TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Numerické modelování problémů mechaniky kontinua s pomocí výpočetního balíku FEniCS

Bakalářská práce

Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy

Autor práce: **Martin Votýpka**
Vedoucí práce: doc. Ing. Petr Šidlof, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Numerické modelování problémů mechaniky kontinua s pomocí výpočetního balíku FEniCS

Jméno a příjmení: **Martin Votýpka**
Osobní číslo: M17000065
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se se základy mechaniky elastických těles.
2. Nastudujte základy dynamiky tekutin, zejména z oblasti externího proudění (obtékání těles).
3. Seznamte se se základními principy metody konečných prvků, osvojte si jednoduché koncepty programování v C++ nebo Python.
4. Na základě benchmarkového problému specifikovaného v publikaci [4] připravte geometrii a výpočetní síť
5. S pomocí volně dostupné knihovny FEniCS realizujte zvlášť numerickou simulaci deformace elastického nosníku a simulaci obtékání tuhého tělesa ve 2D.
6. Navrhněte možné přístupy, jak do budoucna ve FEniCS sestavit numerickou simulaci sdruženého problému interakce proudění a pružného tělesa.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] White F. M. (2006), Fluid Mechanics, McGraw-Hill.

[2] Brennen C. E. (2006), Internet Book on Fluid Dynamics, Caltech, http://brennen.caltech.edu/fluidbook/content.htm (Online).

[3] Schäfer M. (2006), Computational Engineering Introduction to Numerical Methods, Springer-Verlag.

[4] Turek S., Hron J. (2006), Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow, in: Fluid-Structure Interaction, pp. 371-385, DOI 10.1007/3-540-34596-5_15.

[5] FEniCS Project, https://fenicsproject.org/ (Online).

Vedoucí práce:	doc. Ing. Petr Šidlof, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	9.října 2019
Předpokládaný termín odevzdání:	18.května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 17. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. června 2020 Martin Votýpka

Abstrakt

Hlavním cílem této bakalářské práce je numerická simulace obtékání tělesa tekutinou a simulace deformace elastických těles v prostředí FEniCS. Dlouhodobým cílem je řešení problému interakce proudění s pružnými tělesy. V první části této práce je uveden velmi zjednodušený popis metody konečných prvků a také představení výpočetního balíku FEniCS, který je použit pro numerické řešení parciálních diferenciálních rovnic. Práce se dále zabývá numerickou simulací deformací spojitě zatíženého nosníku a numerickou simulací obtékání tělesa tekutinou. V případě ohybu spojitě zatíženého nosníku jsou výsledky simulací ověřeny pomocí analyticky získaného řešení, v případě simulace obtékaní tělesa tekutinou jsou výsledky ověřeny pomocí benchmarkových dat. V poslední části je uveden možný postup při řešení problému interakce proudění s pružnými tělesy.

Klíčová slova: metoda konečných prvků, FEniCS, numerické simulace, deformace pružných těles, Navierovy-Stokesovy rovnice, interakce proudění a pružného tělesa

Abstract

The main goal of this bachelor's thesis is numerical simulation of fluid flow around a body and a simulation of deformation of elastic bodies in the FEniCS environment. The long-term goal is to solve the problem of fluid-structure interaction. The first part of this work presents a very simplified description of the finite element method and also the introduction of the FEniCS libraries, which are used for the numerical solution of partial differential equations. This thesis further deals with the numerical simulation of deformations of a continuously loaded beam and numerical simulation of the fluid flow around a body. In the case of bending of a continuously loaded beam, the results of simulations are verified using an analytically obtained solution, in the case of simulating the fluid flow around a body, the results of simulations are verified using benchmark data. The last part presents a possible method for solving the problem of fluid-structure interaction.

Keywords: finite element method, FEniCS, numerical simulations, deformation of elastic bodies, Navier-Stokes equations, fluid-structure interaction

Obsah

Seznam obrázků

Seznam použitých zkratek a symbolů

ALE	Arbitrary Lagrangian-Eulerian
Δ	Laplaceův operátor
E	Youngův modul
ϵ	symetrický gradient posunutí, tenzor rychlosti deformace
I	jednotková matice
J	moment setrvačnosti
λ	Lamého konstanta
Mo	ohybový moment
μ	smykový modul, dynamická viskozita
n	vnější jednotková normála
∇	nabla operátor
ν	Poissonovo číslo, kinematická viskozita
p	tlak
Re	Reynoldsovo číslo
ρ	hustota
σ	tensor napětí
t	čas
tr	stopa matice
$\boldsymbol{u}$	vektorové pole posunutí, vektor rychlosti
v	testovací funkce

Úvod

Proudění tekutin i deformace pružných těles jsou témata s širokým spektrem uplatnění. S aplikací proudění tekutin se setkáváme například v meteorologii, při testech aerodynamických vlastností těles, například letadel, nebo při proudění tekutin v potrubích. S pružnými deformacemi se setkáváme u každého stroje. Často dochází také ke sdruženému problému, kdy v důsledku proudění tekutiny dochází k deformaci těles a tyto vzniklé deformace následně ovlivňují proudící tekutinu.

Cílem této práce je numerická simulace pružné deformace spojitě zatíženého nosníku a simulace obtékání tělesa tekutinou. Dlouhodobým cílem, tedy možným navázáním na tuto práci, je simulace sdruženého problému. K těmto simulacím byl využit výpočetní balík FEniCS, který disponuje všemi potřebnými funkcemi pro řešení parciálních diferenciálních rovnic metodou konečných prvků.

V první části se věnuji metodě konečných prvků. Je zde uveden velmi zjednodušený popis metody a následně je představena open-source knihovna FEniCS, obsahující funkce založené na metodě konečných prvků.

Druhá část je věnována pružným deformacím spojitě zatíženého vetknutého nosníku. Nejprve jsou uvedeny rovnice popisující deformace pružného tělesa a zároveň úpravy rovnic pro případ rovinné napjatosti. Následně je ukázán analytický výpočet a kód implementující simulaci ohybu spojitě zatíženého nosníku. Nakonec jsou srovnány výsledky získané ze simulací při využití různých výpočetních sítí s analyticky získaným výsledkem. Geometrie i vlastnosti materiálu nosníku odpovídají hodnotám uvedeným v benchmarku [1].

Třetí část je věnována simulaci obtékání tělesa nestlačitelnou vazkou tekutinou. Nejprve jsou představeny rovnice popisující proudění nestlačitelné vazké tekutiny a jejich následné úpravy. Poté jsou uvedeny parametry simulací, které se shodují s parametry v benchmarku, a také výpočetní sítě použité při numerických simulacích. Následně je popsán kód implementující výpočet pomocí knihoven FEniCS. Nakonec je uvedeno srovnání dosažených výsledků s výsledky v benchmarku.

Ve čtvrté a poslední části je uveden možný postup při řešení sdruženého problému, tedy interakce proudění s pružnými tělesy.

1 Metoda konečných prvků

Metoda konečných prvků je numerická metoda pro řešení parciálních diferenciálních rovnic. V současné době nachází využití v širokém spektru aplikací jako je například meteorologie či simulace deformace namáhaných částí strojů. Její princip spočívá v diskretizaci spojité oblasti a převedení diferenciálních rovnic na soustavu lineárních rovnic. Informace použité v této kapitole byly získány z [4] a [8].

Princip metody konečných prvků lze demonstrovat například na Laplaceově rovnici ve 2D

$$-\Delta u = f \quad \text{v} \quad \Omega \quad , \tag{1}$$

s okrajovými podmínkami

$$u = 0 \quad \text{na} \quad \partial\Omega \quad . \tag{2}$$

Nejprve je nutné získat slabou formulaci. Rovnici (1) násobíme libovolnou funkcí v s hodnotou rovnou nule na hranicích oblasti. Výsledná rovnice po úpravě je

$$-\Delta u v = f v \quad . \tag{3}$$

Dále bude potřeba rovnici (3) zintegrovat

$$-\int_\Omega \Delta u v \, dx = \int_\Omega f v \, dx \quad . \tag{4}$$

Pomocí Greenovy věty

$$\int_\Omega f \frac{\partial g}{\partial x_i} = \int_{\partial\Omega} f g n_i - \int_\Omega g \frac{\partial f}{\partial x_i} \tag{5}$$

je možné dále rovnici upravit do tvaru

$$\sum_{i=1}^{2} \left(\int_{\partial\Omega} v \left(-\frac{\partial u}{\partial x_i} \right) n_i - \int_\Omega \left(\frac{\partial v}{\partial x_i} \right) \left(-\frac{\partial u}{\partial x_i} \right) \right) = \int_\Omega f v \quad . \tag{6}$$

Jelikož má v tomto případě funkce v nulovou hodnotu na hranicích oblasti, je výsledek křivkového integrálu vždy rovný nule a je tedy možné rovnici dále upravit do tvaru

$$\int_\Omega \nabla v \cdot \nabla u \, dx = \int_\Omega f v \, dx \quad . \tag{7}$$

Následně provedeme aproximaci slabé formulace. Budeme hledat po částech lineární funkci u_h s hodnotou rovnou nule na hranicích oblasti tak, aby

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, dx = \int_\Omega f \, v_h \, dx \quad , \tag{8}$$

kde v_h jsou po částech lineární funkce splňující okrajovou podmínku.

Funkce v_h a u_h jsou elementy prostoru

$$V_h = \{ v_h : \Omega \to \mathbb{R} \} \quad . \tag{9}$$

V_h je prostor s konečnou dimenzí, jehož bázi lze zapsat jako

$$\{ \varphi_1, \ldots, \varphi_n \} \quad , \tag{10}$$

kde n je dimenze prostoru V_h (pro lineární prvky rovný počtu vnitřních uzlů výpočetní sítě). Funkci u_h je možné vyjádřit jako

$$\sum_{j=1}^{n} \alpha_j \varphi_j \quad , \tag{11}$$

kde $\alpha_1, \ldots, \alpha_n$ jsou neznámé.

Výsledkem této aproximace je soustava lineárních rovnic

$$\begin{gathered} \sum_{j=1}^{n} \alpha_j a_{ij} = b_i \quad , \quad i = 1, \ldots, n \quad \text{, kde} \\ a_{ij} = \int_\Omega \nabla \varphi_j \cdot \nabla \varphi_i \, dx \quad , \quad b_i = \int_\Omega f \, \varphi_i \, dx \quad . \end{gathered} \tag{12}$$

Soustavu je možné také zapsat maticově jako

$$\begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ldots & \vdots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \quad . \tag{13}$$

1.1 FEniCS Project

FEniCS je výpočetní balík pro řešení parciálních diferenciálních rovnic, který uživatelům umožňuje rychlé převedení rovnic do kódu konečných prvků. Je možné jej použít jak na běžných pracovních počítačích, tak na výkonných strojích využívajících paralelního výpočtu. FEniCS byl původně vytvořen v roce 2003 a je vyvíjen za účasti univerzit a výzkumných ústavů po celém světě. Bližší informace ohledně knihoven FEniCS, jejich instalaci a použití lze nalézt v [4].

Přímo lze FEniCS instalovat pouze v operačním systému Linux. Pokud má být FEniCS nainstalován na zařízení s jiným operačním systémem, je nutné využít dalších programů, jako je například Docker, který instalaci a následné používání knihovny FEniCS umožní.

Po úspěšné instalaci balíku FEniCS je možné přistoupit k tvorbě zdrojového kódu. Ten je možné psát buď v programovacím jazyku C++ a nebo v jazyku Python. Osobně jsem pro tvorbu zdrojového kódu zvolil Python. Učinil jsem tak hlavně z důvodu většího množství materiálů, ze kterých bylo možné čerpat potřebné informace.

2 Numerická simulace ohybu spojitě zatíženého nosníku

2.1 Rovnice popisující deformace pružného tělesa

V této kapitole jsou uvedeny rovnice popisující deformace pružného tělesa a úprava rovnic pro případ rovinné napjatosti. Informace byly čerpány z [6] a [4].

