Odprto kodni sistemi za reševanje FVM in FEM na HPC

Univerza *v Ljubljani*
Fakulteta *za pomorstvo in promet*

Aleksander GRM

Maj, 2022

Vsebina

Kratek pregled

Matematično modeliranje realnih sistemov se v večini primerov zoži na matematičen model, ki ga popišejo **Parcialne diferencialne enačbe** (**PDE**).

Primer: popis valovanja na vodni gladini

$$\boxed{\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)} = c^2 \Delta u,$$

kjer imamo višino vala

$$u = u(t, \boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \quad \Omega \subseteq \mathbb{R}^2,$$

podano v smeri z.

V splošnem lahko rešujemo dva tipa problemov

- **začetni problem** (**IVP** - initial value problem):
  čas t je neodvisna spremenljivka problema in rešujemo **časovno odvisen** problem, tako potrebujemo

  začetni pogoj: $u_0 = u(t = t_0, \boldsymbol{x})$, kjer je $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^n$

  (primer valovne enačbe)

- **robni problem** (**BVP** - boundary value problem):
  čas t ni podan kot neodvisna spremenljivka v problemu in rešujemo **časovno neodvisen** problem, tako potrebujemo

  robni pogoj: $u_0 = u(\boldsymbol{x})$, kjer je $\boldsymbol{x} \in \Gamma \subseteq \mathbb{R}^{n-1}$ $(\Gamma = \partial\Omega)$,

  kjer je Γ **rob** računske domene. Velikokrat uporabimo tudi oznako $\partial\Omega$.

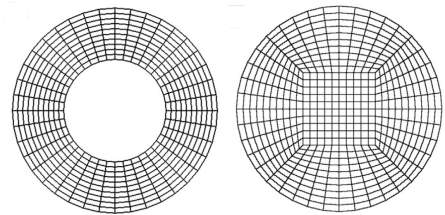
IVP pogoj dobimo lahko tudi z rešitvijo BVP (po potrebi BVP štartamo z intuitivno/enostavno rešitvijo)!

PDE lahko rešimo na različne načine

- **analitični pristopi**: rešujemo linearni problem oz. skoraj linearni problem v enostavnih geometrijah. Pristopi, ki se uporabljajo: separacija, razvoj v vrsto, Perturbacijske metode, Laplaceova transformacija, kompleksna analiza, ...
- **numerični pristopi**: vse, kar se ne da analitično, metode v uporabi:
  - metoda končnih diferenc - **močna** rešitev (FDM - Finite Diference Method)
  - metoda končnih volumnov - **šibka** rešitev (FVM - Finite Volume Method)
  - metoda končnih elementov - **šibka** rešitev (FEM - Finite Element Method)
- **specialni numerični pristopi**: uporaba FDM, FVM in FEM v različnih kombinacijah
  - metoda vložene površine - (IBM - Immersed Boundary Method)
  - brezmrežne metode - (SPH - Smothed Particle Hydrodynamics)

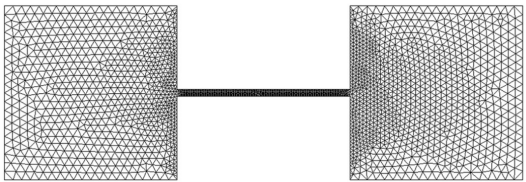
Vse numerične metode, razen SPH, potrebuje mrežo. Mreža razdeli računsko domeno na celice oz. elemente

Strukturirana mreža
(urejena)



dobre konvergenčne lastnosti (fluidi)

