Mezní plastická analýza rovinných rámů

Semestrální práce z předmětu Stavební mechanika 3

Jakub Mareš

Červen 2020

1 Úvod

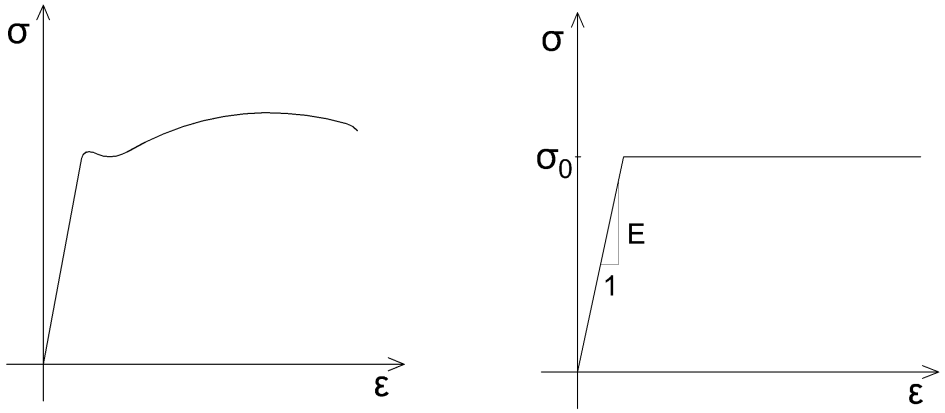
Tento text se bude zabývat výpočtem únosnosti rovinných rámů, s ohledem na předepsané zatížení, s využitím statické neurčitosti konstrukce. To znamená, že se některým částem konstrukce dovoluje dosáhnout mezního plastického stavu. Text je zaměřen na konkrétní způsoby výpočtu s využitím výpočetní techniky a znalostí z teorie pružnosti, pevnosti. Proto je zde pro začátek uvedeno stručné opakování.

V rámci práce jsou skalární veličiny značeny pomocí kurzívy, a, vektory jsou psány kurzívou a tučně, např. $\boldsymbol{x}$, a matice jsou zapisovány pomocí tučného řezu písma, $\mathbf{A}$.

1.1 Opakování z teorie pružnosti

Jak je vám již jistě dobře známo, okamžité napětí materiálu se obvykle uvádí v závislosti na okamžité hodnotě deformace. Zobrazení závislosti průběhu deformace na hodnotě napětí se nazývá pracovní diagram. Průběhy pracovních diagramů se různí dle vlastností zkoušeného materiálu a historie zatěžování.

Běžné pracovní diagramy bývají příliš složité pro následné výpočty, a proto se zavádí diagramy idealizované. Idealizací je obrovské množství a různí se vlastnostmi i odchylkami od skutečného stavu. Pro naše potřeby postačí tzv. ideálně pružnoplastický model.



(a) (b)

Obrázek 1: a) Zjednodušený pracovní diagram oceli, b) Ideálně pružnoplastický model

1.2 Vlastnosti ideálně pružnoplastického modelu

Ideálně pružnoplastický diagram je možno rozdělit na dvě větve: elastickou (pružnou) a plastickou.

Pro pružnou část diagramu platí Hookův zákon

$$\sigma = E\varepsilon \tag{1}$$

kde E je Youngův modul pružnosti, který je definován jako tangens úhlu sevřeného mezi pružnou větví diagramu a vodorovnou osou. Pro tuto část je tedy deformace ε přímo úměrná zvyšujícímu se napětí σ.

Oproti tomu, jak můžete vidět na obrázku 1b, při dosažení meze plasticity σ_0 se materiál prodlužuje nezávisle na napětí. Nelze tedy přesně určit, jak se materiál protáhne pouze z napětí. Pro naše účely navíc předpokládáme, že nedojde k přerušení materiálu ani při sebevětším přetvoření. Pak tedy v průřezu, který dosáhl plastického napětí (a v jeho blízkém okolí) vzniká plastický kloub, který umožňuje pootočení části, ale stále přenáší napětí.

Pružnoplastický model je matematicky zadefinován pomocí tzv. funkce plasticity,

$$f(\sigma) = |\sigma| - \sigma_0 \tag{2}$$

kde σ je okamžité napětí a σ_0 je maximální plastické napětí

Stav napětí je plasticky přípustný pokud:

$$f(\sigma) \leq 0 \tag{3}$$

Pro přetvoření jsou navíc postulovány další podmínky:

$$\varepsilon = \varepsilon_{el} + \varepsilon_{pl} \tag{4}$$

$$\dot{\varepsilon}_{pl} = \dot{\lambda}\text{sgn}(\sigma) \qquad \dot{\lambda} > 0 \tag{5}$$

$$\dot{\lambda} f(\sigma) = 0 \tag{6}$$

Tedy, že celkové přetvoření je rovno součtu elastického a plastického přetvoření. Pokud plastická deformace stoupá (s rychlostí $\dot{\lambda}$) je napětí na kladné mezi kluzu, a pokud klesá je napětí na záporné mezi kluzu. Pokud je $f(\sigma)$ nenulové (tedy pokud není dosaženo meze kluzu), je rychlost plastického přetvoření nulová. To často bývá označeno jako podmínky komplementarity.