Pružné těleso je takové těleso, které se po odstranění vnějších působících sil navrátí do původního stavu. Jeho deformace jsou popsány pomocí rovnice

$$-\nabla\cdot\sigma=f \quad , \tag{14}$$

kde f je síla působící na jednotku objemu deformovaného tělesa a σ je tenzor napětí. Jelikož se zde jedná o případ přímého tenkého nosníku, je možné zanedbat kolmou složku napětí a uvažovat případ rovinné napjatosti, čímž dojde k zjednodušení výsledných rovnic. Pro případ rovinné napjatosti lze tenzor napětí σ zapsat jako

$$\sigma=\begin{bmatrix}\sigma_{11} & \sigma_{12} & 0\\ \sigma_{21} & \sigma_{22} & 0\\ 0 & 0 & 0\end{bmatrix} \quad . \tag{15}$$

Tento tenzor je dán vztahem

$$\sigma=\lambda\cdot tr(\epsilon)\cdot I+2\mu\cdot\epsilon \quad , \tag{16}$$

kde symbol λ a μ jsou Lamého konstanty definované jako

$$\lambda=\frac{E\cdot\nu}{(1+\nu)(1-2\nu)} \quad \text{a} \quad \mu=\frac{E}{2(1+\nu)} \quad , \tag{17}$$

symbol I ve vztahu (16) je jednotková matice a ϵ je symetrický gradient vektorového pole posunutí definovaný pro případ rovinné napjatosti jako

$$\epsilon=\begin{bmatrix}\epsilon_{11} & \epsilon_{12} & 0\\ \epsilon_{21} & \epsilon_{22} & 0\\ 0 & 0 & \epsilon_{33}\end{bmatrix} \quad . \tag{18}$$

Symetrický gradient posunutí je dán vztahem

$$\epsilon(\boldsymbol{u})=\frac{1}{2}(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^{T}) \quad , \tag{19}$$

kde $\boldsymbol{u}$ je vektorové pole posunutí.

Celý problém může být následně zredukován pouze na dva rozměry. Nelze však ignorovat vliv parametru ϵ_{33} . Aby byla redukce možná, je nutné rovnici (16) upravit.

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 \\ \sigma_{21} & \sigma_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \lambda \cdot (\epsilon_{11}+\epsilon_{22}+\epsilon_{33}) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 2\,\mu \cdot \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & 0 \\ \epsilon_{21} & \epsilon_{22} & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix}$$

$$\lambda \cdot (\epsilon_{11}+\epsilon_{22}+\epsilon_{33}) + 2\,\mu \cdot \epsilon_{33} = 0 \tag{20}$$

$$\epsilon_{33} = -\frac{\lambda}{\lambda+2\,\mu}(\epsilon_{11}+\epsilon_{22})$$

Dosazením výsledku úpravy (20) do rovnice (16) lze následně získat

$$\sigma = \lambda \cdot (\epsilon_{11}+\epsilon_{22}-\frac{\lambda}{\lambda+2\,\mu}(\epsilon_{11}+\epsilon_{22})) \cdot I + 2\,\mu \cdot \epsilon$$

$$\sigma = \lambda \cdot ((1-\frac{\lambda}{\lambda+2\,\mu})(\epsilon_{11}+\epsilon_{22})) \cdot I + 2\,\mu \cdot \epsilon \tag{21}$$

$$\sigma = \left(\frac{2\,\mu \cdot \lambda}{\lambda+2\,\mu}\right)(\epsilon_{11}+\epsilon_{22}) \cdot I + 2\,\mu \cdot \epsilon \quad .$$

Při označení

$$\lambda^* = \left(\frac{2\,\mu \cdot \lambda}{\lambda+2\,\mu}\right) \tag{22}$$

je výsledná rovnice pro dvourozměrný problém

$$\sigma = \lambda^* \cdot tr(\epsilon) \cdot I + 2\,\mu \cdot \epsilon \quad . \tag{23}$$

2.2 Parametry výpočtu a simulace

2.2.1. Geometrie nosníku a zvolené konstanty

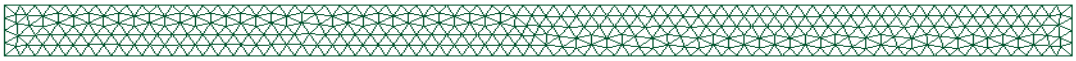
Parametry simulace ohybu spojitě zatíženého nosníku vycházejí z parametrů daných v benchmarku [1]. Jedná se o nosník s délkou 35 centimetrů, výškou 2 centimetry, vyrobený z polypropylenu.

Tabulka 1: hodnoty parametrů výpočtu a simulace ohybu spojitě zatíženého nosníku

Délka	$L = 0{,}35$ m
Výška	$H = 0{,}02$ m
Youngův modul	$E = 9 \cdot 10^8 \ \frac{\text{kg}}{\text{ms}^2}$
Hustota	$\rho = 1100 \ \frac{\text{kg}}{\text{m}^3}$
Poissonovo číslo	$\nu = 0{,}42$

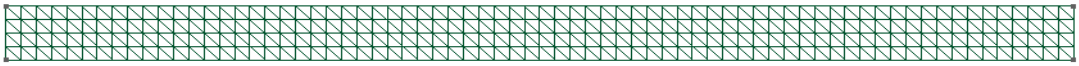
2.2.2. Výpočetní sítě

Při vlastních simulacích pružných deformací nosníku jsem využil dvě výpočetní sítě, každou s jiným uspořádáním prvků.



Obrázek 1: nestrukturovaná síť (858 prvků)

První síť použitá pro simulace byla nestrukturovaná síť, tedy síť s nepravidelným rozložením jednotlivých prvků. Výslednou nestrukturovanou síť je možné vidět na obrázku 1.



Obrázek 2: strukturovaná síť (712 prvků)

Druhá použitá síť byla strukturovaná, tedy síť s pravidelným uspořádáním prvků, kterou je možné vidět na obrázku 2.

Různé výpočetní sítě pro simulaci ohybu nosníku byly použity pro následné porovnání obdržených výsledků.

2.3 Analytický výpočet deformace nosníku

Při výpočtu průhybu spojitě zatíženého nosníku budu vycházet ze vzorce

$$w''(x)=\frac{Mo(x)}{E\cdot Jy} \quad , \tag{24}$$

kde $w(x)$ je průhybová křivka, $Mo(x)$ je ohybový moment, E je Youngův modul pružnosti a Jy je moment setrvačnosti vzhledem k ose y definovaný jako

$$Jy=\int_{-\frac{b}{2}}^{\frac{b}{2}}\int_{-\frac{h}{2}}^{\frac{h}{2}} y^2\,dy\,dz=\int_{-\frac{b}{2}}^{\frac{b}{2}}\frac{h^3}{12}dz=\frac{b\cdot h^3}{12} \quad , \tag{25}$$

kde b je šířka nosníku a h je výška nosníku.

Ohybový moment $Mo(x)$ je definovaný pro spojitě zatížený nosník

$$Mo(x)=m\cdot g\left(x-\frac{x^2}{2L}-\frac{L}{2}\right) \quad , \tag{26}$$

kde $m\cdot g$ je působící zatížení a L je délka nosníku.

Rovnici (24) je následně možné dvakrát zintegrovat, čímž se získá rovnice

$$w(x)=\frac{m\cdot g}{E\cdot Jy}\left(\frac{-x^4}{24L}+\frac{x^3}{6}-\frac{L\cdot x^2}{4}\right) \quad . \tag{27}$$

Při výpočtu maximálního průhybu, tedy průhybu na volném konci nosníku, je možné rovnici (27) dále upravit do tvaru

$$\frac{L\cdot h\cdot b\cdot \rho\cdot g}{E\cdot\frac{1}{12}b\cdot h^3}\cdot\left(-\frac{3}{24}L^3\right)=-\frac{3}{2}\frac{\rho\cdot g\cdot L^4}{E\cdot h^2} \quad . \tag{28}$$

Po dosazení zadaných parametrů je výsledný průhyb na volném konci nosníku

$$w(0{,}35)\approx -0{,}000674719 \text{ m}. \tag{29}$$

Vzorec (24) a jeho následné úpravy byly získány z [7].

2.4 Implementace ve FEniCSu

V této kapitole bude popsán kód implementující výpočet pod vlastní vahou deformovaného pružného nosníku. Velká část kódu byla převzata z [6]. Informace ohledně použitých funkcí byly čerpány z [3] a [6]. Parametry použité při simulaci lze nalézt v kapitole 2.2.

Aby bylo možné pro výpočet diferenciálních rovnic použít funkce FEniCSu, je nejprve nutné knihovnu FEniCS importovat.

```
from fenics import *
```

Knihovna *fenics* obsahuje všechny potřebné funkce pro výpočet parciálních diferenciálních rovnic, založených na metodě konečných prvků. Po jejím importování již bude možné potřebné funkce využít.

Dále bude potřeba nadefinovat konstanty použité při výpočtu.

```
L = 0.35
H = 0.02
E = Constant(9e8)
nu = Constant(0.42)
rho = 1100
g = 9.81
rho_g = rho*g
f = Constant((0,-rho_g))
mu = E/2/(1+nu)
lmbda = E*nu/(1+nu)/(1-2*nu)
lmbda = 2*mu*lmbda/(lmbda+2*mu)
```

Proměnná *L* je délka nosníku, *H* je výška nosníku, konstanta *E* je Youngův modul v pružnosti a *nu* je Poissonovo číslo. Proměnná *rho* je hustota, *g* je gravitační zrychlení a konstanta *f* je zatížení nosníku. mu a lmbda jsou Lamého konstanty odpovídající předpisu (17). Z odvození v kapitole 2.1 je zřejmé, že aby mohla být provedena redukce problému čistě na dva rozměry, je nutné rovnici upravit do tvaru (23). Výsledná hodnota proměnné lmbda odpovídá hodnotě koeficientu λ^* .

Následně je třeba načíst, případně vytvořit výpočetní síť. Při načítání již vytvořené sítě je nutné, aby síť byla uložena ve formátu *xml*. Pokud je síť uložena v jiném formátu, je nutné nejprve provést konverzi. K tomu je možné využít nástroj *dolfin-convertr*, který je implementován v knihovnách FEniCS. Pokud síť není předem vytvořena, je možné ji vytvořit přímo ve FEniCS. Tímto způsobem lze vytvořit jednodušší sítě. Aby mohla být síť vytvořena pomocí knihoven FEniCS, je nejprve nutné importovat komponentu *mshr*, která potřebné funkce pro vytvoření sítě obsahuje.

```
mesh = Mesh("beam.xml")
```

Ve chvíli, kdy je načtena, případně vytvořena výpočetní síť, je možné přistoupit k definici funkčního prostoru a následně bázové a testovací funkce.

```
V = VectorFunctionSpace(mesh, 'Lagrange', degree=2)
u_ = TrialFunction(V)
v_ = TestFunction(V)
```

Funkce *VectorFunctionSpace* přebírá tři vstupní parametry. První parametr je výpočetní síť, druhý parametr značí typ prvku a třetí parametr je stupeň prvku. Proměnná *V* je tedy vektorový funkční prostor s kvadratickými prvky typu *Lagrange*. Funkce *TrialFunction* vytvoří bázovou funkci daného funkčního prostoru. Funkce *TestFunction* vytvoří testovací funkci daného funkčního prostoru.

Aby byl program snáze čitelný je dobré předem nadefinovat dlouhé a často používané výrazy. Z tohoto důvodu je výpočet symetrického gradientu posunutí (19) a tenzoru napětí (16) realizován jako funkce.

```
def epsilon(u):
	return 0.5*(nabla_grad(u)+nabla_grad(u).T)
```

Funkce *nabla_grad* je jednou z funkcí obsažených v knihovnách FEniCS. Vstupním parametrem je vektor, výstupním pak jeho gradient. Parametr T značí transponovanou matici gradientu vstupního vektoru.

```
def sigma(u):
	return lmbda*tr(epsilon(u))*Identity(2) + 2.0*mu*epsilon(u)
```

lmbda je již definovaná proměnná odpovídající hodnotě koeficientu (22). Funkce *tr* vrací stopu vstupní čtvercové matice, tedy součet prvků na hlavní diagonále. Funkce *Identity* vytvoří jednotkovou matici o velikosti závislé na vstupním parametru. V tomto případě se jedná o matici 2x2. Funkce *epsilon* je funkce pro výpočet tenzoru rychlosti deformace (16) popsaná výše.