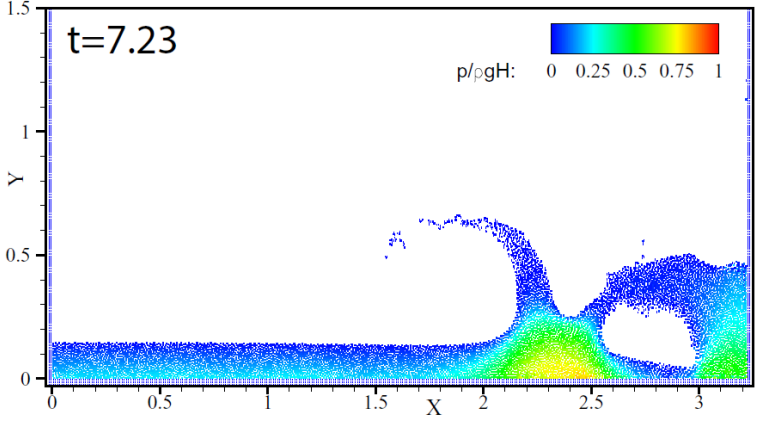
Ne-Strukturirana mreža
(ne urejena)



slabše konvergenčne lastnosti (fluidi)

Pogosto imamo kombinacije obeh tipov mrež!

V primeru brezmrežnih metod uporabljamo delce, ki polnijo domeno in so predvsem v uporabi, kjer se oblika domene hitro spreminja s časom, kot so primeri valovanja, kontinuiranega ulivanja, strjevanja, ipd.



Mrežni generatorji

GMSH je odprto kodni mrežni generator, kjer lahko izdelamo skorajda vsako mrežo.

Vsebuje izdelavo:

- 2D strukturirane/nestrukturirane mreže
- 3D nestrukturirane mreže (3D strukturirane - ?)
- 2D/3D mejne plasti

Elementi, ki so uporabljeni za gradnjo:

- 2D: trikotnik in štirikotnik
- 3D: tetraeder, heksaeder, dodekaeder (? - satovje)

Modelirniki:

- interni jezik
- OpenCascade
- STL rekonstrukcija (zelo dobro)

Primer – Cavity mesh


```
1  // Geo parameters - Parametrised geometry - VERY USEFUL ;)
2  L = 100; // side length
3  G1 = 10;  // element size
4  G2 = 2;  // element size
5
6  // All numbering counterclockwise from bottom-left corner
7  Point(1) = {-L, -L, 0, G2};
8  Point(2) = { L, -L, 0, G2};
9  Point(3) = { L,  L, 0, G1};
10 Point(4) = {-L,  L, 0, G1};
11 Line(1) = {1, 2};                          // bottom line
12 Line(2) = {2, 3};                          // right line
13 Line(3) = {3, 4};                          // top line
14 Line(4) = {4, 1};                          // left line
15 Line Loop(5) = {1, 2, 3, 4};
16 ...
```

Prikaži strukturo in uporabo GMSH cavity primera (`cavity_problem/gmsh`)!

Open source - PDE

Obstaja vrsto odprto kodnih projektov, ki rešujejo specialne PDEje. V matematiki nas zanimajo predvsem projekti, ki nudijo ogrodje za reševanje poljubne PDE ali sisteme PDE.
Pomembno: okolje se mora tekoče posodabljat!

- **OpenFOAM**: splošen FVM paket
  OpenFOAM: `https://openfoam.org`
  FOAM-extend: `https://github.com/Unofficial-Extend-Project-Mirror`
- **deal.II**: splošen FEM paket `https://www.dealii.org`
- **Fenics**: splošen FEM paket `https://fenicsproject.org`
- **Medusa**: splošen SPH paket `http://e6.ijs.si/medusa`
- **Elmer**: specializiran FEM paket `https://www.csc.fi/web/elmer`
- **Vega**: specializiran FEM paket `http://barbic.usc.edu/vega`

OpenFOAM (OF)

Kaj je OpenFOAM?

