MASARYKOVA UNIVERZITA PŘÍRODOVĚDECKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE:

POHYB DISLOKACÍ V TŘÍROZMĚRNÉM KRYSTALU

BRNO, KVĚTEN 2006 TOMÁŠ ZÁLEŽÁK

Prohlašuji, že jsem tuto práci sepsal samostatně a využíval jsem pouze literaturu uvedenou na straně 31.

Tomáš Záležák

V Brně dne 24. května 2006

Poděkování

Chtěl bych poděkovat RNDr. Antonínu Dlouhému, CSc., který déle než rok vedl moji práci, poskytl mi mnoho rad a konzultací, též potřebnou literaturu a řadu doporučení ke zpřesnění textu.

Tomáš Záležák

Anotace

Tato práce se zabývá studiem pohybu křivočaré dislokace v prostoru. Dislokační čára je přibližně vyjádřena pomocí lineárních segmentů. V mezi platnosti lineární teorie elasticity jsou odvozeny vztahy pro silová působení na jednotlivé segmenty pocházející od smyčky samotné, tak i od vnějšího napětí. Součástí práce je i ukázka numerického řešení kontrakce dislokační smyčky. Program je napsán v jazyce C++. Obecný postup řešení však lze uplatnit i pro složitější konfigurace dislokačních čar.

Annotation

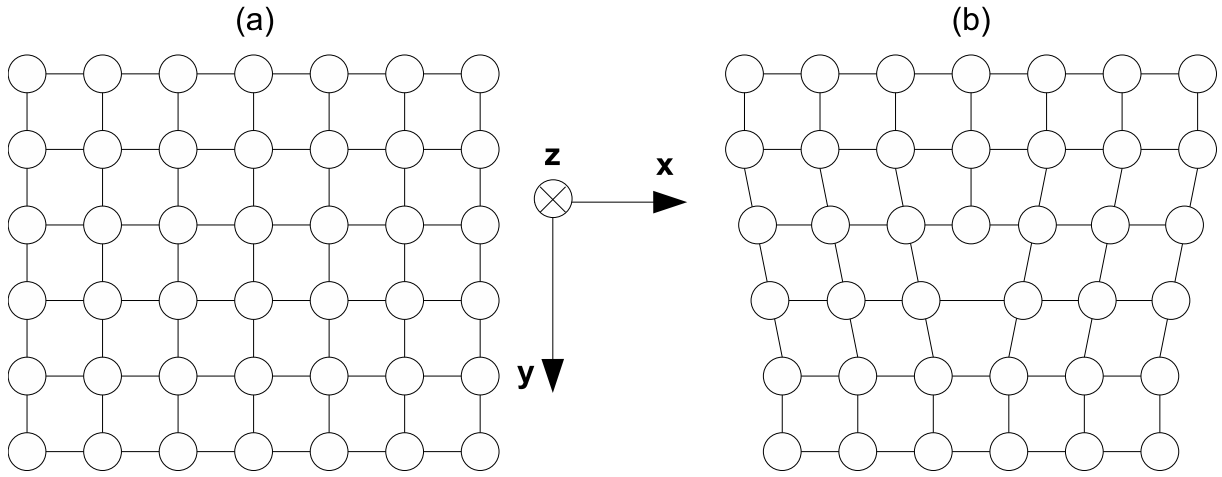
This work describes a motion of curved dislocation in 3-dimensional space. The dislocation line is approximated by straight line segments. Formulas for forces acting on line segments and coming from both, self- and external-stresses have been obtained in terms of the linear elasticity theory. The general solution is implemented for the case study, in which self-stresses cause the shrinkage of a dislocation loop. The numerical solution in C++ language is also included. However, the general solution can be applied to more complex dislocation configurations.

OBSAH

1. ÚVOD

1.1 Plastická deformace a dislokace

Jsou-li reálná pevná tělesa podrobena účinku vnějších sil, dochází ke změně jejich tvaru. Trvá-li změna tvaru i po ukončení silového působení (po neomezeně dlouhou dobu) nazýváme tuto změnu plastickou deformací [1]. Plastická deformace má své příčiny v nevratných dějích probíhajících ve struktuře látek. V případě látek krystalických bývá plastická deformace způsobena pohybem čarových poruch krystalické mříže zvaných dislokace [1]. Na obrázku 1.1 je znázorněna část jednoduché mřížky s kubickou symetrií pro případ ideálního neporušeného krystalu (obr. 1.1a) a pro případ krystalu obsahujícího dislokaci (obr. 1.1b). Lze si představit, že uspořádání atomů v porušeném krystalu na obr. 1.1b je podobné i v dalších rovinných řezech mřížkou ($z = \mathrm{konst.}$), které se nacházejí nad a pod řezem nakresleným na obrázku. Proto můžeme dislokaci jako poruchu pravidelného uspořádání krystalické mřížky charakterizovat rovněž pomocí jedné vložené mřížkové poloroviny (yz) obsahující atomy. Samotnou dislokaci ztotožňujeme s čarou charakterizující hranu uvedené poloroviny v krystalu. V tomto smyslu je dislokační čára na obr. 1.1b orientována ve směru osy z kolmo na rovinu nákresny.



Obrázek 1.1: Řez neporušenou prostorovou kubickou mřížkou (a) a porušenou mřížkou s hranovou dislokací (b).

Uspořádání atomů v okolí přímé dislokační čáry (jádra dislokace) je relativně stabilní a bez vnějšího působení nedochází k jeho změně. Tuto skutečnost lze vyjádřit i tak, že přímočarý segment dislokace se nemůže pohybovat v důsledku napětí, které sám v krystalu vyvolává [2]. K pohybu dislokace dochází zejména díky působení mechanických napětí, jejichž zdroje leží mimo jádro dislokace. Tato napětí vychylují atomy v oblasti okolo jádra z rovnovážných poloh a vedou buďto k posunu nadbytečné poloroviny ve směru osy x (*konzervativní pohyb skluzem*) nebo k posunu dislokační čáry ve směru osy y (*nekonzervativní pohyb šplháním* za asistence *difuze*). Zejména při plastické deformaci za zvýšených a vysokých teplot je pohyb dislokací v krystalu kombinovaný, sestávající jak z posunutí díky skluzu, tak i z posunutí díky šplhání dislokační linie.

Vzhledem k tomu, že vnější napěťové pole působící na přímou dislokaci může být obecně funkcí prostorových souřadnic a může se poměrně výrazně měnit s polohou podél dislokační čáry, lze si poměrně snadno představit, že se jednotlivé části původně přímé dislokace mohou v krystalu pohybovat různými směry a různými rychlostmi. Důsledkem nestejnoměrného pohybu nebude rozpad přímočaré dislokace na části (dislokace nemůže být ukončena uvnitř jinak neporušené krystalové mřížky [1] [2]), ale prohnutí přímočarého segmentu a jeho přeformování na dislokační linii složitějšího tvaru charakterizovanou nenulovými křivostmi. V důsledku tohoto procesu může délka dislokačních linií v krystalu, v němž působí vnější napětí, narůstat. Při pohybu jednotlivých segmentů dislokační linie a při odpovídající změně tvaru linie dochází ke vzájemnému posunu částí krystalu. Konzervativní pohyb dislokační linie (ve směru osy x v situaci na obr. 1.1b) vede k posunu horní části krystalu obsahující nadbytečnou mřížkovou polorovinu ve směru osy x vůči dolní části, a to o vzdálenost odpovídající jisté vzdálenosti v krystalické mřížce. Nekonzervativní pohyb dislokační linie (ve směru y v situaci na obr. 1.1b) má za následek podobné posunutí ve směru osy x, nyní ovšem části krystalu napravo od dislokační linie vůči levé části. Z obr. 1.1b je zřejmé, že v prosté kubické mřížce tato posunutí odpovídají právě mřížkovému parametru. Obecně je vzájemné posunutí částí krystalu při pohybu dislokace popsáno pomocí Burgersova vektoru, jehož směr a velikost určují směr a velikost vzájemného posunutí. Definice a způsob určení *Burgersova vektoru* jsou podrobněji popsány v části 2.1 této práce.

Vzájemné posunutí částí krystalu způsobené pohybem dislokací je základem plastické deformace krystalických látek. Vzhledem k tomu, že posunutí spojené s pohybem jedné dislokace je poměrně malé, řádu $2 \cdot 10^{-10}$ m, je pro makroskopicky pozorovatelnou plastickou deformaci nutné, aby došlo k pohybu značného počtu dislokačních linií. Měříme-li množství dislokačních linií v krystalu pomocí dislokační hustoty, tj. pomocí délky dislokačních linií v jednotce objemu, lze odhadnout, že pro makroskopickou deformaci tělesa řádu 10% (např. pro 10% nevratné prodloužení délky drátu) je nutné přemístění asi $10^{13}\mathrm{m} \cdot \mathrm{m}^{-3}$ dislokačních linií na střední vzdálenost asi 10^{-4} m. Uvedená délka dislokačních linií v $1\ \mathrm{m}^3$ odpovídá zhruba stonásobku vzdálenosti mezi Zemí a Sluncem. Jak již bylo zmíněno, plastická deformace je dějem nevratným. Nevratnost plastické deformace je dána elementárními ději přeskoku atomů mezi rovnovážnými polohami v oblasti jádra dislokace při pohybu dislokační linie. Přeskok každého atomu je spojen s překonáním určité energetické bariéry a energie dodaná např. prostřednictvím vnějšího napěťového pole je po přechodu atomu do nové rovnovážné polohy disipována ve formě kmitů mřížky (tepelných nebo akustických fononů).

Zdrojem napěťového pole pro pohyb dislokace nemusí být jen síly přiložené na deformující se těleso zvenku (*aplikované napětí*). Napěťové pole v oblasti jádra dislokace způsobující pohyb dislokační linie může souviset i s přítomností dalších defektů uvnitř krystalu, např. s jinými dislokacemi, koherentními precipitáty nebo hranicemi zrn v polykrystalických látkách. Napětí pocházející od těchto zdrojů označujeme jako *vnitřní napětí* [1] [2]. Pro výsledný pohyb dislokace je rozhodující celkové napětí působící v oblasti jádra, které je obecně součtem napětí aplikovaného a vnitřního. V tomto smyslu mohou k pohybu daného (přímočarého) segmentu dislokace přispívat i jiné segmenty dislokační linie, které v oblasti jádra daného segmentu přispívají k poli vnitřních napětí. Z uvedeného vyplývá, že matematický popis procesu plastické deformace krystalických látek na úrovni jednotlivých dislokací není jednoduchý. Při tvorbě modelů je třeba přijmout řadu zjednodušujících předpokladů, např. omezit se pouze na popis dějů ve dvou

dimenzích a pro konkrétní řešení využít numerických metod [4].

Na obrázku 1.2 je pro ilustraci reálné konfigurace dislokací prezentován snímek z transmisního elektronového mikroskopu (TEM), který zachycuje stav dislokační struktury v intermetalické slitině $\mathrm{Ti-46Al-2W-0{,}5Si-0{,}6B}$ po vysokoteplotní deformaci v tahu při teplotě $760^{\circ}\mathrm{C}$ a konstantním vnějším napětí $138\ \mathrm{MPa}$. Celkové deformace 18% bylo dosaženo za 18000 hodin a deformace proto byla velmi pomalá (rychlost deformace odpovídá $6 \cdot 10^{-10}\ \mathrm{s}^{-1}$). Nicméně, jak plyne z TEM snímku, i při této pomalé deformaci dochází ke zvyšování dislokační hustoty, k pohybu dislokačních linií a jejich interakci s precipitáty sekundárních fází. Místa interakce jsou na snímku označena šipkami. Dislokační linie přítomné ve střední části snímku mají velmi výrazné zakřivení svědčící o značné proměnlivosti napěťového pole v mikrostruktuře slitiny. Podobnost tvaru jednotlivých dislokací, např. dislokací označených jako A a B, rovněž dává tušit, že mezi jednotlivými dislokacemi existují významné interakce zprostředkované vnitřním napěťovým polem spojeným s dislokačními segmenty. Tento a další experimentální výsledky jasně ukazují, že pohyb dislokací má obecně prostorový charakter a pro jeho korektní modelový popis je nutné vzít v úvahu flexibilitu dislokační linie a vzájemné interakce mezi pohybujícím se segmentem a ostatními dislokacemi přítomnými v krystalu. Jak vyplývá z TEM snímku na obr. 1.2, pohyb dislokačních segmentů může být rovněž významně ovlivněn precipitací sekundárních fází.

