FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Bayesovská inverze pro specifické geofyzikální úlohy s využitím knihovny PyMc

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Ondřej Šimůnek**
Vedoucí práce: Ing. Pavel Exner, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Bayesovská inverze pro specifické geofyzikální úlohy s využitím knihovny PyMc

Jméno a příjmení:	**Ondřej Šimůnek**
Osobní číslo:	M21000142
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

Cíl práce: Zefektivnit aplikaci Bayesovské inverze pro vybranou geofyzikální úlohu pomocí knihovny PyMC.
Zásady vypracování:

1. Implementujte základní verzi algoritmu Metropolis-Hastings. Seznamte se s principy algoritmu na 1D úloze vedení tepla skrze dvouvrstevný materiál.
2. Seznamte se s knihovnou PyMc, implementujte v ní model z bodu 1.
3. Zprovozněte stejnou úlohu pomocí externího volání a s paralellním samplováním na clusteru.
4. Otestujte funkce PyMC pro diagnostiku konvergence samplovacího procesu.
5. Zprovozněte existující hydro-mechanický model ražby tunelu se simulací měření tlaku v monitorovacích vrtech.
6. Řešte inverzní úlohu identifikace parametrů HM modelu pomocí knihovny PyMc s využitím externího řešiče a paralelního samplování.
7. Aplikujte samplovací algoritmus Multi-level Delayed Acceptance (MLDA) v knihovně PyMc pro akceleraci výpočtu.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] PyMC project, online: https://www.pymc.io/welcome.html

[2] Dodwell, Tim & Ketelsen, Chris & Scheichl, Robert & Teckentrup, Aretha. (2019). Multilevel Markov Chain Monte Carlo. SIAM Review. 61. 509-545. https://doi.org/10.1137/19M126966X

[3] PyMC – MLDA introduction, online: https://www.pymc.io/projects/docs/en/v3.11.4/pymc-examples/examples/samplers/MLDA_introduction.html

[4] Cameron Davidson-Pilon : Bayessian Methods for Hackers, Addison-Wesley Professional, 2015, ISBN: 978-0-13-390283-9, online: https://dataorigami.net/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/

[5] Simona Bérešová: Bayesian approach to the identification of parameters of differential equations, PhD Thesis, 2021, online: https://dspace.vsb.cz/bitstream/handle/10084/148521/DOM0015_FEI_P1807_1103V036_2022.pdf?sequence=1&isAllowed=y

Vedoucí práce:	Ing. Pavel Exner, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Chaloupka, Ph.D.
garant studijního programu

V Liberci dne 19. října 2023

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.

11. května 2024 Ondřej Šimůnek

Poděkování

Děkuji vedoucímu práce doktorovi Pavlovi Exnerovi za bohatou podporu, zpětnou vazbu a pravidelné konzultace. Zároveň děkuji docentovi Janovi Březinovi za příležitostné konzultace a nabídku této práce. Nakonec bych rád poděkoval e-INFRA CZ za poskytnutí potřebných výpočetních prostředků pro realizaci práce.

Bayesovská inverze pro specifické geofyzikální úlohy s využitím knihovny PyMc

Abstrakt

Tato bakalářská práce se zaměřuje na implementaci Bayesovské inverze pro určení parametrů hydromechanického modelu. Výsledkem je aplikace propojující hydromechanickou simulaci v Flow123d s Bayesovskou inverzí s využitím moderních knihoven jazyka Python. Kvůli vysoké výpočetní náročnosti těchto modelů je jedním z cílů aplikace vysoká paralelnost a možnost efektivního nasazení na výpočetní cluster. Aplikace nativně podporuje nasazení přes službu Metacentrum a plánovací systém OpenPBS. Výsledek inverze se ukládá do standardizovaného formátu ArviZ InferenceData, který má nativní podporu pro vizualizaci výsledků. Součástí aplikace je rozhraní umožňující ladit parametry inverze a výběr veličin, pro které se inverze má provést. Práce řeší některé nedostatky existující implementace a tvoří použitelnou alternativu.

Klíčová slova: Python, statistika, Bayesovská inverze, hydromechanická simulace, paralelizace, výpočetní cluster

Bayesian inversion for specific geophysical problems using PyMC library

Abstract

This bachelor thesis focuses on implementing Bayesian inversion to determine parameters of a hydromechanical model. The result is an application that connects hydromechanical simulation in Flow123d with Bayesian inversion using modern Python libraries. Due to the high computational demands of these models, one of the application's goals is high parallelism and the ability for efficient deployment on a computing cluster. The application natively supports deployment via the Metacentrum service and the OpenPBS scheduling system. The inversion result is stored in the standardized ArviZ InferenceData format, which has native support for result visualization. The application includes an interface for tuning inversion parameters and selecting variables for which the inversion should be performed. The thesis addresses shortcomings of existing implementations and provides a usable alternative.

Keywords: Python, statistics, Bayesian inversion, hydromechanical simulation, paralelism, computing cluster

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

TUL	Technical University of Liberec
NTI	Institute of New Technologies and Applied Informatics
MCMC	Markov Chain Monte Carlo
MH	Metropolis Hastings
ESS	Effective Sample Size
NUTS	No-U-Turn Sampler
DEMZ	Differential Evolution Metropolis Hastings
MLDA	Multi-Level Delayed Acceptance
OS	Operating System
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
CESNET	Czech Education and Scientific NETwork
YAML	YAML Ain't Markup Language
CSV	Comma Separated Values
SSH	Secure SHell

1 Úvod

Použití Bayesovských modelů představuje silný nástroj, který současně nachází využití v mnoha různých oborech včetně genetiky, strojového učení, předpovědi akciového trhu i počasí. Principem je, že máme nějakou původní domněnku o určitém jevu a tuto domněnku na základě nových informací upravujeme.

Jedním oborem, kde lze tento přístup použít, je při zjišťování vhodnosti podzemního úložiště pro sklad radioaktivního odpadu. Tento proces je velice složitý a skládá se z mnoha dílčích úloh – jednou z těchto úloh je určit, jak se bude hornina kolem úložiště mechanicky chovat *po* uložení radioaktivního odpadu. Máme k tomu informace o tom, jak se hornina chová *před* uložením odpadu.

K simulaci horniny je potřeba znalost hydromechanických parametrů horniny. V této úloze jsou ale hydromechanické parametry neznámé. Máme tedy model, který umí určit chování horniny s určitými parametry, ale my potřebujeme *opačný* model, který umí určit parametry na základě chování horniny.

Zde můžeme použít Bayesovskou inverzi – využít Bayesovského modelu pro zjištění neznámých parametrů. Začneme s nějakým počátečním odhadem pro hodnotu každého neznámého parametru a tyto odhady budeme průběžně vylepšovat, dokud se nedostaneme blízko k výsledku. Tento proces je ovšem výpočetně náročný – budeme chytře navrhovat sady parametrů a testovat, jak moc se shoduje chování těchto parametrů s chováním horniny. I když se sady parametrů vybírají chytře, tak se bude muset provést simulace pro velké množství různých parametrů, než se dosáhne výsledku. Jedna simulace může trvat řádově vteřiny až minuty, takže výpočetní náročnost stovek až tisíců simulací bude velice vysoká.

Pro efektivní použití Bayesovské inverze je potřeba velké množství výpočetního výkonu. Může se zde využít více paralelních vláken, kde každé bude dělat vlastní inverzi a jejich výsledky se na konci sloučí. Tímto způsobem bude možné provozovat paralelně tolik inverzí, co je jader na procesoru. Stejný nápad lze rozšířit dále a využít několika počítačů ve formě počítačového clusteru a provozovat několikanásobně více inverzi paralelně.

Další přístup pro snížení výpočetní náročnosti je použití náhradních modelů. Tyto modely jsou výrazně méně výpočetně náročné a slouží jako filtr, který odfiltruje špatně vybrané sady parametrů. Díky tomu se neplýtvá výpočetní výkon použitím původního modelu u sad parametrů, kde to nedává smysl. Této technice se říká Delayed Acceptance, případně Multi-Layer Delayed Acceptance (MLDA).

Určení správnosti výsledků inverze je důležitou součástí Bayesovské inverze. Prvotní výstup z inverze je velká množina dat, která obsahuje všechny sady parametrů z inverze. Samotná množina bez žádného zpracování moc informací v sobě nenese.

Užitečná informace z této množiny je rozdělení těchto parametrů - jak často jsou v množině parametry z určitých regionů, neboli výpočet histogramu množiny výsledků. Vysoké zastoupení hodnot v histogramu představují hodnoty, kde si je inverze více jistá, že jsou správné.

Pro Bayesovskou inverzi s hydromechanickými modely již existuje jedno hotové řešení. Řešení využívá knihovny surrDAHM pro inverzi a Flow123d pro simulaci horniny. Toto řešení má však několik nedostatků, včetně problémů při použití výpočetního clusteru a složité konfigurace.

Cílem práce je tedy vytvořit aplikaci, která je schopná provést Bayesovskou inverzi pro hydromechanický model. Aplikace by měla být schopná využít výpočetního clusteru pro zrychlení inverze. Zároveň by aplikace měla využívat techniky MLDA pro omezení plýtvání výpočetního výkonu. Výsledky inverze by měli být jednoduše interpretovatelné a dobře vizualizované. Důležitým bodem je také zohlednit nedostatky existujícího řešení, včetně obtížné konfigurace a jiných technických omezení.

Kapitola 2 slouží pro přiblížení Bayesovské inverze a realizace. Použitý model a simulátor Flow123d je více popsaný v kapitole 3. Seznam použitých nástrojů pro tvorbu aplikace a jejich stručný popis je v další kapitole 4. Náplň kapitoly 5 je porovnání existujících implementací Bayesovské inverze na jednoduché úloze. V kapitole 6 je popsaný proces tvorby výsledné aplikace, včetně klíčových bodů jako paralelizace v 6.4 a kontejnerizace v 6.3. Kapitola 7 obsahuje prezentaci výsledků inverze a diskuzi o jejich správnosti. V poslední kapitole 8 je práce porovnaná s existujícím řešení.

2 Bayesovská inverze

Model představuje libovolnou transformaci, jejíž výstup je určen vstupnímy parametry. Pojem *inverze* je postup vytvoření modelu, který představuje opačnou transformaci vůči jinému dopřednému modelu. Inverzní model je tedy schopný určit k jaké vstupní hodnotě koresponduje určitá výstupní hodnota. Vytvoření inverzního modelu je netriviální úloha - jedním z problémů je fakt, že více různých vstupních parametrů může mít stejný výstup. Jaká z hodnot bude tedy výstupem inverzního modelu?

Jeden možný postup je přímá inverze, která hledá jednu nejlepší sadu vstupních parametrů (best fit) s výstupem nejpodobnější ke známému ve zvolené míře.

Bayesovská inverze je alternativa a naopak hledá všechny sady parametrů, které mají výstup dostatečně podobný známé hodnotě. Tato dostatečná podobnost modeluje nejistotu ve známé hodnotě – předpokládá se zatížení výstupu určitou úrovní šumu. Zároveň znalost celého pravděpodobnostního rozdělení vstupních veličin může být často důležitější informace, než znát pouze nejlepší hodnoty.

Bayesovská inverze vychází z Bayesova vzorce a implementuje se pomocí Markov Chain Monte Carlo (MCMC) metod. Výpočetní náročnost těchto metod se odvozuje převážně od náročnosti dopředného modelu. Často se musí dělat velké množství iterací inverze pro dosažení dobrých výsledků, takže s náročným modelem může inverze trvat velice dlouho.

2.1 Bayesův vzorec

Bayesův vzorec definuje vztah mezi podmíněnými pravděpodobnostmi. Lze ho chápat jako způsob aktualizace pravděpodobnosti nějaké domněnky na základě nových informací. Byl původně zformulován Thomasem Bayesem v 18. století a současně nachází široké využití v rámci Bayesovské statistiky a pravděpodobnosti. Příklady aplikací Bayesova vzorce jsou v genetice, umělé inteligenci a předpovědi akciového trhu.