Dále bude potřeba definovat hranice oblasti po pozdější definici okrajových podmínek. V případě deformace vetknutého nosníku bude definován jeden okraj v místě vetknutí.

```
def left(x, on_boundary):
	return near(x[0],0.)
```

Funkce *near* vrací booleovskou hodnotu udávající, zda se hodnota prvního parametru dostatečně blíží hodnotě druhého. Parametr *x[0]* značí souřadnice na ose x.

Po definování okraje je možné definovat okrajovou podmínku.

```
bc = DirichletBC(V, Constant((0.,0.)), left)
```

Funkce DirichletBC je funkce pro definici Dirichletovy okrajové podmínky. Prvním argumentem je funkční prostor, druhý argument je popis chování funkce na okraji a třetí argument je okraj.

Po definování všech potřebných konstant, vztahů a okrajových podmínek je možné přistoupit k výpočtu. Nejprve je nutné získat slabou formulaci. Tu lze získat vynásobením rovnice (14) testovací funkcí, integrací a následnou aplikací Greenovy věty a okrajové podmínky. Rovnice po této úpravě je ve tvaru

$$\int_{\Omega} \sigma(u) \cdot \nabla v \, dx = \int_{\Omega} f \, v \, dx \quad , \tag{30}$$

kde $\sigma(u)$ je tenzor napětí, v je testovací funkce, a f je zatížení nosníku. Jelikož je tenzor $\sigma(u)$ symetrický, tak po skalárním součinu s tenzorem ∇v budou mít vliv na výsledek pouze symetrické části tenzoru ∇v . Rovnici (30) je tedy dále možné upravit do tvaru

$$\int_{\Omega} \sigma(u) \cdot \epsilon(v) dx = \int_{\Omega} f \, v \, dx \quad , \tag{31}$$

kde $\epsilon(v)$ je symetrická část gradientu funkce v . Implementace této rovnice ve FEniCSu vypadá následovně:

```
a = inner(sigma(u_), epsilon(v_))*dx
l = dot(f, v_)*dx
```

Aby mohla být rovnice vyřešena, musí nejdříve být rozdělena na pravou a levou stranu. Proměnná *a* značí levou stranu rovnice (31), proměnná l pravou stranu rovnice (31). Funkce *inner* je funkce pro skalární součin tenzorů, funkce *sigma* a *epsilon* jsou již definované funkce popsané výše. Proměnná *u_* je bázová funkce, proměnná *v_* je testovací funkce. Parametr *dx* je parametr značící integraci. Funkce *dot* je funkcí pro skalární součin vektorů a proměnná *f* je zatížení nosníku.

Následně je již možné pomocí funkcí knihovny *fenics* rovnici vyřešit.

```
u = Function(V, name="Displacement")
solve(a == l, u, bc)
```

Proměnná *u* je funkce, do které bude uložen výsledek simulace. *function* vytvoří funkci v závislosti na vstupním parametru. Vstupní parametr může být funkční prostor, nebo jiná funkce. Pokud je funkce tvořena z jiné funkce, je možné přidat další parametr určující číslo dílčí funkce, která má být extrahována. Parametr *name* je název funkce, který umožní snazší práci při zpracování výsledků. K řešení rovnice je využita funkce *solve*. Tato funkce má celkem čtyři různé způsoby použití. V tomto případě bude řešena rovnice $a = l$, kde *a* je levá strana rovnice (31) a *l* pravá strana. Funkce *u* bude výsledkem výpočtu a *bc* jsou okrajové podmínky.

Po dokončení výpočtu bude možné výsledky uložit a dále zpracovat. Aby mohly být uloženy hodnoty napětí je nejprve nutné promítnout tenzor napětí do vhodného funkčního prostoru.

```
Vsig = TensorFunctionSpace(mesh, "DG", degree=0)
sig = Function(Vsig, name="Stress")
sig.assign(project(sigma(u), Vsig))
```

Funkce *TensorFunctionSpace* je funkce pro vytvoření tenzorového funkčního prostoru. První vstupní parametr funkce je výpočetní síť, druhý parametr je typ prvků, třetí parametr je stupeň prvků. Funkce *project* vrací projekci funkce či předpisu v prvním parametru do prostoru konečných prvků v parametru druhém. Výsledek projekce je nakonec přiřazen do proměnné *sig* pomocí funkce *assign*.

Výsledky jsou nakonec pro další zpracování uloženy.

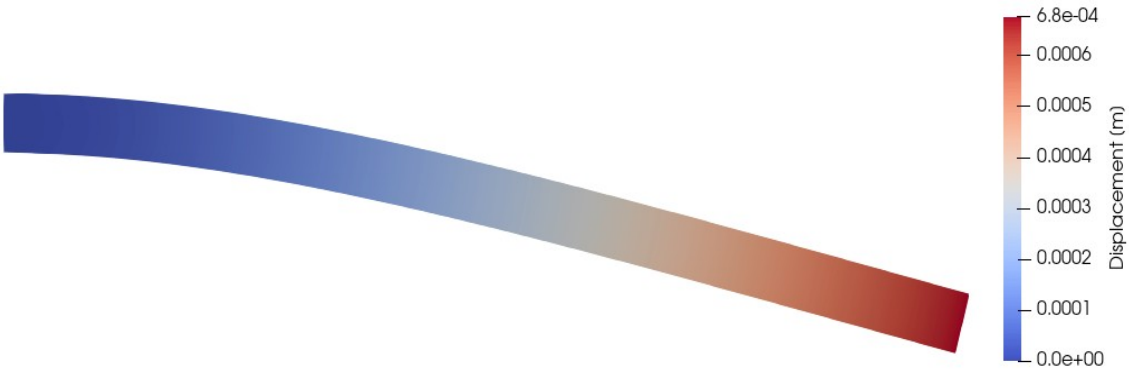
```
file_results = XDMFFile("elasticity_results.xdmf")
file_results.parameters["flush_output"] = True
file_results.parameters["functions_share_mesh"] = True
file_results.write(u, 0.)
file_results.write(sig, 0.)
```

2.5 Výsledky numerických simulací

Při simulacích průhybu nosníku byly použity dvě sítě s odlišným uspořádáním prvků. Výsledky získané při použití jednotlivých sítí se od sebe mírně liší. Výsledné hodnoty průhybu nosníku získané při simulacích a z analytického výpočtu je možné najít v tabulce 2. Grafické zpracování výsledků simulací je možné vidět na obrázku 3.

Tabulka 2: Výsledné hodnoty průhybu na volném konci nosníku

Nestrukturovaná síť:	0,0006752933 m
Strukturovaná síť:	0,0006750631 m
Analytický výpočet:	0,0006747185 m



Obrázek 3: průhyb spojitě zatíženého nosníku

Výsledky numerických simulací souhlasí s analyticky získaným výsledkem. Hodnota vychýlení nosníku na jeho konci se při využití nestrukturované sítě liší od analyticky získaného výsledku o necelých 0,09%. U strukturované sítě se výsledná hodnota simulace od analytického výsledku liší o přibližně 0,05%. Ze získaných hodnot je patrné, že byť strukturovaná síť obsahuje nižší počet prvků a šlo by ji tedy považovat za méně detailní, tak díky vhodném uspořádání prvků vzhledem k řešenému problému podává v tomto případě přesnější výsledky než síť nestrukturovaná.

3 Numerická simulace obtékání tělesa tekutinou

3.1 Navierovy-Stokesovy rovnice

S jistou formou proudění se každý z nás setkává denně a není tedy divu, že se o popsání tohoto jevu vědci dlouhá léta snažili. První, kdo rovnice popisující proudění vazké nestlačitelné tekutiny odvodil, byl Claude Louis Navier a o několik let později George Gabriel Stokes. Navierovy-Stokesovy rovnice jsou nejznámější model proudění nestlačitelné viskózní tekutiny a mají velmi široké využití od zkoumání počasí po proudění tekutin v potrubí [5].

Navierovy-Stokesovy rovnice vyplývají z užití Newtonova druhého zákona. Ten lze pro proudící tekutinu zapsat ve tvaru (viz [2], [4] a [5])

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=f+\nabla\cdot\sigma(\boldsymbol{u},p) \quad , \tag{32}$$

Kde symbol ρ je hustota tekutiny, symbol $\boldsymbol{u}$ je vektor rychlosti, t je čas, f představuje objemovou sílu, která je často volena rovna nule a $\sigma(\boldsymbol{u},p)$ je tensor napětí, který je pro Newtonovské kapaliny dán vztahem

$$\sigma(\boldsymbol{u},p)=2\mu\epsilon(\boldsymbol{u})-pI \quad . \tag{33}$$

Symbol I značí jednotkovou matici, symbol μ je dynamická viskozita proudící tekutiny a symbolem $\epsilon(\boldsymbol{u})$ je označen tensor rychlosti deformace, který je dán vztahem

$$\epsilon(u)=\frac{1}{2}\left(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^{T}\right) \tag{34}$$

neboli

$$\epsilon(u)=\frac{1}{2}\left(\begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{bmatrix}+\begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix}\right) \quad . \tag{35}$$

Po dosazení vztahu (34) do rovnice (33), lze tenzor napětí zapsat

$$\sigma(\boldsymbol{u},p)=\mu\left(\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^{T}\right)-pI \quad . \tag{36}$$

Výsledný tvar tenzoru napětí (36) je poté možné dosadit do rovnice (32), čímž se získá vektorový tvar Navierových-Stokesových rovnic.

Vektorový tvar Navierovy-Stokesovy rovnice je

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \nu \Delta \boldsymbol{u} - \frac{1}{\rho} \nabla p + f \quad , \tag{37}$$

kde symbol ν je kinematická viskozita

$$\nu = \frac{\mu}{\rho} \quad . \tag{38}$$

Neznámými v rovnici (37) jsou rychlost $\boldsymbol{u}$ a tlak p. Soustavu je nutné doplnit o rovnici kontinuity, která vyjadřuje zákon zachování hmoty. V diferenciálním tvaru lze rovnici kontinuity zapsat

$$\nabla \cdot \boldsymbol{u} = 0 \quad . \tag{39}$$

Výsledná soustava rovnic popisuje proudění nestlačitelné newtonovské tekutiny, což znamená, že libovolná část tekutiny zachovává svůj objem [5].

Soustavu rovnic (37) a (39) je nutné ještě doplnit o počáteční a okrajové podmínky.

Jako počáteční podmínku volíme hodnotu rychlosti a tlaku v počátečním čase. Tato hodnota je často volena rovna nule.

Okrajové podmínky lze charakterizovat jako popis hledané funkce v mezních bodech. Nejčastější předepisovanou okrajovou podmínkou je Dirichletova okrajová podmínka (okrajová podmínka prvního typu)

$$v = g \quad , \tag{40}$$

kde v je neznámá funkce a g je známá skalární funkce udávající chování funkce neznámé na okrajích (často $g = 0$).

Dalším příkladem okrajové podmínky je Neumannova okrajová podmínka (okrajová podmínka druhého typu)

$$\frac{\partial v}{\partial n} = g \quad , \tag{41}$$

kde n je vnější jednotková normála k hranici oblasti.

Další tvar Navierových-Stokesových rovnic, který bude v této práci uveden, je tzv. bezrozměrný tvar. Ten je možné získat zavedením parametru charakteristického rozměru L a parametru charakteristické rychlosti U a následnou úpravou jednotlivých členů rovnice na bezrozměrné proměnné [2]

$$u' = \frac{u}{U} \qquad \nabla' = L\nabla$$

$$x' = \frac{x}{L} \qquad y' = \frac{y}{L} \qquad z' = \frac{z}{L} \quad . \tag{42}$$

$$t' = \frac{t\,U}{L} \qquad p' = \frac{p}{\rho \cdot U^2}$$

Výsledné bezrozměrné rovnice jsou

$$\nabla' \cdot u' = 0$$

$$\frac{\partial u'}{\partial t'} = -\nabla' p' + \frac{1}{Re}\nabla'^2 u' \quad , \tag{43}$$

kde parametr Re je Reynoldsovo číslo dané vztahem

$$Re = \frac{\rho\, U\, L}{\mu} \quad . \tag{44}$$

Reynoldsovo číslo je bezrozměrný parametr, který charakterizuje chování proudící tekutiny. Pomocí znalosti Reynoldsova čísla lze usuzovat, zda bude proudění laminární či turbulentní. Hodnota Reynoldsova čísla, při které dochází k přechodu z laminárního proudění do turbulentního, se nazývá kritická hodnota. Tato hodnota se pro různé druhy uspořádání liší a zjišťuje se experimentálně.

