0
				n <sub>5</sub> *	0

Obr. 3.20: Podpory - šířky, kari sítě a příložky

V další části definujeme průřez stropu, jak bylo zmíněno na začátku, jedná se o strop tloušťky 250 mm s osovou vzdáleností trámů 625 mm.

**Rozměry stropu**

Osová vzdálenost trámu vlevo [mm]:\*

Osová vzdálenost trámu vpravo [mm]:\*

Typ stropu:\*

Výška vložky [mm]:\*

Výška nadbetonávky [mm]:\*

Tloušťka stropu [mm]:\*

**Obr. 3.21:** Rozměry stropu

Materiál uvažujeme jako beton C20/25. Protože řešíme mezní stav únosnosti, součinitel dotvarování betonu necháme na defaultní hodnotě.

**Materiálové vlastnosti**

Třída betonu:\*

Součinitel dotvarování betonu:\*

**Limitní hodnoty**

Poměr rozpětí prvku ku limitnímu průhybu:\*

**Obr. 3.22:** Materiálové vlastnosti

Dále zadáme přitížení stropu, lze zadat zatížení plošné (např. od podlahy, užité zatížení) a liniové (např. příčka rovnoběžná s osou nosníku). Pokud chceme, aby program vypočítal i průhyb, musíme vyplnit charakteristickou hodnotu. V tomto případě si vystačíme s hodnotou návrhovou. Přestože máme pouze jedno pole, na stránce se zobrazují šedě zabarvená okna pro ostatní pole.

**Přítížení stropu (zatížení bez vlastní tíhy zmonolitněné konstrukce)**

Char. plošná zatížení [kN/m <sup>2</sup> ]	Návr. plošná zatížení [kN/m <sup>2</sup> ]	Char. liniová zatížení [kN/m]	Návr. liniová zatížení [kN/m]
$f_{p,k,1}$ :* <input type="text" value="0"/>	$f_{p,d,1}$ :* <input type="text" value="0"/>	$f_{l,k,1}$ :* <input type="text" value="0"/>	$f_{l,d,1}$ :* <input type="text" value="0"/>
$f_{p,k,2}$ :* <input type="text" value="0"/>	$f_{p,d,2}$ :* <input type="text" value="5.78"/>	$f_{l,k,2}$ :* <input type="text" value="0"/>	$f_{l,d,2}$ :* <input type="text" value="0"/>
$f_{p,k,3}$ :* <input type="text" value="0"/>	$f_{p,d,3}$ :* <input type="text" value="0"/>	$f_{l,k,3}$ :* <input type="text" value="0"/>	$f_{l,d,3}$ :* <input type="text" value="0"/>
$f_{p,k,4}$ :* <input type="text" value="0"/>	$f_{p,d,4}$ :* <input type="text" value="0"/>	$f_{l,k,4}$ :* <input type="text" value="0"/>	$f_{l,d,4}$ :* <input type="text" value="0"/>
$f_{p,k,k2}$ :* <input type="text" value="0"/>	$f_{p,d,k2}$ :* <input type="text" value="0"/>	$f_{l,k,k2}$ :* <input type="text" value="0"/>	$f_{l,d,k2}$ :* <input type="text" value="0"/>

**Obr. 3.23:** Přítížení stropu (zatížení bez vlastní tíhy zmonolitněné konstrukce)

V další části lze zadat bodová zatížení, podobně jako u plošného zatížení máme na výběr z typu Sloup a typu Příčná stěna. Opět je nutné zadat zvlášť charakteristickou i návrhovou hodnotu. Tímto končí zadávání dat a je možné spustit výpočet pomocí tlačítka Spustit posouzení stropu.

**Bodová zatížení na nosníku**

Vzdálenost síly od levého kraje nosníku [m]	Typ síly (zatížení od sloupu nebo od stěny)	Charakteristická hodnota zatížení [kN] nebo [kN/m]	Návrhová hodnota zatížení [kN] nebo [kN/m]
$x_{F1}$ :*	Typ:*	$F_{1,k}$ :*	$F_{1,d}$ :*
$x_{F2}$ :*	Typ:*	$F_{2,k}$ :*	$F_{2,d}$ :*
$x_{F3}$ :*	Typ:*	$F_{3,k}$ :*	$F_{3,d}$ :*
$x_{F4}$ :*	Typ:*	$F_{4,k}$ :*	$F_{4,d}$ :*
$x_{F5}$ :*	Typ:*	$F_{5,k}$ :*	$F_{5,d}$ :*
$x_{F6}$ :*	Typ:*	$F_{6,k}$ :*	$F_{6,d}$ :*

Poznámka:  
Charakteristické hodnoty zatížení jsou využívány pro výpočet a posouzení MSP (průhyb). Návrhové hodnoty zatížení jsou využívány pro výpočet a posouzení MSÚ.