Znění vzorce je následující: Nechť R, G jsou náhodné jevy, $P(G)$ je pravděpodobnost jevu G, $P(G|R)$ je podmíněná pravděpodobnost jevu G za předpokladu, že jev R nastane. Pak lze definovat $P(G|R)$ následovně:

$$P\left(G|R\right) = \frac{P\left(R|G\right) * P\left(G\right)}{P\left(R\right)} \tag{2.1}$$

Neznámá R představuje původní domněnku a G představuje novou informaci, co tuto domněnku upravuje. $P\left(G\right)$ se jmenuje *proposal* a představuje pravděpodobnost nových informací, $P\left(R\right)$ je *prior* a představuje pravděpodobnost původní domněnky. Podmíněná pravděpodobnost $P\left(R|G\right)$ je pak *likelihood*, který představuje v kontextu inverze úroveň šumu modelu.

2.2 Markov Chain Monte Carlo metody

Markov chain Monte Carlo metody (zkráceně MCMC metody) jsou algoritmy, které využívají rozdělení stavů Markovova řetězce pro aproximaci náhodného rozdělení. Používají se typicky pro aproximaci složitých a často vícerozměrných náhodných rozdělení.

Metropolis-Hastings (zkráceně MH) je jedna z často používaných MCMC metod. Její princip spočívá v aproximaci neznámého rozdělení pomocí rozdělení stavů Markovova řetězce. Předpokladem je, že tento použitý řetězec bude mít stacionární rozdělení stavů úměrné neznámému rozdělení, neboli stejné až na normalizační konstantu. Pak lze s dostatečným množstvím iterací algoritmu se limitně přibližovat neznámému rozdělení.

Máme naměřená data O odpovídající výstupu nějakého modelu. Zároveň máme dopředný model F, který umí pro sadu vstupních parametrů S vytvořit obdobný výstup $F\left(S\right)$. Tyto dvě sady hodnot lze porovnávat a lze určit jejich podobnost. Můžeme tedy pro libovolnou sadu vstupních parametrů modelu říci, jak moc je jejich výstup modelu podobný naměřeným hodnotám. K tomu máme náhodné rozdělení původního odhadu R. Nakonec je potřeba rozdělení G, ze kterého je možné vygenerovat novou sadu vstupních parametrů v závislosti na jiné sadě vstupních parametrů.

Na začátku algoritmu se určí počáteční odhad S^0 z apriorního rozdělení R. V každém dalším kroku t se určí nový návrh vstupních parametrů $S^{'}$, který je z návrhové hustoty G a závislý na současných parametrech $S^{(t)}$:

$$S^0 \in R$$
$$S^{'} \in G\left(S^{(t)}\right)$$

Pro navržené vstupní parametry $S^{'}$ a poslední přijaté vstupní parametry $S^{(t)}$ se vypočítá jejich pravděpodobnostní hustota $G_{pdf}\left(S^{'}\right)$ a $G_{pdf}\left(S^{(t)}\right)$ v návrhové hustotě G. Pro obě sady vstupních parametrů se také vypočítá výsledek z dopředného modelu a vyčíslí se jejich úroveň šumu v $P_{pdf}\left(F\left(S^{'}\right)\right)$ a $P_{pdf}\left(F\left(S^{(t)}\right)\right)$. Z těchto informací pak lze vypočítat pravděpodobnost, se kterou se navržené vstupní parametry přijmou. Vzorec pro tuto pravděpodobnost lze vidět v 2.2.

$$p_{accept} = \min\left(1, \frac{G_{pdf}\left(S'\right)}{G_{pdf}\left(S^{(t)}\right)} \frac{P_{pdf}\left(F\left(S'\right)\right)}{P_{pdf}\left(F\left(S^{(t)}\right)\right)}\right) \tag{2.2}$$

Existují v tomto vzorci také souvislosti s Bayesovým vzorcem v 2.1 – úroveň šumu vstupních parametrů $P_{pdf}\left(F\left(S'\right)\right)$ je *likelihood* a vynecháním členu $P_{pdf}\left(F\left(S^{(t)}\right)\right)$ bude zlomek představovat *posterior*. S' představuje člen G ve vzorci 2.1 a $S^{(t)}$ představuje R ve stejném vzorci.

Hodnota p_{accept} z 2.2 se porovná s náhodnou pravděpodobností $U_{draw}\left(0,1\right)$ a pokud je náhodná pravděpodobnost menší, návrh se přijme a nastaví se $S^{(t+1)} = S'$. V opačném případě se návrh zamítne a poslední přijatý návrh se nemění, tzn. $S^{(t+1)} = S^{(t)}$. Hodnoty $S^{(t+1)}$ v každé iteraci si algoritmus ukládá a z nich se pak vytváří aproximace neznámého rozdělení. Algoritmus pokračuje do dosažení nějaké koncové podmínky, typicky množství přijatých návrhů nebo množství všech návrhů.

Způsob, jakým se vybírá v každé iteraci návrh, se jmenuje návrhová hustota. Různé návrhové hustoty mohou výrazně změnit chování a výsledky inverze. Některé návrhové hustoty mohou být adaptivní – během inverze mohou průběžně měnit své parametry (většinou rozptyl) na základě aktuálních výsledků.

Výsledkem je množina sad parametrů, jejíž rozdělení hodnot je do určité míry podobné neznámému rozdělení. Míra podobnosti záleží na mnoha parametrech jako velikost množiny, množství šumu, původním odhadu atd. výsledky před dosažením stacionárního stavu nejsou žádoucí, proto je cílem tyto hodnoty vyřadit z výsledku (burn-in).

Určit správnou konvergenci algoritmu je netriviální úloha a může se k tomu použít různých metod a metrik. Jednou z těchto metrik je $\hat{R}$ (r-hat)[1], který vyčísluje míru korelace a autokorelace v rámci rozdělení. Další podobnou metrikou je Effective Sample Size (ESS) – určuje, kolika návrhů z kompletně nezávislého náhodného rozdělení odpovídá výsledkům, neboli jak moc korelace snižuje množství informací ve výsledcích. Další možností je vizualizace dat pomocí relevantních grafů.

Algorithm 1 Metropolis-Hastings

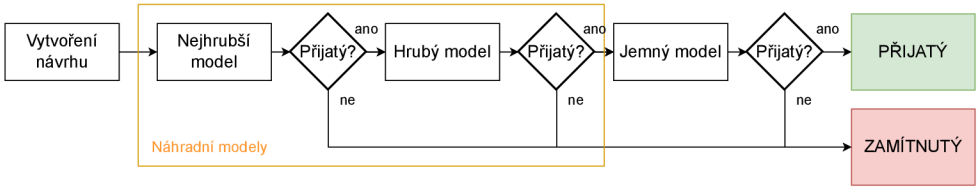
Require: $S^0 \in R$, $Draws > 1$

$S^{(t)} \leftarrow S^0$
for $draw = 1 : Draws$ **do**
  $S' \in G\left(S^{(t)}\right)$
  $p_{accept} \leftarrow \min\left(1, \frac{G_{pdf}\left(S'\right)}{G_{pdf}\left(S^{(t)}\right)} \frac{P_{pdf}\left(F\left(S'\right)\right)}{P_{pdf}\left(F\left(S^{(t)}\right)\right)}\right)$
  if $U_{draw}\left(0,1\right) < p_{accept}$ **then**
    $S^{(t+1)} \leftarrow S'$
  else
    $S^{(t+1)} \leftarrow S^{(t)}$
  end if
end for

2.3 MLDA

Delayed Acceptance (DA) a Multi-Level Delayed Acceptance (MLDA) jsou pokročilé metody Bayesovské inverze [2, 3, 4]. Metody se využívají převážně u výpočetně náročných modelů, kde jejich použití výrazně zvyšuje efektivitu inverze. Princip metod je využít náhradní model(y), který je méně výpočetně náročný a umí zamítnout většinu špatných návrhů. Původní dopředný model se v tomto kontextu jmenuje jemný model, náhradní modely se jmenují hrubé modely. V rámci náhradních modelů je hierarchie – nejméně náročný (nejhrubší) model se testuje jako první. V rámci hierarchie modelů postupně stoupá výpočetní náročnost – nejhrubší model má nejnižší výpočetní náročnost, jemný model má nejvyšší výpočetní náročnost.

Pokud nejhrubší model nezamítne návrh, tak se návrh propaguje hierarchií modelů a v každé další úrovni se testuje. Pokud se v žádném modelu nezamítne a dostane se až na jemný model, návrh se může přijmout. Hrubé modely slouží jako filtr a protože jsou méně výpočetně náročné, tak to vylepšuje efektivitu inverze – neplýtvá se výpočetní výkon použitím jemného modelu na návrhy, které jsou určitě špatné.



Obrázek 2.1: MLDA diagram pro tři úrovně – 2 hrubé modely a jemný model

Samozřejmě MLDA není zadarmo a přináší několik vlastních problémů. Prvním problémem je určit, co vlastně budou náhradní modely. Zde existují dva přístupy – náhradní modely, které souvisí s jemným modelem a náhradní modely co málo souvisí nebo nesouvisí s jemným modelem.

Příklad prvního přístupu je například použití hrubší sítě pro konečno-prvkovou simulaci. Simuluje se stejný jev, jako v jemném modelu, ale díky menší výpočetní síti je výpočet rychlejší. Výsledek simulace bude jiný, ale pořád bude relativně podobný jemnému modelu. Zde jsou rozdíly výpočetní náročnosti mezi jemným a náhradním modelem typicky menší oproti druhé metodě.

Druhý přístup už je složitější, protože se musí definovat model co bude efektivně zamítat špatné návrhy, ale nebude moc propojený s existujícím jemným modelem. Těmto modelům se říká *meta* modely. Motivace tohoto přístupu je, že tyto modely mohou být daleko výpočetně rychlejší oproti prvnímu přístupu. Příklad takového přístupu je polynomiální aproximace v [5] nebo použití neuronových sítí.

Další výzva s MLDA je potřeba ladit větší množství parametrů. Nejenom že musíme definovat více modelů se svými parametry, ale ke každému mudelu také korespondují parametry k zamítání a přijímání návrhů. Je možné použít pro všechny modely parametry stejné, ale povede to na horší výsledky.

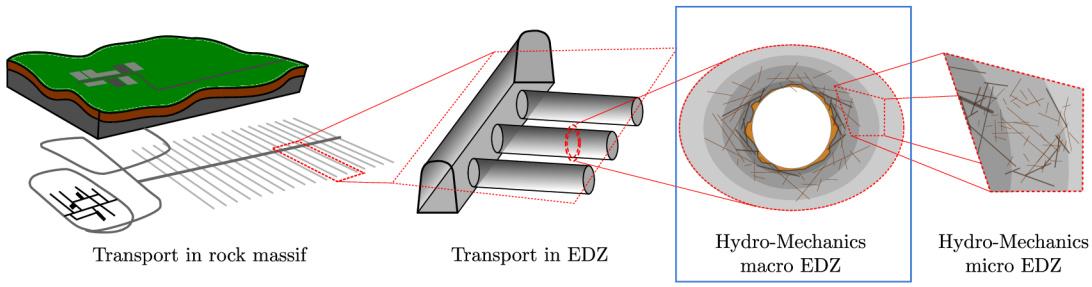
MLDA je silný nástroj, ale jeho použití není vždy žádoucí. Naopak v extrémních případech může být jeho použití detrimentální k inverzi a vést ke zpomalení a zhor-

šení výsledků. I v případech, kde jeho použití může být příznivé, může špatný výběr parametrů pro MLDA inverzi zhoršit.

3 Flow123d a kontext modelu

Modely, kterými se tato práce bude zabývat, budou simulace pomocí software Flow123d [6]. Flow123 je simulátor pro simulaci porézních podzemních hornin. Využije se pro simulaci hydromechanické úlohy, která bude představovat dopředný model.