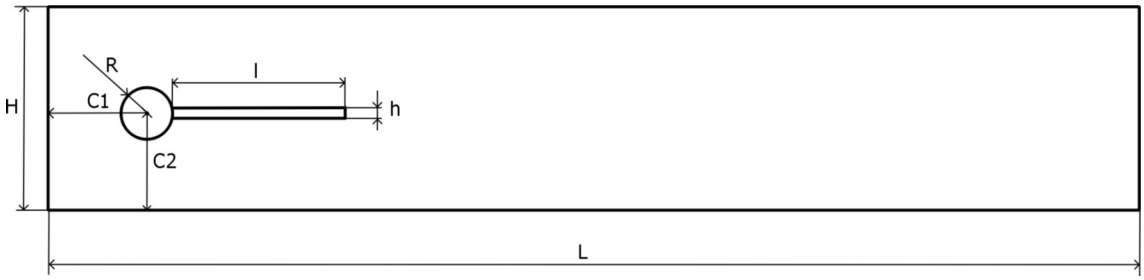
Reynoldsovo číslo má velký význam při studiu proudění. U některých těles, jako jsou například letadla, není možné kvůli rozměru tělesa měřit působení odporových sil a je nutné tyto síly získat z modelu. Získané hodnoty budou použitelné pro originál za předpokladu, že Reynoldsovo číslo zůstane stejné [9].

3.2 Parametry simulace

Geometrie výpočetní oblasti a okrajové podmínky jsou nastaveny podle specifikace benchmarkové úlohy [1].

3.2.1. Geometrie oblasti

Geometrie oblasti, ve které tekutina proudí, je dvourozměrný kanál šířky 0,41 metrů, délky 2,5 metrů. V kanálu se na souřadnicích 0,2 metrů a 0,2 metrů nachází překážka válcového tvaru s průměrem 0,05 metrů s uprostřed vetknutým nosníkem délky 0,35 metrů a šířky 0,02 metrů. Výslednou oblast je možné vidět na obrázku 4. Hodnoty všech parametrů jsou v tabulce 3.



Obrázek 4: geometrie oblasti

Tabulka 3: hodnoty parametrů simulace obtékání tělesa tekutinou

Šířka kanálu	$H = 0{,}41$ m
Délka kanálu	$L = 2{,}5$ m
Poloměr válce	$R = 0{,}05$ m
Poloha středu válce na vodorovné ose	$C1 = 0{,}2$ m
Poloha středu válce na svislé ose	$C2 = 0{,}2$ m
Šířka nosníku	$h = 0{,}02$ m
Délka nosníku	$l = 0{,}35$ m

3.2.2. Parametry proudící tekutiny

Jako proudící tekutina byly zvolena tekutina s hustotou $\rho = 1000\ \frac{\mathrm{kg}}{\mathrm{m}^3}$ a kinematickou viskozitou $\nu = 0{,}001\ \frac{\mathrm{m}^2}{\mathrm{s}}$. Jednotlivé parametry lze vidět v tabulce 4.

Tabulka 4: parametry proudící tekutiny

Hustota proudící kapaliny	$\rho = 1000\ \frac{\mathrm{kg}}{\mathrm{m}^3}$
Kinematická viskozita	$\nu = 0{,}001\ \frac{\mathrm{m}^2}{\mathrm{s}}$

3.2.3. Okrajové podmínky

Jako rychlostní profil na vstupu byl zvolen kvadratický profil s maximem ve středu kanálu popsaný vztahem

$$u(0, y) = 3\frac{y(H-y)}{\left(\frac{H}{2}\right)^2} \quad . \tag{45}$$

Pro zadanou šířku kanálu H je výsledná rychlost na vstupu

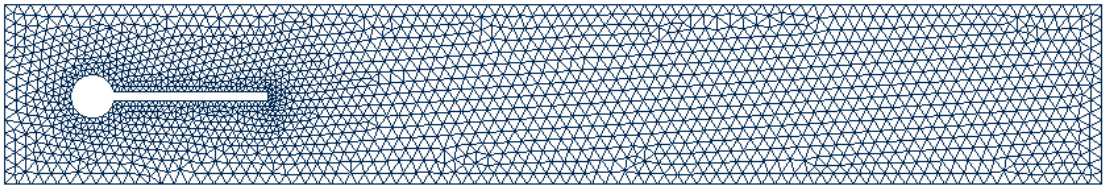
$$u(0, y) = \frac{12}{0{,}1681}\, y(0{,}41 - y) \quad . \tag{46}$$

Jako okrajová podmínka na výstupu se volí referenční hodnota tlaku. V případě simulace proudění nestlačitelné kapaliny může tato hodnota být volena libovolně. V případě proudění stlačitelné tekutiny má tato hodnota vliv na napětí [1]. V tomto případě má zvolená referenční hodnota tlaku na výstupu nulovou průměrnou hodnotu.

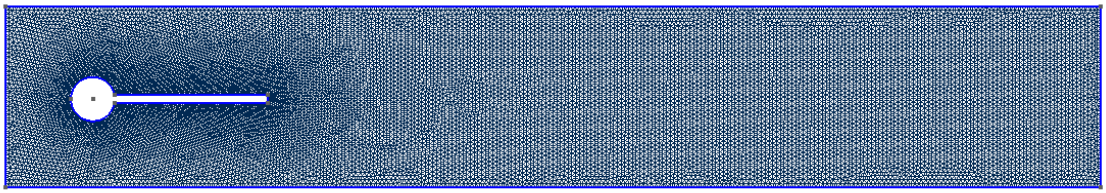
Na zbylých okrajích, tedy na horní a spodní stěně a na okrajích obtékaného tělesa, je volena nulová rychlost tekutiny vzhledem k hranicím oblasti.

3.2.4. Výpočetní sítě

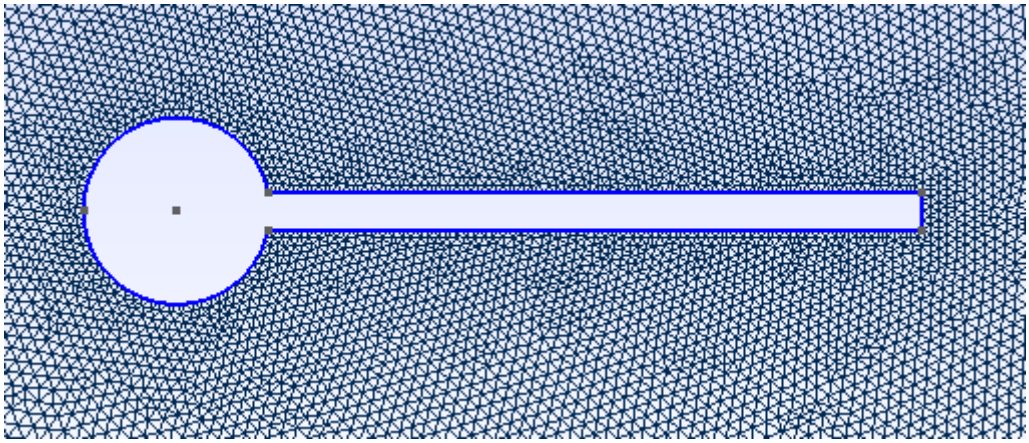
Výpočetní sítě, na kterých vlastní simulace proudění nestlačitelné viskózní tekutiny probíhaly, byly dvě. První síť, kterou jsem pro simulace použil, byla příliš hrubá a tak bylo nutné síť výrazně zjemnit. Druhá použitá síť obsahovala přibližně osminásobné množství prvků sítě první, čímž bylo dosaženo výrazně vyšší přesnosti simulace, avšak za cenu výrazného zvýšení doby trvání celé simulace. Hrubou výpočetní síť je možné vidět na obrázku 5, jemnou výpočetní síť na obrázku 6.



Obrázek 5: hrubá výpočetní síť (3794 prvků)



Obrázek 6: jemná výpočetní síť (30850 prvků)



Obrázek 7: jemná síť v okolí obtékaného tělesa

3.3 Implementace ve FEniCSu

V následujícím textu bude popsán kód implementující výpočet obtékání válce s deskou nestlačitelnou newtonovskou kapalinou. Velká část kódu byla převzata z [4]. Informace ohledně výpočtu aerodynamických sil a implementaci tohoto výpočtu ve FEniCSu byly získány z [11]. Informace ohledně použitých funkcí byly čerpány z [3]. Parametry simulace je možné nalézt v kapitole 3.2.

Nejprve je nutné importovat potřebné knihovny:

```
from fenics import *
from mshr import *
```

Knihovna fenics obsahuje všechny potřebné funkce pro výpočet parciálních diferenciálních rovnic založených na metodě konečných prvků. mshr je komponenta FEniCSu pro generování sítí.

Dále je potřeba nadefinovat konstanty využité při výpočtu:

```
T = 10
num_steps = 50000
dt = T / num_steps
mu = 1
rho = 1000
inflow_profile = ('0.2*1.5*4*x[1]*(0.41 - x[1]) / 0.1681', '0')
```

Konstanta *T* je celkový čas simulace, *num_steps* je počet kroků, *dt* je délka jednoho kroku, *mu* je dynamická viskozita, *rho* je hustota a *inflow_profile* je rychlost na vstupu do oblasti.

Následně je třeba načíst, případně vytvořit výpočetní síť. Jak již bylo v kapitole 2.4 zmíněno, při načítání již vytvořené sítě je nutné, aby síť byla uložena ve formátu *xml*. Pokud je síť uložena v jiném formátu, je nutné nejprve provést konverzi pomocí nástroje *dolfin-convertr*, který je implementován v knihovnách FEniCS. Pokud síť není předem vytvořena, je možné ji vytvořit přímo ve FEniCS.

```
mesh = Mesh("meshLD.xml")
bndry = MeshFunction("size_t", mesh)
s = Measure("ds", subdomain_data=bndry)
I = Identity(mesh.geometry().dim())
```

Pomocí funkce *MeshFunction* bude možné přistupovat k jednotlivým součástem sítě. To bude třeba při definování hranic pro budoucí výpočet aerodynamických sil. K výpočtu aerodynamických sil bude dále potřeba funkce *Measure*, která je použita pro integraci přes povrch. Proměnnou *s* tak bude možné použít pro integraci přes povrch, kde vstupní parametr bude značit index plochy. Funkce Identity vytvoří jednotkovou matici. Rozměr matice je dán dimenzí úlohy. V tomto případě se jedná a matici 2x2.

Po načtení, popřípadě vytvoření sítě je třeba vytvořit funkční prostory. Proměnná *V* je vektorový funkční prostor pro rychlost, proměnná *Q* je skalární funkční prostor pro tlak. Pro rychlost jsou zvoleny po částech kvadratické prvky, pro tlak po částech lineární prvky.

```
V = VectorFunctionSpace(mesh, 'P', 2)
Q = FunctionSpace(mesh, 'P', 1)
```

Funkce *FunctionSpace* přebírá tři vstupní parametry. První parametr je výpočetní síť, druhý parametr značí typ prvku a třetí parametr je stupeň prvku.