Spustit posouzení stropu

Obr. 3.24: Bodová zatížení na nosníku

Po spuštění stropu server vygeneruje unikátní identifikační kód. Po kliknutí na tlačítko Přejít k zobrazení výsledků posouzení budeme přesměrováni na stránku s výsledky, která má identické pole jako stránka pro zadávání, rozdíl je však v tom, že hodnoty už nelze upravovat, jsou zde pouze vypsány.

**Výsledky posouzení**

Poznamenejte si níže uvedený kód! Budete jej potřebovat pro přístup k výsledkům na další stránce.

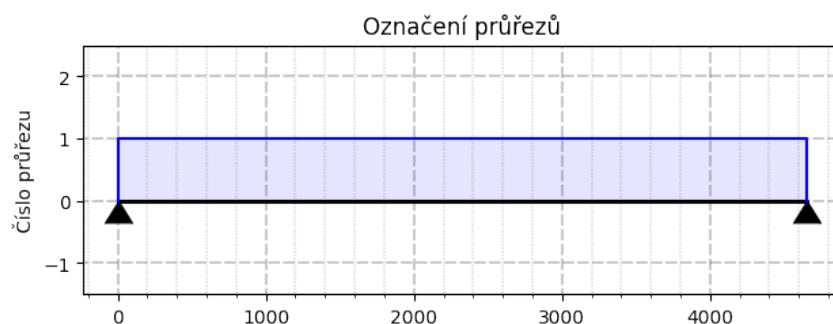
Identifikační kód posudku:\*

posouzeni\_33cb42766b4442fc93205f95572846e5

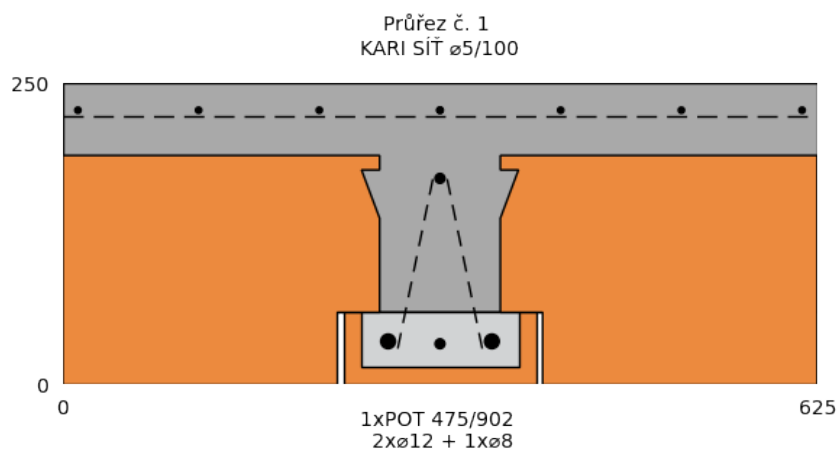
Přejít k zobrazení výsledků posouzení

Obr. 3.25: Výsledky posouzení

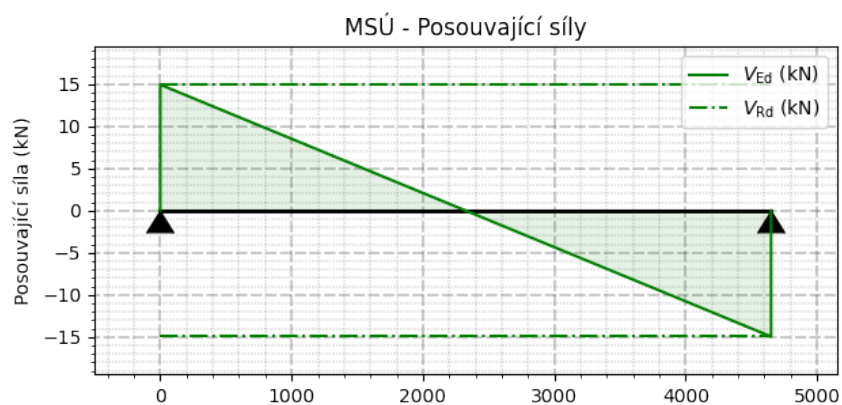
Aplikace STROP zároveň vygeneruje statické obrázky, které jsou na stránce s posouzením.