Širší kontext tohoto modelu je skladování radioaktivního odpadu. Dílčí úloha tohoto procesu je určení hydromechanických parametrů v blízkém okolí ukládacích chodeb. Simulace napodobuje opravdovou ražbu, která v minulosti proběhla. Z této ražby jsou známé hodnoty dokumentované v [7] ve tvaru vývoje tlakové výšky v čase. Data jsou z několika bodů v hornině a použijí se v inverzi. Zároveň jsou známé rozměry a tvar tunelu, které se použijí pro simulaci, a budou tvořit dopředný model – vstupem jsou hydromechanické parametry horniny, výstupem časová řada tlakových výšek v určitém bodě horniny.



Obrázek 3.1: Širší kontext modelu [8]

Součástí práce není studie použitého modelu – lze se na model dívat pouze jako obecnou transformaci, u které není známé jak, vnitřně funguje. Každopádně popis modelu je užitečná informace pro uvedení práce do širšího kontextu.

Konkrétní řešenou úlohou je ražba tunelu modelována prostřednictvím okrajové podmínky na stěně tunelu. Ražba probíhala prvních 17 dní, poté následuje relaxace horninového masivu (odezva změny způsobené ražbou). Nad tunelem dochází k výraznému nárůstu hydraulického tlaku vlivem komprese, na boku naopak k poklesu. Používáme Rutqvistův [7] vztah pro závislost permeability na napětí, avšak s fyzikálně interpretovatelnými parametry (3.1):

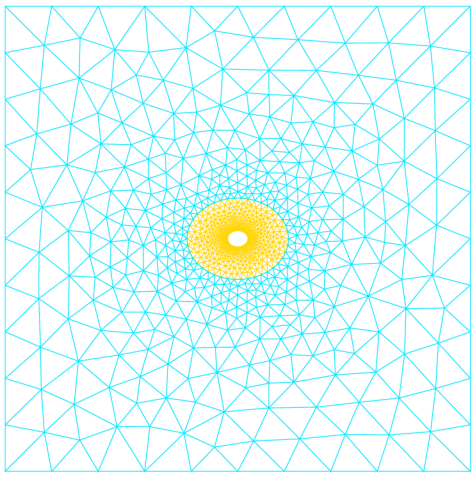
$$\kappa = \left[\kappa_r + \kappa_\delta \exp\left(\log\left(\frac{\kappa_0 - \kappa_r}{\kappa_\delta}\right)\frac{\sigma_m}{\sigma_0}\right)\right] \exp\left(\gamma \frac{\max\{0, \sigma_{\mathrm{VM}c} - \sigma_c\}}{\sigma_c}\right), \tag{3.1}$$

kde:

- σ_0 je počáteční napětí,
- σ_m je objemové napětí,
- σ_{VM} - von Misesovo napětí,
- σ_c - kritické smykové napětí

a parametry nelineární permeability:

- κ_0 - permeabilita neporušené horniny (před ražbou)
- κ_δ - teoretická permeabilita horniny při nulovém zatížení
- κ_r - minimální (reziduální) permeabilita
- γ - bezrozměrný faktor určující velikost vlivu smykového napětí (při překročení kritického)



Obrázek 3.2: Výpočetní síť modelující hydromechanickou úlohu

Model je relativně výpočetně náročný – dokončení jedné simulace trvá řádově desítky sekund. Náročnost výrazně kolísá na základě volby vstupních parametrů kvůli použití iterativního řešiče v modelu – pro některé sady parametrů simulace rychle konverguje, pro jiné to trvá až několikanásobně déle. Velká výpočetní náročnost vede na řešení s optimalizacemi ve formě MLDA a paralelizace.

Inverzi lze provádět pro vybranou množinu hydromechanických parametrů. Čím více parametrů je v inverzi, tím větší je rozměr náhodného prostoru a tím náročnější inverze bude. Vyšší náročnost se zde projeví ve větším množství potřebných iterací pro dosažení stacionárního rozdělení.

4 Použité nástroje

V této sekci jsou popsané všechny důležité nástroje použité pro vytvoření této práce. Je potřeba implementovat inverzi, aplikaci zabalit do kontejneru a nasadit na výpočetní cluster. Kontejnerizace aplikace je žádoucí z důvodu zahrnutí knihovny Flow123d a ostatních závislostí do jednotného prostředí, které pak půjde efektivně nasadit i na výpočetní cluster. Detaily jsou v sekci 6.3. Potřeba využít výpočetní cluster vychází z výpočetní náročnosti modelu, detaily paralelizace aplikace jsou v 6.4.

4.1 Implementace inverze

`PyMC`[9] je jedna ze dvou použitých Python knihoven, které obsahují hotovou implementaci Bayesovské inverze. Knihova vnitřně využívá PyTensor pro optimalizaci výpočtů a obsahuje několik robustních algoritmů pro inverzi jako NUTS [10] nebo DEMZ [11, 12]. Příklad použití knihovny je vidět v 5.3 a také v návodech a příkladech přímo od PyMC.

Současně je nejnovější verze PyMC 5.13, ale podpora MLDA existovala pouze do verze 4.0. Po této verzi se MLDA mělo přesunout do vedlejší větve PyMC Experimental, ale dosud se tak nestalo a podpora MLDA v současných verzích není. Tento fakt je hlavní důvod proč se PyMC nakonec nepoužil pro realizaci práce, i když to původně bylo v zadání.

`TinyDA`[13] je druhá použitá knihovna s implementací Bayesovské inverze. Je relativně nová (vytvořena začátek 2021) a tvůrcem knihovny je jeden z vývojářů PyMC. Knihovna představuje minimalistickou implementaci inverze, ale pořád obsahuje všechny potřebná rozhraní pro realizaci této práce. Narozdíl od PyMC plně podporuje MLDA i jiné pokročilé metody inverze. Příklad použití knihovny je dostupný v 5.4 nebo GitHubu knihovny.

Knihovna `Arviz`[14] slouží pro vizualizaci a diagnostiku Bayesovských modelů. Vizualizace je dostupná ve formě mnoha různých grafů – například `posterior_plot` zobrazí výsledné rozdělění parametrů z inverze a `trace_plot` zobrazí cesty řetězců v náhodném prostoru a jakých dosahovali hodnot. Grafy lze generovat s použitím `matplotlib` nebo `bokeh`.

Další důležitý nástroj, který ArviZ poskytuje, je formát dat `InferenceData`. Tento formát je navržený přímo pro výsledky z Bayesovské inverze a slouží pro definici společného rozhraní pro aplikace, které generují výsledky Bayesovských modelů. Plné specifikace formátu jsou dostupné na [15].

4.2 Kontejnerizace aplikace

`Docker` je platforma na virtualizaci a kontejnerizaci aplikací. Výhodou kontejnerizované aplikace je oddělení závislostí aplikací od operačního systému, na kterém aplikace běží. Aplikace může běžet na libovolném systému za předpokladu, že systém podporuje Docker – není nutné řešit kompatibilitu aplikace a jejích závislostí s jednotlivými OS.

V práci je použitý vlastní Docker obraz, který se odvozuje od jednoho z Flow123d obrazů. Referenční docker obraz obsahuje knihovnu Flow123d již nainstalovanou a nakonfigurovanou.

`Singularity`, obdobně jako Docker, je nástroj pro kontejnerizaci aplikací. Narozdíl od Dockeru jsou ale cílová skupina výpočetně náročné aplikace, které se typicky spouští na výpočetních clusterech. Singularity umí pracovat s Docker obrazy a umí komunikovat s Dockerhubem a jinými repozitáři. Hlavní rozdíly vůči Dockeru jsou:

- Omezená práva – výpočetní clustery mají velké množství uživatelů, nelze mít root práva
- Nepoužívá daemon – lze lépe monitorovat CPU/RAM využití, prakticky nutné u plánovacích systémů pro vynucení limitů u výpočetních prostředků

4.3 Nasazení a paralelizace aplikace

Knihovna `Ray`[16] slouží pro škálování strojového učení a Python aplikací obecně. TinyDA tuto knihovnu interně používá pro umožnění běhu inverze na více jádrech. Ray umožňuje specifikovat množství výpočetních prostředků, včetně CPU, RAM a i GPU. V případě použití více počítačů k spuštění aplikace umí Ray navázat komunikaci mezi počítači na určeném portu a skupinu počítačů koordinovat. Na každém počítači opět lze omezit množství dostupných výpočetních prostředků.

Relevantním pojmem je zde koncept objektů typu `Actor`. Tento pojem označuje obal nad Python objektem, který umožňuje tento objekt oddělit do vlastního vlákna. S tímto vláknem lze následně komunikovat a s objektem pracovat. Komunikace je asynchronní a funguje pomocí systému zpráv, které se postupně vykonávají v pořadí, ve kterém přišly. Na odpověď actora lze počkat v případě, že je nutné odpověď znát před pokračováním. TinyDA používá jednoho actora na každý řetězec, takže každá instance beží ve vlastním vlákně.

`Metacentrum`[17] je služba poskytovaná společnosti CESNET. Služba poskytuje uživatelům přístup k výpočetním prostředkům ve formě výpočetních clusterů. Uživatelé jsou zaměstnanci a studenti akademických a výzkumných organizací v České Republice.

Služba obsahuje několik komponent, které dohromady tvoří jeden systém:

- Frontendy - části výpočetních clusterů, které umožňují uživatelům se přihlásit do systému a zpřístupňují uživatelům dlouhodobé úložiště

- Systém PBS - umožuje uživatelům žádat o výpočetní prostředky, specifikuje co se má po přidělení prostředků spustit/udělat
- PBS servery - servery, které se starají o sbírání uživatelských žádostí o výpočetní prostředky a přidělování těchto prostředků na výpočetních clusterech
- Fronty úloh - místa, kde se řadí uživatelské žádosti o výpočetní prostředky
- Výpočetní prostředky - výpočetní clustery dostupné pro uživatelské využití, některé jsou s omezeným přístupem

`Charon` je výpočetní cluster, který vlastní ústav NTI na faktultě FM TUL. Cluster je součástí služby Metacentrum. Obsahuje celkem 24 uzlů – 20 uzlů je veřejně dostupných a 4 jsou dostupné pouze pro určité uživatele. Každý uzel obsahuje dva procesory Intel Xeon Silver 4114 (10+10 jader), 96 GB RAM a 480 GB SSD úložiště. Uzly jsou propojené technologií Intel InfiniBand.

Ke clusteru jsou na Metacentru přístupné dvě fronty úloh. Fronta `charon` poskytuje uživatelům prioritní přístup k veřejným uzlům a fronta `charon_2h` zpřístupňuje uživatelům 4 neveřejné uzly. U druhé fronty jsou úlohy omezeny na výpočetní čas do dvou hodin. Přístup k těmto frontám musí být uživatelům udělen individuálně. Díky těmto frontám je nasazování aplikace na cluster výrazně jednoduší – čas na přidělení prostředků je výrazně zmenšený a ladění takto složitých výpočtů výrazně rychlejší.

5 Porovnání implementací inverze

V této kapitole je obsaženo porovnání dvou existujících implementací Bayesovské inverze v jazyku `Python`. Knihovny jsou `PyMC` a `tinyDA`. K porovnání je ještě zahrnutá jednoduchá vlastní implementace. Kapitola slouží pro propojení teorie Bayesovské inverze s praktickou úlohou. Jednoduchá povaha úlohy umožňuje vytvořit několik různých implementací za využití různých knihoven, variací algoritmu a parametrů. Výsledky různých implementací lze pak porovnat a diskutovat o jejich interpretaci a významu.

5.1 Zadání

Zadání úlohy je z práce [5] v sekci 3.2 a doplněná tabulkou 3.1. Úloha řeší tok tepla v jednorozměrném řezu chladičem. Chladič je složen ze dvou dotýkajících se materiálů s různou tepelnou vodivostí u_1 a u_2. Na jednom konci chladiče do něj proudí teplo, na druhém se měří teplota G. Zanedbává se přenos tepla z chladiče do okolního vzduchu. Cílem úlohy je zjistit pro jaké hodnoty vodivostí u_1 a u_2 je výstup z dopředného modelu dostatečně podobný známé naměřené hodnotě. Jinak řečeno – jaké asi byly hodnoty vodivosti u chladiče, ze kterého máme naměřená data.