Aby bylo možné simulovat proudění nestlačitelné viskózní tekutiny, je nutné správně definovat hranice oblasti, ve které bude kapalina proudit. Celkově se jedná o čtyři hranice, z nichž první je přítok, druhá je odtok, třetí jsou stěny kanálu a poslední je těleso obtékané kapalinou. Hranice obtékaného tělesa může být zadána přesně, ale lze použít funkce FEniCSu, u kterých stačí zadat část výpočetní sítě a funkce hranice rozpozná automaticky.

```
inflow = 'near(x[0], 0)'
outflow = 'near(x[0], 2.5)'
walls = 'near(x[1], 0) || near(x[1], 0.41)'
cylinder = 'on_boundary && x[0]>0.1 && x[0]<0.61 && x[1]>0.1 && x[1]<0.3'
```

Funkce *near* vrací booleovskou hodnotu udávající, zda se hodnota prvního parametru dostatečně blíží hodnotě druhého. Parametr *x[0]* značí souřadnice na ose x, parametr *x[1]* je souřadnice na ose y. *on_boundary* při definici hranice obtékaného tělesa dává funkci najevo, že hranice má hledat v síti v definované oblasti.

Další možný způsob definování hranic je pomocí přímého přístupu k součástem sítě. Zde je možné k jednotlivým částem oblasti přiřadit indexy. Toho je využito při výpočtu aerodynamických sil působících na obtékané těleso.

```
eps = 1e-10
for f in facets(mesh):
        mp = f.midpoint()
        if near(mp[0], 0.0):
                bndry[f] = 1
        elif near(mp[0], 2.5):
                bndry[f] = 2
        elif near(mp[1], 0.0) or near(mp[1], 0.41):
                bndry[f] = 3
        elif mp.distance(Point(0.2, 0.2)) <= (0.05 + eps) or (0.20 <= mp[0] <= (0.6 +
eps) and (0.19 - eps) <= mp[1] <= (0.21 + eps)):
                bndry[f] = 5
```

Proměnná *eps* slouží k eliminaci chyb způsobených v důsledku zaokrouhlování čísel v počítačové aritmetice. Další možný způsob eliminace takto vzniklých chyb je užití funkce near. Pokud by případné chyby nebyly eliminovány, mohlo by dojít k vynechání některých bodů a tím ke znehodnocení výsledků simulace.

Po definování jednotlivých okrajů a počáteční podmínky pro rychlost je možné přistoupit k definování okrajových podmínek.

```
bcu_inflow = DirichletBC(V, Expression(inflow_profile, degree=2), inflow)
bcu_walls = DirichletBC(V, Constant((0, 0)), walls)
bcu_cylinder = DirichletBC(V, Constant((0, 0)), cylinder)
bcp_outflow = DirichletBC(Q, Constant(0), outflow)
bcu = [bcu_inflow, bcu_walls, bcu_cylinder]
bcp = [bcp_outflow]
```

Funkce DirichletBC je funkce pro definici Dirichletovy okrajové podmínky. Prvním argumentem je funkční prostor, druhý argument je popis chování funkce na okraji a třetí argument je hranice. Všechny okrajové podmínky jsou následně uloženy do listu pro snazší přístup a úspornější zápis.

Jelikož jsou definovány dva různé funkční prostory, je nutné také vytvořit dvě sady bázových a testovacích funkcí

```
u = TrialFunction(V)
v = TestFunction(V)
p = TrialFunction(Q)
q = TestFunction(Q)
```

Funkce *TrialFunction* vytvoří bázovou funkci daného funkčního prostoru. Funkce *TestFunction* vytvoří testovací funkci daného funkčního prostoru.

Dále je nutné vytvořit funkce reprezentující výsledky jednotlivých výpočtů. Jak pro rychlost tak pro tlak bude potřeba vytvořit dvě funkce. První funkce v sobě ponese informaci o předchozím výsledku, druhá funkce bude výsledkem v novém kroku.

```
u_i = Function(V)
u_ = Function(V)
p_i = Function(Q)
p_ = Function(Q)
```

function vytvoří funkci v závislosti na vstupním parametru. Vstupní parametr může být funkční prostor nebo jiná funkce. Pokud je funkce tvořena z jiné funkce, je možné přidat druhý parametr určující číslo dílčí funkce, která má být extrahována.

Dále bude nutné definovat výrazy použité při výpočtu.

```
U = 0.5*(u_i + u)
n = FacetNormal(mesh)
f = Constant((0,0))
k = Constant(dt)
mu = Constant(mu)
rho = Constant(rho)
```

U je proměnná určující rychlost v polovině intervalu dána vzorcem

$$u^{i+\frac{1}{2}} \approx \frac{1}{2}\left(u^{i}+u^{*}\right) \quad , \tag{47}$$

kde i je předchozí časový krok a u^{*} je hodnota předběžné rychlosti, tedy hodnota získaná výpočtem z hodnot tlaku v předchozím časovém kroku.
Proměnná *n* je vnější jednotková normála, *f* je konstanta udávající objemovou sílu, *k* je délka jednoho časového kroku, *mu* je dynamická viskozita a *rho* je hustota.

Pro snazší práci s programem je dobré nadefinovat často používané výrazy. Výsledný program je poté kratší, snáze čitelný a při případné chybě snáze opravitelný. Z těchto důvodů je výpočet tenzoru rychlosti deformace (34) realizován jako funkce, která může být v případě potřeby využita a není tak nutné psát celý výraz znovu.

```
def epsilon(u):
        return 0.5*(nabla_grad(u)+nabla_grad(u).T)
```

Funkce *nabla_grad* je jednou z funkcí obsažených v knihovnách FEniCS. Vstupním parametrem je vektor, výstupním pak jeho gradient. Parametr T značí transponovanou matici gradientu vstupního vektoru.

Ze stejných důvodů jako u tenzoru rychlosti deformace (34) je i výpočet tenzoru napětí (33) realizován jako funkce.

```
def sigma(u, p):
        return 2*mu*epsilon(u) - p*I
```

Funkce *epsilon* je funkce pro výpočet tenzoru rychlosti deformace (34) popsaná výše. Proměnná *I* je již definovaná jednotková matice.

Stejně jako výpočet tenzoru rychlosti deformace (34) a tenzoru napětí v kapalině (33) je i výpočet aerodynamických sil a jejich následné uložení realizováno pomocí funkce. Vstupním parametrem do této funkce je soubor, do kterého mají být hodnoty uloženy a čas, ve kterém síly na obtékané těleso působí.
Síla, která působí na povrch obtékaného tělesa, je dána jako integrál

$$F = \int_S T \cdot n \quad , \tag{48}$$

kde S je povrch obtékaného tělesa, T je tensor napětí a n je jednotkový normálový vektor k ploše S .

```
def Lift_Drag(LDFile,t):
        T2 = sigma(u_,p_)
        force = -dot(T2, n)
        D = (force[0])*s(5)
        L = (force[1])*s(5)
        drag = assemble(D)
        lift = assemble(L)
        LDFile.write("%e;%e;%e\n"%(t,lift,drag))
        return
```

Proměnná *T2* je tenzor napětí. K jeho výpočtu je využita již definovaná funkce popsaná výše. Proměnná *force* značí vnitřek integrálu (48). Proměnná *n* je opačně orientovaná vnější normála. Proměnná *D* odpovídá aerodynamické odporové síle, proměnná *L* aerodynamické vztlakové síle. Parametr *s* značí integraci přes povrch, přičemž číslo *5* je identifikátor plochy. Daný předpis pro výpočet sil musí být nejdřív sestaven. K tomu slouží funkce *assemble*, která předpis sestaví a vrátí odpovídající tenzor. Následně už je jen třeba získané výsledky uložit. K tomu slouží funkce *write*, která do předem otevřeného souboru uloží dané hodnoty.

Dále bude potřeba definovat rovnice popisující daný problém. Celý výpočet rychlosti a tlaku je rozdělen do tří částí. V první části je z předchozí hodnoty rychlosti a tlaku vypočtena předběžná rychlost. Výsledná rovnice pro první krok je

$$\int_\Omega \rho \frac{(u^* - u^i)}{\Delta t} \cdot v \, dx + \int_\Omega \sigma(u^{i+\frac{1}{2}}, p^i) \cdot \epsilon(v) \, dx = \int_\Omega f^{i+1} \cdot v \, dx$$
$$- \int_\Omega \rho \cdot u^i \cdot \nabla u^i \cdot v \, dx - \int_{\partial\Omega} p^i \cdot n \cdot v \, ds + \int_{\partial\Omega} \mu \cdot \nabla u^{i+\frac{1}{2}} \cdot n \cdot v \, ds \quad , \tag{49}$$

kde symbol u^* je neznámá předběžná hodnota rychlosti, u^i a p^i jsou hodnoty rychlosti a tlaku v předchozím časovém kroku, Δt je velikost jednoho časového kroku, *v* je testovací funkce a $u^{i+\frac{1}{2}}$ je hodnota rychlosti v polovině intervalu aproximovaná aritmetickou hodnotou (47). Celý výraz lze získat vynásobením rovnice (32) testovací funkcí *v* a následnou integrací. Výsledná implementace prvního kroku ve FEniCSu vypadá následovně.

```
F1 = rho*dot((u - u_i) / k, v)*dx \
       + rho*dot(dot(u_i, nabla_grad(u_i)), v)*dx \
       + inner(sigma(U, p_i), epsilon(v))*dx \
       + dot(p_i*n, v)*ds - dot(mu*nabla_grad(U)*n, v)*ds \
       - dot(f, v)*dx
a1 = lhs(F1)
L1 = rhs(F1)
```

Proměnná *u* je neznámá rychlost, proměnná *v* je testovací funkce. *P*roměnná *u_i* je rychlost v minulém časovém kroku, proměnná *p_i* je tlak v předchozím čase, *U* je hodnota rychlosti v polovině intervalu (47), *k* je definovaná velikost jednoho časového kroku, *n* je vnější jednotková normála a *f* je objemová síla. Parametr *dx* značí integraci. Pro integraci přes povrch je použit parametr *ds.* Jelikož jsou napětí a rychlost deformace tensory, nemůže být pro výpočet integrálu $\int_\Omega \sigma(u^{i+\frac{1}{2}}, p^i) \cdot \epsilon(v) dx$ použita funkce *dot*. Místo ní je použita funkce *inner*. Pro vektory vrací funkce *inner* stejnou hodnotu jako funkce *dot*. Nakonec je nutné separovat levou a pravou stranu rovnice. K tomu jsou užity funkce *lhs* a *rhs*, které rozdělení rovnice provedou automaticky.

V dalším kroku bude z výsledné rychlosti z prvního kroku vypočten tlak v novém čase. Výsledná rovnice pro tlak v čase i+1 je

$$\int_\Omega \nabla p^{i+1} \cdot \nabla q \, dx = \int_\Omega \nabla p^i \cdot \nabla q \, dx - \frac{1}{\Delta t} \int_\Omega \nabla \cdot u^* \cdot q \, dx \quad . \tag{50}$$

Symbol q v rovnici (50) je skalární testovací funkce z prostoru tlaku.

```
a2 = dot(nabla_grad(p), nabla_grad(q))*dx
L2 = dot(nabla_grad(p_i), nabla_grad(q))*dx - (1/k)*div(u_)*q*dx
```

V předchozím kroku byly jednotlivé strany rovnice získány funkcí *lhs* a *rhs*. V tomto případě je levá a pravá strana rovnice separována rovnou.