2 Metody plastické analýzy

Cílem plastické analýzy je určit při jaké hodnotě předem daného zatížení konstrukce dosáhne svého plastického limitu, tedy stavu, kdy už není schopna dál přenášet zatížení. Je nutno podotknout, že plastického stavu může průřez konstrukce dosáhnout pouze v případě, že se jedná o průřez třídy 1, nebo 2. V ostatních případech dojde ke ztrátě stability ještě před dosažením meze kluzu.

Metody výpočtu plastické meze prostého nostíku již dobře známe z teorie pružnosti, na rámu je však problém o trochu složitější. Pro zjednodušení výpočtu tedy můžeme zanedbat vliv normálových sil na rozložení napětí na průřezu.

2.1 Přírustková metoda

Přírustková metoda je zcela nejuniverzálnějším způsobem výpočtu plastických limitů konstrukce. Zde jí však pouze nastíním, protože její náročnost stoupá se stupněm statické neurčitosti konstrukce a tak může být velmi zdlouhavá.

Zadáním je konstrukce s přesně danou geometrií prutů i zatížení, hodnoty zatížení jsou naše neznámé.

Prvním krokem je zjištění statické neurčitosti konstrukce. Stupeň statické neurčitosti nám udává, kolik plastických kloubů může vzniknout, než se konstrukce zhroutí (musíme však dávat pozor na vznik výjimkových případů podepření a mechanismů). Poté se, s využitím deformační nebo silové metody, zjistí průběh vnitřních sil na konstrukci. Najde se maximum, případně maxima, momentů a to místo (místa) se označí jako místo vzniku plastických kloubů.

Mezní plastický moment průřezu je nám již znám z materiálových a průřezových charakteristik. Proto nám už zbývá pouze vyčíslit hodnotu zatížení.

Nyní jsme určili hodnotu zatížení, při kterém vznikne první plastický kloub. Ověříme hodnotu statické neurčitosti a pokud je stále menší než nula, budeme dál určovat změnu zatížení, tedy hodnotu přírustku k prvnímu stavu.

Dál budeme postupovat stejně jako v předchozím případě, ale zatížení budeme určovat na již změněné konstrukci. Určíme přírustek zatížení do vzniku dalšího kloubu.

Tak postupujeme opakovaně, dokud nedosáhneme staticky přeurčité konstrukce. Poslední stav je tedy námy hledaná hodnota zatížení.

Zmíněná metoda je jednoduchá a vcelku bez přemýšlení nás dovede k požadovanému cíly. Za to zaplatíme pracností. Proto je na místě hledat rychlejší a efektivnější způsoby výpočtu.

2.2 Limitní analýza

V tomto odstavci vysvětlím postup výpočtu, již s využitím programu MATLAB®.

2.2.1 Definování základních principů a matematického principu úlohy

Nejprve pro zjednodušení zápisu definuji základní proměnné:

$$\boldsymbol{d} = \left[u_1; w_1; \varphi_1; u_2; w_2; \varphi_2 \dots u_i; w_i; \varphi_i\right]^T \qquad \boldsymbol{f} = \left[F_{x1}; F_{z1}; M_1; F_{x2}; F_{z2}; M_2 \dots F_{xi}; F_{zi}; M_i\right]^T \tag{7}$$

$$\boldsymbol{e} = \left[\Delta L_1; \Phi_{12}; \Phi_{21}; \Delta L_2; \Phi_{23}; \Phi_{32}; \dots \Delta L_i; \Phi_{ij}; \Phi_{ji}\right]^T \quad \boldsymbol{s} = \left[N_1; M_{12}; M_{21}; N_2; M_{23}; M_{32}; \dots N_i; M_{ij}; M_{ji}\right]^T$$

$\boldsymbol{d}$ je vektor přemístění, obsahuje přemístění jednotlivých styčníků. u_i je posun styčníku i ve vodorovném směru, w_i je posun styčníku ve svislém směru a φ_i je pootočení styčníku proti směru hodinových ručiček.

$\boldsymbol{e}$, neboli vektor deformací, obsahuje změny délek jednotlivých prutů a pootočení konců prutů. Vektor vnějších sil je $\boldsymbol{f}$ a $\boldsymbol{s}$ je vektorem vniřních sil.

Jak již víme, z vlastností ideálně pružnoplastického diagramu (viz. rovnice (2)) vyplývá, že zatěžovací stav je plasticky přípustný, pokud

$$-\sigma_0 \leq \sigma \leq \sigma_0 \tag{8}$$

Tento vztah lze zobecnit i na vektor zatížení

$$-\boldsymbol{s_0} \leq \boldsymbol{s} \leq \boldsymbol{s_0} \tag{9}$$

a pro rámy pak i na jednotlivé plastické momenty

$$-M_{0i} < M_i < M_{0i} \qquad\qquad i = 1, 2, ...n \tag{10}$$

kde M je moment jednoho určitého průřezu.

Budeme pokračovat využitím bežných geometrických a statických podmínek konstrukce.

$$\dot{\boldsymbol{e}} = \mathbf{B}\dot{\boldsymbol{d}} \tag{11}$$

$$\mathbf{B^T}\boldsymbol{s} = \boldsymbol{f} \tag{12}$$

kde $\mathbf{B}$ je geometrická matice prutu (viz níže).