- **finite volume** okolje za CFD
- izdelan v c++, Python support (`pyFoam`)
- izvaja **paralelno** računanje (shared and distributed memory platforms - MPI paralelizacija)
- vključuje že narejene solverje (tekočin in trdnine), različne robne pogoje, pre in post procesiranje, ...
- tekstovno orientiran zagon (OpenFOAM-GUI specialni projekti ali komercialna uporaba)
- **paraview** je GUI za post-procesiranje (`paraFoam` ukaz)
- je Open Source **!!! ZASTONJ !!!**
- vse o OpenFOAM:
  → `www.openfoam.org`
  → `foam-extend@github`

Linux

- naravno okolje

Windows Subsystem for Linux - WSL

- se izvaja nativno (hitro)
- pazi na distribucijo (Debian, Ubuntu)
- ni nativne X podpore

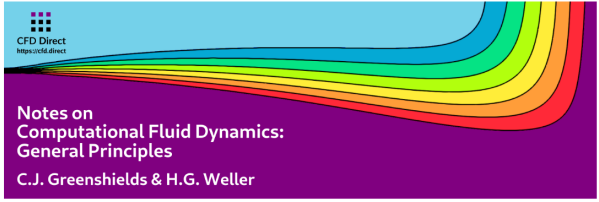
Docker Engine

- dela nativno (Docker)
- problem z izvajanjem X aplikacij

Virtual Machine

- teče nekaj počasneje (VMWare ali Virtualbox)

link to the book



About the Book

Notes on Computational Fluid Dynamics (CFD) was written for people who use CFD in their work, research or study, providing essential *knowledge* to perform CFD analysis with confidence. It offers a modern perspective on CFD with the finite volume method, as implemented in OpenFOAM and other popular general-purpose CFD software. Fluid dynamics, turbulence modelling and boundary conditions are presented alongside the numerical methods and algorithms in a series of short, digestible topics, or *notes*, that contain complete, concise and relevant information. The book benefits from the experience of the authors: Henry Weller, core developer of OpenFOAM since writing its first lines in 1989; and, Chris Greenshields, who has delivered over 650 days of CFD training with OpenFOAM.

ISBN 978-1-3999-2078-0, 291 pages.

link to the book

OF raznolikost

- nabor turbulentnih modelov (k-ε, k-ω, k-ω-SST,...)
- nabor robnih pogojev
- podpora za več-fazno modeliranje
- modeli za notranje izgorevanje
- DNS
- napetostna analiza + FSI
- in še veliko več ...
- poglej na `www.openfoam.org` - User Guide

Primer osnovne uporabe za primer - Cavity flow

1. izdelava mreže: `blockMesh` (interni OF ukaz)
2. preveri kvaliteto mreže: `checkMesh` (interni OF ukaz)
3. poženi časovno shemo: `icoFoam` (interni OF ukaz)
4. oglej si rezultat: `paraFoam` (interni OF ukaz)

Primer napredne uporabe za primer - Cavity flow

- rešimo osnovni problem (prejšnja prosojnica)
- mapiramo polja na drugo mrežo
- stabiliziramo pogoje reševanja za novo mrežo
- izvedemo poljubno gostitev mreže (glede na markerje celic/elementov)
- sedaj lahko rešimo problem za visoke Re

Osnovna struktura OF problema

- root je določen kot ime problema, imenujemo ga recimo **case**
- znotraj imamo pod mape (slika spodaj)
  - **ibc**: začetni in robni pogoji
  - **constant**: vse vrednosti konstant in vse za manipulacijo mreže
  - **system**: vsi pogoji za vodenje računskega procesa

```
case
├── 0 - ibc
├── constant - constant properties
└── system - computational properties
```

Dimenzije so v OF nastavljive z vektorjem

No.	Property	SI unit	USCS unit
1	Mass	kilogram (kg)	pound-mass (lbm)
2	Length	metre (m)	foot (ft)
3	Time	second (s)	second (s)
4	Temperature	Kelvin (K)	degree Rankine (°R)
5	Quantity	mole (mol)	mole (mol)
6	Current	ampere (A)	ampere (A)
7	Luminous intensity	candela (cd)	candela (cd)

Primer: kinematična viskoznost ν $[\mathrm{m}^2/\mathrm{s}]$

- vrednost je določena v datoteki `constants/transportProperties`

```
nu          [0 2 -1 0 0 0 0]        0.01;
```