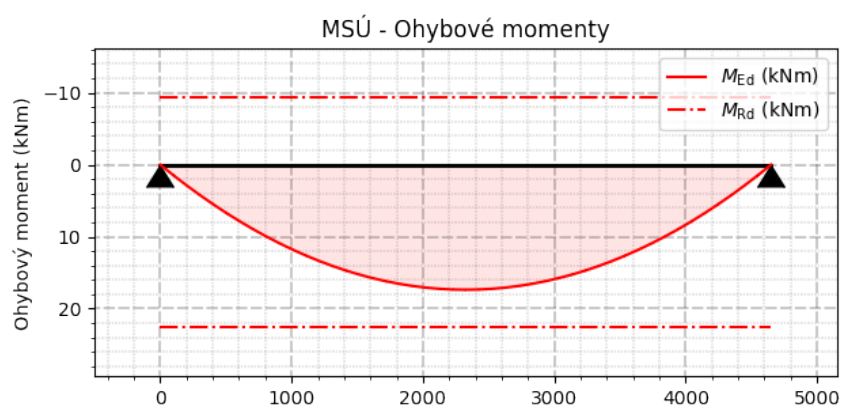
Obr. 3.26: Označení průřezů po délce nosníku



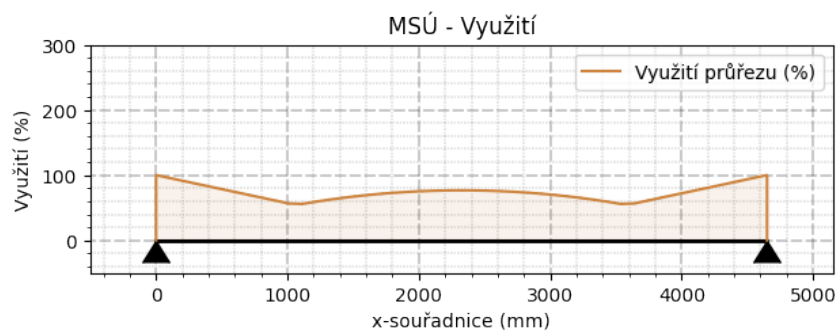
Obr. 3.27: Průřez stropní konstrukce



Obr. 3.28: Posouvající síla



Obr. 3.29: Ohybový moment



**Obr. 3.30:** Využití průřezů

Z obrázku 3.30 je patrné, že konstrukce je využita na 100%, což znamená, že program dává shodné výsledky s tabulkou uvedenou v *Podkladu pro navrhování* [27].

## Závěr a diskuze

V první kapitole byly představeny základy teorie pružnosti, včetně veličin pro popis chování libovolného pružného tělesa a vztahů mezi nimi. Byl uveden princip virtuálních prací, ze kterého vychází princip virtuálních posunutí. Dále jsme se zabývali přibližným řešením úlohy dle teorie pružnosti, zobecněnými podmínkami rovnováhy a výpočtem matice tuhosti a vektoru zatížení. Poté jsme se zaměřili na 1D napjatost a popis pružného chování prutových prvků. Pomocí knihovny SymPy byly odvozeny matice tuhostí a vektory zatížení pro nejběžnější prutové prvky: tažený-tlačený prut, ohýbaný prut bez vlivu smyku a ohýbaný prut s vlivem smyku. Následně byl představen princip statické kondenzace a transformace vztahů z lokální do globální souřadné soustavy.

Dále byla představena objektově orientovaná knihovna pro výpočty prutových konstrukcí, vytvořená v rámci této práce. Byly popsány externí open-source nástroje použité při vývoji, způsob instalace knihovny, včetně tvorby virtuálního prostředí, a jednotlivé funkce programu. Každý modul byl podrobně rozebrán a byly uvedeny obecné předpoklady přijaté při vývoji knihovny. Následně byl popsán algoritmus výpočtu celého modelu, použití řídkých matic a jejich implementace. Bylo také vysvětleno, jakým způsobem je knihovna testována, včetně významu unit testů a verifikačních testů. Jeden z verifikačních testů byl detailně rozebrán v prostředí JupyterLab, kde byly hodnoty porovnány s referenčními hodnotami ze sbírky příkladů stavební mechaniky.

V další kapitole byla představena knihovna pro navrhování konstrukcí podle Eurokódů, včetně instalace, architektury, detailního rozboru jednotlivých modulů a integrace s knihovnou pro výpočty prutových konstrukcí. Byl uveden příklad použití pro vygenerování kombinací zatížení podle ČSN EN 1990.