Analytický model tohoto problému pro teplotu G a vektor vodivostí $u = [u_1, u_2]$ je následující:

$$G(u) = -\frac{1}{80}\left(\frac{3}{\exp(u_1)} + \frac{1}{\exp(u_2)}\right)$$

Tato informace představuje dopředný model F. Zároveň je známá naměřená hodnota $O = -10^{-3}$ a úroveň šumu, která je modelovaná přes náhodné rozdělění $P = \mathcal{N}(\mu = 0,\ \sigma = 2 \cdot 10^{-4})$. Nakonec je známé rozdělění původního odhadu R pro vektor vodivosti u:

$$R : \mathcal{N}_2(\mu, \Sigma)$$
$$\mu = \begin{bmatrix} 5 & 3 \end{bmatrix}$$
$$\Sigma = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}$$

5.2 Porovnání implementací

Porovnají se celkem 4 implementace – Metropolis-Hastings z PyMC pod názvem `PyMC MH`, vlastní implementace Metropolis-Hastings pod názvem `Custom MH` a dvě varianty Metropolis-Hastings z tinyDA; IndependenceSampler pod názvem `tinyDA IS` a GaussianRandomWalk pod názvem `tinyDA RW`.

Pro porovnání různých implementací se použijí 2 různé původní odhady vodivostí. První odhad R_1 bude stejný jako původní odhad v zadání. Druhý odhad R_2 bude mít střední hodnotu dále od regionu vysoké pravděpodobnostní hustoty dopředného modelu a zároveň větší rozptyl jako kompenzaci.

$$R_1 : \mathcal{N}_2\left(\mu_1, \Sigma_1\right), \ \mu_1 = \begin{bmatrix} 5 & 3 \end{bmatrix}, \ \Sigma_1 = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}$$

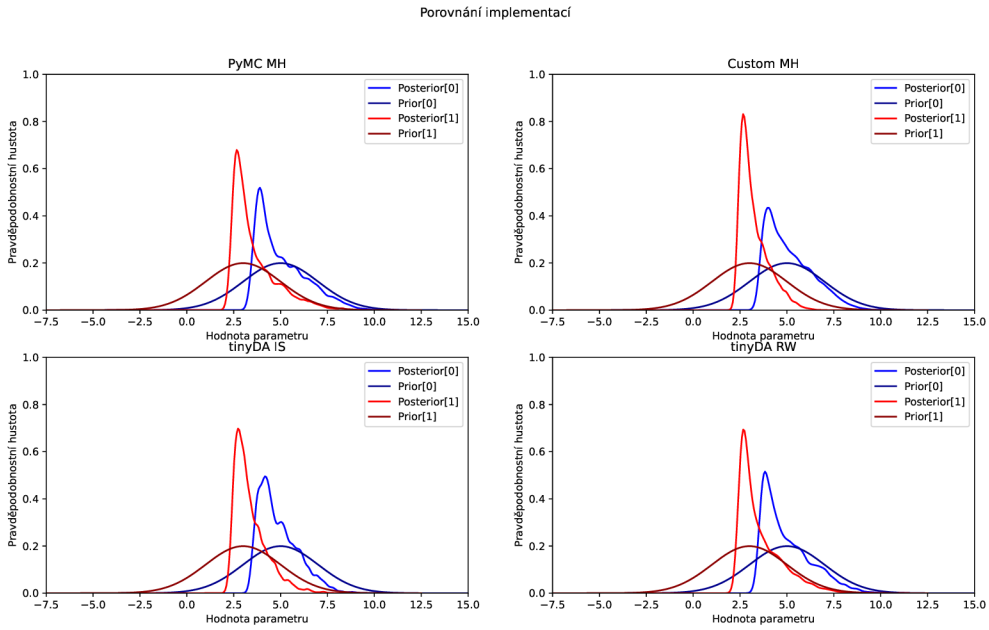
$$R_2 : \mathcal{N}_2\left(\mu_2, \Sigma_2\right), \ \mu_2 = \begin{bmatrix} 8 & 6 \end{bmatrix}, \ \Sigma_2 = \begin{bmatrix} 16 & -2 \\ -2 & 16 \end{bmatrix}$$

Odhad R_1 představuje jistější odhad, R_2 je odhad s menší jistotou. Úroveň jistoty se zde odvozuje od rozptylu rozdělění – větší rozptyl pokryje větší část náhodného prostoru a umožní algoritmu dosáhnout většího množství hodnot, ale pravděpdobnost výběru jakékoli možné hodnoty je menší.

Výsledkem porovnání jsou 2 sady grafů pro každý odhad.

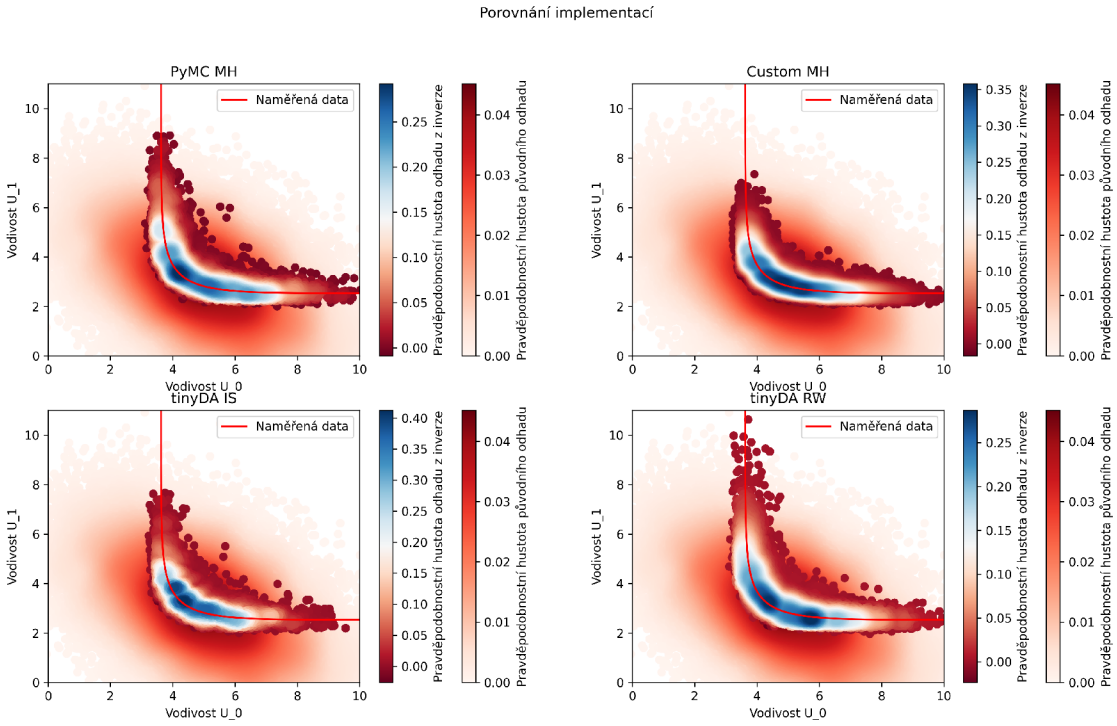
V první sadě grafů 5.1 a 5.3 lze v každém grafu vidět původní odhad pro obě vodivosti `Prior[0]` a `Prior[1]` a výsledek inverze pro obě vodivosti `Posterior[0]` a `Posterior[1]`. Původní odhad a výsledek inverze pro stejný parametr je obarven stejnou barvou. Výsledné křivky jsou vytvořeny interpolací jednotlivých hodnot parametrů z inverze. Tyto rozdělění představujou marginální rozdělění, tzn. nelze vidět korelaci parametrů – viditelné v druhé sadě grafů 5.2 a 5.4. Vodorovná osa představuje hodnotu vodivosti, svislá osa představuje pravděpodobnostní hustotu.

Každý graf v druhé sadě grafů 5.2 a 5.4 obsahuje v pozadí pravděpodobnostní hustotu původního odhadu, v popředí pravděpodobnostní hustotu výsledku inverze a k tomu doprovázející konturu, která zobrazuje vodivosti, pro které je hodnota dopředného modelu přesně rovna naměřeným datům. Tato kontura slouží pro znázornění faktu, že algoritmy se snaží této hodnotě přibližovat a pravděpodobnostní hustota inverze kolem této kontury dosahuje maxima v celém grafu. Vodorovná osa jsou hodnoty první vodivosti, svislá osa jsou hodnoty druhé vodivosti. Graf je obarven pravděpodobnostní hustotou v daném okolí grafu. Samotné hodnoty pravděpodobnostní hustoty nejsou tak relevantní, spíš jejich vzájemný poměr mezi různými částmi grafu je důležitý.



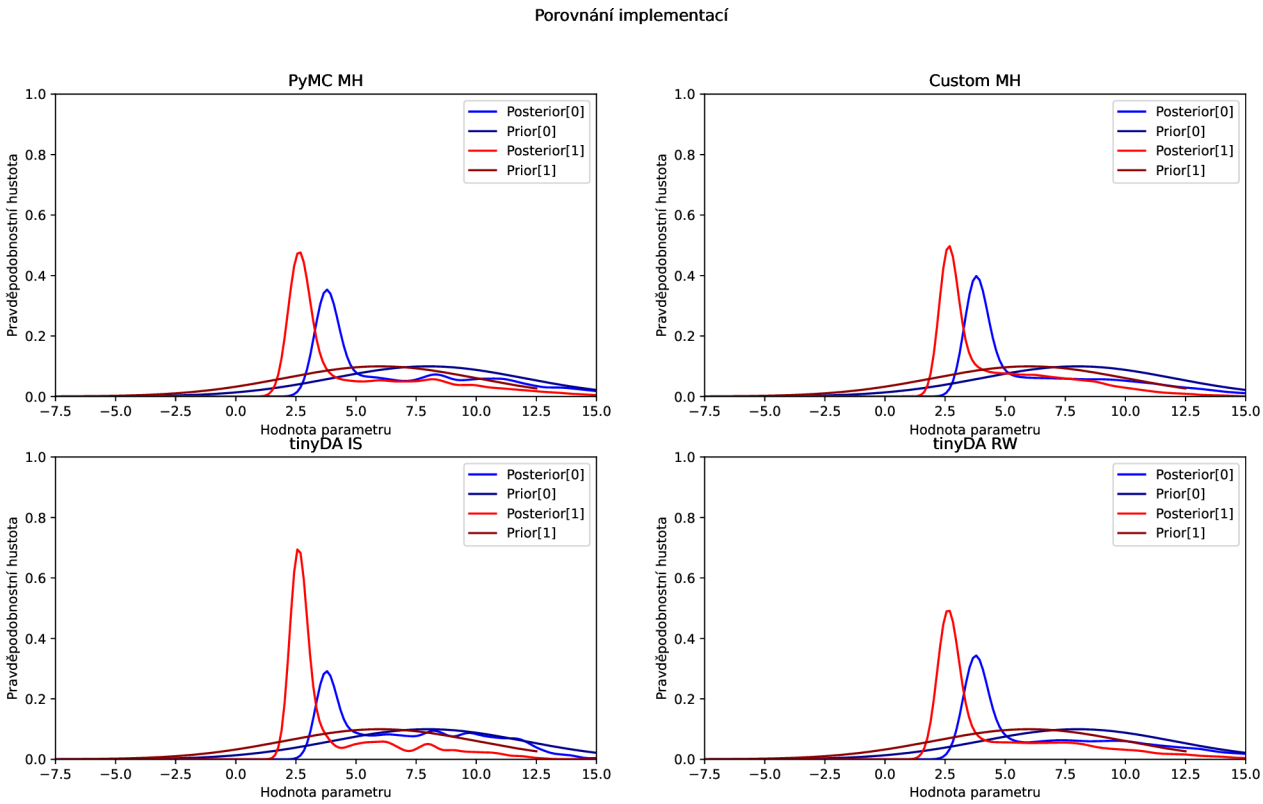
Obrázek 5.1: Marginální rozdělění inverze R1

U prvního odhadu R1 lze vidět, že algoritmy se chovají relativně podobně. Ve všech implementacích je odhad z inverze výrazně zúžený oproti původnímu odhadu. Zároveň je pravděpodobnostní hustota výsledku z inverze výrazně vyšší kolem kontury. Největší odchylkou je vlastní implementace s inverzí druhé vodivosti, kde je výrazně vyšší špička pravděpodobnostní hustoty. Všechny algoritmy ale mají v inverzi dvě špičky, v každé vodivosti jednu. Špička druhé vodivosti je vždy vyšší, než u první.