Začetni in robni pogoji za primer turbulentnega toka s transportom za T

```
0
├── U - velocity
├── p - pressure
├── k - turbulent
├── epsilon - turbulent
└── T - scalar transport
```

Constants

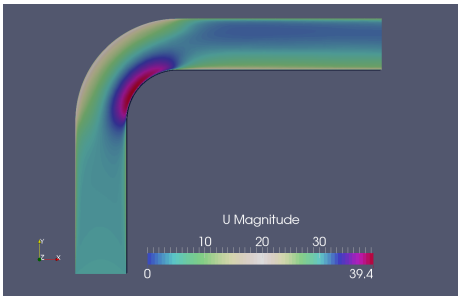
- `constant`
  - `turbulenceProperties:` lastnosti turbulentnega modela
  - `transportProperties:` viskoznost in tip toka
  - `polyMesh:` računska mreža
    - `boundary`
    - `points`
    - `faces`
    - `owner`
    - `neighbour`

Sistemske nastavitve

`system`
- `controlDict:` kontrola simulacije
- `fvSchemes:` diskretizacijske sheme
- `fvSolution:` način reševanja
- `decomposeParDict:` dekompozicij in paralelizacija
- `residuals:` filtriranje residualov za analizo procesa
- ...

Naloga: Reši tok tekočine v cevi, kjer imamo topno cev (raztapljanje)

- $r = 0.25\,\mathrm{cm}$
- $U_{\mathrm{inlet}} = 30\,\mathrm{m/s}$
- nestisljiva tekočina ($\nabla \cdot \boldsymbol{v} = 0$)
- izotermni pojav
- stena se topi
- transport snovi stene (coupling)



Tok tekočine je modeliran kot k-ϵ turbulentni model, kjer je transport skalarne količine ϕ določen z modelom

$$\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi = \nu \Delta \phi$$

OF potrebni koraki za izračun transporta ϕ v ravnovesnem stanju

1. določi hitrostno polje $\boldsymbol{v}$ (turbulent)
2. transport sklalarja ϕ
3. če so residuali $\leq$ EPS končaj, nasprotno GOTO 1

- tok tekočine je turbulenten, rešujemo s k-ϵ model,
- "no-slip"pogoj: na robu je hitrost nič ($\boldsymbol{v}$ @ rob $= 0$),
- raztopljeni material ϕ je prenešen s tokom ($\boldsymbol{v} \cdot \nabla\phi$) in hkrati poteka tudi difuzija materiala po tekočini ($\nu\Delta\phi$)
- iščemo ravnovesno stanje
- robni pogoj za ϕ je določen s tokom raztopljenega materiala

$$\frac{d\phi}{dt\, dS} = \kappa|\phi_{\text{wall}} - \phi_{\text{bulk}}|,$$

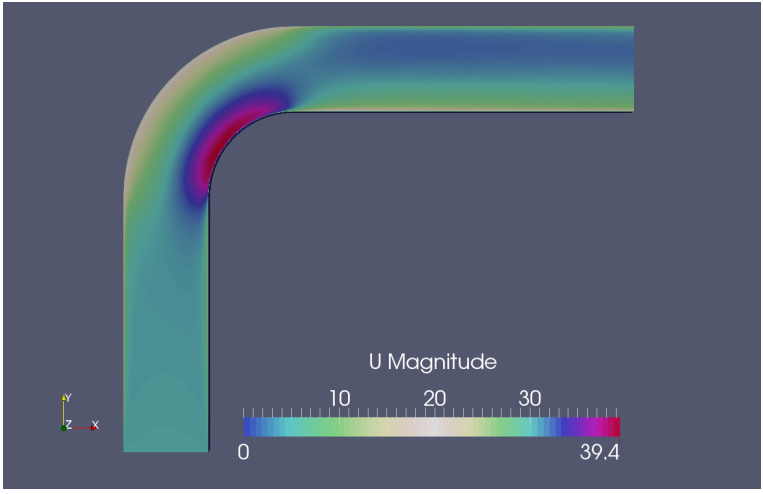
- ϕ_{wall} je koncentracija na robu (vrednost za ϕ v prvi celici ob robu)
- ϕ_{bulk} je saturacijska konstanta
- κ je konstanta topljenja