V poslední části byla představena webová aplikace vyvinutá v rámci projektu *Vývoj komplexního softwaru pro optimalizaci návrhu a posouzení střešních a stropních konstrukcí*, která prošla výrazným zlepšením funkčnosti. Nejprve byla představena aplikace STŘECHA, určená pro předběžný návrh konstrukčních prvků krovu se střešními krytinami a skladbami Tondach. Do aplikace STŘECHA bylo implementováno nové uživatelské prostředí a výpočetní nástroje představené v prvních dvou kapitolách této práce. Byl znovu vyřešen příklad ze sbírky [15] a porovnány výsledky s referenčními průběhy vnitřních sil. Dále byla na krátkém případě předvedena funkčnost aplikace STROP.

V rámci této diplomové práce bylo dosaženo několika významných cílů. Byly vyvinuty a verifikovány výpočetní nástroje (knihovny), které umožňují detailní analýzu prutových konstrukcí. Tyto nástroje lze použít pro analýzu

jakýchkoliv prutových střešních a stropních konstrukcí. V rámci práce bylo popsáno využití těchto knihoven pro zlepšení uživatelského rozhraní aplikace STŘECHA. Tato aplikace nyní představuje intuitivní a efektivní nástroj pro inženýry a projektanty. V rámci pokračující práce na představené webové aplikaci budou tyto knihovny v budoucnu integrovány i do aplikace STROP.

Hlavním přínosem jsou nově vytvořené knihovny pro analýzu prutových konstrukcí a vylepšení webové aplikace pro předběžné posouzení střešních a stropních konstrukcí. Neméně přínosný je i ucelený popis problematiky teorie pružnosti a představení knihovny SymPy pro symbolické výpočty.

Budoucí směřování práce zahrnuje dokončení integrace nově vytvořených knihoven do webové aplikace a tvorba nového uživatelského rozhraní pro aplikaci STROP. Důležitým krokem bude také implementace dalších verifikačních testů a srovnání s dalšími referenčními hodnotami.

## Bibliografie

- [1] American Wood Council (AWC). *DESIGN AID No. 6 — BEAM DESIGN FORMULAR WITH SHEAR AND MOMENT DIAGRAM*. Dostupné z <https://awc.org/wp-content/uploads/2021/12/AWC-DA6-BeamFormulas-0710.pdf>. 2007.
- [2] ČSN EN 1991-1-1:2004. *Eurokód 1: Zatížení konstrukcí - Část 1-1: Obecná zatížení – Objemové tíhy, vlastní tíha a užitná zatížení pozemních staveb*. Praha: Český normalizační institut. Břez. 2004.
- [3] ČSN EN 1995-1-1:2006. *Eurokód 5: Navrhování dřevěných konstrukcí - Část 1-1: Obecná pravidla – Společná pravidla a pravidla pro pozemní stavby*. Praha: Český normalizační institut. Pros. 2006.
- [4] D. Beránek. *Nelineární analýza stropních systémů z trámů a vložek*. Bakalářská práce. Dostupné z [https://dspace.cvut.cz/bitstream/handle/10467/102674/F1-BP-2022-Beranek-Daniel-220515\\_BAKALARKA.pdf](https://dspace.cvut.cz/bitstream/handle/10467/102674/F1-BP-2022-Beranek-Daniel-220515_BAKALARKA.pdf). Květen 2022.
- [5] Bittnar, Z.; Šejnoha, J. *Numerické metody mechaniky 1*. Vydavatelství ČVUT, Praha, 1992. ISBN: 80-01-00855-X.
- [6] Black contributors. *Black code formatter*. [software]. [cit. 2024-04-20]. Dostupné z <https://black.readthedocs.io/en/stable/>.
- [7] Brožovský, J.; Materna, A. *Základy matematické teorie pružnosti*. Online. [cit. 2024-04-10]. Dostupné z [https://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/mat\\_teorie\\_pruznosti.pdf](https://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/mat_teorie_pruznosti.pdf). 2012.
- [8] ČSN EN 1991-1-4 ed.2:2020. *Eurokód 1: Zatížení konstrukcí - Část 1-3: Obecná zatížení – Zatížení větrem*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. Lis. 2020.
- [9] ČSN EN 1990 ed.2:2021. *Eurokód: Zásady navrhování konstrukcí*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. Ún. 2021.
- [10] ČSN EN 1991-1-3 ed.2:2022. *Eurokód 1: Zatížení konstrukcí - Část 1-3: Obecná zatížení – Zatížení sněhem*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. Led. 2022.
- [11] git contributors. *git - distributed version control system*. [software]. [cit. 2024-04-20]. Dostupné z <https://git-scm.com/>.
- [12] GitHub contributors. *GitHub*. [software]. [cit. 2024-04-20]. Dostupné z <https://github.com/>.