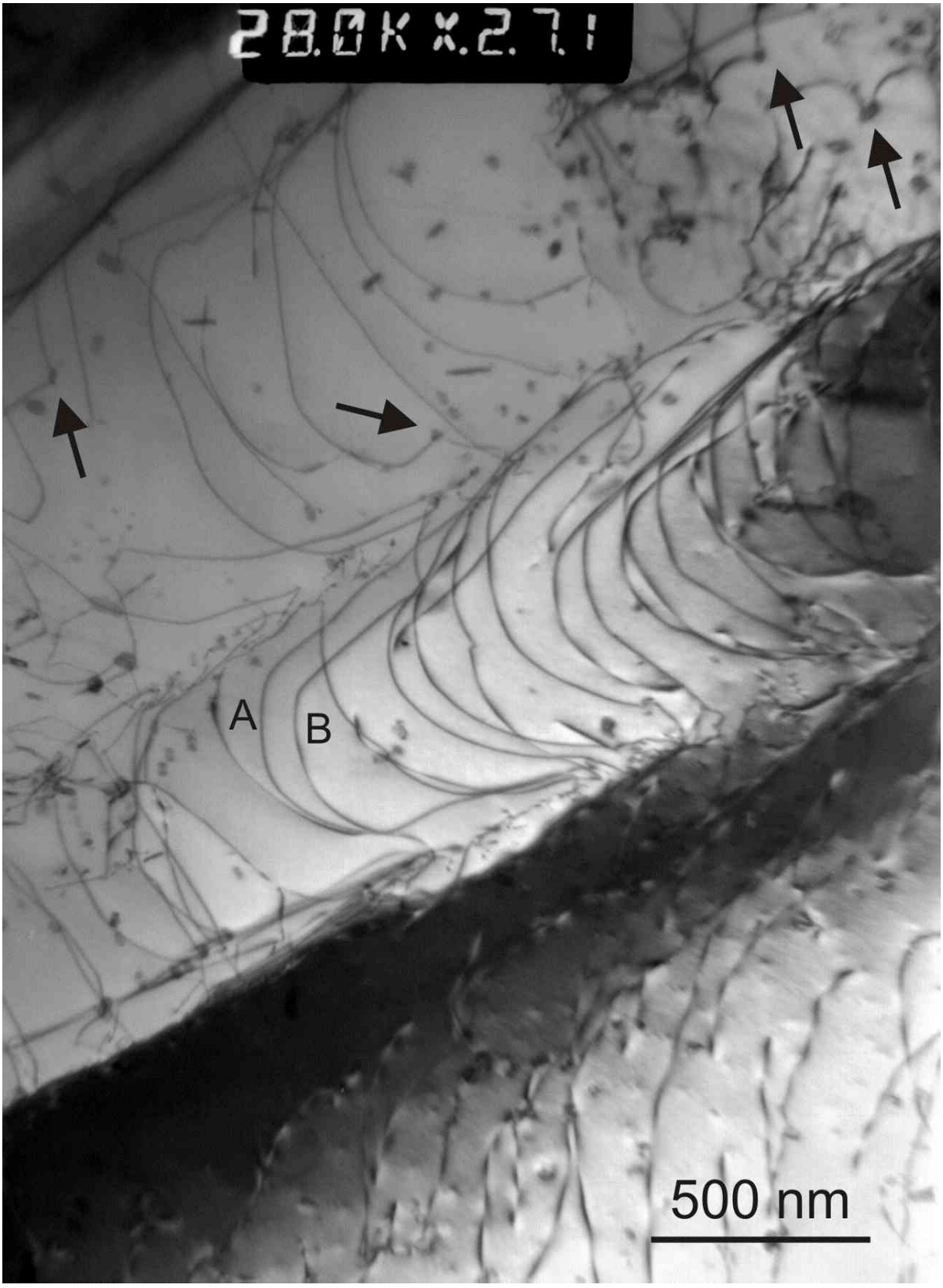
1.2 Pohyb dislokací v třírozměrném prostoru

Dislokace se v krystalu pohybují díky silovému působení, jež pochází od napěťového pole. Toto napěťové pole mohou vytvářet jak dislokační linie uvnitř materiálu, tak vnější síly působící na těleso. Vyjádření napěťového pole, které v krystalu indukuje dislokační segment, závisí na tvaru segmentu. Pro rovné dislokace jsou vztahy poměrně jednoduché, pro obecně zakřivené dislokace, na které je zaměřena tato práce, je určení napěťového pole dosti náročné. Výpočet lze výrazně zjednodušit, pokud je postaven na vztazích pro rovné dislokační segmenty, do nichž lze jakoukoli obecnou křivku rozdělit (např. jako na obr. 1.3). Každý takový segment kolem sebe vytváří vlastní napěťové pole, zároveň však pociťuje okolní napěťové pole od okolních segmentů a případně vnějších sil.

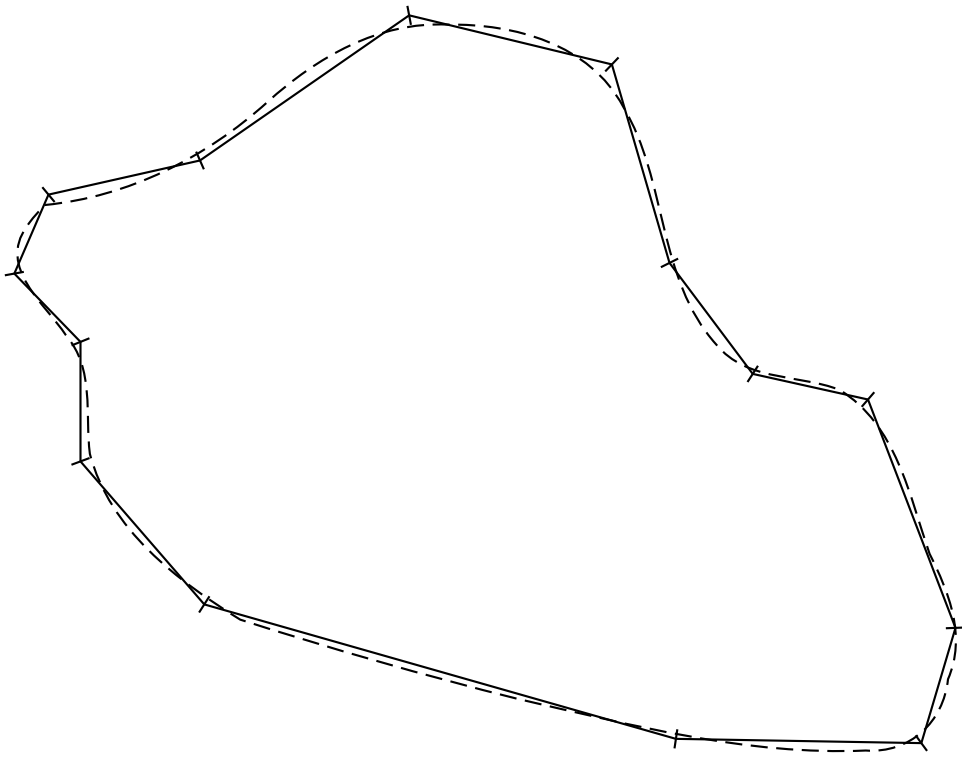
Vzorce pro tenzor popisující napěťové pole dislokačního segmentu (úsečky) jsou odvozeny pro speciální případ, kdy úsečka leží na ose z kartézského souřadného systému. To u spojité dislokační čáry převedené na čáru lomenou zřejmě není možné splnit pro všechny úsečky. Proto je třeba lineárně zobrazit vstupní parametry (Burgersův vektor a polohový vektor místa, v němž je pole vyčíslováno) ze souřadné soustavy, v níž je popsána lomená čára, do takové souřadné soustavy, v níž úsečka leží na ose z. K vytvoření takového lineárního zobrazení stačí dva jednotkové vektory – směrový vektor dislokačního segmentu a vektor určující směr osy z. Doplněním obou vektorů na ortonormální báze lze získat potřebné transformační matice. Po vyčíslení napěťového pole je třeba jej ze souřadné soustavy segmentu zobrazit zpět do souřadné soustavy lomené čáry.

Protože je zde použita lineární teorie elasticity založená na malých deformacích, je možné příspěvky k napěťovému poli od jednotlivých segmentů a vnějších sil sčítat, získat celkové napěťové pole a použít jej k výpočtu síly působící na segment.

Po vypočítání sil působících na jednotlivé segmenty je možné s pomocí pohybové rovnice zjistit, jak se budou segmenty posouvat. Po jejich posunutí je možné znovu najít



Obrázek 1.2: TEM snímek dislokační struktury ve slitině $Ti-46Al-2W-0,5Si-0,6B$ po creepu při teplotě $760°C$ a konstantním vnějším napětí 138 MPa. Místa interakcí dislokačních linií s precipitáty jsou označena šipkami. Dislokační linie A a B mají velmi podobný tvar.



Obrázek 1.3: Dislokační smyčka a její nahrazení lomenou čarou.

matice přechodu, vypočítat napěťová pole, určit síly, z nich opět posunutí a sledovat tak pohyb dislokační čáry obecného tvaru.

1.3 Cíl práce

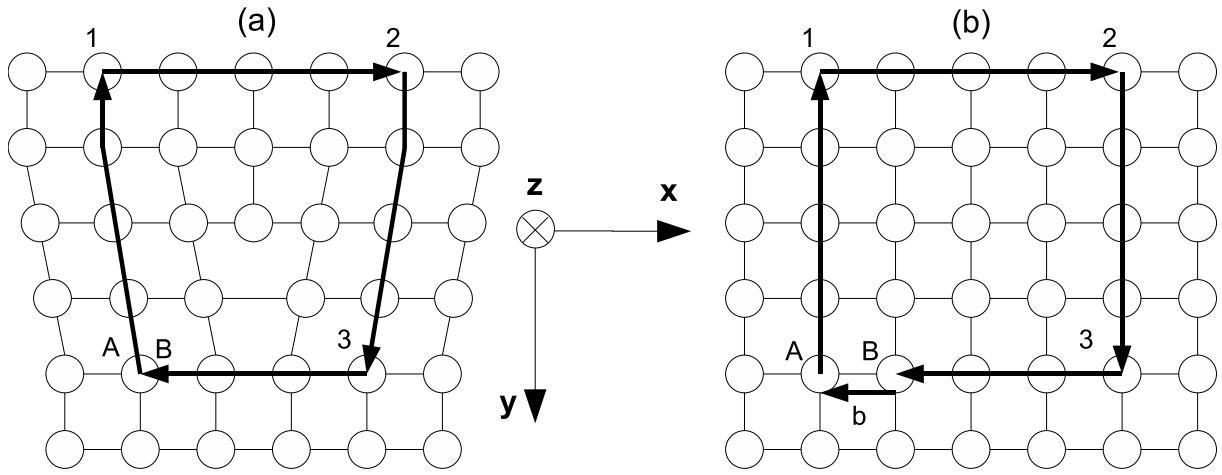
V předchozím textu bylo podrobněji charakterizováno téma této práce. Jejím cílem je vypracovat postup umožňující popsat napěťové pole dislokační čáry obecného tvaru pomocí aproximace metodou přímkových dislokačních segmentů. Problém bude řešen numericky na počítači. K napsání vlastního programu je třeba nejprve zpracovat potřebné pasáže teorie dislokací týkající se rovných dislokací v prostoru, jejich napěťových polí a sil na ně působících. Dále je cílem navrhnout zpracování zadané křivky do podoby lomené čáry a prozkoumat její další vývoj. Výsledný aparát bude konfrontován s jednoduchými případy dislokačních linií, jako je např. samovolně kontrahující dislokační smyčka bez vnějšího silového působení.

2. ZÁKLADY TEORIE DISLOKACÍ

2.1 Burgersův vektor

Velmi důležitým pojmem v teorii dislokací je *Burgersův vektor* $\vec{b}$. Tento vektor popisuje zdroj silového působení v krystalu, kterým je čarová porucha – *dislokace*. Rovinný řez porušeným krystalem kolmo na dislokační čáru lze znázornit pomocí obrázku 1.1a. V krystalu „přebývá" jedna rovina, která náhle končí, její okraj tvoří *hranovou dislokaci*. Směr dislokační čáry je určen jednotkovým vektorem $\vec{\xi}$, jenž zde míří kolmo za nákresnu.

Kolem dislokace lze vytvořit uzavřený okruh $A123B$ procházející neporušenou oblastí krystalu, kde došlo jen k malým posunutím atomů z mřížkových poloh (obr. 2.1a). Deformace odpovídající těmto posunutím jsou malé a lze je popsat jako lineární funkce složek Burgersova vektoru. Přesuneme-li nyní smyčku $A123B$ do neporušeného krystalu (obrázek 2.1b), okruh se změní, počáteční bod A nesplývá s konečným B. Okruh lze uzavřít vektorem $\vec{b}$, tento postup je proto užit k jeho definici. Tato definice se jmenuje Burgersova-Frankova.