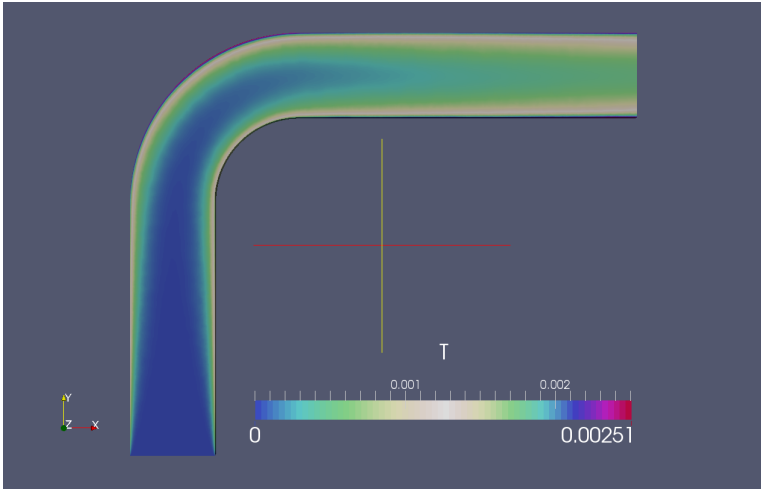
Poganjanje

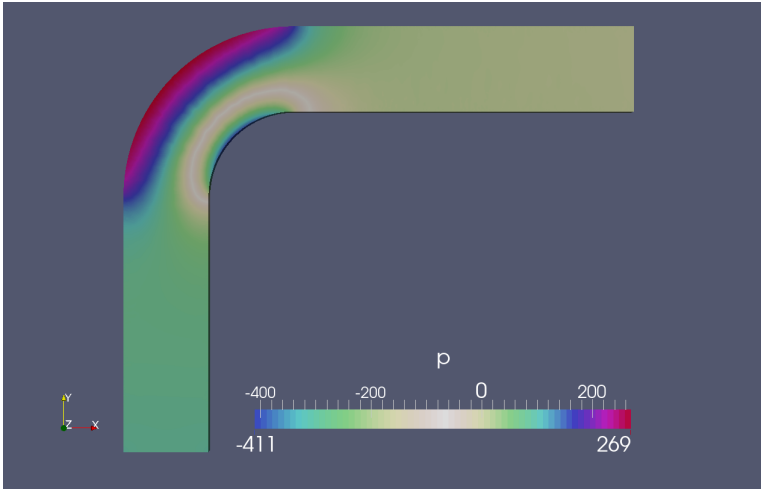
- prevedemo nov robni pogoj
- prevedemo nov solver
- importamo mrežo (fix patches for wall, empty) ali uporabimo staro
- poženemo solver `simpleScalarFoam`
- monitoriramo residuale
  `foamMonitor -l postProcessing/residuals/0/residuals.dat`

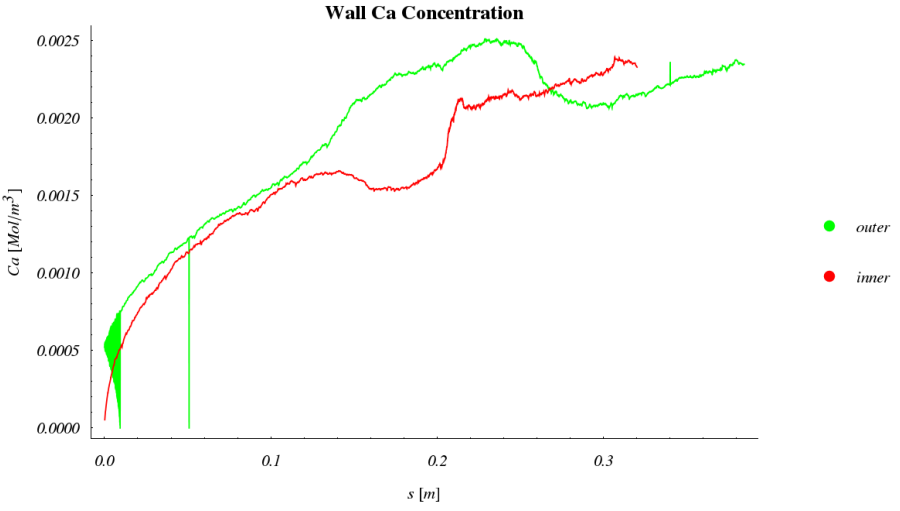
Post procesiranje - analiza novih možnosti