Vektor $\boldsymbol{f}$ je vektor pouze referenčního zatížení a naším problémem je, kolikrát se může zvětšit, než dosáhne mezního stavu. Zvolíme tedy plastický násobitel μ_k a upravíme statickou podmínku do tvaru

$$\mathbf{B^T}\boldsymbol{s} = \mu_k\overline{\boldsymbol{f}} \tag{13}$$

Naší neznámou se tedy stává μ_k.

Práce vnějších sil je vykonávána silami na přemístěních a momenty na pootočeních. V našem případě je časová změna práce vnějších sil (tedy výkon) nenulová a lze ji zapsat jako:

$$\dot{W}_{\text{ext}} = \mu_k\overline{\boldsymbol{f}}^{\boldsymbol{T}}\dot{\boldsymbol{d}} \tag{14}$$

Výkon skutečného napětí na skutečné rychlosti deformací je nazýván plastickou disipací. Princip maxima plastické disipace nám říká, že ze všech plasticky přípustných napětí je skutečné napětí právě to napětí, které maximalizuje plastickou disipaci. Pak můžeme disipaci zapsat jako:

$$\mathcal{D} = \max \sigma^*\dot{\epsilon}_p = \sigma_0|\dot{\epsilon}_p| \tag{15}$$

Celková disipace konstrukce je pak součet disipací na jednotlivých prutech, proto můžeme pomocí geometrických charakteristik prutu zapsat celkovou disipaci konstrukce jako:

$$D_{\text{int}} = \boldsymbol{s_0^T}|\dot{\boldsymbol{e}}| \tag{16}$$

Základní předpoklad mechaniky, vyplívající ze zákona zachování energie, říká, že výkon vnějších sil na posunech a pootočeních je roven výkonu vnitřních sil na deformaci (tedy disipaci).

$$\dot{W}_{\text{ext}} = D_{\text{int}} \tag{17}$$

$$\mu_k\overline{\boldsymbol{f}}^{\boldsymbol{T}}\dot{\boldsymbol{d}} = \boldsymbol{s_0^T}|\dot{\boldsymbol{e}}| \tag{18}$$

A pokud vyjádříme neznámou μ_k, vznikne:

$$\mu_k = \frac{\boldsymbol{s_0^T}|\dot{\boldsymbol{e}}|}{\overline{\boldsymbol{f}}^{\boldsymbol{T}}\dot{\boldsymbol{d}}} \tag{19}$$

Tím vznikne stěžejní výraz celého problému.

2.2.2 Řešení problému s pomocí programu Matlab®

V konstrukci obvykle může vznikat víc stavů, kdy se práce vnějších sil rovná vnitřní disipaci. Záleží na rozložení kloubů na geometrii konstrukce. Hledáme tedy takový násobitel μ_k, pro nějž bude práce vnějších sil co nejmenší a zároveň bude splněna rovnost vnějších a vnitřních prací. Hledáme tedy nejmenší přípustné μ_k.

Pokud vynásobíme $\dot{\boldsymbol{d}}$ a $\dot{\boldsymbol{e}}$ stejným kladným číslem, dostaneme pro pozměněné hodnoty stejný násobitel μ_k, proto zavedeme podmínku:

$$\overline{\boldsymbol{f}}^{\boldsymbol{T}}\dot{\boldsymbol{d}} = 1 \tag{20}$$

a tedy dosáhneme zjednodušení:

$$\mu_k = \boldsymbol{s_0^T}|\dot{\boldsymbol{e}}| \tag{21}$$

Cílem našeho počínání je převést problém na úlohu tzv. lineárního programování. Proto musí být vstupní funkce včetně omezujících podmínek lineární. Toho dosáhneme rozdělením absolutní hodnoty na kladnou a zápornou část.

$$|\dot{\boldsymbol{e}}| = \dot{\boldsymbol{e}}^+ - \dot{\boldsymbol{e}}^- \tag{22}$$

kde $\dot{\boldsymbol{e}}^+ = (|\dot{\boldsymbol{e}}| + \dot{\boldsymbol{e}})/2 \geq 0$ a $\dot{\boldsymbol{e}}^- = (|\dot{\boldsymbol{e}}| - \dot{\boldsymbol{e}})/2 \geq 0$.

Budeme využívat výpočetního programu, proto je důležité převést rovnici do standardní formy. Ta předpokládá, že jsou všechny členy rovnice kladné, To však nesplňuje náš vektor přemístění $\dot{\boldsymbol{d}}$, jehož prvky mohou být i záporné. Proto uděláme jednoduchou úpravu.

$$\dot{\boldsymbol{d}} = \dot{\boldsymbol{d}}^+ - \dot{\boldsymbol{d}}^- \qquad \dot{\boldsymbol{d}}^+, \dot{\boldsymbol{d}}^- \geq 0 \tag{23}$$

Shrneme-li tedy všechny podmínky a dosadíme-li právě upravené neznámé dostaneme vztahy pro úlohu lineárního programování:

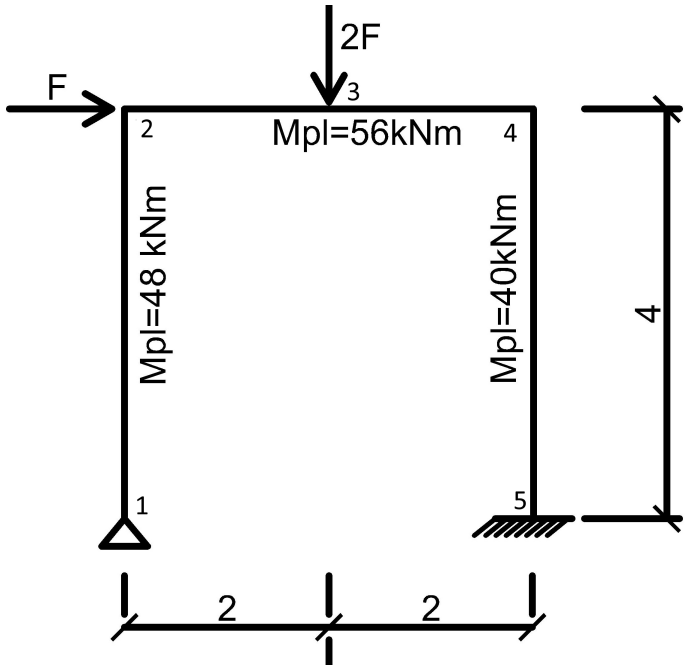
$$\text{Minimize} \quad \mu_k(\dot{\boldsymbol{e}}^+, \dot{\boldsymbol{e}}^-, \dot{\boldsymbol{d}}^+, \dot{\boldsymbol{d}}^-) = \boldsymbol{s_0^T}\dot{\boldsymbol{e}}^+ + \boldsymbol{s_0^T}\dot{\boldsymbol{e}}^- \tag{24}$$

$$\dot{\boldsymbol{e}}^+ - \dot{\boldsymbol{e}}^- - \mathbf{B}\dot{\boldsymbol{d}}^+ + \mathbf{B}\dot{\boldsymbol{d}}^- = 0 \tag{25}$$

$$\overline{\boldsymbol{f}}^{\boldsymbol{T}}\dot{\boldsymbol{d}} = 1 \tag{26}$$

$$\dot{\boldsymbol{e}}^+ \geq 0 \qquad \dot{\boldsymbol{e}}^- \geq 0 \tag{27}$$

Pro názornost bude další postup řešen na příkladu.



Obrázek 2: Příklad

Prvním krokem je sestavení geometrické matice **B**. Vyjdeme z již zmíněné závislosti deformací na přemístění.

$$\dot{\boldsymbol{e}} = \mathbf{B}\dot{\boldsymbol{d}} \tag{28}$$

Matici **B** lze rozdělit na dvě části, rovnice obsahující matici $\mathbf{B_a}$ popisuje vztah mezi osovými prodlouženími a přemístěními styčníků a rovnice s maticí $\mathbf{B_b}$ popisuje vztah mezi případnými ryzími pootočeními na koncích prutů a přemístěními styčníků.

$$\dot{\boldsymbol{e}}_{\boldsymbol{a}} = \mathbf{B_a}\dot{\boldsymbol{d}} \tag{29}$$

$$\dot{\boldsymbol{e}}_{\boldsymbol{b}} = \mathbf{B_b}\dot{\boldsymbol{d}} \tag{30}$$

Je jasné, že $\mathbf{B_a}$ bude mít počet řádků stejný jako je počet prutů a celá matice **B** bude mít tolik sloupců, kolik je na konstrukci volných přemístění. Matice $\mathbf{B_b}$ bude mít dvojnásobnobný počet řádků, než je počet prutů, neboť na každém konci každého prutu může dojít ke vzniku plastického kloubu a k následnému pootočení. Z toho vyplývá, že **B** má vždy větší počet řádků než sloupců, pokud je konstrukce staticky neurčitá.

Zvyšuje-li se x směrem zleva doprava a z směrem ze shora dolů a je-li α odchylka prutu od vodorovné osy měřená po směru hodinových ručiček. Pak pro sestavení členů **B** na jednom prutu ij platí vztah

$$\begin{array}{ccccc} \dot{\boldsymbol{e}}_{\boldsymbol{ij}} & = & & \mathbf{B} & & & \dot{\boldsymbol{d}}_{\boldsymbol{ij}} \end{array}$$

$$\begin{bmatrix} \Delta\dot{L}_{ij} \\ \dot{\Phi}_{ij} \\ \dot{\Phi}_{ji} \end{bmatrix} = \begin{bmatrix} -c & -s & 0 & c & s & 0 \\ s/l & -c/l & 1 & -s/l & c/l & 0 \\ s/l & -c/l & 0 & -s/l & c/l & 1 \end{bmatrix} \begin{bmatrix} \dot{u}_i \\ \dot{w}_i \\ \dot{\varphi}_i \\ \dot{u}_j \\ \dot{w}_j \\ \dot{\varphi}_j \end{bmatrix} \tag{31}$$

kde c =cos(α) a s =sin(α), ΔL_{ij} je osové prodloužení prutu a Φ_{ij} je pootočení těsně za styčníkem i směrem k j. (Podrobnější zadefinování matice **B** viz. [1])