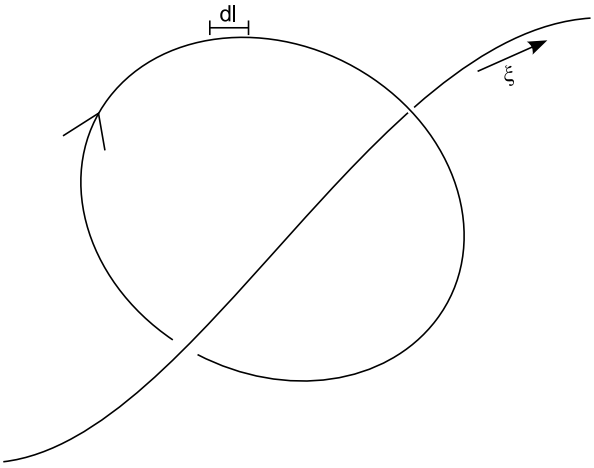


Obrázek 2.1: Řez porušenou prostorovou kubickou mřížkou s hranovou dislokací (a), přenos smyčky do neporušeného krystalu (b).

Jiná definice Burgersova vektoru využívá mechaniky kontinua (podle vztahu 2.1):

$$\vec{b} = \oint_{\mathcal{C}} \frac{\partial \vec{u}}{\partial l} \mathrm{d}l \tag{2.1}$$

Vektorové pole $\vec{u}$ popisuje posunutí kontinua v okolí dislokace. Výsledkem integrálu podél uzavřené křivky $\mathcal{C}$ je Burgersův vektor $\vec{b}$. Pro hranovou dislokaci znázorněnou na obr. 2.1a je $\vec{b}$ kolmý na směr dislokace, tj. $\vec{b} \cdot \vec{\xi} = 0$. Druhým mezním případem jsou šroubové dislokace vznikající smykem krystalových rovin, jejich Burgersův vektor je rovnoběžný se směrem dislokace, tedy $|\vec{b} \cdot \vec{\xi}| = b$.



Obrázek 2.2: K definici Burgersova vektoru pomocí mechaniky kontinua.

Obecná dislokace má složku hranovou $\vec{b}_e$ a složku šroubovou $\vec{b}_s$:

$$\begin{aligned} \vec{b}_e &= \vec{\xi} \times (\vec{b} \times \vec{\xi}) \\ \vec{b}_s &= (\vec{b} \cdot \vec{\xi})\, \vec{\xi} \end{aligned} \tag{2.2}$$

2.2 Tenzory napětí a deformace

Stav napjatosti popisuje tenzor napětí σ (v literatuře bývá někdy značen jako τ) [5]. Odpovídající geometrie kontinua je pak popsána tenzorem deformace ε. Pokud jsou deformace malé, tj. zabýváme-li se lineární teorií elasticity, lze vztah mezi vektorovým polem posunutí $\vec{u}$ a tenzorem deformace ε vyjádřit jako[1]:

$$\varepsilon_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial j} + \frac{\partial u_j}{\partial i}\right], \quad i,j \in \{x,y,z\} \tag{2.3}$$

Mezi složkami tenzoru deformace a napětí platí vztahy:

$$\begin{aligned} \varepsilon_{xx} &= \frac{1}{E}\left[\sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz})\right] & \varepsilon_{yz} &= \frac{1}{2\mu}\sigma_{yz} \\ \varepsilon_{yy} &= \frac{1}{E}\left[\sigma_{yy} - \nu(\sigma_{xx} + \sigma_{zz})\right] & \varepsilon_{xz} &= \frac{1}{2\mu}\sigma_{xz} \\ \varepsilon_{zz} &= \frac{1}{E}\left[\sigma_{xx} - \nu(\sigma_{xx} + \sigma_{yy})\right] & \varepsilon_{xy} &= \frac{1}{2\mu}\sigma_{xy} \end{aligned} \tag{2.4}$$

[1] Ze vztahu je zřejmé, že tenzor deformace ε je symetrický.

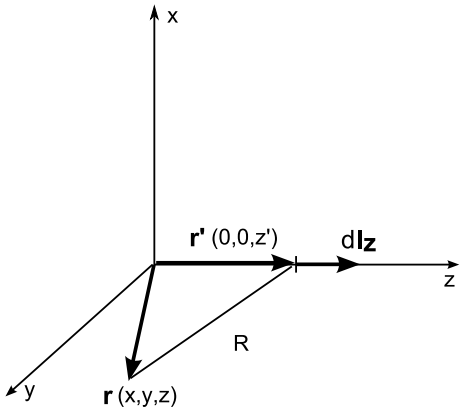
3. MATEMATICKÉ ODVOZENÍ

3.1 Napěťové pole kolem rovného úseku dislokace

Pro složky tenzoru napětí v okolí dislokace tvaru obecné křivky platí Peachův-Koehlerův vztah

$$\begin{aligned}\sigma_{\alpha\beta} &= -\frac{\mu}{8\pi}\oint_{\mathcal{C}} b_m\epsilon_{im\alpha}\frac{\partial}{\partial x'_i}\Delta' R\mathrm{d}x'_\beta - \frac{\mu}{8\pi}\oint_{\mathcal{C}} b_m\epsilon_{im\beta}\frac{\partial}{\partial x'_i}\Delta' R\mathrm{d}x'_\alpha \\ &\quad -\frac{\mu}{4\pi(1-\nu)}\oint_{\mathcal{C}} b_m\epsilon_{imk}\left(\frac{\partial^3 R}{\partial x'_i\partial x'_\alpha\partial x'_\beta} - \delta_{\alpha\beta}\frac{\partial}{\partial x'_i}\Delta' R\right)\mathrm{d}x'_k\,, \end{aligned} \tag{3.1}$$

kde $R^2 = x^2 + y^2 + (z - z')^2$ a $\alpha, \beta \in \{1, 2, 3\}$. Uspořádání odpovídá obrázku 3.1. Nečárkovanými souřadnicemi je vyjádřena poloha, v níž hledáme napětí (na obrázku vektor $\vec{r} = (x, y, z)$, čárkovanými souřadnicemi jsou určeny polohy infinitezimálních segmentů rovného úseku dislokace (vektor $\vec{r}' = (0, 0, z')$).



Obrázek 3.1: Souřadná soustava k výpočtu napěťového pole dislokačního segmentu.

Úsek dislokace je tedy úsečka ležící na ose z, nechť je vymezena souřadnicemi z'_2 a z'_1. Potom je napěťové pole v místě $\vec{r}$ kolem úsečky určeno vztahem

$$\sigma_{ij}(\vec{r}) = \sigma_{ij}(z'_2) - \sigma_{ij}(z'_1) \tag{3.2}$$

Protože platí $\partial/\partial x_i = -\partial/\partial x'_i$, lze jednotlivé složky tenzoru napětí získat úpravou obec-

ného vztahu 3.1 ve tvaru:

$$\begin{aligned}
\sigma_{xx} &= \frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial x^2}-\frac{\partial}{\partial x_i}\Delta R\right)\mathrm{d}z' \\
&= \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial x^2\partial y}+\frac{\partial}{\partial y}\Delta R\right)+b_y\left(\frac{\partial^3 R}{\partial x^3}-\frac{\partial}{\partial x}\Delta R\right)\right]\mathrm{d}z' \\
\sigma_{yy} &= \frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial y^2}-\frac{\partial}{\partial x_i}\Delta R\right)\mathrm{d}z' \\
&= \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial y^3}+\frac{\partial}{\partial y}\Delta R\right)+b_y\left(\frac{\partial^3 R}{\partial x\partial y^2}-\frac{\partial}{\partial x}\Delta R\right)\right]\mathrm{d}z' \\
\sigma_{zz} &= \frac{\mu}{4\pi}\int b_m\epsilon_{imz}\frac{\partial}{\partial x_i}\Delta R\,\mathrm{d}z'+\frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial z^2}-\frac{\partial}{\partial x_i}\Delta R\right)\mathrm{d}z' \\
&= \frac{\mu}{4\pi}\int\left[b_x\left(-\frac{\partial}{\partial y}\Delta R\right)+b_y\left(\frac{\partial}{\partial x}\Delta R\right)\right]\mathrm{d}z'+ \\
&+ \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial y\partial z^2}+\frac{\partial}{\partial y}\Delta R\right)+b_y\left(\frac{\partial^3 R}{\partial x\partial z^2}-\frac{\partial}{\partial x}\Delta R\right)\right]\mathrm{d}z' \\
\sigma_{xy} &= \frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial x\partial y}\right)\mathrm{d}z' \\
&= \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial x\partial y^2}\right)+b_y\left(\frac{\partial^3 R}{\partial x^2\partial y}\right)\right]\mathrm{d}z' \\
\sigma_{xz} &= \frac{\mu}{8\pi}\int b_m\epsilon_{imx}\frac{\partial}{\partial x_i}\Delta R\,\mathrm{d}z'+\frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial x\partial z}\right)\mathrm{d}z' \\
&= \frac{\mu}{8\pi}\int\left[b_y\left(-\frac{\partial}{\partial z}\Delta R\right)+b_z\left(\frac{\partial}{\partial y}\Delta R\right)\right]\mathrm{d}z'+ \\
&+ \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial x\partial y\partial z}\right)+b_y\left(\frac{\partial^3 R}{\partial x^2\partial z}\right)\right]\mathrm{d}z' \\
\sigma_{yz} &= \frac{\mu}{8\pi}\int b_m\epsilon_{imy}\frac{\partial}{\partial x_i}\Delta R\,\mathrm{d}z'+\frac{\mu}{4\pi(1-\nu)}\int b_m\epsilon_{imz}\left(\frac{\partial^3 R}{\partial x_i\partial y\partial z}\right)\mathrm{d}z' \\
&= \frac{\mu}{8\pi}\int\left[b_x\left(\frac{\partial}{\partial z}\Delta R\right)+b_z\left(-\frac{\partial}{\partial x}\Delta R\right)\right]\mathrm{d}z'+ \\
&+ \frac{\mu}{4\pi(1-\nu)}\int\left[b_x\left(-\frac{\partial^3 R}{\partial y^2\partial z}\right)+b_y\left(\frac{\partial^3 R}{\partial x\partial y\partial z}\right)\right]\mathrm{d}z' \qquad (3.3)
\end{aligned}$$

Rovnice 3.3 lze integrovat s využitím vztahů:

$$\begin{aligned}
\Delta R &= 2/R & \frac{\partial}{\partial x_i}\Delta R &= -2x_i/R^3 \\
\int\frac{\mathrm{d}x}{(x^2\pm a^2)^{3/2}} &= \pm\frac{x}{a^2\sqrt{x^2\pm a^2}} & \int\frac{\mathrm{d}x}{(x^2\pm a^2)^{5/2}} &= \pm\frac{3a^2x+2x^3}{3a^4(x^2\pm a^2)^{3/2}}
\end{aligned} \qquad (3.4)$$

Po integraci vyjdou tyto vztahy pro jednotlivé složky:

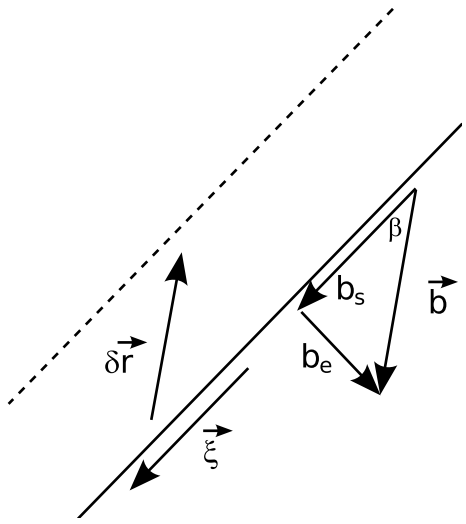
$$\begin{aligned}
\frac{\sigma_{xx}}{\sigma_0} &= b_x \frac{y}{R(R+\lambda)}\left[1+\frac{x^2}{R^2}+\frac{x^2}{R(R+\lambda)}\right] + b_y \frac{x}{R(R+\lambda)}\left[1-\frac{x^2}{R^2}-\frac{x^2}{R(R+\lambda)}\right] \\
\frac{\sigma_{yy}}{\sigma_0} &= -b_x \frac{y}{R(R+\lambda)}\left[1-\frac{y^2}{R^2}-\frac{y^2}{R(R+\lambda)}\right] - b_y \frac{x}{R(R+\lambda)}\left[1+\frac{y^2}{R^2}+\frac{y^2}{R(R+\lambda)}\right] \\
\frac{\sigma_{zz}}{\sigma_0} &= b_x\left[\frac{2\nu y}{R(R+\lambda)}+\frac{y\lambda}{R^3}\right] + b_y\left[-\frac{2\nu x}{R(R+\lambda)}-\frac{x\lambda}{R^3}\right] \\
\frac{\sigma_{xy}}{\sigma_0} &= -b_x \frac{x}{R(R+\lambda)}\left[1-\frac{y^2}{R^2}-\frac{y^2}{R(R+\lambda)}\right] + b_y \frac{y}{R(R+\lambda)}\left[1-\frac{x^2}{R^2}-\frac{x^2}{R(R+\lambda)}\right] \\
\frac{\sigma_{xz}}{\sigma_0} &= -b_x \frac{xy}{R^3} + b_y\left[-\frac{\nu}{R}+\frac{x^2}{R^3}\right] + b_z \frac{y(1-\nu)}{R(R+\lambda)} \\
\frac{\sigma_{yz}}{\sigma_0} &= b_x\left[\frac{\nu}{R}-\frac{y^2}{R^3}\right] + b_y \frac{xy}{R^3} - b_z \frac{x(1-\nu)}{R(R+\lambda)} \\
&\text{kde } \sigma_0 = \frac{\mu}{4\pi(1-\nu)} \text{ a } \lambda = z' - z
\end{aligned} \tag{3.5}$$

Podrobnosti k odvození těchto vztahů lze nalézt v [2].