V posledním kroku je spočtena výsledná rychlost v čase i+1. Vzorec pro výpočet rychlosti je

$$\int_\Omega u^{i+1}\cdot v\,dx=\int_\Omega u^{*}\cdot v\,dx-\Delta t\int_\Omega \nabla\left(p^{i+1}-p^{i}\right)\cdot v\,dx \quad . \tag{51}$$

```
a3 = dot(u, v)*dx
L3 = dot(u_, v)*dx - k*dot(nabla_grad(p_ - p_i), v)*dx
```

Jelikož jsou levé strany všech tří rovnic časově nezávislé, není nutné je sestavovat v každém časovém kroku a stačí tak učinit pouze jednou.

```
A1 = assemble(a1)
A2 = assemble(a2)
A3 = assemble(a3)
```

Nakonec je nutné na výsledné matice použít definované okrajové podmínky. Pro matici získanou v první kroku jsou aplikovány okrajové podmínky určující rychlost tekutiny na stěnách a okraji obtékaného předmětu. Na matici získanou v druhém kroku je aplikována okrajová podmínka určující referenční tlak na výstupu. Třetí matice je počítána z hodnot získaných v prvním a druhém kroku a není tedy již nutné okrajové podmínky aplikovat.

```
[bc.apply(A1) for bc in bcu]
[bc.apply(A2) for bc in bcp]
```

Aby bylo možné s výsledky simulace dále pracovat, bude nutné je uložit. Výsledky budou ukládány do vybraných složek a souborů.

```
ufile = File("results/velocity.pvd")
pfile = File("results/pressure.pvd")
File('navier_stokes_cylinder/cylinder.xml.gz') << mesh
LDFile = open("results/LiftDrag.txt","a")
```

Před začátkem cyklu, ve kterém bude počítán průběh proudící kapaliny, je vytvořena proměnná udávající aktuální čas simulace.

```
t = 0
for x in range(num_steps):
```

Nejprve je k hodnotě aktuálního času přičtena hodnota časového kroku.

```
t += dt
```

Následně je zjištěn aktuální pokrok v simulaci a při dosažení další mezní hodnoty je tato skutečnost signalizována. Tato část kódu slouží čistě pro signalizace pokroku v simulaci.

```
if (x % (num_steps/100)) == 0 :
        print("Done:",100*x/num_steps,"%")
```

V prvním kroku je nejprve sestavena pravá strana rovnice (49). Poté jsou na výsledný vektor použity okrajové podmínky. Nakonec je možné rovnici vyřešit.

```
b1 = assemble(L1)
[bc.apply(b1) for bc in bcu]
solve(A1, u_.vector(), b1, 'bicgstab', 'hypre_amg')
```

K řešení rovnice je využita funkce *solve*. Tato funkce má celkem čtyři různé způsoby použití. V tomto případě je využita pro řešení lineárního systému, který je pro tento určitý případ

$$A1 \cdot u\_.vector() = b1 \quad , \tag{52}$$

kde $A1$ je matice a $u\_.vector()$ a $b1$ jsou vektory. Ostatní parametry jsou volitelné a určují způsob a podmínky řešení, což má vliv na dobu výpočtu a využití paměti.

Stejným způsobem jako probíhal první krok výpočtu probíhají i kroky další.

```
b2 = assemble(L2)
[bc.apply(b2) for bc in bcp]
solve(A2, p_.vector(), b2, 'bicgstab', 'hypre_amg')
b3 = assemble(L3)
solve(A3, u_.vector(), b3, 'cg', 'sor')
```

Po vyřešení všech rovnic je možné výsledky uložit. Jelikož je však jednotlivých kroků velké množství, nejsou výsledky ukládány v každém kroku. V tomto případě jsou výsledky uloženy v každém dvacátém průchodu cyklem.

```
if (x % 20) == 0 :
        Lift_Drag(LDFile,t)
        ufile << u_
        pfile << p_
```

Funkce *Lift_Drag* je funkce pro výpočet a uložení hodnot aerodynamických odporových sil popsaná výše. *ufile* je soubor pro ukládání hodnot rychlosti, *pfile* je soubor pro ukládání hodnot tlaku.

Nakonec je ještě nutné aktualizovat hodnoty rychlosti a tlaku v minulém kroku.

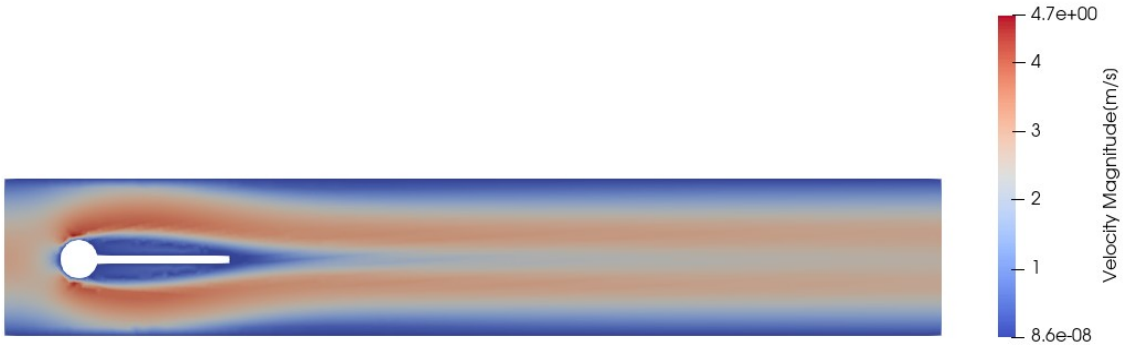
```
u_i.assign(u_)
p_i.assign(p_)
```

Funkce *assign* přiřadí funkci, která je vstupním parametrem funkci druhé.

3.4 Výsledky numerických simulací

3.4.1. Výsledky simulace při použití hrubší výpočetní sítě

Na obrázcích 8 a 9 je možné vidět proudová pole získaná při použití hrubé sítě .

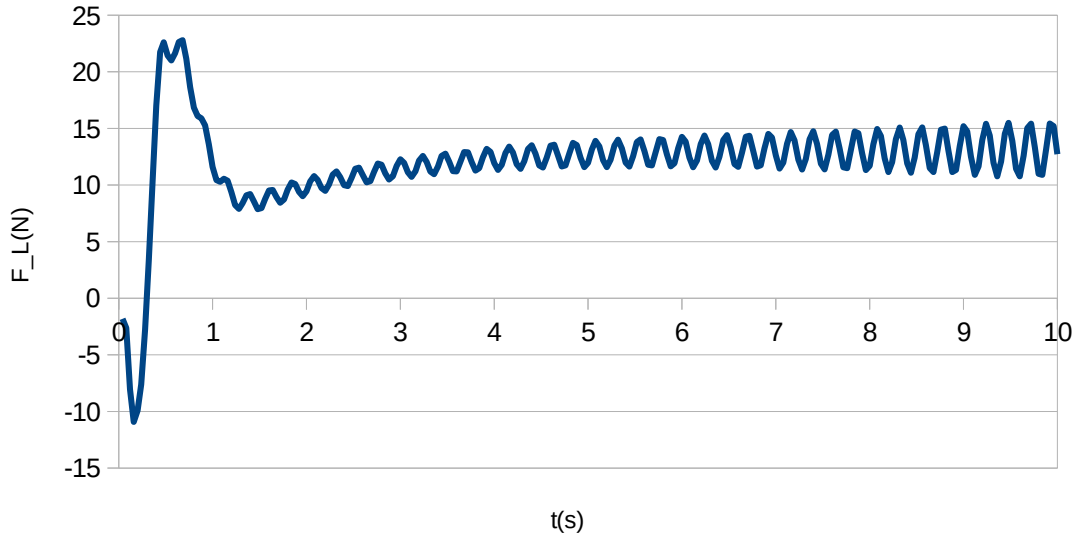


Obrázek 8: rychlost tekutiny obtékající těleso v čase 9 sekund

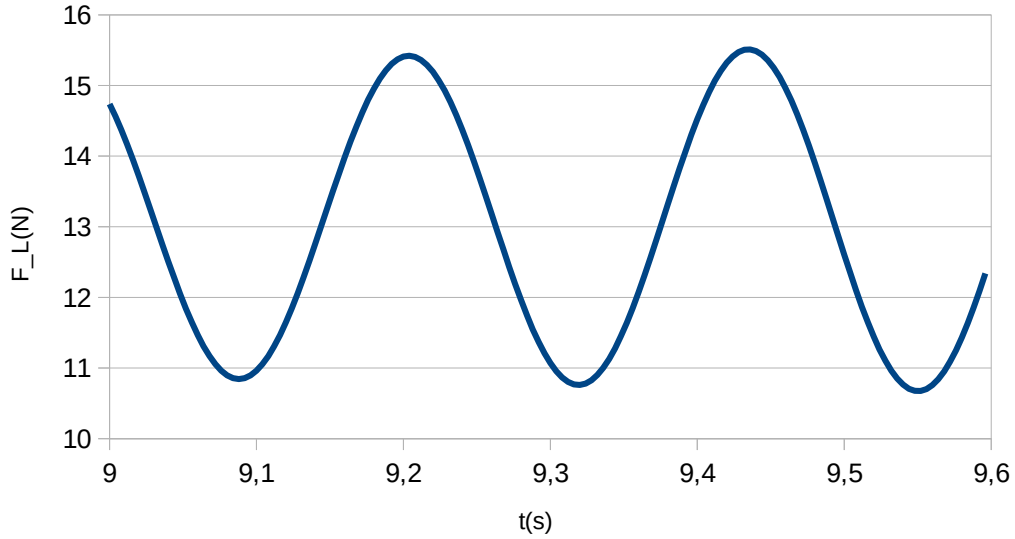
Obrázek 9: tlak tekutiny obtékající těleso v čase 9 sekund

Z obrázku je patrné, že téměř nedochází k oscilacím úplavu za obtékaným tělesem. Tato skutečnost je zapříčiněna příliš nízkým počtem prvků výpočetní sítě. Při užití sítě jemnější, tedy sítě s více jak osminásobným počtem prvků než má síť použitá v tomto případě, již k oscilacím dochází. Výsledky numerických simulací s využitím jemnější sítě lze nalézt v kapitole 3.4.2.

Na obrázcích 10 až 13 jsou vykresleny průběhy aerodynamické odporové a vztlakové síly působící na obtékané těleso.