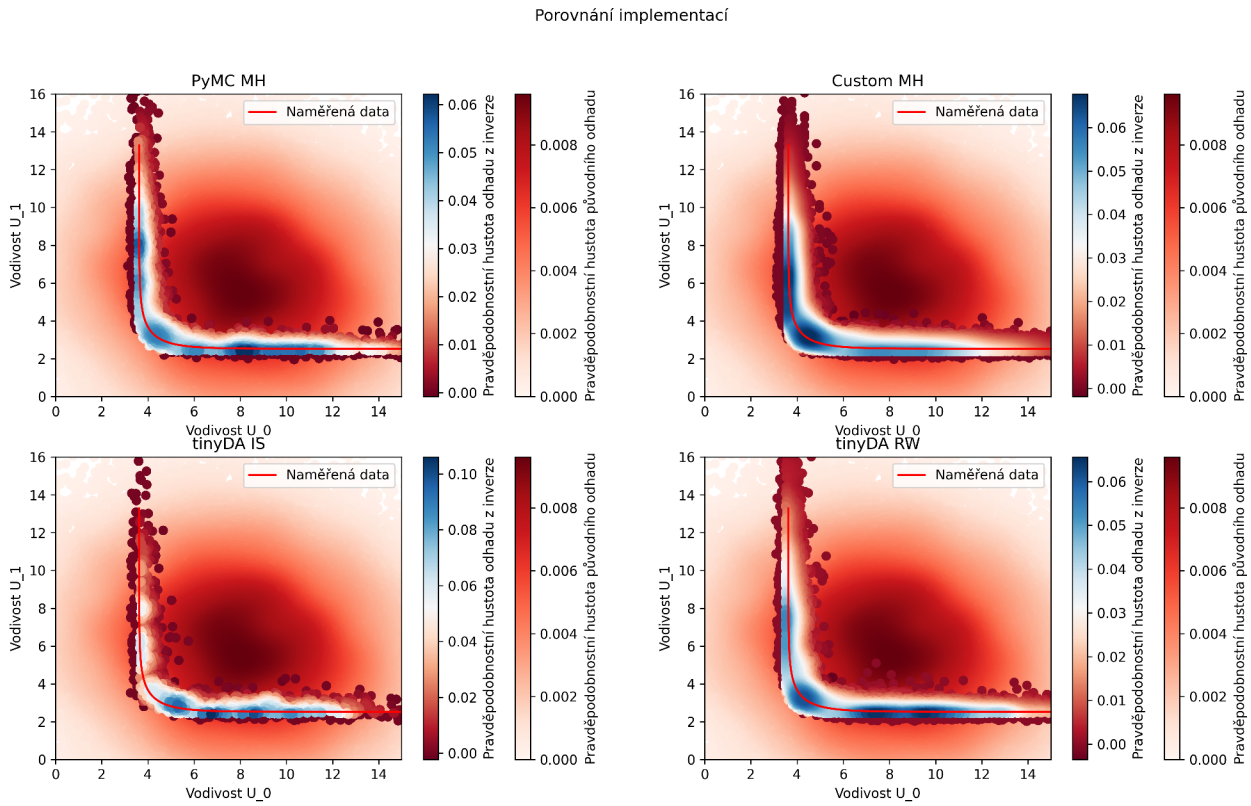
Obrázek 5.2: 2D rozdělění inverze R1

V druhé sadě grafů lze vidět větší rozdíly v chování algoritmů. Vlastní implementace má opět největší odchylku, kde pro druhou vodivost na svislé ose má výrazně nižší pokrytí – dosahuje maximální hodnoty pouze přibližně 7.6, ale ostatní implementace až 8 nebo víc. Naopak tinyDA RW je přesný opak – dosahuje hodnot druhé vodivosti až 10. Jinak se všechny algoritmy drží převážně kolem zahnuté části kontury, kde dosahují maximální hustoty. Dalším rozdílem je maximální dosažená hustota v regionu, kde PyMC MH a tinyDA RW mají podobná maxima a ostatní mají vyšší maxima. Tento výsledek dává smysl, jelikož PyMC MH je prakticky stejný algoritmus jako tinyDA RW.



Obrázek 5.3: Marginální rozdělění inveze R2

U druhého odhadu lze vidět, že rozptyl původního odhadu je výrazně větší a zároveň jeho pravděpodobnostní hustota je průměrně menší. U výsledků inverze je největší odchylka vidět u `tinyDA IS`, kde hodnota druhé vodivosti je více koncentrovaná ve špičce. Opět je špička druhé vodivosti vždy vyšší, než u první.



Obrázek 5.4: 2D rozdělení inverze R2

V druhé sadě grafů lze vidět u `tinyDA IS` opět výrazný rozdíl – kolem svislé části kontury je pravděpodobnostní hustota výrazně menší oproti ostatním implementacím. Toto potvrzuje data z první sady grafů, kde bylo obdobné chování viditelné. Lze tu také vidět rozšířený původní odhad, který také způsobil, že barevný gradient inverze je více nerovnoměrný. Pro dosažení lepšího gradientu by bylo nutné udělat více iterací algoritmu.

5.3 Implementace v PyMC

V PyMC se provádí veškeré definice proměnných pro inverzi v rámci instance třídy `pymc.Model`. Lze tu definovat náhodné veličiny, deterministické veličiny a zároveň spustit samotný algoritmus a provádět zpracování dat.

Náhodné veličiny lze definovat přes PyMC funkce korespondující k typu náhodné veličiny. např. veličinu typu normálního rozdělení lze definovat přes funkci `pymc.Normal()`. Prvním parametrem je vždy řetězec, který PyMC vnitřně používá pro identifikaci proměnných. Dále se musí specifikovat parametry náhodné veličiny, v případě normálního rozdělení střední hodnota μ a směrodatná odchylka σ.

```
G = pm.Normal('G', mu=G_mean, sigma=2e-4)}
```

Deterministické veličiny se definují buď explicitně přes `pymc.Deterministic()`, nebo implicitně přes standardní python výraz. Tyto veličiny lze použít jako pomocné pro mezivýpočty.

```
G_mean = pm.Deterministic('G_mean',
    -1 / 80 * (3 / np.exp(U[0]) + 1 / np.exp(U[1])))
```

Neznámé vodivosti v úloze lze modelovat pomocí funkce `pymc.MvNormal()` a použítím známého vektoru středních hodnot a kovarianční matice. Dopředný model lze modelovat přes `pymc.Deterministic()` a přepsání analytického vzorce. Model se odkazuje na vodivosti, takže výstup funkce `MvNormal()` se musí uložit do proměnné a použít. Pro začlenění šumu k naměřené vodivosti se musí udělat další náhodná veličina. Tato náhodná veličina bude typu normálního rozdělění s parametry podle rozdělění P ze zadání. Klíčová věc navíc tu je nastavení parametru `observed` na hodnotu O ze zadání.

Dalším krokem je spustit algoritmus a získat výsledky inverze – pro toto slouží metoda `pymc.Sample()`. Lze tu specifikovat počet iterací algoritmu, kolik iterací od začátku zahodit, kolik nezávislých řetězců iterovat a kolik procesorových jader použít. PyMC poskytuje několik různých návrhových hustot, které lze určit přes parametr `step`. Příkladem jsou metody Metropolis-Hastings, No-U-Turn Sampler, Differential Evolution MH a další. Tato funkce zároveň vrací objekt typu `arviz.InferenceData` s výsledkem inverze.

Výsledný objekt `InferenceData` lze dále modifikovat, buď přes vestavěné PyMC funkce, nebo přes funkce vlastní. PyMC v základu neuvádí veškeré informace o mezivýsledcích algoritmu do tohoto objektu. Pro zahrnutí pravděpodobnostní hustoty návrhů z návrhové hustoty lze použít `pymc.compute_log_likelihood`. Po zavolání `pymc.sample_prior_predictive` se do objektu také zahrnou hodnoty z původního odhadu, což může být užitečné pro diagnostiku a vizualizaci výsledků inverze.

5.4 Implementace v tinyDA

Rozhraní tinyDA je více modulární a jednotlivé části algoritmu jsou oddělené. Umožňuje prakticky neomezenou volnost v implementaci jednotlivých částí algoritmu, ale zároveň je struktura algoritmu relativně jednoduchá. Knihovna opět umí pracovat se strukturou dat `arviz.InferenceData`.

Nejprve se musí definovat instance třídy `tinyda.Posterior`. Tento objekt v sobě zahrnuje původní odhad, dopředný model a určení úrovně šumu. Implementace těchto funkcí je kompletně na uživateli. Musí se dodržet formáty dat – výstup původního odhadu musí mít stejný formát jako vstup dopředného modelu a výstup dopředného modelu musí mít stejný formát jako vstup určení šumu.

Pro původní odhad lze použít `scipy.multivariate_normal`, jelikož jediný požadavek zde je že objekt má funkci `rvs()`. V dopředného modelu se implementuje transformace podle vzorce dopředného modelu Fv zadání úlohy. Vstupem dopředného modelu je dvouprvkový vektor vodivostí, výstupem je skalár odpovídající hodnotě operátoru Fv zadání. Nakonec vyčíslení šumu vypočítá úroveň šumu pro výstup dopředného modelu vůči naměřeným hodnotám.

```
prior = multivariate_normal(mean=prior_mean, cov=prior_cov)
forward_model = lambda params:
  -1 / 80 * (3 / np.exp(params[0]) + 1 / np.exp(params[1]))
likelihood = lambda data:
  tda.GaussianLogLike(data, covariance=1e-3)
posterior = tda.Posterior(prior, forward_model, likelihood)
```

Dále je nutné vybrat návrhovou hustotu. Toto je ekvivalentní k určení parametru `step` v `pymc.Sample()`. Zde není co implementovat, pouze vybrat jednu z definovaných metod a určit pro ní parametry. Nejjednodušší možností je `tinyda.IndependenceSampler()`, který v každé iteraci použije původní odhad pro získání nového návrhu.

Spuštění algoritmu provede příkaz `tinyda.sample()`, kterému se předají již popsané parametry posterioru a návrhové hustoty. Dále se musí jako v PyMC určit počet iterací algoritmu, počet iterací k zahození a počet řetězců. Výstupem funkce je jedna nebo více instancí třídy `tinyda.Chain`, počet se odvozuje podle specifikovaného počtu řetězců. Tento objekt pak pomocí `tinyda.to_inference_data()` lze převést na objekt `InferenceData`. Pro dosažení stejného výsledku jako z PyMC je ještě potřeba pojmenovat parametry inverze, jelikož tinyDA je interpretuje jako jeden celek - parametry se identifikují pouze pořadím ve vektoru.

5.5 Implementace vlastní

Vlastní implementace se liší od PyMC a tinyDA převážně v nutnosti definovat proces návrhu, přijmutí nebo zamítnutí a zpracování výsledků. Obdobně jako v PyMC a tinyDA se musí definovat náhodné a deterministické veličiny. Pro definici náhodných veličin se použila knihovna `scipy.stats`, pro deterministické se použily standardní python výrazy.