3.2 Síla působící na segment dislokace

Je-li přímá dislokace posunuta o $\delta\vec{r}$, je kolmo na skluzovou rovinu přemístěna hmota o objemu $\delta\nu$ vztažená na délku dislokační linie L. Situace je znázorněna obrázkem 3.2.

$$\frac{\delta\nu}{L} = b_e \delta h$$



Obrázek 3.2: Posunutí dislokace o $\vec{r}$.

Vzdálenost δh je určena projekcí vektoru $\delta\vec{r}$ do normály skluzové roviny $\vec{e}_n = \vec{\xi}\times\vec{b}_e/b_e$,

tedy platí $\delta h = \vec{e}_n \cdot \delta\vec{r}$ a z toho:

$$\frac{\delta\nu}{L} = \delta\vec{r} \cdot (\vec{\xi} \times \vec{b}_e) = \delta\vec{r} \cdot (\vec{\xi} \times \vec{b}) \tag{3.6}$$

Při posunu přímé dislokace je vykonána práce.

$$\frac{\delta W}{L} = \frac{\vec{F}}{L} \cdot \delta\vec{r} \tag{3.7}$$

Tato práce je určena vztahem

$$\frac{\vec{F}}{L} \cdot \delta r = \left[\vec{\sigma} \cdot (\vec{\xi} \times \delta\vec{r})\right] \cdot \vec{b} \tag{3.8}$$

Po přepsání[1] tento vztah nabude podoby

$$\frac{\vec{F}}{L} \cdot \delta\vec{r} = (\vec{b} \cdot \vec{\sigma}) \cdot (\vec{\xi} \times \delta\vec{r}) = \left[(\vec{b} \cdot \vec{\sigma}) \times \vec{\xi}\right] \cdot \delta\vec{r}$$

Z toho plyne obecný tvar pro sílu napěťového pole působící na jednotkovou délku přímé dislokace:

$$\frac{\vec{F}}{L} = (\vec{b} \cdot \vec{\sigma}) \times \vec{\xi} \tag{3.9}$$

3.3 Pohybové rovnice dislokačního segmentu

Ze znalosti síly působící na jednotku délky dislokačního segmentu 3.9 je možné zjistit, jak se tento segment bude pohybovat. Nelze však užít Newtonův zákon síly, neboť dislokace nemá žádnou hmotnost. Je možné ukázat (např. v [2]), že rychlost pohybu dislokace kolmo na skluzovou rovinu (ve směru $\vec{b} \times \vec{\xi}$) je

$$v_\perp = \frac{D_s \Omega}{b_e^2 kT} \frac{F_\perp}{L} = B \frac{F_\perp}{L}, \tag{3.10}$$

Ω je objem iontového páru (zde nahrazen objemem připadajícím na jeden atom), b_e je velikost hranové složky Burgersova vektoru, k Boltzmannova konstanta, T teplota a D_s je součinitel samodifuze určený vztahem

$$D_s = D_0 \mathrm{e}^{-\frac{Q}{RT}}, \tag{3.11}$$

Q je aktivační energie samodifuze a $R = kN_A$ je univerzální plynová konstanta, N_A je Avogadrova konstanta.

Pohyb ve skluzové rovině může být rovněž lineární:

$$v_\parallel = A \frac{F_\parallel}{L} \tag{3.12}$$

Pohyb ve skluzové rovině je snazší než pohyb kolmý (šplhání), zvolme proto

$$A = 10B \tag{3.13}$$

[1] S využitím $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}, \quad \vec{a} \cdot (\vec{b} \times \vec{c}) = \vec{c} \cdot (\vec{a} \times \vec{b}), \quad a, b, c \in \mathbb{R}^3$.

Obecné určení tohoto součinitele je složité.

Složením výše uvedených vztahů 3.10 a 3.12 vyjde:

$$\begin{aligned} \vec{F}_\perp &= \frac{\vec{F}\cdot(\vec{b}\times\vec{\xi})}{|\vec{b}\times\vec{\xi}|^2}(\vec{b}\times\vec{\xi}) \\ \vec{F}_\parallel &= \vec{F}-\vec{F}_\perp \\ \vec{v} &= A\frac{\vec{F}_\parallel}{L}+B\frac{\vec{F}_\perp}{L} \end{aligned} \tag{3.14}$$

Z tohoto vztahu pro rychlost lze již snadno spočítat posunutí dislokačního segmentu:

$$\Delta\vec{r}=\vec{v}\Delta t \tag{3.15}$$

Délka jednoho kroku Δt je jedním z parametrů určujících rychlost a přesnost výpočtu.

3.4 Skládání napěťových polí

Vztahy pro napěťové pole odvozené v předešlém textu platí pouze pro dislokační úseky ležící na ose $\vec{z}$. Mají-li být použity k výpočtu napěťového pole v okolí lomené čáry v prostoru (která slouží k přibližnému vyjádření obecné křivky), je třeba tuto čáru rozložit na jednotlivé úsečky a najít vhodné lineární zobrazení úsečky ležící na ose $\vec{z}$ na úsečku ležící na lomené čáře v prostoru. Je zřejmé, že to bude zobrazení složené z posunutí a otočení. Nejprve najděme vztah pro otočení.

3.4.1 Otočení ve dvou rozměrech

Pro otočení ve dvou rozměrech platí tento vztah:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \overbrace{\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}}^{\hat{O}} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.16}$$

Úhel φ je úhel otočení a roste proti směru hodinových ručiček. Jde o ortogonální zobrazení ($\hat{O}^T=\hat{O}^{-1}$), zachovává tedy úhly (skalární součin), délky, plochu i objem (má jednotkový determinant).

Toto zobrazení lze převést i do třech rozměrů tak, že prvky matice odpovídající rovině otáčení budou mít stejný tvar, jako výše uvedené zobrazení $\hat{O}$, prvek odpovídající ose otáčení bude roven jedné.

$$\hat{O}_{\vec{z}}=\begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \hat{O}_{\vec{x}}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{pmatrix}, \hat{O}_{\vec{y}}=\begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} \tag{3.17}$$

3.4.2 Obecné otočení ve třech rozměrech

Původní souřadná soustava nechť je x, y, z a otočená x', y', z'. Dále označme průsečnici rovin xy a $x'y'$ jako p. Nejprve otočme kolem z o úhel φ_1 – $\hat{O}_{\vec{z}}(\varphi_1)$ – a osa x se zobrazí na

p. Nyní otočme kolem p o ϑ – $\hat{O}_{\vec{p}}(\vartheta)$ – a z přejde v z'. Nakonec otočme kolem z' o φ_2 – $\hat{O}_{\vec{z}'}(\varphi_2)$ – a p se otočí na x'. Vznikne tak obecné otočení

$$\hat{O}(\phi) = \hat{O}_{\vec{z}'}(\varphi_2)\hat{O}_{\vec{p}}(\vartheta)\hat{O}_{\vec{z}}(\varphi_1) \tag{3.18}$$

Lze ukázat, že otočení kolem p a z' lze získat přímo otočením kolem os x a z, a to tak, že požadované otočení kolem p a z' složíme z otočení kolem původních os a otočení, která převádějí osy p a z' na osy původní a zpět.

$$\begin{aligned} \hat{O}_{\vec{p}}(\vartheta) &= \hat{O}_{\vec{z}}(\varphi_1)\hat{O}_{\vec{x}}(\vartheta)\hat{O}_{\vec{z}}(\varphi_1)^{-1} \\ \hat{O}_{\vec{z}'}(\varphi_2) &= \left[\hat{O}_{\vec{p}}(\vartheta)\hat{O}_{\vec{z}}(\varphi_1)\right]\hat{O}_{\vec{z}}(\varphi_2)\left[\hat{O}_{\vec{p}}(\vartheta)\hat{O}_{\vec{z}}(\varphi_1)\right]^{-1} \end{aligned} \tag{3.19}$$

Když tato otočení dosadíme do předchozího vztahu 3.19, tak vyjde:

$$\hat{O}(\phi) = \hat{O}_{\vec{z}}(\varphi_1)\hat{O}_{\vec{x}}(\vartheta)\hat{O}_{\vec{z}}(\varphi_2) \tag{3.20}$$

Toto otočení je složeno z elementárních otočení podle původních os, jen jsou tato otočení seřazena opačně. Tento vztah však můžeme vyjádřit i ve složkách vynásobením matic jednotlivých otočení:

$$\hat{O}(\phi) = \begin{pmatrix} \cos\varphi_1\cos\varphi_2 - \sin\varphi_1\sin\varphi_2\cos\vartheta & \cos\varphi_1\sin\varphi_2 + \sin\varphi_1\cos\varphi_2\cos\vartheta & \sin\varphi_1\sin\vartheta \\ -\sin\varphi_1\cos\varphi_2 - \cos\varphi_1\sin\varphi_2\cos\vartheta & -\sin\varphi_1\sin\varphi_2 + \cos\varphi_2\cos\varphi_2\cos\vartheta & \cos\varphi_1\sin\vartheta \\ \sin\varphi_2\sin\vartheta & -\cos\varphi_2\sin\vartheta & \cos\vartheta \end{pmatrix} \tag{3.21}$$

Z matice otočení ve třech rozměrech lze snadno určit také úhel otočení ϕ. Matice otočení zachovává délku vektoru, takže její vlastní hodnoty budou komplexní jednotky. Protože jde o kořeny charakteristické rovnice $\lambda^3 = 1$, budou hodnoty rovny $\lambda_1 = 1, \lambda_2 = \mathrm{e}^{\mathrm{i}\phi}$ a $\lambda_3 = \mathrm{e}^{-\mathrm{i}\phi}$. Po nalezení vlastních hodnot lze matici otočení diagonalizovat a získat tak matici $L = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$, která je původní matici podobná. Rovněž lze matici převést podobnostní transformací do tvaru $\hat{O}_{\vec{x}}(\phi)$. Všechny tyto matice mají stejnou stopu, ta se podobnostní transformací nemění. Platí tedy:

$$\mathrm{Tr}\, O(\phi) = \mathrm{Tr}\, L = \lambda_1 + \lambda_2 + \lambda_3 = 1 + \mathrm{e}^{\mathrm{i}\phi} + \mathrm{e}^{-\mathrm{i}\phi} = 1 + 2\cos\phi$$

Podobnost s maticí $\hat{O}_{\vec{x}}(\phi)$ zaručuje, že zde jde opravdu o úhel otočení. Jiným vyjádřením dostaneme vztah

$$\cos\phi = \frac{\mathrm{Tr}\, O(\phi) - 1}{2} \tag{3.22}$$

Podrobnější informace o otočení lze nalézt v [6].