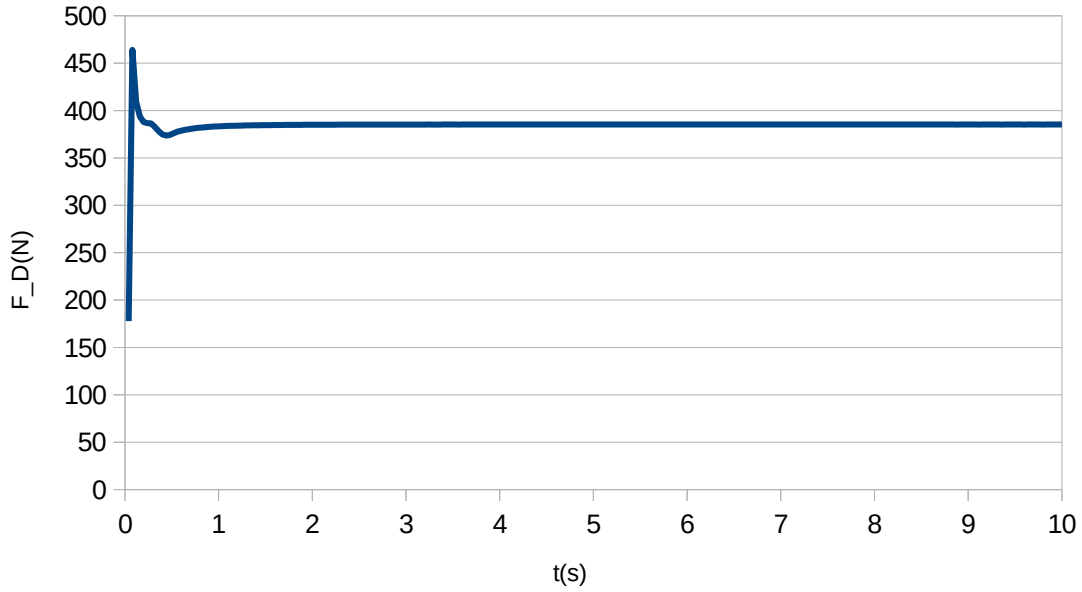


Obrázek 10: graf aerodynamické vztlakové síly v čase 0 až 10 sekund

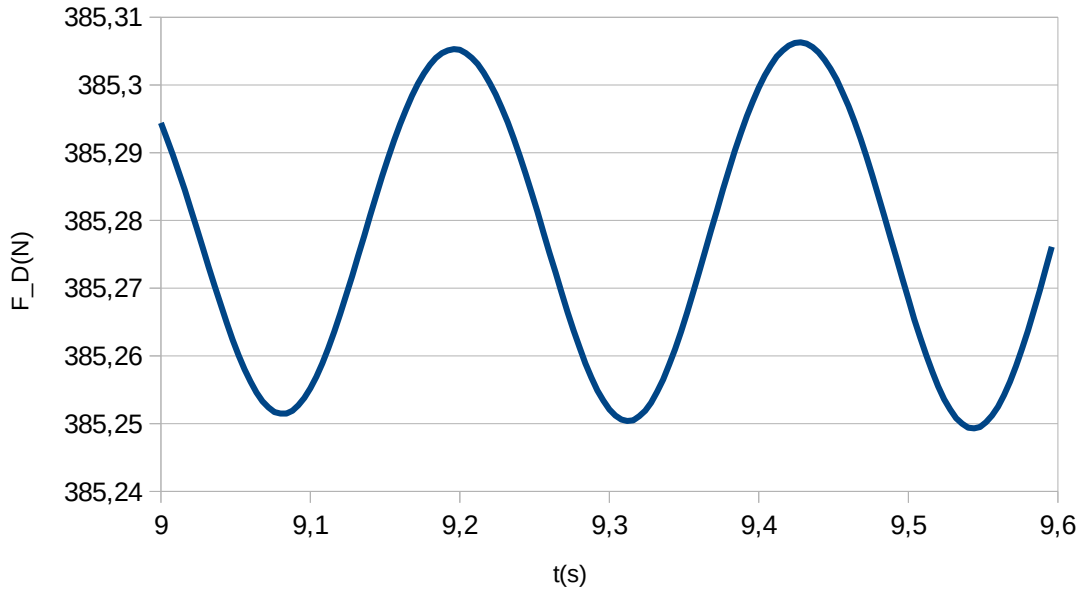


Obrázek 11: graf aerodynamické vztlakové síly v čase 9 až 9,6 sekund

Z grafu 10 je patrné, že i když z proudového pole by se dalo soudit, že k žádným oscilacím úplavu za obtékanou překážkou nedochází, tak úplav lehce osciluje. Tyto kmity mají mírně vzestupnou tendenci. Z detailnějšího zobrazení průběhu aerodynamické vztlakové síly v grafu 11 je patrné, že střední hodnota průběhu síly v čase 9 až 9,6 sekund se pohybuje přibližně kolem 13 N.



Obrázek 12: graf aerodynamické odporové síly v čase 0 až 10 sekund

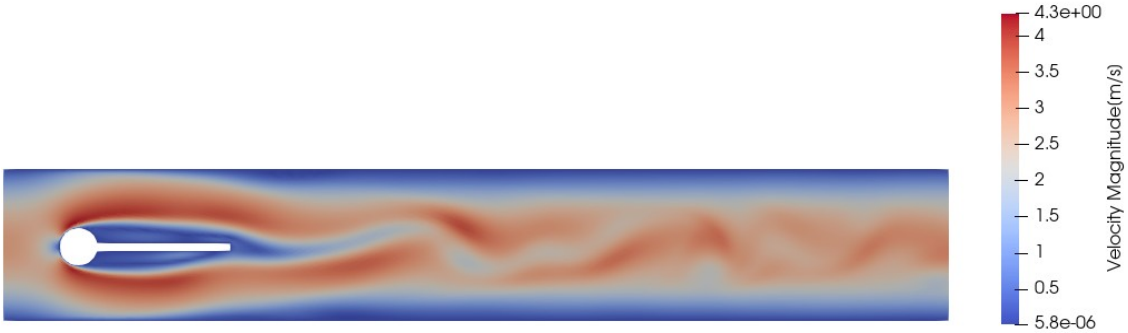


Obrázek 13: graf aerodynamické odporové síly v čase 9 až 9,6 sekund

Z grafů 12 a 13 je zřejmé, že u aerodynamické odporové síly k oscilacím téměř nedochází. Síla se přibližně po jedné sekundě stabilizuje. Z grafu 13 lze vyčíst, že minima a maxima síly v čase 9 až 9,6 sekund se od sebe liší v průměru o přibližně 0,055 N.

3.4.2. Výsledky simulace při použití jemnější výpočetní sítě

Na obrázcích 14 a 15 je možné vidět proudová pole získaná při použití jemnější výpočetní sítě .

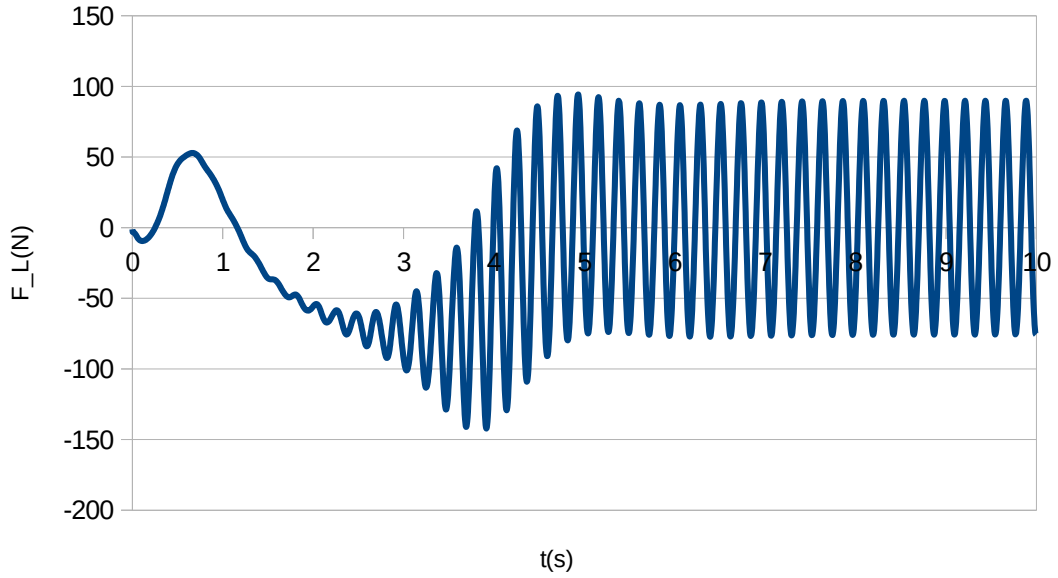


Obrázek 14: rychlost tekutiny obtékající těleso v čase 9 sekund

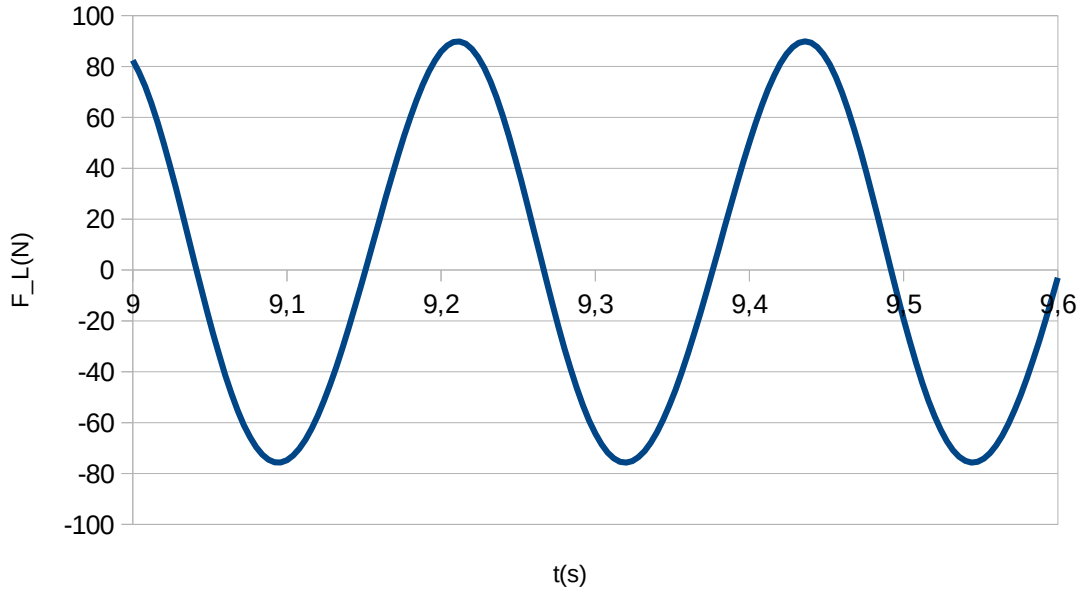
Obrázek 15: tlak tekutiny obtékající těleso v čase 9 sekund

Z obrázku 14 je zřejmé, že při využití jemnější výpočetní sítě již úplav začne zřetelně kmitat. Skutečnost, že ve výsledcích získaných při použití sítě hrubé úplav téměř nekmitá a při využití sítě jemné jsou kmity dobře zřetelné, poukazuje na důležitost správného výběru výpočetní sítě.

Na obrázcích 16 až 19 jsou vykresleny průběhy aerodynamické odporové a vztlakové síly působící na obtékané těleso.

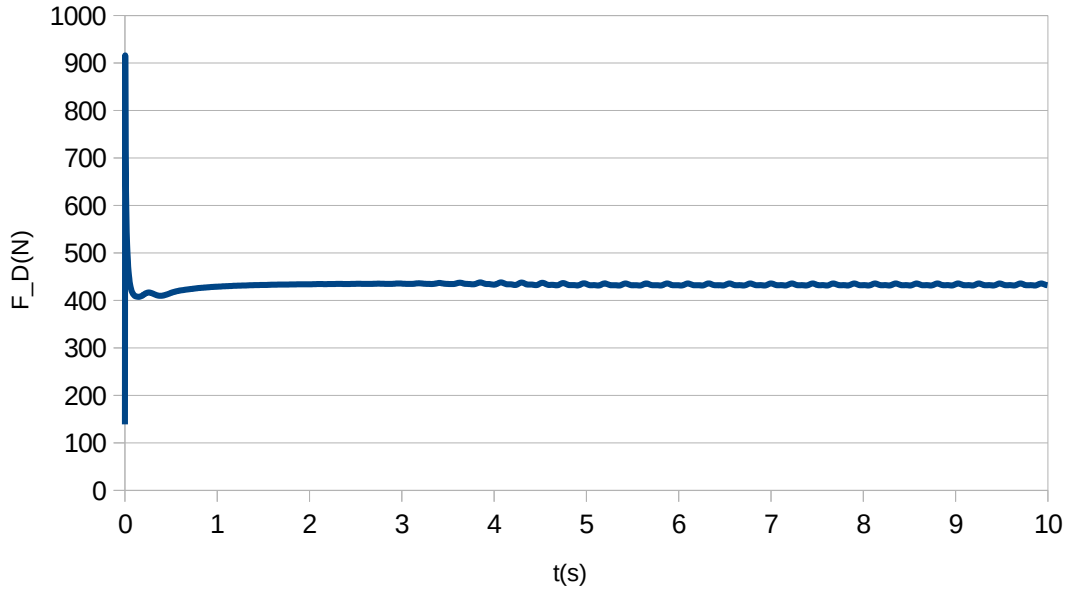


Obrázek 16: graf aerodynamické vztlakové síly v čase 0 až 10 sekund

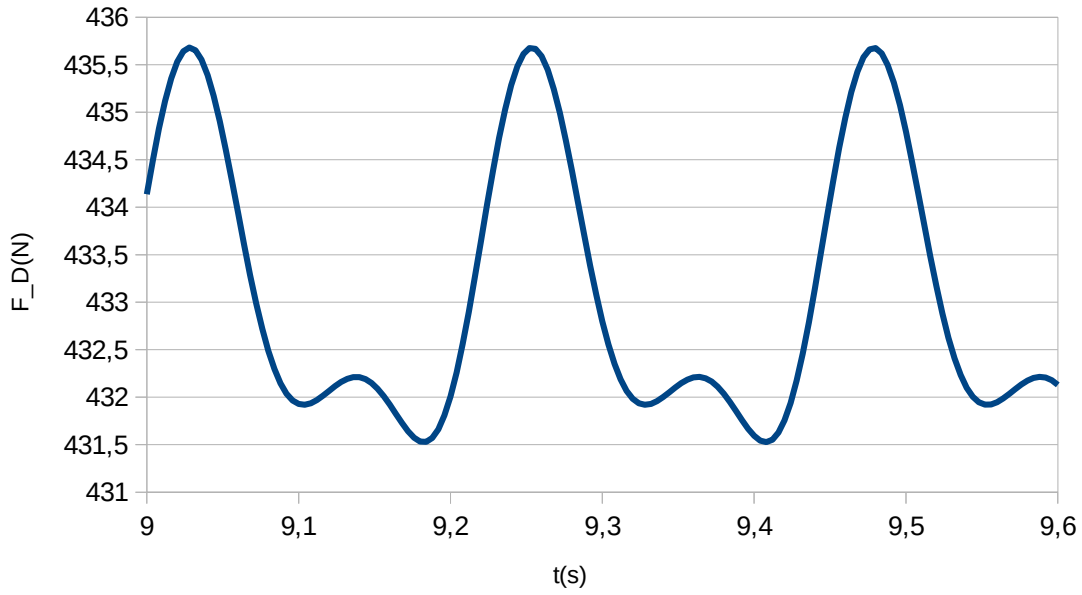


Obrázek 17: graf aerodynamické vztlakové síly v čase 9 až 9,6 sekund

Z grafu je patrné, že střední hodnota aerodynamické vztlakové síly se přibližně po pěti sekundách stabilizuje. Z detailnějšího zobrazení v grafu 17 lze vyčíst, že střední hodnota síly se po ustálení blíží nulové hodnotě. Rozdíly mezi maximy a minimy jsou přibližně 165N.