Samotný algoritmus běží ve smyčce, dokud počet přijatých návrhů nepřekročí určenou hranici. Ve smyčce se z návrhové hustoty určí návrh a vypočítá se pro něj dopředný model. Zároveň si algoritmus pamatuje poslední přijatý návrh a hodnotu dopředného modelu. Z těchto hodnot se pak počítá pravděpodobnost přijmutí návrhu. Ta se porovná vůči náhodné pravděpodobnosti a pokud je vyšší, návrh se přijme. Při přijmutí návrhu se uloží jeho hodnota a hodnota dopředného modelu, dopočítá se hustota pravděpodobnosti z návrhové hustoty a inkrementuje se počet přijatých návrhů. Při zamítnutí se nic nestane, algoritmus se pouze vrátí na začátek cyklu.

Tato implementace neobsahuje podporu pro více nezávislých řetězců, ani pro použití více procesorových jader. Oba tyto požadavky nejsou potřeba k efektivnímu porovnání této implementace vůči PyMC a tinyDA.

Návrhová hustota použitá ve vlastní implementaci je prakticky ekvivalentní k návrhové hustotě `tinyDA.IndependenceSampler` – v každé iteraci se z původního odhadu vezme nový návrh. Je to nejjednodušší algoritmus z pohledu implementace, ale zároveň je relativně neefektivní. V rámci této úlohy toto však není velký problém, jelikož máme malý počet neznámých a málo výpočetně náročný model.

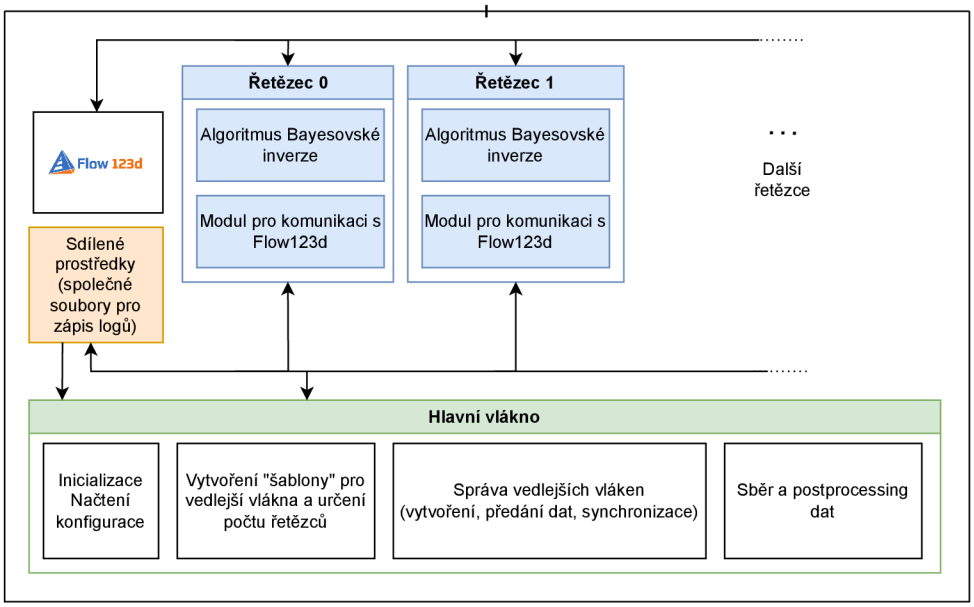
Z pohledu obsahu, který se ukládá z algoritmu do objektu InferenceData, je vlastní implementace jednoznačně nejvíc omezená. Ukládají se pouze hodnoty návrhů, jejich hodnoty dopředného modelu a pravděpodobností hustoty z návrhové hustoty.

Všechny tyto nedostatky nejsou kritické, jelikož cílem vlastní implementace je pouze si algoritmus lépe přiblížit. Každopádně mít nedostatky na vědomí je užitečné pro odůvodnění výsledků při porovnání implementací.

6 Tvorba aplikace

Tato kapitola slouží pro shrnutí procesu tvorby aplikace pro inverzi. Součástí kapitoly je přehled hlavních funkcí aplikace a několik podkapitol s detaily důležitých části aplikace. Aplikace je ve formě knihovny, ke které jsou přidané příklady použití včetně nasazení na výpočetní cluster.

6.1 Přehled struktury aplikace



Obrázek 6.1: Diagram aplikace z pohledu obsahu vláken

V obrázku 6.1 lze vidět přehled aplikace, která je rozdělená na jednotlivá vlákna. Každý blok představuje jedno nezávislé vlákno se specifickým účelem.

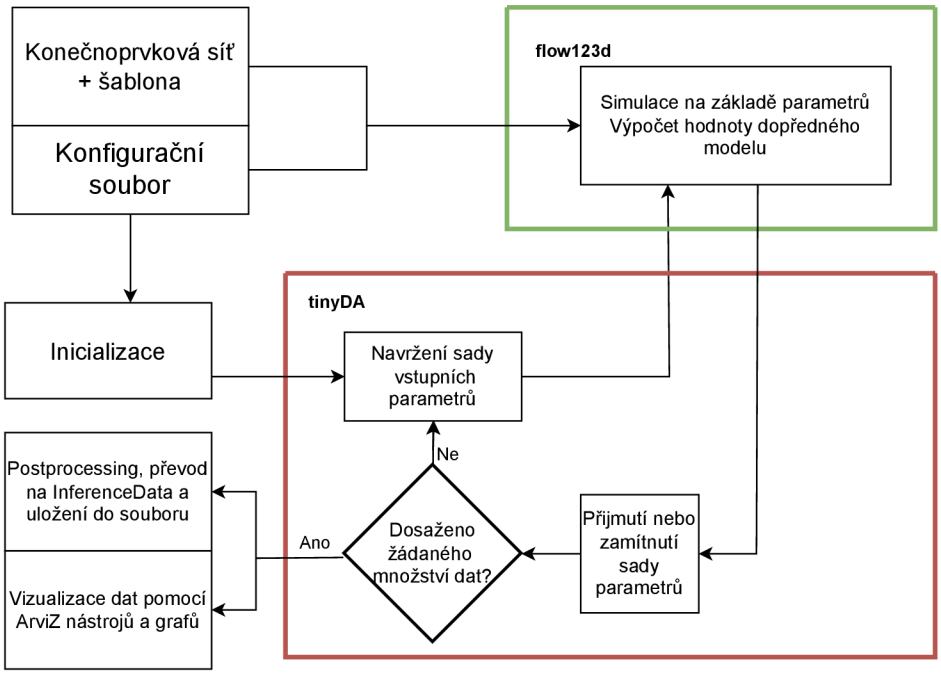
Hlavní vlákno je zeleně a je první, které se vytvoří. Ze začátku se stará o inicializaci inverze, načtení konfigurace a vytvoření ”šablony” pro vlákna, která budou provádět inverzi. Během inverze je vlákno málo používané a většinu práce dělají vedlejší vlákna. Po ukončení inverze hlavní vlákno posbírá data od vedlejších vláken a udělá veškeré zpracování dat jako převod na InferenceData, uložení do souboru a vytvoření a uložení grafů.

Vedlejší vlákna jsou navzájem nezávislá a každé spravuje svůj řetězec inverze. Každé vlákno si také vytváří vlastní Flow123d proces v dalším vlákně, které není v diagramu zobrazené. Všechny vedlejší vlákna jsou instance od stejné třídy se stejným obsahem. Toto vychází z omezení tinyDA – parametrizace vláken je velice minimální.

Jedno další vedlejší vlákno slouží pro sběr ladících dat během inverze. Všechny vedlejší vlákna s tímto vláknem během inverze komunikují a informace mu poskytují. Na konci inverze toto vlákno pošle veškeré uložené informace do hlavního vlákna, kde se data zpracují.

6.2 Propojení s Flow123d

Připojení inverze k simulátoru Flow123d je stěžejní součást práce. V rámci procesu inverze je Flow123d potřeba ve výpočtu dopředného modelu, kde se přes rozhraní předá sada parametrů k simulaci a s výsledkem simulace se dál pracuje. V následujícím obrázku 6.2 lze vidět, jak se Flow123d integruje do procesu inverze.

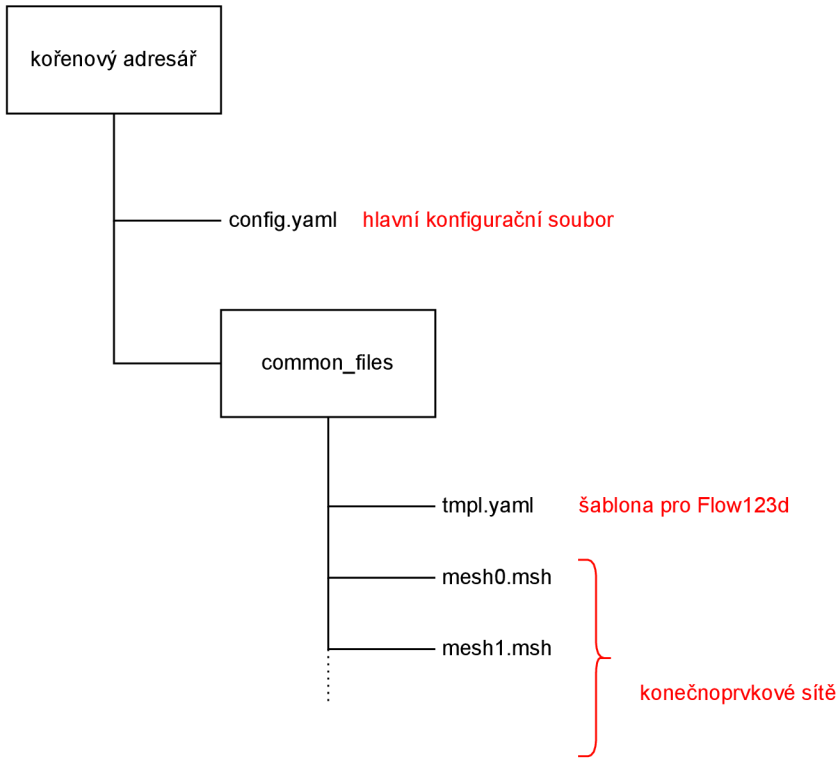


Obrázek 6.2: Diagram integrace Flow123d do inverze

6.2.1 Adresářová struktura

Pro definici úlohy je potřeba několik souborů uspořádaných v určité adresářové struktuře. Přehled struktury lze vidět v 6.3. Konfigurační soubor `config.yaml` je de-

tailněji popsaný v 6.2.2 a představuje místo, kde lze inverzi ladit. Šablona `tmpl.yaml` a konečno-prvkové sítě `meshx.msh` tvoří vstupní data do Flow123d. Každý parametr v šabloně je během každé iterace inverze vyplněn navrženou hodnotou nebo výchozí hodnotou podle toho, jestli pro parametr děláme inverzi. Výchozí hodnoty jsou definované v konfiguračním souboru.



Obrázek 6.3: Struktura adresáře pro definici úlohy

6.2.2 Konfigurační soubor

Konfigurační soubor představuje zdroj informací, které jsou součástí definice řešené úlohy. Soubor je ve formátu YAML. Definuje všechny veličiny, pro které se má dělat inverze. Zároveň slouží pro vyplnění šablony Flow123d a definici doplňujících údajů. Ve stejném konfiguračním souboru jsou dále definovány parametry algoritmu inverze. Tyto parametry přesně definují chování inverze.

V sekci `parameters` se definují invertované veličiny a jejich původní odhad. Sekce je list, kde každý prvek obsahuje 3 údaje – jméno veličiny, typ původního odhadu a hodnotu původního odhadu. Typem odhadu je řetězec, který odpovídá typu náhodného rozdělení. Podporované typy jsou `norm` pro normální rozdělení, `lognorm` pro logaritmicko-normální rozdělení a `truncnorm` pro omezené normální rozdělení.

Hodnotou původního odhadu jsou parametry náhodného rozdělení, typicky střední hodnota a směrodatná odchylka.