- [13] Charles R. Harris et al. „Array programming with NumPy“. In: *Nature* 585.7825 (zář. 2020), s. 357–362. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [14] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), s. 90–95. DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- [15] Jíra, A.; Jandeková, D.; Novotná, E.; Hájková, P., ed. *Sbírka příkladů stavební mechaniky: princip virtuálních sil, silová metoda, deformační metoda*. Dostupné z [https://mech.fsv.cvut.cz/wiki/images/9/9e/Sbirka\\_prikladu\\_SNK.pdf](https://mech.fsv.cvut.cz/wiki/images/9/9e/Sbirka_prikladu_SNK.pdf). ČVUT v Praze, 2019. ISBN: 978-80-01-06677-5.
- [16] O. Jiroušek. *Metoda konečných prvků - poznámky k přednáškám*. Online. [cit. 2024-03-14]. Dostupné z [http://mech.fd.cvut.cz/members/jirousek/download/k618y2m1/ymkp\\_fem.pdf](http://mech.fd.cvut.cz/members/jirousek/download/k618y2m1/ymkp_fem.pdf). 2006.
- [17] Lourenço, P. B.; Gaetani, A. *Finite Element Analysis for Building Assessment: Advanced Use and Practical Recommendations*. Routledge, New York, 2022. DOI: [10.1201/9780429341564](https://doi.org/10.1201/9780429341564).
- [18] Aaron Meurer et al. „SymPy: symbolic computing in Python“. In: *PeerJ Computer Science* 3 (led. 2017), e103. ISSN: 2376-5992. DOI: [10.7717/peerj-cs.103](https://doi.org/10.7717/peerj-cs.103). URL: <https://doi.org/10.7717/peerj-cs.103>.
- [19] MyPy contributors. *MyPy type checker*. [software]. [cit. 2024-04-20]. Dostupné z <https://www.mypy-lang.org/>.
- [20] Nox contributors. *Nox*. [software]. [cit. 2024-04-20]. Dostupné z <https://nox.thea.codes/en/stable/>.
- [21] Poetry contributors. *Poetry*. [software]. [cit. 2024-04-20]. Dostupné z <https://python-poetry.org/>.
- [22] pre-commit contributors. *pre-commit*. [software]. [cit. 2024-04-20]. Dostupné z <https://pre-commit.com/>.
- [23] J.S. Przemieniecki. *Theory of Matrix Structural Analysis*. Dover Civil and Mechanical Engineering, Dover, 1985. ISBN: 9780486649481.
- [24] Pytest contributors. *Pytest framework*. [software]. [cit. 2024-04-20]. Dostupné z <https://docs.pytest.org/en/8.2.x/>.
- [25] Rafael Lopez Rangel a Luiz Fernando Martha. „LESM—An object-oriented MATLAB program for structural analysis of linear element models“. In: *Computer Applications in Engineering Education* 27.3 (2019), s. 553–571. DOI: <https://doi.org/10.1002/cae.22097>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.22097>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22097>.
- [26] Ruff contributors. *Ruff linter*. [software]. [cit. 2024-04-20]. Dostupné z <https://docs.astral.sh/ruff/>.
- [27] Wienerberger s.r.o. *Podklad pro navrhování Porotherm*. Led. 2024.
- [28] SonarCloud contributors. *SonarCloud*. [software]. [cit. 2024-04-20]. Dostupné z <https://sonarcloud.io/>.
- [29] Sphinx contributors. *Sphinx documentation generator*. [software]. [cit. 2024-04-20]. Dostupné z <https://www.sphinx-doc.org/en/master/>.

- [30] Šejnoha, J.; Bittnarová, J. *Pružnost a pevnost 10*. Vydavatelství ČVUT, Praha, 1997.
- [31] Šejnoha, J.; Bittnarová, J. *Pružnost a pevnost 20*. Vydavatelství ČVUT, Praha, 1998. ISBN: 80-01-02709-0.
- [32] Štefan, R.; Holan, J.; Beránek, D.; Petrášek, I.; Jirkovský, V. *Komplexní software pro optimalizaci návrhu a posouzení střešních a stropních konstrukcí*. 2023. URL: <https://people.fsv.cvut.cz/www/holanjak/software/wienerberger>.
- [33] Pauli Virtanen et al. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. In: *Nature Methods* 17 (2020), s. 261–272. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).