V naší plastické analýze zanedbáváme vliv normálových sil a tedy můžeme říct, že $\mathbf{B_a}\dot{\boldsymbol{d}}$ se bude rovnat nulové matici.Zároveň nebereme v úvahu žádná plastická protažení prutů, tedy bereme pruty jako neprotažitelné. To znamená, že všechny členy ΔL_{ij}, tedy $\dot{\boldsymbol{e}}_{\boldsymbol{a}}$,se rovnají nule a můžeme je tedy vynechat a brát v potaz pouze vliv pootočení. Pak rovnice (29) a (30) dostávají tvar:

$$\mathbf{O}\dot{\boldsymbol{e}}_{\boldsymbol{b}} = \mathbf{B_a}\dot{\boldsymbol{d}} \tag{32}$$

$$\mathbf{I}\dot{\boldsymbol{e}}_{\boldsymbol{b}} = \mathbf{B_b}\dot{\boldsymbol{d}} \tag{33}$$

kde $\mathbf{O}$ je nulová matice a $\mathbf{I}$ jednotková diagonální čtvercová matice

Pro náš příklad tedy platí:

$$\begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix} \begin{bmatrix} \Phi_{12}\\ \Phi_{21}\\ \Phi_{23}\\ \Phi_{32}\\ \Phi_{34}\\ \Phi_{43}\\ \Phi_{45}\\ \Phi_{54} \end{bmatrix} = \begin{bmatrix} 0&0&-0.25&0&0&0&0&0&0&0\\ 0&-0.5&0&0&0.5&0&0&0&0&0\\ 0&0&0&0&-0.5&0&0&0.5&0&0\\ 0&0&0&0&0&0&0&0&-0.25&0\\ 1&0.25&0&0&0&0&0&0&0&0\\ 0&0.25&0&1&0&0&0&0&0&0\\ 0&0&-0.5&1&0&0.5&0&0&0&0\\ 0&0&-0.5&0&0&0.5&1&0&0&0\\ 0&0&0&0&0&-0.5&1&0&0.5&0\\ 0&0&0&0&0&-0.5&0&0&0.5&1\\ 0&0&0&0&0&0&0&0.25&0&1\\ 0&0&0&0&0&0&0&0.25&0&0 \end{bmatrix} \begin{bmatrix} \varphi_1\\ u_2\\ w_2\\ \varphi_2\\ u_3\\ w_3\\ \varphi_3\\ u_4\\ w_4\\ \varphi_4 \end{bmatrix} \tag{34}$$

Dalším krokem je vyjádření změny vektoru přemístění $\boldsymbol{d}$,

$$\mathbf{P^T}\dot{\boldsymbol{e}}_{\boldsymbol{b}} = \dot{\boldsymbol{d}} \tag{35}$$

$$\mathbf{S^T}\dot{\boldsymbol{e}}_{\boldsymbol{b}} = \mathbf{0} \tag{36}$$

Toho lze dosáhnout pomocí Gaussovy eliminace, nebo pomocí rozkladu SVD

Singulární rozklad (SVD-Singular Value Decomposition) je algoritmus užívaný k rozkladu matice $\mathbf{B}$ na součin 3 matic $\mathbf{U}$,$\mathbf{S}$ a $\mathbf{V}$. Má-li $\mathbf{B}$ m řádků a n sloupců, je matice $\mathbf{U}$ velikosti $m \times m$ a zároveň je unitární. To znamená, že se jedná o komplexní(nebo reálnou) matici, jejíž komplexně sdružená a transponovaná matice je zároveň maticí inverzní. $\mathbf{V}$ je komplexní nebo reálná unitární matice $n \times n$ a $\mathbf{S}$ je diagonální obdelníková matice rozměru $m \times n$, na jejíž hlavní diagonále jsou tzv. singulární hodnoty matice $\mathbf{B}$. Dále platí $\mathbf{UU^T} = \mathbf{I}$ a $\mathbf{VV^T} = \mathbf{I}$. Výpočetní náročnost úkonu sice roste s třetí mocninou rozměrů matic, ale s využitím MATLAB®u se tímto problémem zabývat nemusíme.

$$\mathbf{B} = \mathbf{USV^T} \tag{37}$$

sloučíme-li rovnice (32) a (33) a použijeme-li rozklad SVD dostáváme vztah:

$$\mathbf{J}\dot{\boldsymbol{e}} = \mathbf{USV^T}\dot{\boldsymbol{d}} \tag{38}$$

kde $\mathbf{J}$ je matice, jejíž horní část je tvořena maticí $\mathbf{O}$ a spodní část maticí $\mathbf{I}$

$$\mathbf{U^T J}\dot{\boldsymbol{e}} = \mathbf{SV^T}\dot{\boldsymbol{d}} \tag{39}$$

V tuto chvíli je vhodné se opět podívat na vlastnosti vzniklých matic. Pro námi zvolený příklad se $\mathbf{SV^T}$ rovná:

$$\begin{bmatrix}
-0.015 & -0.102 & 0.406 & -0.923 & 0.000 & -0.792 & -0.000 & 0.100 & 0.405 & 0.922 \\
-0.014 & -0.078 & 0.464 & -0.604 & 0.020 & 0.001 & -1.115 & -0.077 & -0.464 & -0.605 \\
-0.032 & -0.131 & -0.023 & -0.791 & 0.042 & 0.001 & 0.858 & -0.126 & 0.022 & -0.792 \\
0.944 & 0.385 & 0.083 & 0.005 & -0.171 & -0.118 & 0.023 & 0.026 & 0.042 & -0.096 \\
-0.155 & 0.064 & 0.304 & 0.386 & 0.098 & -0.581 & -0.008 & -0.219 & 0.321 & -0.359 \\
0.264 & -0.269 & -0.075 & -0.000 & 0.662 & 0.061 & -0.033 & -0.381 & 0.008 & 0.097 \\
0.111 & -0.319 & 0.051 & 0.078 & 0.066 & -0.057 & 0.008 & 0.378 & 0.023 & -0.078 \\
0.026 & -0.087 & -0.248 & -0.055 & -0.100 & -0.003 & -0.157 & -0.045 & 0.252 & -0.063 \\
-0.000 & 0.000 & -0.115 & 0.002 & 0.002 & -0.121 & -0.000 & 0.003 & -0.114 & -0.002 \\
-0.022 & 0.086 & -0.030 & -0.019 & 0.081 & 0.002 & -0.015 & 0.068 & 0.031 & -0.016 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{40}$$