3.5 Hledání matice otočení

Při vlastním výpočtu bude vždy třeba otočit úsečku ležící na ose z tak, aby měla směr úsečky ležící na zadané lomené čáře. Je tedy třeba najít takové otočení, které otočí vektor $\vec{z}$ do směru nějakého obecného vektoru $\vec{\xi}$

$$\vec{\xi} = \hat{B}\vec{z}, \quad \text{kde } z = (0,0,1)^T \text{ a } \vec{\xi} \text{ je obecný vektor}$$

Je známo, že libovolné lineární zobrazení $\hat{B}$ je jednoznačně určeno obrazy báze. Vektor $\vec{z}$ spolu s vektory $\vec{x}$, $\vec{y}$ tvoří kanonickou bázi. Ta je ortonormální. Vektor $\vec{v}_3$, který vznikl normováním vektoru $\vec{\xi}$, tedy $\vec{v}_3 = \vec{\xi}/\xi$, je však jediný a sám bázi tvořit nemůže. Lze jej však na bázi doplnit dvěma lineárně nezávislými vektory. Není-li báze ortogonální, lze ji takovou učinit Gram-Schmidtovým ortogonalizačním procesem. Lze ji však snadno ortogonální vytvořit s využitím dvou vektorových součinů. Následným normováním vznikne ortonormální báze $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$, kde $\vec{v}_3 = \vec{\xi}/\xi$. Protože původní báze $(\vec{x}, \vec{y}, \vec{z})$ je kanonická, tvoří nová ortonormální báze přímo matici zobrazení $\hat{B} = (\vec{v}_1, \vec{v}_2, \vec{v}_3)$.

Zobrazení $\hat{B}$ je jistě nějaké otočení, ale nemusí to být to, které hledáme. Zatímco pořadí vektoru $\vec{v}_3$ v bázi B je pevně dáno tak, jako je dáno pořadí vektoru $\vec{z}$ v bázi kanonické, u vektorů $\vec{v}_1$ a $\vec{v}_2$ není po ortogonalizaci zaručeno správné pořadí. To lze zjistit výpočtem determinantu a prověřením jeho znaménka. Je-li záporné, stačí vektory zaměnit a vznikne tak matice vlastního otočení bez zrcadlení. Je-li báze vytvořena vektorovými součiny, tento problém odpadá.

3.6 Hledání matice posunutí

Matice posunutí má tu vlastnost, že musí mít jeden rozměr navíc oproti prostoru, v němž má posouvat. Pokud rozšíříme třírozměrný vektor $\vec{r}$ o čtvrtou složku, která bude rovna jedné, dostaneme tak $\vec{r} = (x, y, z, 1)^T$. Zobrazení do posunutých souřadnic s počátkem $O' = (x_0, y_0, z_0)$ lze vyjádřit maticově takto:

$$\hat{P} = \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.23}$$

Přímým výpočtem se lze přesvědčit, že platí $\vec{r}' = \hat{P}\vec{r} = (x - x_0, y - y_0, z - z_0, 1)$.

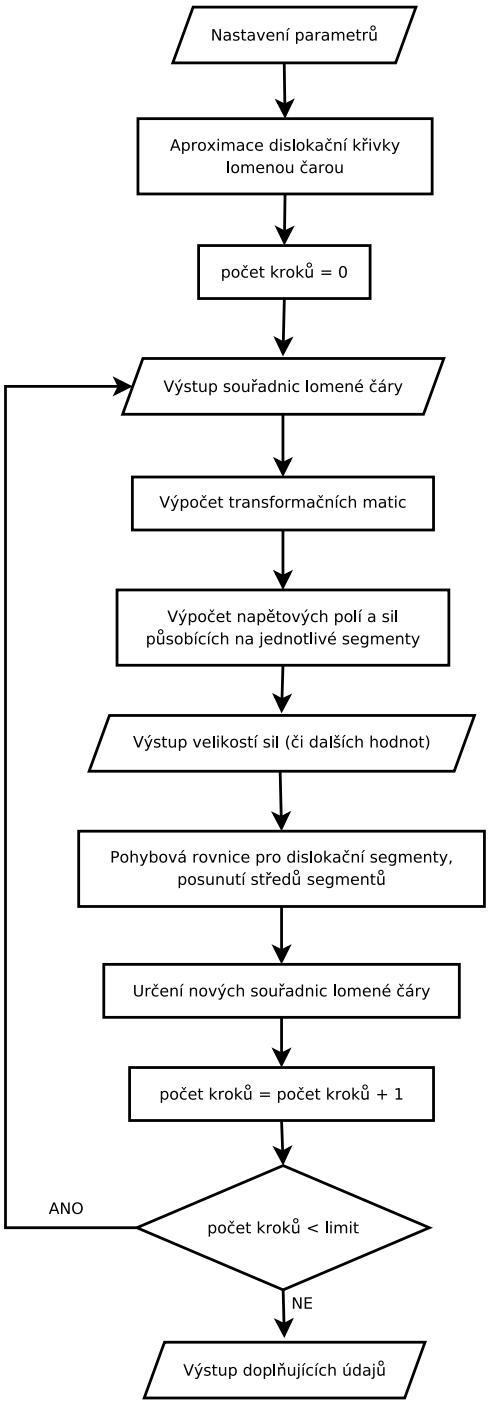
3.7 Výsledná transformace

Pokud si lomenou čáru rozložíme na jednotlivé úsečky – vektory, lze přímo odečíst souřadnice středů jednotlivých úseček a získat tak posunutí $\hat{P}$. Použitím postupu uvedeného v odstavci 3.5 lze určit matici otočení $\hat{O}$. Jejich složením vzniknou matice transformací souřadnic umožňující převádět napěťová pole mezi souřadnými soustavami. Snadnou úvahou lze dospět k tomu, že transformace mezi souřadnou soustavou lomené čáry ($\vec{r}$) a souřadnou soustavou úsečky ($\vec{r}'$) bude mít tuto podobu:

$$\vec{r} = \hat{T}\vec{r}' = \hat{P}\hat{O}\vec{r}', \quad \vec{r}' = \hat{T}^{-1}\vec{r} = \hat{O}^T\hat{P}^{-1}\vec{r} \tag{3.24}$$

Tato složená transformace je však potřeba pouze k zobrazení polohy, v níž hledáme napěťové pole. Burgersův vektor je vektorové pole, napěťové pole je tenzorové a stačí je pouze otočit.

4. Napěťové pole a pohyb křivočaré dislokace

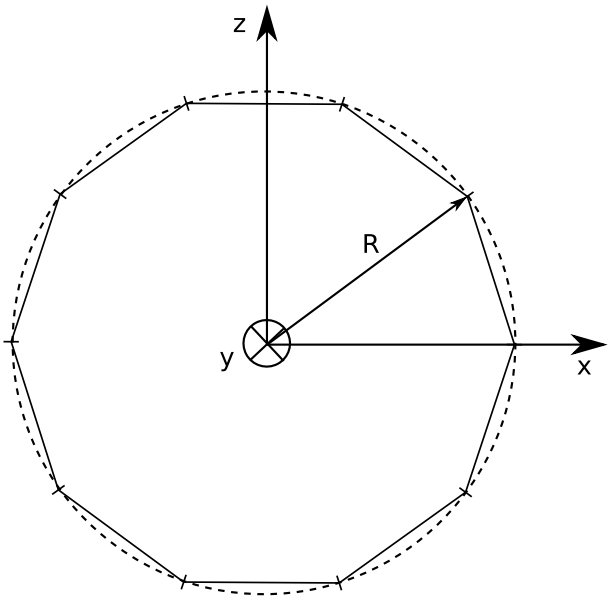


Obrázek 4.1: Blokové schéma programu

5. NUMERICKÉ ŘEŠENÍ

5.1 Aplikace obecného postupu

Obecný postup modelování napěťových polí a pohybu křivočaré dislokace formulovaný v předcházejících částech práce je demonstrován na příkladu kontrakce dislokační smyčky v poli vlastních napětí.



Obrázek 5.1: Dislokační smyčka a její aproximace lomenou čarou.

Kruhová smyčka ležící v rovině xz je nahrazena mnohoúhelníkem s N vrcholy vepsaným do této kružnice. Jednotlivé souřadnice určuje vzorec

$$X_i = \left[R\cos\frac{2\pi(i-1)}{N}, 0, R\sin\frac{2\pi(i-1)}{N} \right], \quad i = 1 \ldots N \tag{5.1}$$

5.2 Použité nástroje

Problém byl řešen programem napsaným v jazyce C++ a s využitím numerické knihovny Blitz ve verzi 0.9, která je dostupná na adrese `http://www.oonumerics.org/blitz`. Tato knihovna umožňuje snadno pracovat s vektory a maticemi. Pro lepší srozumitelnost zdrojového textu je zde uveden malý příklad použití této knihovny při násobení matice vektorem.

```
#include <iostream>
#include <blitz/array.h>
using namespace std;
using namespace blitz;
```

```
int main() {
  Array<double, 2> M(3, 3); Array<double, 1> a(3), b(3);
  firstIndex I; secondIndex J;
  M = 1, 2, 3,
      4, 5, 6,
      7, 8, 9;
  a = 9,10,11;
  cout << (b = sum(M(I, J), J)) << endl;
  return 0;
}
```

Je proveden výpočet

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad a = \begin{pmatrix} 9 \\ 10 \\ 11 \end{pmatrix}, \quad b_i = M_{ij}a_j \quad \Rightarrow \quad b = \begin{pmatrix} 62 \\ 152 \\ 242 \end{pmatrix}$$

5.3 Výpis zdrojového kódu

5.3.1 Univerzální část kódu

Kód uvedený v této části je univerzální a lze jej použít nezávisle na příkladu 5.3.2.

Na začátku programu je třeba uvést použité hlavičkové soubory.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cmath>
#include <blitz/array.h>
```

Dále bylo potřeba doplnit funkce pro vektorový a skalární součin `cross` a `dot` ve třech rozměrech.

```
template<typename FT> blitz::Array<FT, 1> cross(const blitz::Array<FT, 1>& v1,
  const blitz::Array<FT, 1>& v2) {
  blitz::Array<FT, 1> v(3);
  v = v1(1)*v2(2) - v1(2)*v2(1), v1(2)*v2(0) - v1(0)*v2(2), v1(0)*v2(1) - v1(1)*v2(0);
  return v;
}

template<typename FT> FT dot(const blitz::Array<FT, 1>& v1,
  const blitz::Array<FT, 1>& v2) {
  return v1(0) * v2(0) + v1(1) * v2(1) + v1(2) * v2(2);
}
```

Poté byla zavedena třída `StraightSegment3D`, v níž jsou uloženy potřebné souřadnice, matice přechodu a transformovaný Burgersův vektor. V této třídě zavedená metoda `ComputeDLTransformation3D` počítá matici přechodu podle 3.5 pomocí vektorových součinů:

$$\hat{O} = (\vec{v}_1, \vec{v}_2, \vec{v}_3), \quad \vec{v}_3 = \vec{\xi}, \quad \vec{v}_2 = \vec{v}_3 \times \frac{\vec{b}}{b}, \quad \vec{v}_1 = \vec{v}_2 \times \vec{v}_3 \tag{5.2}$$

Pokud jde náhodou o čistě šroubový segment, je zřejmě $\vec{b} = b\vec{\xi} \Rightarrow \vec{\xi} \times \vec{b} = b(\vec{\xi} \times \vec{\xi}) = \vec{0}$. Je tedy nutné použít jiný vektor, např. $\vec{b'} = (b_2, -b_3, b_1)^T$ či jakýkoli jiný. Pak jistě platí

$$\hat{O} = (\vec{v}_1, \vec{v}_2, \vec{v}_3), \quad \vec{v}_3 = \vec{\xi}, \quad \vec{v}_2 = \vec{v}_3 \times \frac{\vec{b'}}{b'}, \quad \vec{v}_1 = \vec{v}_2 \times \vec{v}_3 \tag{5.3}$$

Nejde-li o šroubový segment, je $\vec{v}_2 = \vec{\xi} \times \vec{b} \neq \vec{0}$ a platí

$$\hat{O}\vec{b} = (b_e, 0, b_s) \tag{5.4}$$

Jde-li o šroubový segment, platí

$$\hat{O}\vec{b} = (0, 0, b) \tag{5.5}$$

Veličiny b_e a b_s v rovnici 5.4 jsou velikosti hranové a šroubové složky Burgersova vektoru podle vztahu 2.2.

Z rovnic 5.4 a 5.5 plyne, že po transformaci souřadnic vypadne složka příslušná ose ve směru $\vec{v}_2$, takže lze vypustit všechny členy obsahující b_y ve vztahu 3.5, čímž lze urychlit běh programu.