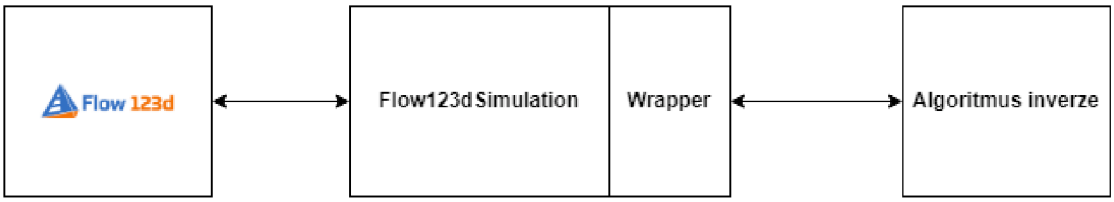
Sekce `sampler_parameters` slouží pro ladění algoritmu inverze. Lze tu definovat počet iterací na řetězec, počet zahozených iterací v rámci ladění a počet řetězců. Dále lze ovlivnit způsob, jakým bude algoritmus navrhovat hodnoty veličin pro simulování. Je tu na výběr z několika různých metod a parametrů, včetně přepínače pro adaptivní návrhovou hustotu. Nakonec tu je přepínač na použití MLDA a kolik úrovní MLDA se má použít. Tuto sekci lze buď částečně nebo úplně vynechat, dosadí se při tom výchozí hodnoty. Každopádně správný výběr těchto parametrů je důležitý a spoléhat na výchozí hodnoty není doporučené.

6.2.3 Flow123d Python rozhraní

Prakticky celé toto rozhraní nevzniklo v rámci práce – vychází z existujícího řešení a je pouze lehce upravené a navázané k práci. Každopádně to je součást práce a je nutné popsat fungování hlavních částí tohoto rozhraní.

Pro komunikaci s Flow123d existuje rozhraní definované ve třídě `Wrapper`. Hlavní částí jsou metody pro předání navržených parametrů do Flow123d a získání výsledků simulace pro dané parametry. Interně toto rozhraní má v sobě instanci třídy Flow123d simulace a chová se pouze jako obal.

Třída `Flow123dSimulation` se pak stará o správné načtení konfigurace, připravení konečno-prvkové sítě, připravení pracovního adresáře a samotné zavolání Flow123d se správnými parametry. Také slouží pro zpětné získání výsledků po dokončení simulace včetně statusu simulace a informací ohledně selhání nebo jiných událostí.



Obrázek 6.4: Komunikace mezi Flow123d a inverzí

Poslední součást rozhraní je systém pro načítání naměřených dat ze souborů. Předpokládá se s formátem souboru s naměřenými daty `.csv`. Soubor obsahuje data z několika bodů v hornině. Rozhraní tyto data zpřístupňuje a umožňuje určit z jakých bodů v hornině se mají data přečíst, případně v jakém pořadí.

6.3 Kontejnerizace aplikace

Použitý kontejner vychází z Docker obrazu `flow123d/endorse_ci`, který obsahuje již nainstalovaný Flow123d. Při tvorbě kontejneru se navíc zkopíruje soubor `requirements.txt` s vypsanými závislostmi aplikace a tento soubor se použije pro

nainstalování všech závislostí uvnitř kontejneru. Závislosti se nainstalují do virtuálního prostřední, které je přístupné i mimo kontejner. Nakonec se nainstaluje balík SSHPass, který je potřeba pro paralelizace aplikace na výpočetním clusteru. Celý tento proces je definovaný v souboru `dockerfile` a kontejner se sestaví pomocí skriptu `build-image.sh`.

Po sestavení kontejneru je možné se dostat do jeho příkazové řádky pomocí skriptů `fterm` a `fterm_sing`. První z těchto skriptů je pro spuštění přes Docker, druhý je pro vytvoření Singularity obrazu (pokud neexistuje) a spuštění přes Singularity.

Pro spuštění aplikace v rámci kontejneru je potřeba přepnout se do kontejneru, aktivovat virtuální prostředí a spustit aplikaci.

6.4 Paralelizace aplikace

Přechod z jedno-vláknové aplikace je částečně hotový díky integraci Ray v tinyDA, ale pořád existuje několik problematických bodů, které se musí vyřešit. Zároveň se ladění paralelní aplikace stane složitější oproti jednov-láknové a ještě složitější při testování na výpočetním clusteru. Každopádně je paralelizace prakticky vyžadovaná kvůli výpočetní náročnosti modelu.

6.4.1 V rámci jednoho počítače

Hlavním problémem u vícevláknové implementace na jednom počítači je, že Flow123d potřebuje mít specifikovanou adresu, kde se bude nacházet pracovní adresář. Zatím to nezní jako problém, ale důležitým kontextem jsou následující 2 fakty – původní varianta rozhraní Flow123d rozlišovala různé adresáře pomocí ID instance rozhraní a tinyDA neumožňuje jednoduchou parametrizaci řetězců, takže všechny používají stejnou instanci rozhraní. Toto způsobuje, že všechny řetězce chtějí používát stejný adresář jako pracovní a nastává konflikt, kde více různých simulací se snaží výsledky ukládat do stejných souborů.

Tato problematika měla dvě různé řešení s různým přístupem. První řešení se zaměřilo na přidělení různých adresářů pro každý řetězec s Flow123d rozhraním. Adresáře se museli přidělovali dynamicky a celkově bylo řešení zbytečně složité.

Druhá varianta řešení této problematiky se zaměřuje více na modifikaci Flow123d rozhraní (na rozdíl od první varianty, která se zaměřovala na modifikaci tinyDA části). Cílem této implementace je umožnit více instancím Flow123d pracovat ve stejném pracovním adresáři.

Zde byl hlavní problém definovat smysluplný způsob pojmenování adresářů – mezi názvy nesmí nastat konflikt a mělo by být možné určit, k jakým parametrům adresář patří. Řešením tedy bylo vzít vstupní parametry simulace a vytvořit z nich hash. Toto bude fungovat, protože šance výběru dvou stejných sad parametrů je prakticky 0. Každopádně součástí řešení je i kontrola kolize hashů.

Jedinou nevýhodou této varianty je omezení na jediný, společný pracovní adresář. Mít výstupy řetězců oddělené do různých adresářů může být užitečné pro ladění a diagnostiku.

6.4.2 Na výpočetním clusteru

Knihovna Ray nativně podporuje škálování aplikací na výpočetní clustery, ale protože každý cluster má jinou strukturu a používá jiné systémy, tak je potřeba provést původní inicializaci za něj. Pro provedení inicializace je nutné přes Ray-CLI na každém počítači spustit příkaz `ray init`. Na hlavním uzlu se pouze specifikuje port, který se má použít pro komunikaci, a adresář, který se použije pro dočasné soubory. Na vedlejších uzlech se musí určit IP adresa a port, které odpovídají údajům hlavního uzlu.

Aby bylo možné inicializovat vedlejší uzly, musí existovat přístup k jejich příkazové řádce. Pro tento účel existuje SSH, ale SSH vyžaduje heslo nebo SSH klíč pro umožnění připojení. SSH klíče je nepraktické používat, jelikož musí být dostupné na všech uzlech používaného clusteru. Zároveň pokud se používaný cluster mění, tak se u každého nově použitého clusteru musí opět klíč přenést a heslo by se muselo zadat. Použití pouze hesla je tedy jednoduší. U hesla je zase jiný problém – jakým způsobem se skript dozví, jaké heslo má použít při přihlášení.

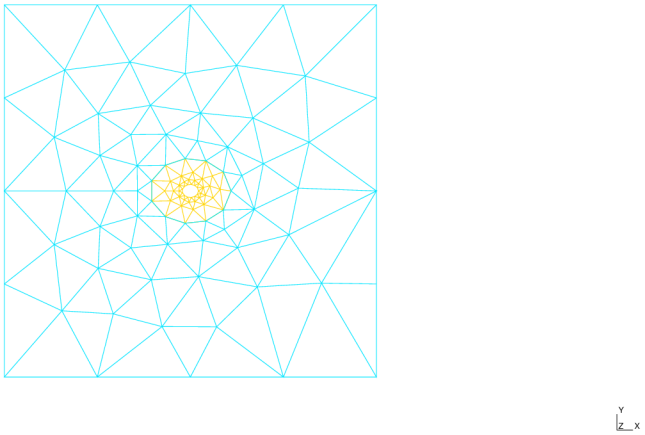
Zde se nabízí několik možností, které se dělí na dvě hlavní kategorie – uživatel zadá heslo jednou a nebo uživatel zadá heslo pokaždé co chce inverzi spustit. Druhá varianta, kdy se heslo zadává pokaždé, je bezpečnější a méně praktická. V obou případech se heslo ukládá jako proměnná v prostředí, kterou může přečíst pouze uživatel, kterému heslo patří, a root. Z bezpečnostního hlediska se liší v tom, že v první variantě heslo musí být uvedené v souboru `.bashrc` aby se při přihlášení proměnná inicializovala.

S přístupem k heslu lze realizovat inicializaci. Skript pro inicializaci začne na hlavním uzlu, kde se přepne do Singularity, aktivuje virtuální prostředí a inicializuje Ray. Pak si zjistí adresy vedlejších uzlů a na každý uzel se přes SSHPass přihlásí. Po přihlášení se opět přepne do Singularity, aktivuje virtuální prostředí a inicializuje Ray. Všechny uzly spouští stejný Singularity obraz.

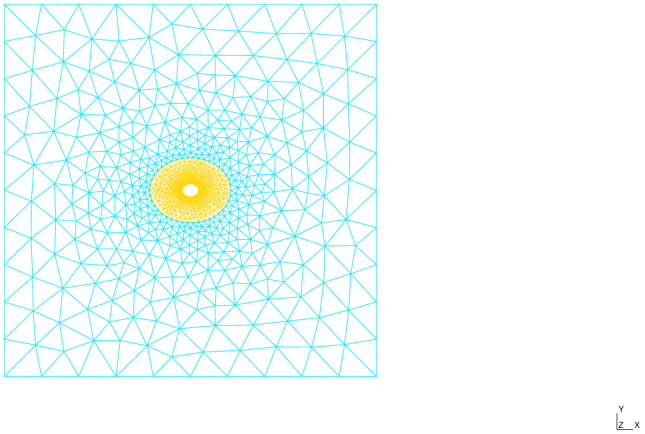
Důležitým detailem je, jak se použije Singularity kontejner. Pokud by se použil pouze `singularity exec`, tak se spustí kontejner, vykoná se inicializace, a kontejner se zavře. Toto je samozřejmě špatně, jelikož Ray potřebuje kontejner mít pořád zapnutý, aby mohl s tím uzlem vůbec komunikovat. Proto se musí použít `singularity instance` a u `singularity exec` se na tuto instanci odkázat. Tímto způsobem zůstane kontejner pořád otevřený i po dokončení inicializace.

6.5 Implementace MLDA

V rámci implementace MLDA se použil první přístup definovaný v 2.3 a vytvořila se hrubší konečno-prvková síť. Původní síť má 1020 vrcholů a 1950 trojúhelníků, náhradní hrubší síť má pouhých 110 vrcholů a 191 trojúhelníků, neboli asi 9-10x menší. Obě sítě modelují stejnou úlohu, ale s různou úrovní přesnosti. Výrazné zjednodušení výpočetní sítě se téměř stejně odrazilo na zmenšení výpočetní náročnosti – jedna simulace jemného modelu trvá průměrně 90 vteřin, naopak jedna simulace hrubého modelu trvá pouhých 10 vteřin.



Obrázek 6.5: Hrubší náhradní konečnoprvková síť



Obrázek 6.6: Původní jemná konečnoprvková síť

Každá síť je definovaná jedním souborem s příponou `.msh` a oba soubory jsou v adresáři `common_files`, který je součástí definice úlohy. Na tyto soubory dále existuje odkaz v konfiguračním souboru popsaném v 6.2.2, kde sítě musí být seřazeny od nejhrubější až po jemnou. Tyto informace se pak během inverze použijí, aby se simulaci specifikovalo, jakou síť má použít.