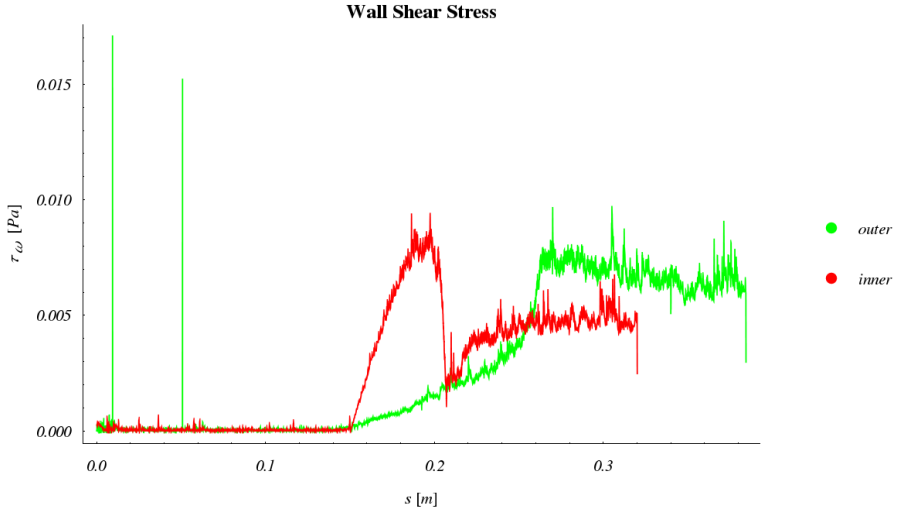
- določitev strižne napetosti na robu (WSS)
- je mogoče erozija ali abrazija povezana z WSS?


U Magnitude
10
20
30
0
39.4
Y
Z
X


T
0.001
0.002
0
0.00251
Y
Z
X


p
-400
-200
0
200
-411
269
Y
Z
X


Wall Ca Concentration
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Ca [Mol/m^3]
0.0
0.1
0.2
0.3
s [m]
outer
inner


Wall Shear Stress
0.015
0.010
0.005
0.000
τ_ω [Pa]
0.0
0.1
0.2
0.3
s [m]
outer
inner

Fenics-project

Kaj je Fenics?

FenicsProject je okolje za reševanje PDE z metodo končnih elementov (FEM).

- **jezik**: razvoj v `c++`, vožnja v `python` (hitrost in enostavna uporaba)
- **paralelizacija**: domain-decomposition + MPI; LA z uporabo PETSc
- reševanje PDE na nivoju variacijske forme (direktna implementacija bilinearne forme)
- širok nabor FE funkcijskih prostorov (izdelava je preprosta, saj je narejen math-engine za delanje novih)
- naravno okolje za testiranje novih idej za FEM
- izjemno dobro prilagojen za računanje na velikih HPC sistemih, ker je vsaka komponenta optimizirana za paralelno računanje!