Všimněte si, že součin $\mathbf{SV^T}$ tvoří matici, kterou lze rozdelit na čtvercovou maticí $\mathbf{SV_c}$ a několik nulových řádků $\mathbf{SV_0}$. Proto můžeme rovnici (38) opět rozdělit na dvě a určit $\mathbf{P^T}$ a $\mathbf{S^T}$.

Nechť $\mathbf{U^T J_c}$ má stejnou hodnost jako $\mathbf{SV_c}$ a je tvořená prvními několika řádky součinu $\mathbf{U^T J}$ a $\mathbf{U^T J_0}$ jsou všechny zbylé řádky součinu $\mathbf{U^T J}$, potom:

$$\mathbf{P^T} = (\mathbf{SV_c})^{-1}\mathbf{U^T J_c} \tag{41}$$

$$\mathbf{S^T} = \mathbf{U^T J_0} \tag{42}$$

S využitím programu MATLAB® by to mohlo vypadat nějak takto:

Skript 1: Vstupní soubor pro MATLAB® využití SVD

```
% nstruts is number of struts and ncsecrtions is number of critical sections what is 2*nstruts
% B=USV'
J = [zeros(nstruts,ncsections); eye(ncsections)];
[U,S,V]=svd(B);
C=U'*J;
m=size(S*V',2);
PT=V*(S(1:m,1:m)\ C(1:m,:));
ST=C(m+1:end,:);
```

Ale zpět k rovnicím (35) a (36). Vyjdeme z podmínky $\boldsymbol{f^T \dot{d}} = 1$ a odstraníme $\dot{\boldsymbol{d}}$

$$\mathbf{p} = \mathbf{P}\boldsymbol{f} \qquad \Longrightarrow \qquad \mathbf{p^T}\dot{\boldsymbol{e}} = 1 \tag{43}$$

pak už stačí rovnice jen převést do standardní formy nastíněné v rovnici (25) a výsledkem jsou rovnice připravené pro řešení simplexovým algoritmem.

$$\begin{bmatrix} \mathbf{S^T} & -\mathbf{S^T} \\ \mathbf{p^T} & -\mathbf{p^T} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{e}}^+ \\ \dot{\boldsymbol{e}}^- \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \dot{\boldsymbol{e}}^+, \dot{\boldsymbol{e}}^- \geq 0 \tag{44}$$

Simplexový algoritmus je metoda lineárního programování. Úkolem lineárního programování je nalézt extrém lineární funkce za předem daných lineárních omezujících podmínek. Nejdřív získáme výchozí přípustné řešení úlohy tak, že všechny přídatné omezující proměné položíme rovné nule. Ostatní nenulové proměné jsou našimi bazickými proměnými. Úloha má optimální maximum (minimum) pokud dosáhneme stavu, kdy jsou všechny redukované proměné kladné (záporné).

Pro naše účely postačí opět si vypomoct MATLAB®em.

Skript 2: Vstupní soubor pro MATLAB® použití příkazu linprog

```
PTST=[pT,-pT;ST,-ST];
N=[1;zeros(size(ST,1),1)];
[x,mi] = linprog(Mpl,[],[],PTST,N,zeros(ncsections*2,1),[])
```