```
template<class FT> class StraightSegment3D {
public:
  StraightSegment3D(const blitz::Array<FT, 1>& c, bool _f = false) :
    fixed(_f), coordinates(c), dir(3), center(3), base(3, 3), burgers(3) { }
  StraightSegment3D(FT cx, FT cy, FT cz, bool _f = false) :
    fixed(_f), coordinates(3), dir(3), center(3), base(3, 3), burgers(3) {
    blitz::Array<FT, 1> coordinates;
    coordinates = cx, cy, cz;
  }
  const blitz::Array<FT, 1>& Fixed() const { return fixed; }
  const blitz::Array<FT, 1>& Coordinates() const { return coordinates; }
  blitz::Array<FT, 1>& Coordinates() { return coordinates; }
  const blitz::Array<FT, 1>& Center() const { return center; }
  blitz::Array<FT, 1>& Center() { return center; }
  FT Len() const { return len; }
  const blitz::Array<FT, 1>& Dir() const { return dir; }
  const blitz::Array<FT, 2>& Base() const { return base; }
  const blitz::Array<FT, 1>& Burgers() const { return burgers; }
  void ComputeTransformation3D(const StraightSegment3D<FT>& prev,
    const blitz::Array<FT, 1>& _burgers);
private:
  bool fixed;                       /* bod je nehybný? */
  blitz::Array<FT, 1> coordinates; /* souřadnice bodu */
  FT len;                           /* délka úsečky */
  blitz::Array<FT, 1> dir;         /* směr úsečky */
  blitz::Array<FT, 1> center;      /* střed úsečky */
  blitz::Array<FT, 2> base;        /* souřadné soustava úsečky */
  blitz::Array<FT, 1> burgers;     /* Burgersův vektor v této souřadné soustavě */
};

/* Tato funkce vytváří matici otočení, totiž matici přechodu ze souřadné soustavy úsečky
 * lomené čáry  do souřadné soustavy rovnoběžné se soustavou lomené čáry.
 * Je třeba zadat předchozí souřadnici pro vytvoření úsečky a Burgersův vektor.
 */
template<class FT> void StraightSegment3D<FT>::ComputeTransformation3D(
  const StraightSegment3D<FT>& prev, const blitz::Array<FT, 1>& _burgers) {
  FT size;
  blitz::Array<FT, 1> tmp(3);
  blitz::firstIndex I; blitz::secondIndex J;
  center = (Coordinates()(0) + prev.Coordinates()(0)) / 2, /* střed úsečky */
           (Coordinates()(1) + prev.Coordinates()(1)) / 2,
           (Coordinates()(2) + prev.Coordinates()(2)) / 2;
  /* base(2) je směrový vektor určující novou osu 'z' */
```

```
  dir = Coordinates() - prev.Coordinates();
  dir /= (len = sqrt(sum(sqr(dir))));
  base(blitz::Range::all(), 2) = dir;
  /* toto je normovaný Burgersův vektor */
  tmp = _burgers / sqrt(sum(sqr(_burgers)));
  /* vektorový součin směrového a Burgersova vektoru dá další prvek báze */
  base(blitz::Range::all(), 1) = cross(base(blitz::Range::all(), 2), tmp);
  /* Je-li Burgersův vektor rovnoběžný se směrovým vektorem, je třeba postupovat jinak. */
  if ((size = sqrt(sum(sqr(base(blitz::Range::all(), 1))))) < 1e-7) {
    /* Tento zápis není dobrý, leč vysvětluje tři řádky kódu níže. */
    /* tmp = base(1, 2), -base(2, 2), base(0, 2);
     * base(1, blitz::Range::all()) = cross(base(2, Range::all()), tmp); */
    base(0, 1) =  base(0, 2) * base(1, 2) + base(2, 2) * base(2, 2);
    base(1, 1) =  base(1, 2) * base(2, 2) - base(0, 2) * base(0, 2);
    base(2, 1) = -base(2, 2) * base(0, 2) - base(1, 2) * base(1, 2);
    base(blitz::Range::all(), 1) /= sqrt(sum(sqr(base(blitz::Range::all(), 1))));
  }
  /* Vektorovým součinem bázových vektorů odpovídajících ose 'y' a 'z' vznikne ten
   * příslušný ose 'x', tady je vhodné postupovat stejně. */
  base(blitz::Range::all(), 0) = cross(base(blitz::Range::all(), 1),
    base(blitz::Range::all(), 2));
  base(blitz::Range::all(), 0) /= sqrt(sum(sqr(base(blitz::Range::all(), 0))));
  /* Je třeba ještě opravit normování. */
  base(blitz::Range::all(), 1) /= sqrt(sum(sqr(base(blitz::Range::all(), 1))));

  /* Nyní je hotova matice přechodu ze souřadné soustavy úsečky do souřadné soustavy,
   * v níž je úsečka určena. Opačnou matici přechodu není třeba vytvářet, stačí ji
   * transponovat či jen vyměnit její indexy. */

  /* Je vhodné určit Burgersův vektor v souřadné soustavě úsečky. */
  burgers = sum(base(J, I) * _burgers(J), J);
}

/* Tato funkce projde výčet souřadnich určujících lomenou čáru a vytvoří pro
 * jednotlivé segmenty pomocí předchozí funkce matice přechodu a Burgersův vektor. */
template<typename FT> void ComputeDLTransformation3D(
  std::vector<StraightSegment3D<FT> >& straightsegments,
  const blitz::Array<FT, 1>& burgers) {
  typename std::vector<StraightSegment3D<FT> >::iterator cc, cn;
  cn = straightsegments.begin();
  cc = cn++;
  for (; cn != straightsegments.end(); ++cn) {
    cn->ComputeTransformation3D(*cc, burgers);
    cc = cn;
  }
}
```

Třída `StressField` obstarává výpočet napěťového pole podle vztahu 3.5 s vynecháním členů b_y dosazeného do rozdílu 3.2.

```
template<class FT> class StressField {
public:
  StressField() : _r(3), _b(3), S(3, 3) { }
  StressField(const blitz::Array<FT, 1>& r_, FT z2_, FT mu_, FT nu_,
    const blitz::Array<FT, 1>& b_) :
    _z2(z2_), _mu(mu_), _nu(nu_),
    _b(b_), _r(r_), S(3, 3) { }
  FT x() const { return _r(0); } FT& x() { return _r(0); }
  FT y() const { return _r(1); } FT& y() { return _r(1); }
  FT z() const { return _r(2); } FT& z() { return _r(2); }
  FT z2() const { return _z2; } FT& z2() { return _z2; }
  FT mu() const { return _mu; } FT& mu() { return _mu; }
  FT nu() const { return _nu; } FT& nu() { return _nu; }
  const blitz::Array<FT, 1>& b() const { return _b; }
  blitz::Array<FT, 1>& b() { return _b; }
  FT XX() const {
    return b()(0) * y() / RRlambda() * xplus();
  }
```

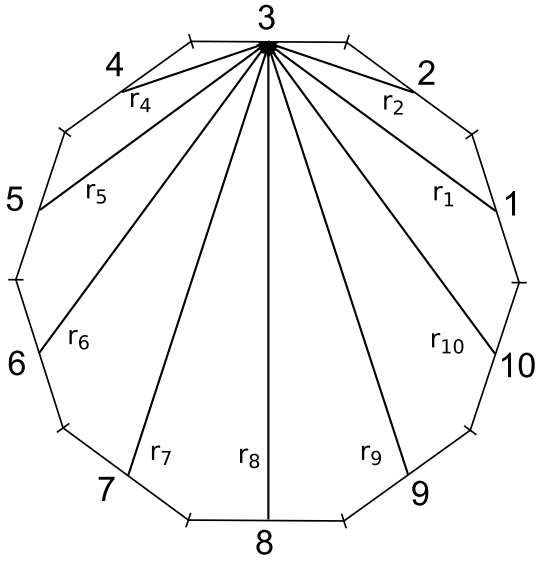
```
  FT YY() const {
    return -b()(0) * y() / RRlambda() * yminus();
  }
  FT ZZ() const {
    return b()(0) * (2 * nu() * y() / RRlambda() + y() * lambda() / (R2()*R()));
  }
  FT XY() const {
    return -b()(0) * x() / RRlambda() * yminus();
  }
  FT XZ() const {
    return -b()(0) * x() * y() / (R2()*R())
      + b()(2) * y() * (1 - nu()) / RRlambda();
  }
  FT YZ() const {
    return b()(0) * (nu() / R() - y() * y() / (R2()*R()))
      - b()(2) * x() * (1 - nu()) / RRlambda();
  }
  const blitz::Array<FT, 2>& operator()() {
    S = sigma0() * XX(), sigma0() * XY(), sigma0() * XZ(),
        sigma0() * XY(), sigma0() * YY(), sigma0() * YZ(),
        sigma0() * XZ(), sigma0() * YZ(), sigma0() * ZZ();
    return S;
  }
private:
  FT _z2, _mu, _nu;
  blitz::Array<FT, 1> _b, _r;
  blitz::Array<FT, 2> S;

  FT R2() const { return x()*x()+y()*y()+(z2() - z())*(z2() - z()); }
  FT R() const { return sqrt(R2()); }
  FT lambda() const { return z2() - z(); }
  FT RRlambda() const { return R()*(R()+lambda()); }
  FT xplus() const { return 1 + x()*x() / R2() + x()*x() / RRlambda(); }
  FT yminus() const { return 1 - y()*y() / R2() - y()*y() / RRlambda(); }
  FT sigma0() const {
    return mu() / (4 * M_PI * (1 - nu()));
  }
};
```

Aby bylo možné spočítat sílu působící na segment dislokace a posléze pohyb tohoto segmentu, je třeba ve středu každého segmentu vypočítat napěťová pole pocházející od ostatních segmentů a sečíst je (podle obr. 5.2):

$$\sigma_{(i)} = \sum_{\substack{j=1 \\ i \neq j}}^{N} \sigma(\vec{r}_j) \tag{5.6}$$

Napěťová pole jednotlivých segmentů počítá funkce `GetStress`, výsledný součet vrací funkce `SumStresses`.

```
/* Vypočítá napěťové pole v bodě 'r' u n. segmentu dislonační čáry. */
template<typename FT> blitz::Array<FT, 2> GetStress(
  const std::vector<StraightSegment3D<FT> >& segments,
  const blitz::Array<FT, 1>& burgers,
  unsigned int n,
  const blitz::Array<FT, 1>& r, FT mu, FT nu) {
  /* napěťové pole */
  blitz::Array<FT, 2> stress(3, 3), stress2(3, 3);
  /* indexy */
  blitz::firstIndex I; blitz::secondIndex J; blitz::thirdIndex K;
  /* transformované vektory */
  blitz::Array<FT, 1> r_trans(3), tmp(3);
  /* r_i' = T_ji r_j */
  tmp           = r - segments[n].Center();
```



Obrázek 5.2: Ke sčítání napěťových polí jednotlivých segmentů. Zde $i = 3, N = 10$.

```
  r_trans       = sum(segments[n].Base()(J, I) * tmp(J), J);
  /* Nyní je možné dosadit vektory vyjádřené v bázi segmentu. */
  /* Výpočet napěťového pole. */
  StressField<FT> S(r_trans, segments[n].Len()/2, mu, nu, segments[n].Burgers());
  stress = S();
  S.z2() = -segments[n].Len()/2;
  stress -= S();
  /* Tenzor napětí je v bázi segmentu, je třeba jej převést do báze lomené čáry. */
  stress2 = sum(segments[n].Base()(I, K) * stress(K, J), K);
  stress = sum(stress2(I, K) * segments[n].Base()(J, K), K);
  return stress;
}

/* Vypočítá napěťové pole ve středu n. segmentu vzniklé součtem polí ostatních. */
template<typename FT> blitz::Array<FT, 2> SumStresses(
  const std::vector<StraightSegment3D<FT> >& segments,
  const blitz::Array<FT, 1>& burgers,
  unsigned int n, FT mu, FT nu) {
  unsigned int i;
  /* Napěťové pole */
  blitz::Array<FT, 2> stress(3, 3);
  stress = 0; /* Nulování je potřeba :-) */
  for (i = 1; i < segments.size(); i++)
    if (i != n)
      stress += GetStress<FT>(segments, burgers, i, segments[n].Center(), mu, nu);
  return stress;
}
```

Z napěťového pole lze nyní určit sílu působící na jednotkovou délku dislokačního segmentu podle vztahu 3.9. Funkce se jmenuje `GetForce`.