Enostaven tok tekočine lahko popišemo s Stokesovo enačbo

$$\nabla \cdot \boldsymbol{u} = 0, \quad \text{(nestisljivost)}$$
$$\nu \nabla^2 \boldsymbol{u} - \nabla p + \boldsymbol{f} = 0, \quad \text{(ohranitev gibalne količine)}$$

kjer je $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}) \in [L^2(\Omega)]$ zunanja sila z robnim pogojem

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{0}, & \boldsymbol{x} &\in \Gamma_{\mathrm{b}} \cup \Gamma_{\mathrm{wall}}, \\ \boldsymbol{u} &= \boldsymbol{u}_{\mathrm{in}}, & \boldsymbol{x} &\in \Gamma_{\mathrm{in}}, \\ \boldsymbol{u} &= \boldsymbol{u}_{\mathrm{out}}, & \boldsymbol{x} &\in \Gamma_{\mathrm{out}}. \end{aligned}$$

Primer – toke tekočine (Stokes)

Za reševanje z metodo končnih elementov moramo PDE zapisati v šibki formulaciji.

Problem Poišči takšen $\boldsymbol{u} \in [H_0^1(\Omega)]^d$ in $p \in L_0^2(\Omega)$ da bo zadoščeno

$$\begin{aligned} (\nabla \boldsymbol{u}, \nabla \boldsymbol{v}) - (\nabla \boldsymbol{v}, p) &= (\boldsymbol{f}, \boldsymbol{v}), \quad && \forall \boldsymbol{v} \in [H_0^1(\Omega)]^d, \\ (\nabla \boldsymbol{u}, q) &= 0, && \forall q \in L_0^2(\Omega), \end{aligned}$$

kjer imamo forme določene z

$$(\nabla \boldsymbol{u}, \nabla \boldsymbol{v}) = \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \, \mathrm{d}x, \qquad (\nabla \boldsymbol{v}, p) = \int_\Omega p \, \nabla \cdot \boldsymbol{v} \, \mathrm{d}x,$$

$$(\boldsymbol{f}, \boldsymbol{v}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}x, \qquad (\nabla \boldsymbol{u}, q) = \int_\Omega q \, \nabla \cdot \boldsymbol{u} \, \mathrm{d}x.$$

Primer – toke tekočine (Stokes)

V Fenics okolju sedaj samo zapišemo (s še nekaj dodatki)

```
# Define function spaces
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V * Q

# Define variational problem
(u, p) = TrialFunctions(W)
(v, q) = TestFunctions(W)
f = Constant((0.0, 0.0, 0.0))
a = inner(grad(u), grad(v))*dx + div(v)*p*dx + q*div(u)*dx
L = inner(f, v)*dx

# Compute solution
w = Function(W)
solve(a == L, w, [bc1, bc2])
```

Stokes flow Example @ Fenics

Iz tekočin preidimo sedaj na trdnine, kjer rešujemo problem raztegovanja tanke kvadratne plošče s kvadratno luknjo. Trdnino modeliramo kot linearno elastičen medij, popisan z

$$\begin{aligned} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) &= \boldsymbol{f}, && \text{v } \Omega \\ \boldsymbol{\sigma}(\boldsymbol{u}) &= 2\mu\, \boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I}, && \text{v } \Omega \\ \boldsymbol{\sigma}(\boldsymbol{u}) \cdot \boldsymbol{n} &= \boldsymbol{h}, && \text{na } \Gamma \text{ (Neumann problem!)} \end{aligned}$$

kjer sta $\mu > 0$ in $\lambda \geq 0$ Lamejevi konstanti, $\boldsymbol{n}$ normala na robu, $\boldsymbol{\sigma}$ napetostni tenzor in $\boldsymbol{\epsilon}$ deformacijski tenzor, ki je določen z

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\big(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\top}\big)$$

Kot je razvidno iz robnega pogoja imamo za primer določen samo Neumannov robni pogoj (Traction problem). V tem primeru, ker nimamo Dirichlejevega pogoja, ki bi ploščo stabiliziral, potrebujemo še kompatibilnostna pogoja za vsoto zunanjih sil in navorov

$$\int_\Omega \boldsymbol{f}\,\mathrm{d}x + \int_\Gamma \boldsymbol{h}\,\mathrm{d}x = 0,$$
$$\int_\Omega \boldsymbol{f}\times\boldsymbol{r}\,\mathrm{d}x + \int_\Gamma \boldsymbol{h}\times\boldsymbol{r}\,\mathrm{d}x = 0.$$