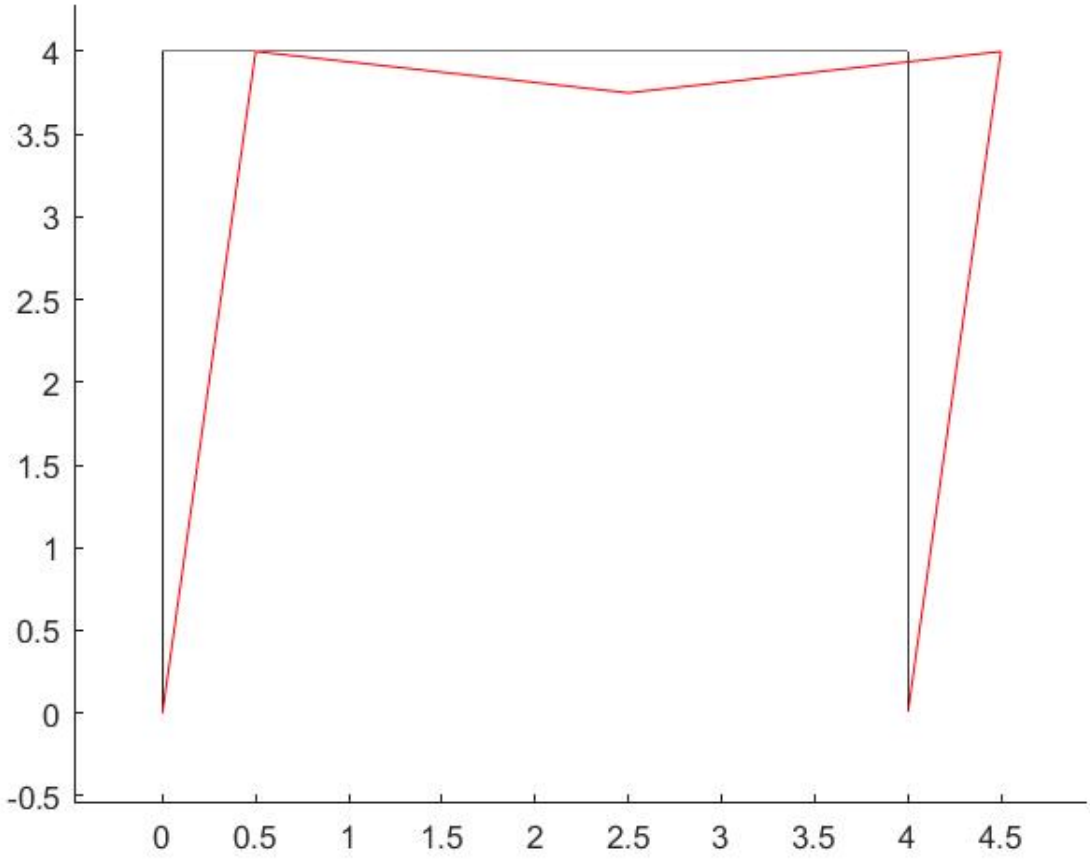
kde M_{pl} je vektor mezních plastických momentů konajících práci na pootočení plastických kloubů, a lb je spodní omezení úlohy. $lb > 0$

Výstupem je

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.2500 \\ 0 \\ 0 \\ 0 \\ 0.1250 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.2500 \\ 0 \\ 0 \end{bmatrix} \qquad mi = 29.0000 \tag{45}$$

kde $x = \dot{\boldsymbol{e}}^+ - \dot{\boldsymbol{e}}^- = \dot{\boldsymbol{e}}$. Naším hledáným násobitelem μ_k je tedy 29. Hodnota mezního zatížení před plastickým kolapsem konstrukce je tedy 29 kN.

Zjištěných parametrů $\mathbf{P^T}$ a $\dot{\boldsymbol{e}}$ můžeme využít k následnému vyjádření deformací a k vykreslení deformovaného tvaru.