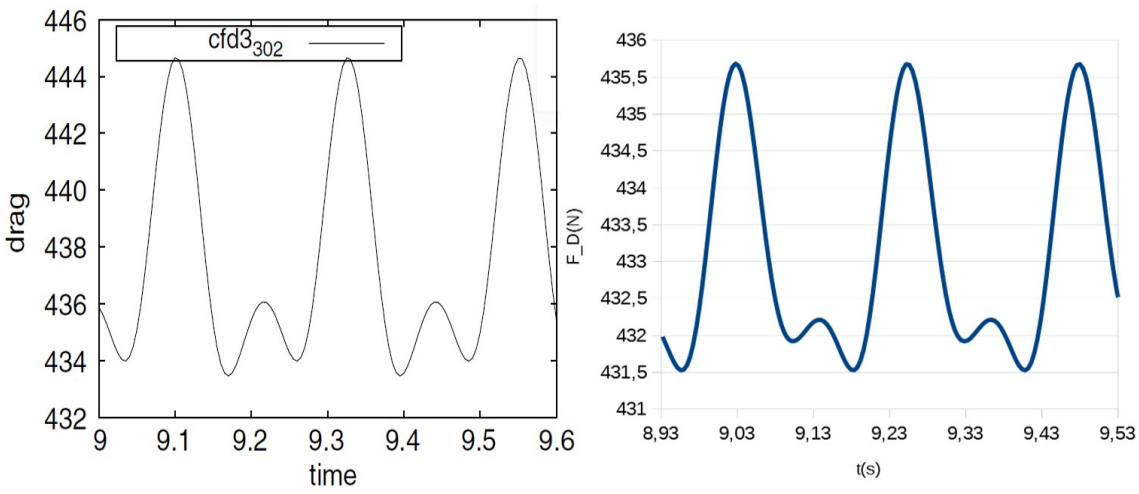
Obrázek 18: graf aerodynamické odporové síly v čase 0 až 10 sekund



Obrázek 19: graf aerodynamické odporové síly v čase 9 až 9,6 sekund

Z grafu 18 je patrné, že přibližně v jedné sekundě dojde k ustálení síly. Stejně jako u aerodynamické vztlakové síly, tak i u aerodynamické odporové síly se při využití jemnější sítě objeví oscilace, což je vidět v grafu 19. Rozdíl mezi maximy a minimy je přibližně 4,1 N a střední hodnota síly v čase 9 až 9,6 sekund se zhruba pohybuje kolem 433 N.

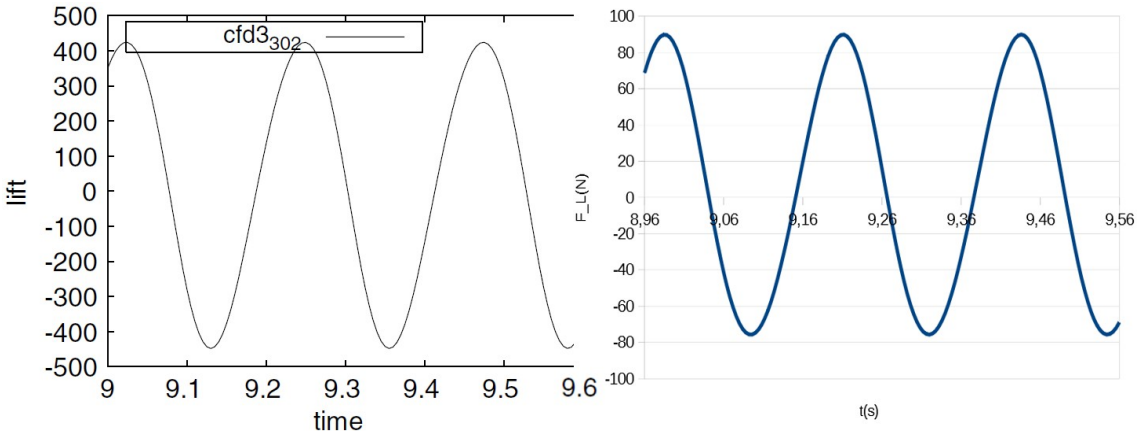
Srovnání výsledků aerodynamické odporové síly s benchmarkovými výsledky [1] je možné vidět na obrázku 20.



Obrázek 20: srovnání výsledků aerodynamické odporové síly z benchmarku [1] a ze simulací

Frekvence kmitů aerodynamické odporové *síly* se shoduje s frekvencí kmitů v benchmarku. Její maxima jsou přibližně o 2% menší než maxima síly v benchmarku a mírně se liší i celkový průběh síly v čase.

Srovnání výsledků aerodynamické vztlakové síly s benchmarkovými výsledky je možné vidět na obrázku 21.



Obrázek 21: srovnání výsledků aerodynamické vztlakové síly z benchmarku [1] a ze simulací

U aerodynamické vztlakové síly jsou rozdíly mezi benchmarkem a výsledkem vlastních simulací podstatně větší. Frekvence kmitů z výsledků simulace souhlasí s benchmarkovými výsledky. Maxima síly jsou však téměř pětkrát menší. Důvodem těchto rozdílů je pravděpodobně stále nedostatečná hustota sítě.

Výsledky simulace jsou na hustotě výpočetní sítě závislé pouze do určitého momentu, kdy i zvýšení hustoty sítě zásadně neovlivní výsledek simulace. S nejhustší mnou použitou výpočetní sítí se čas potřebný k dokončení celé simulace blížil deseti hodinám. Pokud by mělo být dosaženo výsledků, které by dostatečně přesně souhlasily s výsledky v benchmarku, byla by výpočetní síť tak hustá, že na výsledky simulace by bylo nutno čekat několik dní, případně využít paralelního výpočtu na výkonnějším stroji. S vlivem výpočetní sítě na přesnost výsledku a dobu trvání simulací je nutné počítat a vždy volit správnou síť vzhledem k danému problému a požadované přesnosti výsledků.

4 Interakce proudění a pružného tělesa

Pro simulace deformace pružného tělesa v důsledku proudění tekutiny, bude zapotřebí spojit kód implementující deformaci pružného tělesa a kód implementující obtékání pevného tělesa tekutinou dohromady.

Nejprve bude nutné zjistit silové účinky proudící kapaliny na pružné těleso. Tyto aerodynamické síly bude možné získat obdobným způsobem jako probíhalo jejich zjišťování v případě pevného tělesa, s tím rozdílem, že nyní bude nutné získat síly působící výhradně na pružný vetknutý nosník. V případě nosníku deformovaného svojí vlastní vahou bylo zatížení definováno jako konstanta. V případě sdruženého problému bude výsledné zatížení nosníku odpovídat působení aerodynamických sil. V důsledku působících sil se bude nosník deformovat a tyto deformace budou následně ovlivňovat proudící tekutinu. Z tohoto důvodu bude nutné v závislosti na vypočtené deformaci nosníku posunout dané vrcholy výpočetní sítě. To lze učinit pomocí funkce *move* implementované v knihovnách FEniCS. Ta přebírá síť, u které má dojít k posunu a funkci definující posun. Aby bylo možné použít posun sítě v závislosti na deformaci nosníku, bude zároveň potřeba přeformulovat rovnice pomocí ALE metody (Arbitrary Lagrangian-Eulerian method). U této formulace není systém fixovaný v prostoru a je možné s výpočetní sítí pohybovat. Více informací ohledně metody ALE je možné nalézt v [10].

5 Závěr

Tato bakalářská práce byla věnována simulaci ohybu spojitě zatíženého nosníku a simulaci obtékaní tělesa nestlačitelnou vazkou tekutinou s využitím knihovny FEniCS založené na metodě konečných prvků. V případě ohybu nosníku bylo dosaženo odchylky mezi výsledky získaných z analytického výpočtu a výsledky numerických simulací v řádu desetin promile. V případě simulace obtékání tělesa tekutinou se naprosté shody výsledků vlastních numerických simulací a výsledků simulací v benchmarku [1] dosáhnout nepodařilo. Důvodem byla velká časová náročnost výpočtů a nedostatečný výkon stroje, na kterém výpočty probíhaly. Jako další možné pokračování této práce by byl přechod od návrhu postupu k realizaci simulace problému interakce proudění s pružnými tělesy.

Použitá literatura

[1] TUREK, Stefan a Jaroslav HRON. *Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow*. Vol 53. Berlin, Heidelberg: Springer, 2006. ISBN 978-3-540-34595-4.

[2] WHITE, Frank M. *Fluid Mechanics*. 7. New York: McGraw-Hill, 2011. ISBN 978-0-07-352934-9.

[3] LOGG, Anders, Kent-Andre MARDAL, Garth N. WELLS a další. *Automated Solution of Differential Equations by the Finite Element Method* [online]. Berlin Heidelberg: Springer, 2012 [cit. 2020-05-24]. DOI: 10.1007/978-3-642-23099-8. ISBN 978-3-642-23098-1. Dostupné z: https://launchpadlibrarian.net/83776282/fenics-book-2011-10-27-final.pdf

[4] LANGTANGEN, Hans Petter a Anders LOGG. *Solving PDEs in Python* [online]. Berlin Heidelberg: Springer, 2017 [cit. 2020-05-24]. DOI: 10.1007/978-3-319-52462-7. ISBN 978-3-319-52461-0. Dostupné z: https://fenicsproject.org/pub/tutorial/pdf/fenics-tutorial-vol1.pdf

[5] *Matematické modely proudění nestlačitelné tekutiny s různými typy okrajových podmínek* [online]. Praha, 2018 [cit. 2020-05-24]. Dostupné z: https://mat.fsv.cvut.cz/doktorandi/files/JP.pdf. Disertační práce. České vysoké učení technické v Praze.

[6] 2D linear elasticity. *Numerical tours of continuum mechanics using FEniCS* [online]. Jeremy Bleyer, 2016 [cit. 2020-05-24]. Dostupné z: https://comet-fenics.readthedocs.io/en/latest/demo/elasticity/2D_elasticity.py.html

[7] *Mechanika pro ÚZS: Cvičení 7 – ohyb*. Technická univerzita v Liberci: Petr Šidlof, 2010.

[8] *Metoda konečných prvků*. Technická univerzita v Liberci: Jan Stebel, 2020.

[9] *Reynoldsovo číslo* [online], posední aktualizace 18. 8. 2019 v 20:42[cit. 2020-05-24],Wikipedie. Dostupné z: https://cs.wikipedia.org/wiki/Reynoldsovo_%C4%8D%C3%ADslo

[10] *Arbitrary Lagrangian-Eulerian Method* [online]. University of South Carolina: Jianzheng Zuo [cit. 2020-05-24]. Dostupné z:http://www.me.sc.edu/research/jzuo/Contents/ALE/ALE_1.htm

[11] BLECHTA, Jan a Jaroslav HRON. Stokes equation. *FEniCS tutorial 1.5.0 documentation* [online]. Blechta, 2015 [cit. 2020-05-30]. Dostupné z: https://www.karlin.mff.cuni.cz/~blechta/fenics-tutorial/stokes/doc.html