Če zadostimo kompatibilnostnim pogojem je rešitev sistema **singularna**, ker je vsaka rešitev lahko tudi rešitev, ki ji prištejemo gibanje togega telesa

$$\boldsymbol{u} \to \boldsymbol{u} + \boldsymbol{w}, \quad \boldsymbol{w} = \boldsymbol{c}_1 + \boldsymbol{R}\boldsymbol{c}_2; \quad \boldsymbol{\epsilon}(\boldsymbol{w}) = 0.$$

Nalogo rešimo lahko z uporabo Lagrangejevega multiplikatorja. Določimo prostor $Z \subset V = [H^1(\Omega)^3]$, kot prostor vseh togih gibanj v Ω. Vse rešitve u morajo biti linearno neodvisne od funkcij iz Z. Uvedemo lagrangejev multiplikator $p \in Z$ tako, da so vse rešitve u ortogonalne glede na Z. Sedaj imamo celotno formo določeno kot

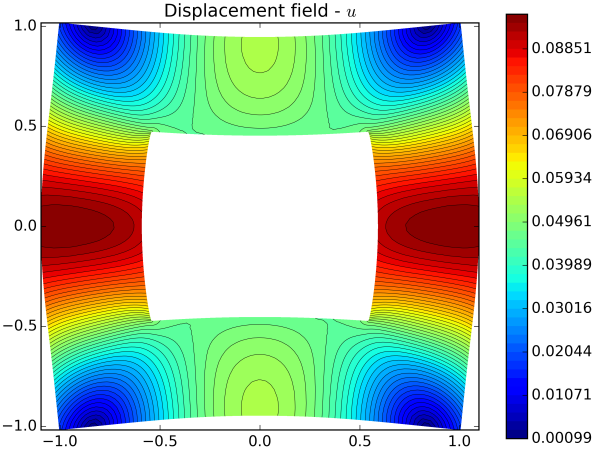
$$\begin{aligned} 2\mu(\epsilon(u), \epsilon(v)) + \lambda(\nabla \cdot u, \nabla \cdot v) - (p, v) &= (f, v) + (h, v)_\Gamma \quad && \forall v \in V \\ -(u, q) &= 0 && \forall q \in Z, \end{aligned}$$

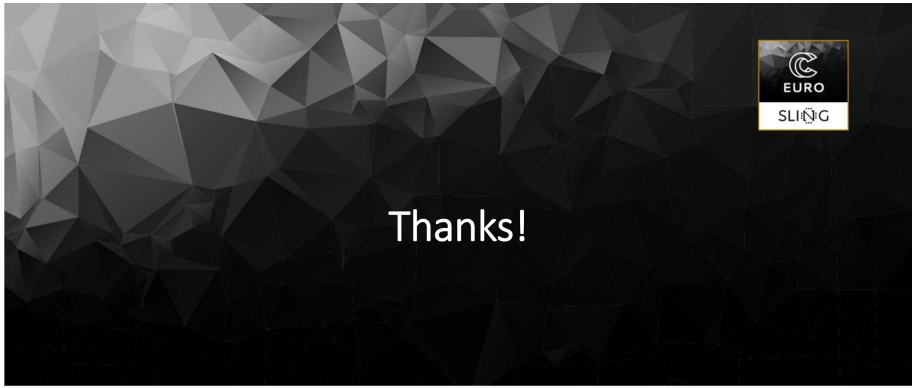
Zgornji sistem se izrazi kot ”saddle-point” problem in za tem stoji izrek, da obstaja rešitev in ta rešitev je edina.

Ne preostane drugega kot uporaba FEM@Fenics in poiskat rešitev! ☺

Prikaz izvedbe sledi uporabi na Jupyter serverju!

jhub@baracuda





This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, United Kingdom, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, Republic of North Macedonia, Iceland, Montenegro

EuroHPC
Joint Undertaking