```
/* Vypočítá sílu působící na n. segment vyvolanou poli ostatních segmentů. */
template<typename FT> blitz::Array<FT, 1> GetForce(
  const std::vector<StraightSegment3D<FT> >& segments,
  const blitz::Array<FT, 1>& burgers,
  unsigned int n, FT mu, FT nu) {
  /* síla */
  blitz::Array<FT, 1> force(3);
  blitz::Array<FT, 2> stress(3, 3);
  /* napěťové pole */
```

```
  stress = SumStresses(segments, burgers, n, mu, nu);
  /* indexy */
  blitz::firstIndex I; blitz::secondIndex J;
  blitz::Array<FT, 1> tmp(3);
  tmp = sum(stress(I, J) * burgers(J), J);
  /* síla */
  force = cross(tmp, segments[n].Dir());
  return force;
}
```

K výpočtu rychlosti segmentu podle vztahu 3.14 je třeba určit konstanty A a B, což činí třída `SegmentMotion3D` pomocí metod `AGlide` pro skluzovou konstantu A a `CClimb` pro šplhovou konstantu B.

```
#ifdef _R_
#undef _R_
#endif
#define _R_ 8.314 /* univerzální plynová konstanta */

#ifdef _k_
#undef _k_
#endif
#define _k_ 1.381e-23 /* Boltzmannova konstanta */

template<class FT> class SegmentMotion3D {
public:
  SegmentMotion3D(FT _D0, FT _Q, FT _T, FT _Omega,
    const blitz::Array<FT, 1>& burgers) {
    FT bb = sum(sqr(burgers));
    b0 = sqrt(bb);
    K = _D0 * exp(-_Q/(_R_*_T)) * _Omega / (bb * _k_ * _T);
    aglide = 10 * K;
  }
  FT AGlide() const { return aglide; }
  FT CClimb(FT be, FT bs) const {
    return K * (be + 10 * bs) / b0;
  }
private:
  FT b0, K, aglide;
};

#undef _k_
#undef _R_
```

Nyní již byly probrány téměř všechny funkce, které budou potřeba k výpočtu pohybu dislokační čáry. Pohyb jednotlivých segmentů bude dán vztahy 3.14 a 3.15, tedy bude známo posunutí středů jednotlivých segmentů, z poloh posunutých středů je poté potřeba znovu určit vrcholy lomené čáry. Nejsnazší se zdá být prosté spojení posunutých středů, tento postup je však možné použít jen u křivek, které jsou přibližně z poloviny konvexní a druhé poloviny konkávní. Např. u pravidelného mnohoúhelníku nahrazujícího kružnici (tj. čistě konvexní křivky) by jen opakované počítání středů a jejich spojování vedlo k jeho kontrakci. Je tedy nutné užít složitější postup.

Jednotlivé segmenty je možné popsat i parametricky pomocí jejich středů $\vec{r}_i$, směrových vektorů $\vec{\xi}_i$ a délky L_i:

$$X_i = \vec{r}_i + t_i \vec{\xi}_i \quad , t_i \in \left\langle -\frac{L_i}{2}; +\frac{L_i}{2} \right\rangle \tag{5.7}$$

Segmenty leží v přímkách, které lze zadat stejně, jen $t_i \in \mathbb{R}$. Posunutí segmentů je provedeno posunutím jejich středů, čímž jsou posunuty i odpovídající přímky. Jejich

průsečíky musí splňovat tuto rovnici:

$$\vec{r}_{i-1} + \vec{v}_{i-1}\,\Delta t + t_{i-1}\,\vec{\xi}_{i-1} = \vec{r}_i + \vec{v}_i\,\Delta t + t_i\,\vec{\xi}_i \tag{5.8}$$

Jelikož jde o rovnici vektorovou, jde o soustavu tří rovnic pro dvě neznámé t_{i-1} a t_i. Nelze obecně určit, které rovnice budou lineárně závislé, nejsnazší je nalézt řešení s využitím Gaussovy eliminační metody. Průsečík je pak možné zjistit třeba dosazením kořene t_i do 5.7.

Je tedy na místě ještě doplnit funkci pro hledání kořenů lineárních rovnic `LinearEquation`.

```
#ifdef __linearequation_min
#undef __linearequation_min
#endif
#define __linearequation_min 1e-18

template<typename FT> void LinearEquation(blitz::Array<FT, 2>& equation,
  int cols) {
  int i, j, k, limit = equation.cols() < cols ? equation.cols() : cols;
  blitz::Array<FT, 1> tmp(equation.cols());
  /* první cyklus */
  for (i = 0; i < limit; i++) {
    for (j = i; j < equation.rows(); j++) {
      if (fabs(equation(j, i)) < __linearequation_min
        && j + 1 < equation.rows()) { /* posuňme nuly na nižší řádky */
        tmp = equation(j, blitz::Range::all());
        equation(j, blitz::Range::all()) = equation(j + 1, blitz::Range::all());
        equation(j + 1, blitz::Range::all()) = tmp;
      }
      if (fabs(equation(j, i)) >= __linearequation_min) { /* prověř znovu */
        equation(j, blitz::Range::all()) /= equation(j, i);
      }
    }
    for (j = i + 1; j < equation.rows(); j++) /* odečti tento řádek od všech dalších */
      if (fabs(equation(j, i)) > __linearequation_min)
        equation(j, blitz::Range::all()) -= equation(i, blitz::Range::all());
  }
  /* horní trojúhelníková matice - lze již vyřešit dosazením */
  for (i = limit - 1; i >= 0; i--) {
    for (j = i - 1; j >= 0; j--)
      for (k = limit; k < equation.cols(); k++) {
        equation(j, k) -= equation(j, i) *
          equation(i, k);
      }
  }
  k = limit < equation.rows() ? limit : equation.rows();
  /* přepsání jednotkovou maticí - jen pro úplnost */
  for (i = 0; i < k; i++)
    for (j = 0; j < k; j++)
      equation(i, j) = i == j;
}
#undef __linearequation_min
```

Funkce `Intersection` najde průsečíky dvou parametricky zadaných přímek.

```
template<typename FT> blitz::Array<FT, 1> Intersection(
  const blitz::Array<FT, 1>& center1, const blitz::Array<FT, 1>& dir1,
  const blitz::Array<FT, 1>& center2, const blitz::Array<FT, 1>& dir2) {
  blitz::Array<FT, 2> equation(3, 3);
  equation = dir1(0), -dir2(0), center2(0) - center1(0),
             dir1(1), -dir2(1), center2(1) - center1(1),
             dir1(2), -dir2(2), center2(2) - center1(2);
  LinearEquation(equation, 2);
  blitz::Array<FT, 1> result(3);
  result = center1 + dir1 * equation(0, 2);
  return result;
}
```

5.3.2 Kontrakce kruhové smyčky

Doplníme-li níže uvedený kód k tomu z části 5.3.1, lze jím počítat kontrakci kruhové smyčky.

```
using namespace std;
using namespace blitz;

/* Funkce vytvářející pravidelný mnohoúhelník se zadaným počtem vrcholů
 * a vepsaným do kružnice zvoleného poloměru. */
template<typename FT> vector<StraightSegment3D<FT> > CreateLoop(
  FT radius, int segments) {
  vector<StraightSegment3D<FT> > line;
  Array<FT, 1> v(3);
  for (int i = 0; i < segments; i++) {
    v = radius * cos(2 * M_PI * i / segments),
        0,
        radius * sin(2 * M_PI * i / segments);
    line.push_back(v.copy());
  }
  line.push_back(StraightSegment3D<FT>(line[0])); /* uzavřená křivka */
  return line;
}

/* hlavní program */
int main() {
  vector<StraightSegment3D<double> > line;
  vector<Array<double, 1> > ncl;
  unsigned int i, j;

  double mu, nu;
  Array<double, 1> burgers(3);
  Array<double, 2> stress(3, 3);
  Array<double, 1> force(3), gforce(3), cforce(3);
  Array<double, 2> base2(3, 3);
  Array<double, 1> tmp(3);
  firstIndex I; secondIndex J;
  double timestep;
  long int N_iterations;

  /* Načteme parametry výpočtu ze standardního vstupu. */
  cout << "mí               : ";
  cin >> mu;
  cout << "ný               : ";
  cin >> nu;
  cout << "Burgersův vektor : ";
  cin >> burgers;
  cout << "Délka kroku      : ";
  cin >> timestep;
  cout << "Počet kroků      : ";
  cin >> N_iterations;

  double D0, Q, T, Omega;
  cout << "D0               : ";
  cin >> D0;
  cout << "Q                : ";
  cin >> Q;
  cout << "T                : ";
  cin >> T;
  cout << "Omega            : ";
  cin >> Omega;

  double radius;
  long int segments;
  cout << "Poloměr smyčky   : ";
  cin >> radius;
  cout << "Počet úseček     : ";
  cin >> segments;
```

```
Array<double, 1> be(3), bs(3), nc(3);

vector<double> spaces;
vector<double>::const_iterator vdi;
double space;
vector<double> forces;

ofstream output;
ostringstream name;

SegmentMotion3D<double> segm(D0, Q, T, Omega, burgers);

/* vytvoření dislokační smyčky */
line = CreateLoop<double>(radius, segments);

long int iterations;
/* hlavní cyklus */
while (iterations < N_iterations) {
  cout << iterations << ". krok" << endl;
  cout << "Zápis souřadnic" << endl;
  name.str("");
  name << "segments" << setw(5) << setfill('0') << iterations;
  output.open(name.str().c_str(), ios::binary | ios::out);
  if (output.good()) {
    for (i = 0; i < line.size(); i++)
      output << line[i].Coordinates()(0) << " "
             << line[i].Coordinates()(1) << " "
             << line[i].Coordinates()(2) << endl;
    output.close();
    if (output.bad()) {
      cout << "Chyba - zápis se nezdařil." << endl;
    }
  } else {
    cout << "Chyba - nelze vytvořit soubor." << endl;
  }

  if (line[0].Coordinates()(0) == line[line.size()-1].Coordinates()(0) &&
    line[0].Coordinates()(1) == line[line.size()-1].Coordinates()(1) &&
    line[0].Coordinates()(2) == line[line.size()-1].Coordinates()(2)) {
    cout << "Výpočet plochy mnohoúhelníka: ";
    space = 0;
    for (i = 1; i < line.size(); i++) {
      space += sqrt(sum(sqr(cross(line[i - 1].Coordinates(), line[i].Coordinates()))));
    }
    cout << space << endl;
    spaces.push_back(space);
  }

  /* výpočet transformační matice */
  cout << "Výpočet transformace" << endl;
  ComputeDLTransformation3D(line, burgers);

  cout << "Pohyb středů" << endl;
  /* Move segment centers according to computed force. */
  ncl.clear();
  forces.clear();
  for (i = 1; i < line.size(); i++) {
    /* Compute force */
    force = GetForce(line, burgers, i, mu, nu);
    forces.push_back(sqrt(sum(sqr(force))));
    /* Compute screw and edge components of Burgers vector. */
    bs = dot(burgers, line[i].Dir()) * line[i].Dir();
    be = burgers - bs;
    /* Compute normal vector to the glide plane. */
    tmp = cross(burgers, line[i].Dir());
    if (sum(sqr(tmp)) < 1e-14)
      tmp = cross(line[i].Dir(), line[i - 1].Dir());
    tmp /= sqrt(sum(sqr(tmp)));
```