Obrázek 3: Vykreslení deformovaného tvaru konstrukce

3 Příloha-Kompletní Matlab® script

Skript 3: Vstupní soubor pro MATLAB®- kompletní skript

```
% frame
% construction
%-----------------------------------------------------------------------------
% nodes(1)=struct('coords',[0;0],'fixed',[1,1,1],'load',[0,0,0]);
% nodes(2)=struct('coords',[3;0],'fixed',[0,0,0],'load',[0,1,0]);
% nodes(3)=struct('coords',[6;0],'fixed',[1,1,0],'load',[0,0,0]);
%
% struts(1)=struct('nodesIds', [1;2], 'Mpl',51.935);
% struts(2)=struct('nodesIds', [2;3], 'Mpl',51.935);
%-----------------------------------------------------------------------------
nodes(1)=struct('coords',[0;0],'fixed',[1,1,0],'load',[0,0,0]);
nodes(2)=struct('coords',[0;4],'fixed',[0,0,0],'load',[1,0,0]);
nodes(3)=struct('coords',[2;4],'fixed',[0,0,0],'load',[0,2,0]);
nodes(4)=struct('coords',[4;4],'fixed',[0,0,0],'load',[0,0,0]);
nodes(5)=struct('coords',[4;0],'fixed',[1,1,1],'load',[0,0,0]);


struts(1)=struct('nodesIds', [1;2], 'Mpl',48);
struts(2)=struct('nodesIds', [2;3], 'Mpl',56);
struts(3)=struct('nodesIds', [3;4], 'Mpl',56);
struts(4)=struct('nodesIds', [4;5], 'Mpl',40);
% -----------------------------------------------------------------------------
% nodes(1)=struct('coords',[0;0],'fixed',[1,1,1],'load',[0,0,0]);
% nodes(2)=struct('coords',[0;1],'fixed',[0,0,0],'load',[0,0,-0.75]);
% nodes(3)=struct('coords',[1;1.75],'fixed',[0,0,0],'load',[0,4,0]);
% nodes(4)=struct('coords',[2;1.75],'fixed',[0,1,1],'load',[0,0,0]);
%
%
% struts(1)=struct('nodesIds', [1;2], 'Mpl',1);
% struts(2)=struct('nodesIds', [2;3], 'Mpl',1);
% struts(3)=struct('nodesIds', [3;4], 'Mpl',1);




nnodes=length(nodes);
nstruts=length(struts);

hold on

L=zeros(1,nstruts);
for n=1:nstruts
   Z= struts(n).nodesIds;
   %sketch frame
    plot([nodes(Z(1)).coords(1),nodes(Z(2)).coords(1)],...
    [nodes(Z(1)).coords(2),nodes(Z(2)).coords(2)],'k');
   %give lenght of struts
   L(n)=((nodes(Z(1)).coords(1)-nodes(Z(2)).coords(1))^2+(nodes(Z(1)).coords(2)-nodes(Z(2)).
     coords(2))^2)^(1/2);
end
L=L';

%number of statical indeterminacy
s=0;
nunknows=0;
nsteps=0;
s3=0;
for s1=1:nnodes
    for s2=1:3
      s=s+ nodes(s1).fixed(s2);
      nsteps=nsteps+1;
      %solve numbers of unknows
        if nodes(s1).fixed(s2)==0
            nunknows=nunknows+1;
            %typ of unknows
            typunknows(nunknows)=s2;
            unknows(nunknows)=nsteps;
        end
    end
end
s=3-s;
```

```
%solve number of critical sections
ncsections=2*nstruts;
csection=zeros(ncsections,2);
ncsections=0;
for n=1:nstruts
    Z= struts(n).nodesIds;
    for i=1:2
        ncsections= ncsections+1;
            if i==1
                csection(ncsections,1)=[Z(1)];
                csection(ncsections,2)=[Z(2)];
            end
            if i==2
                csection(ncsections,1)=[Z(2)];
                csection(ncsections,2)=[Z(1)];
            end
    end
end

% Do matrix B
Ba=zeros(nstruts,nsteps);
for n=1:nstruts
     Z= struts(n).nodesIds;
     n1=Z(1)*3;
     n2=n1-2;
     n3=Z(2)*3;
     n4=n3-2;
     sin=(nodes(Z(2)).coords(2)-nodes(Z(1)).coords(2))/L(n);
     cos=-(nodes(Z(2)).coords(1)-nodes(Z(1)).coords(1))/L(n);
     Ba(n,n2:n1)=[cos/L(n),sin/L(n),0];
     Ba(n,n4:n3)=[-cos/L(n),-sin/L(n),0];
end
Ba=Ba(:,unknows);

Bb=zeros(ncsections,nsteps);
for nc=1:ncsections
    csectionA=csection(nc,1);
    csectionB=csection(nc,2);
    fiA=0;
    if nodes(csectionA).fixed(3)==0
        fiA=1;
    end

    Ll=abs(((nodes(csectionA).coords(1)-nodes(csectionB).coords(1))^2+(nodes(csectionA).coords
    (2)-nodes(csectionB).coords(2))^2)^(1/2);

    sin=(-nodes(csectionA).coords(2)+nodes(csectionB).coords(2))/Ll;
    cos=-(-nodes(csectionA).coords(1)+nodes(csectionB).coords(1))/Ll;
    psi1=sin/Ll;
    psi2=-cos/Ll;

    n1=csectionA*3;
    n2=n1-2;
    n3=csectionB*3;
    n4=n3-2;
    Bb(nc,n2:n1)=[-psi1,-psi2,fiA];
    Bb(nc,n4:n3)=[psi1,psi2,0];

end
Bb=Bb(:,unknows);
B=[Ba;Bb];

%Do load matrix FT
F=zeros(1,nsteps);
nsteps=0;
for s1=1:nnodes
    for s2=1:3
        nsteps=nsteps+1;
       FT(1,nsteps)=nodes(s1).load(s2);
    end
end
FT=FT(1,unknows);

Mpl = [[struts.Mpl], [struts.Mpl], [struts.Mpl], [struts.Mpl]];

% B=U*S*V'

J = [zeros(nstruts,ncsections); eye(ncsections)];
[U,S,V]=svd(B);
C=U'*J;


```

```
m=size(S*V',2);
PT=V*(S(1:m,1:m)\ C(1:m,:));
ST=C(m+1:end,:);
pT=FT*PT;

PTST=[pT,-pT;ST,-ST];
N=[1;zeros(size(ST,1),1)];
[x,mu] = linprog(Mpl,[],[],PTST,N,zeros(ncsections*2,1),[]);
fprintf('Min. plastic multiplier \\mu_{K} = %f\n', mu);

%sketsch deformed shape of frame
trueX = x(1:2*nstruts)-x(2*nstruts+(1:2*nstruts));
trueD=zeros(nnodes*3,1);
trueD(unknows)=PT*trueX;

for n=1:nstruts
     Z= struts(n).nodesIds;
     za=Z(1);
     zb=Z(2);
     plot([nodes(Z(1)).coords(1)+trueD(za*3-2),nodes(Z(2)).coords(1)+trueD(zb*3-2)],...
     [nodes(Z(1)).coords(2)-trueD(za*3-1),nodes(Z(2)).coords(2)-trueD(zb*3-1)],'r')
end
```

Pozn.: Program řeší zadání s bodovým zatížením na konstrukci s nezakřivenými pruty. Zadání je třeba zadat tak aby souřadnice x vzrůstaly směrem v pravo a souřadnice z směrem vzhůru. Skript by jistě snesl drobnou optimalizaci.

4 Závěr

Tato úloha je dobrá pro ukázání využití základních optimalizačních metod, jako je simplexový algoritmus. Je vhodná pro učení se základů programování pomocí MATLAB®u. Rozšířila mé znalosti o mezním chování konstrukcí, a zopakovala znalosti z kurzu Pevnost pružnost. Jako bonus jsem se naučil s LaTeXem.

Tímto děkuji za rady svému cvičícímu Ing. Martinu Doškářovi

Reference

[1] Milan Jirásek and Z. P. Bažant. *Inelastic analysis of structures.* Wiley, Chichester, West Sussex, England, 2002.