```
      /* Compute glide and climb force */
      cforce = dot(force, tmp) * tmp;
      gforce = force - cforce;
      nc = timestep * (gforce * segm.AGlide() +
        cforce * segm.CClimb(sqrt(sum(sqr(be))), sqrt(sum(sqr(bs)))));
      nc += line[i].Center();
      ncl.push_back(nc.copy());
    }
    cout << "Výpočet souřadnic" << endl;
    /* Hledání průsečíků přímek jednotlivých segmentů. */
    for (i = 1; i < line.size(); i++) {
      j = i - 1; /* previous segment */
      if (j < 1) j = line.size() - 1; /* zavedení okrajové podmínky */
      cout << ncl[j-1] << endl;
      cout << line[j].Dir() << endl;
      cout << ncl[i-1] << endl;
      cout << line[i].Dir() << endl;*/
      line[j].Coordinates() = Intersection(ncl[j-1], line[j].Dir(),
        ncl[i-1], line[i].Dir());
    }
    cout << "Zápis sil: " << endl;
    name.str("");
    name << "forces" << setw(5) << setfill('0') << iterations;
    output.open(name.str().c_str(), ios::binary | ios::out);
    if (output.good()) {
      i = 1;
      for (vdi = forces.begin(); vdi != forces.end(); ++vdi)
        output << i++ << " " << *vdi << endl;
      output.close();
      cout << "OK" << endl;
    } else {
      cout << "zápis se nezdařil." << endl;
    }
    cout << "Konec kroku" << endl;
    iterations++;
  }

  cout << "Konec výpočtu. Nyní budou zapsány plochy do souboru: ";
  output.open("spaces", ios::binary | ios::out);
  if (output.good()) {
    i = 1;
    for (vdi = spaces.begin(); vdi != spaces.end(); ++vdi)
      output << i++ << " " << *vdi << endl;
    output.close();
    cout << "OK" << endl;
  } else {
    cout << "zápis se nezdařil." << endl;
  }

  return 0;
}
```

6. VÝSLEDKY

6.1 Parametry a konstanty užité při výpočtu

μ	=	$0{,}8 \cdot 10^{11}\ \mathrm{Nm}^{-2}$	modul pružnosti ve smyku
ν	=	$0{,}3$	Poissonova konstanta
D_0	=	$2 \cdot 10^{-4}\ \mathrm{m}^2\mathrm{s}^{-1}$	difuzní koeficient pro nulové Q
Q	=	$240\ \mathrm{kJ\,mol}^{-1}$	aktivační energie samodifuze
T	=	$873\ \mathrm{K}$	teplota simulovaného děje
Ω	=	$(3{,}5 \cdot 10^{-10}\ \mathrm{m})^3 = 4{,}2875 \cdot 10^{-29}\ \mathrm{m}^3$	objem atomu
$\vec{b}$	=	$(2, 0, 0) \cdot 10^{-10}\ \mathrm{m}$	Burgersův vektor
R	=	$500\ \mathrm{nm}$	poloměr kruhové dislokační smyčky
N	=	22	počet vrcholů mnohoúhelníka
Δt	=	$3\ \mathrm{s}$	délka časového kroku

6.2 Výstup simulace

Byla provedena simulace kontrakce kruhové smyčky s použitím parametrů uvedených v oddíle 6.1. Na obrázku 6.1 lze pozorovat průběh kontrakce, síly působící na jednotlivé segmenty smyčky[1]. a závislost plochy smyčky na čase. Na segmenty šroubového typu, v nichž převládá složka $\vec{b}_s$ Burgersova vektoru $\vec{b}$, působí zpočátku větší síla, kruhová smyčka se proto zplošťuje do elipsy. Bylo zjištěno, že vypočítaná síla vždy leží v rovině smyčky a působí kolmo na dislokační segment.

Zajímavé je též sledovat plochu smyčky. Ta byla vypočítána v každém kroku vztahem, v němž vystupují pouze vektory $\vec{X}_i$ určující polohu vrcholů mnohoúhelníku.

$$S = \sum_{i=1}^{N} |\vec{X}_{i-1} \times X_i| \tag{6.1}$$

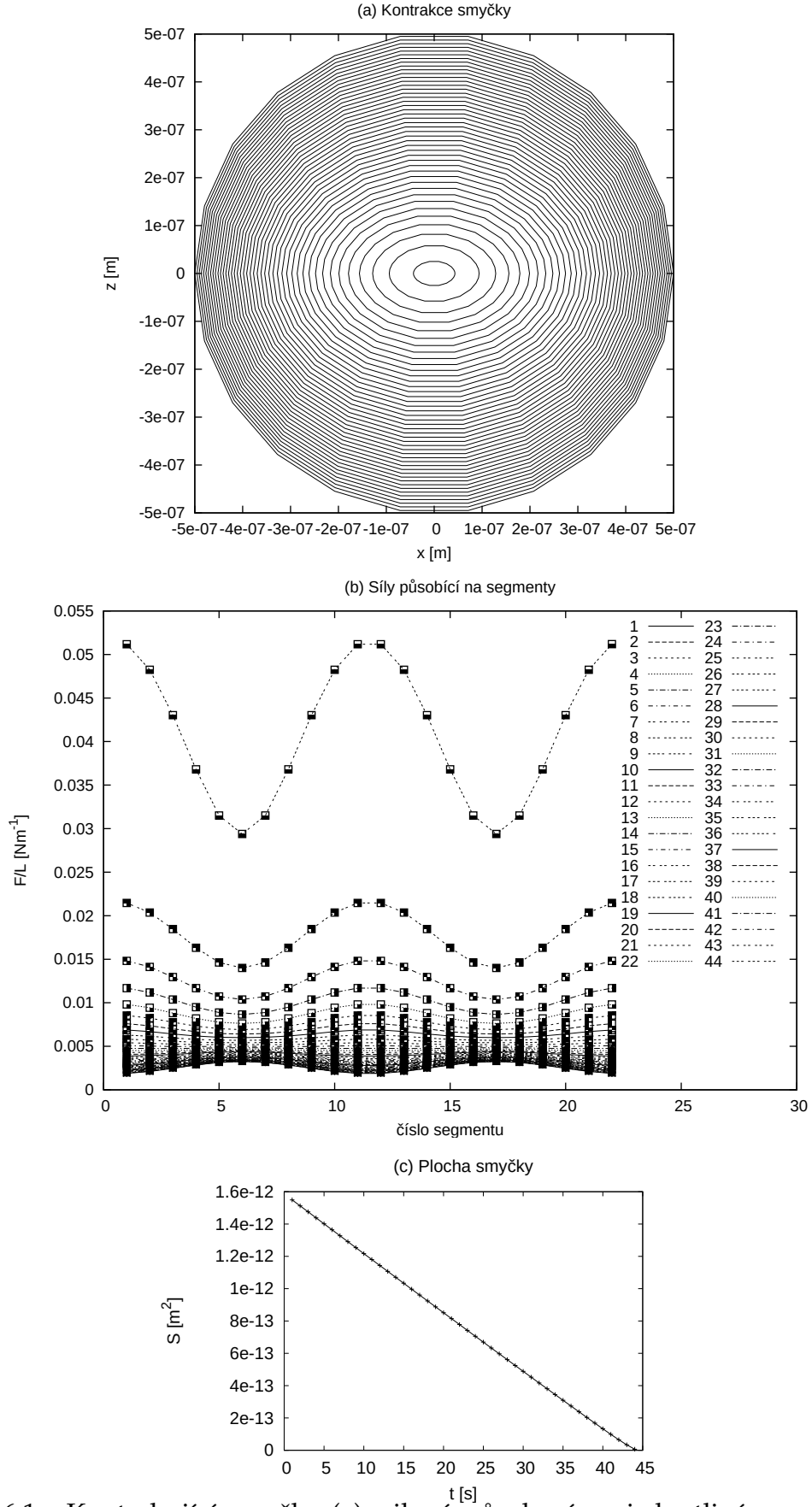
Z obrázku 6.1c je patrné, že plocha s časem lineárně klesá:

$$S(t) = S(0) - Kt \tag{6.2}$$

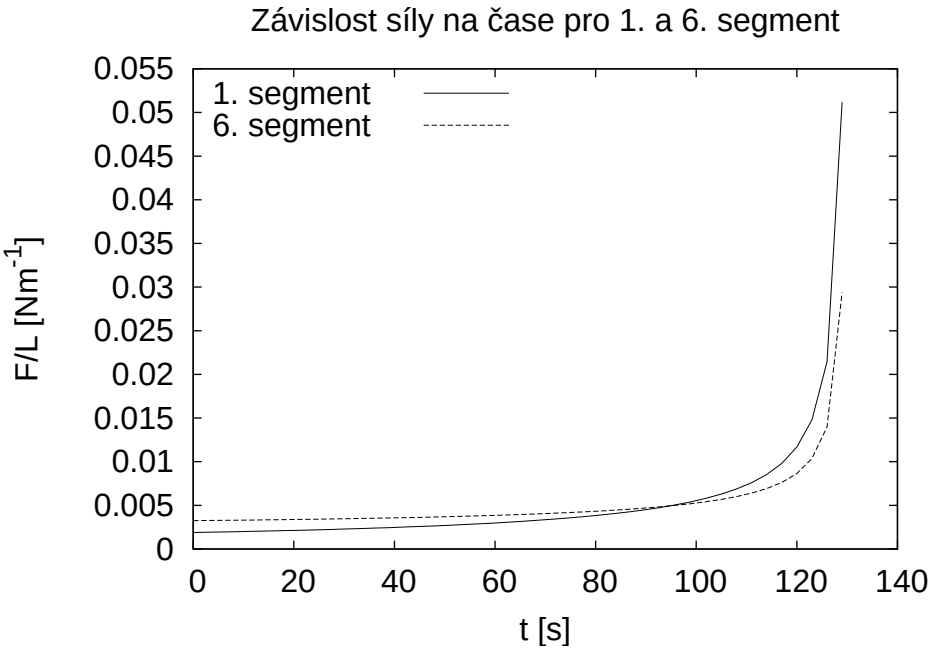
Bylo rovněž zjištěno, že rozložení sil působících na jednotlivé segmenty se s časem mění. Síla působící na šroubové segmenty postupně narůstá, po čase je však převýšena silou působící na segmenty hranové (obr. 6.2).

Výpočet se ovšem po čase stal nestabilním. Po větším počtu kroků došlo ke zkrácení segmentů blízkých k hranové orientaci. Rozdílná délka jednotlivých segmentů může vést k nestabilitě výpočtu, neboť v těchto situacích snadno dochází k posunu segmentů přes sebe, čímž se smyčka poruší a neodpovídá nadále reálné konfiguraci dislokační čáry v krystalu. První možnost, která může tomuto procesu zabránit, je vypuštění vrcholů v oblasti krátkých segmentů, což výpočet sice znepřesní, leč dovolí postoupit o několik

[1] Číslování segmentů je provedeno obdobně, jako na obr. 5.2 podle vztahu 5.1. Z bodu $[R, 0, 0]$ jde nahoru segment č. 1, nahoře je segment č. 6, dole segment č. 17.



Obrázek 6.1: Kontrahující smyčka (a), silové působení na jednotlivé segmenty (síly rostou s klesajícím poloměrem smyčky, na 1. krok připadá spodní křivka) (b), vývoj plochy smyčky (c).



Obrázek 6.2: Průběh sil působících na 1. a 6. segment.

dalších kroků. Druhou, přesnější možnost, představuje užití polynomů 3. stupně (kubických splajnů) k popisu křivky, z nichž lze poté vybrat nové vrcholy v libovolném místě dislokační čáry. Nedocházelo by potom k překřížení segmentů, ke kterému nutně vede použití vzorce 5.8.

Na druhou stranu je třeba poznamenat, že po 44 krocích smyčka zmenšila svoji velikost tak, že hlavní poloosa činila přibližně $a \approx 43$ nm a vedlejší pouze $b \approx 25$ nm. Původní poloměr kružnice činil pro srovnání $R = 500$ nm. V této oblasti přestávají být deformace lineární, což je ovšem nutný předpoklad pro veškeré předcházející výpočty. Při další kontrakci by proto bylo nutné opustit lineární teorii elasticity a pro simulaci použít jiné prostředky [2]. Při poklesu poloměru smyčky pod cca 1 nm by se stále více projevoval diskrétní charakter krystalu a bylo by nutno přejít k metodám kvantové mechaniky.

7. ZÁVĚR

V této práci byl navržen model pohybu obecné dislokační křivky v třírozměrném krystalu. Spojitá dislokační čára byla aproximována lineárními segmenty. Na základě výsledků teorie dislokací uvedených v kapitolách 2 a 3 bylo možné vypočítat napěťová pole působící na jednotlivé segmenty a z nich plynoucí síly. Síly působící na segmenty vstupují do pohybových rovnic a umožňují modelovat pohyb křivočaré dislokace. V 5. kapitole bylo ukázáno obecné numerické řešení, které bylo následně předvedeno na jednoduchém případu rovinné dislokační smyčky. Obecné řešení však umožňuje prozkoumat vývoj složitějších dislokačních linií a zahrnout nejen jejich působení na sebe, ale i působení napětí aplikovaného z vnějšku či interakci s precipitáty nacházejícími se v krystalu.

Dále byla vymezena platnost modelu pouze pro oblast lineární elasticity, v níž je však stále možné dosáhnout dalšího zvýšení přesnosti výpočtu dokonalejším nahrazením obecné křivky segmenty.

LITERATURA

[1] Kratochvíl, P., Lukáč, P., Sprušil, B.: Úvod do fyziky kovů I, SNTL, Praha, 1984

[2] Hirth, J. P., Lothe, J.: Theory of Dislocations (Second Edition), Krieger Publishing Company, 1982.

[3] Čadek, J.: Creep in Metallic Materials, Elsevier, Amsterdam, 1988

[4] Holec, D.: Pohyb dislokačních hranic v precipitačně zpevněných systémech, bakalářská práce, Přírodovědecká fakulta Masarykovy univerzity, Brno, 2003.

[5] Brdička, M., Samek, L., Sopko, B.: Mechanika kontinua, Academia, 2000

[6] Výborný, K., Zahradník, M. a kol.: `http://www.kolej.mff.cuni.cz/~lmotm275/cgi-bin/toASCII.cgi/~lmotm275/skripta/sbirka/karel-milos.pdf` (skripta lineární algebry)