Na straně tinyDA je pro spuštění MLDA potřeba definovat několik `posterior` objektů. Pro připomenutí `posterior` slouží pro zapouzdření původního odhadu `prior`, výpočtu dopředného modelu `forward_model` a výpočet úrovně šumu `likelihood`. Ve všech posteriorech se použije stejný původní odhad, jelikož nedává smysl mít různé odhady pro různé úrovně MLDA. Dopředný model bude jiný, ale jediný rozdíl je příznak, který simulaci dá vědět, jakou konečnoprvkovou síť použít.

Vyčíslení šumu se také změní, konkrétně se mění rozptyl šumu. Důvod je jednoduchý – hrubá síť sice simuluje stejnou úlohu, ale její výsledky budou dále od opravdové hodnoty, oproti jemnému modelu. Proto je smysluplné použít vyšší úroveň šumu, aby hrubý model byl méně přísný na zamítání návrhů a nezamítal dobré návrhy.

7 Výsledky z inverze

V rámci této kapitoly se porovnají 2 vybrané datasety z inverze a přiblíží se problémy během inverze a ladění. Pro porovnání se použije několik grafů, ve kterých budou v lepším formátu vidět výsledky inverze. Také se využije několika statistických metrik dostupných v ArviZ summary, které vyčíslují výsledky inverze a kvalitu inverze. Jeden dataset je bez použití MLDA s kratšímy řetězci a druhý MLDA použil a má delší řetězce. Všechny rozdíly jsou popsané v 7.1. Výsledky jednotlivých datasetů jsou pak důkladně analyzovány v 7.2.1 a 7.2.2.

7.1 Parametry inverze

V rámci ladění aplikace se otestovalo velké množství různých kombinací parametrů pro dosažení výsledků. Pokaždé se inverze prováděla s 10 řetězci a 500 nebo 1000 iteracemi na řetězec. Ve všech případech se použilo adaptivní návrhové hustoty. Hlavní proměnné parametry byly celkem 4, 3 z nich souvisí s návrhovou hustotou. Parametr s největším vlivem na výsledky je `noise_std`, který určuje jak přísná má inverzi být u přijímání návrhů. Potom `proposal_scaling`, který určuje maximální velikost skoku ze současné hodnoty na novou. Nakonec `proposal_gamma` a `proposal_period` slouží pro určení, jak často a o kolik se má návrhová hustota adaptovat. V následující tabulce lze vidět použité parametry pro dvě iterace inverze, jedna s MLDA a druhá bez.

Číslo datasetu	1	2
Počet iterací na řetězec	500	3000
Počet řetězců	10	10
Počet iterací k zahození	1	1
Rozptyl šumu	1000	1000
Koeficient velikost skoku	0.2	0.5
Perioda adaptivity	5	10
Škálování adaptivity	1.02	1.05
MLDA	Ne	Ano
Návrhová hustota	GaussianRandomWalk	GaussianRandomWalk

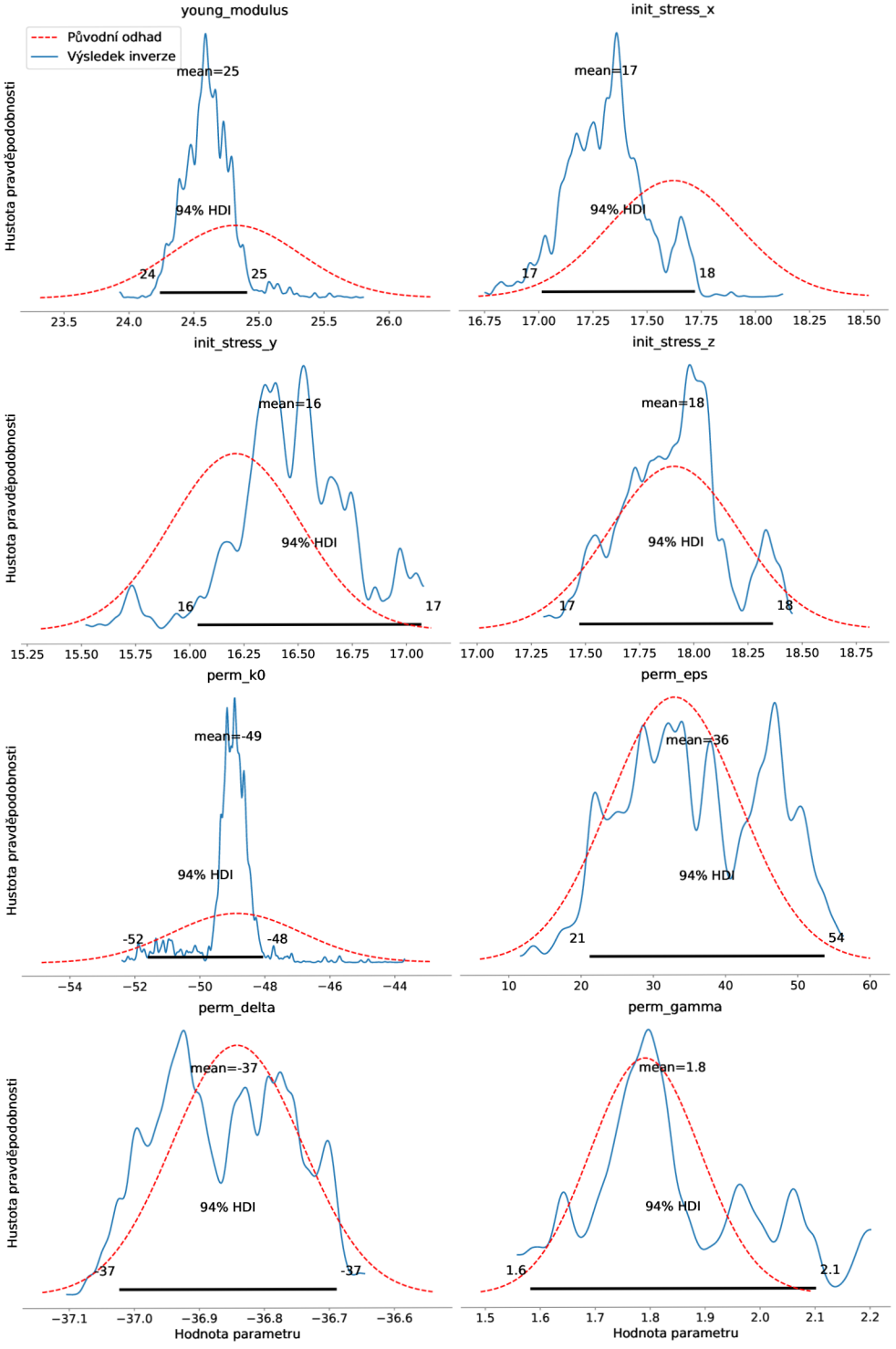
Tabulka 7.1: Přehled parametrů inverze u porovnaných datasetů

//	V 1. datasetu?	V 2. datasetu?	Typ rozdělění	Parametry
storativity	Ne	Ano	log–normální	$\mu = -16.4$, $\sigma = 2$
young_modulus	Ano	Ano	log–normální	$\mu = 24.8$, $\sigma = 0.5$
init_stress_x	Ano	Ano	log–normální	$\mu = 17.6$, $\sigma = 0.3$
init_stress_y	Ano	Ano	log–normální	$\mu = 16.2$, $\sigma = 0.3$
init_stress_z	Ano	Ne	log–normální	$\mu = 17.9$, $\sigma = 0.3$
perm_k0	Ano	Ano	log–normální	$\mu = -48.8$, $\sigma = 2$
perm_eps	Ano	Ano	omez. normální	$\mu = 33$, $\sigma = 9$ $a = 1.1$, $b = 1000$
perm_delta	Ano	Ano	log–normální	$\mu = -36.8$, $\sigma = 0.1$
perm_gamma	Ano	Ano	log–normální	$\mu = 1.79$, $\sigma = 0.1$

Tabulka 7.2: Hydromechanické parametry, pro které se inverze u datasetů dělala

7.2 Rozbor datasetů

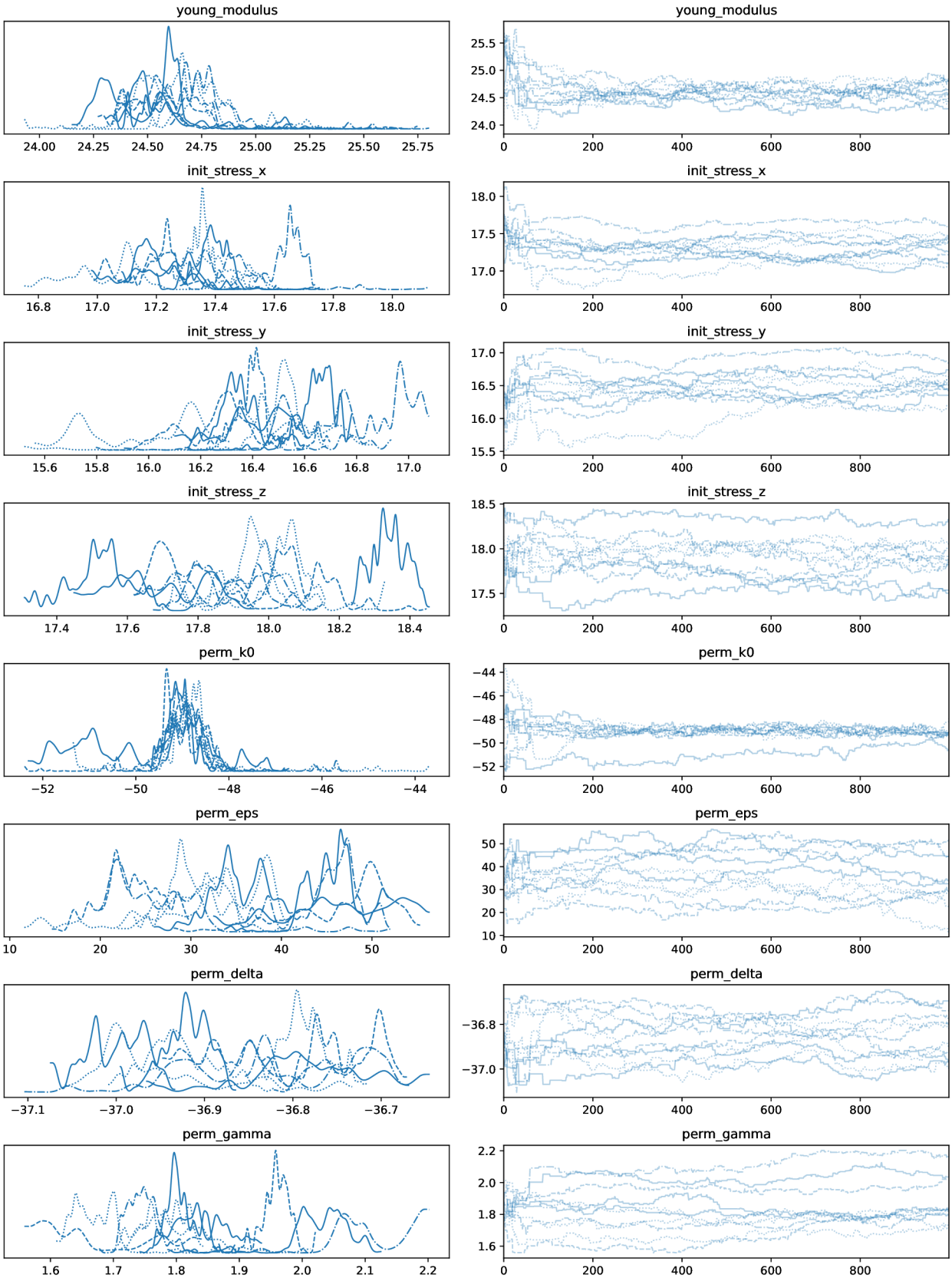
7.2.1 Dataset 1



Obrázek 7.1: Vizualizace prvního datasetu - marginální rozdělění

V prvním sadě grafů 7.1 lze vidět graf vygenerovaný přes `arviz.plot_posterior()`, který byl upravený zahrnutím původního odhadu. Graf zobrazuje marginální rozdělení z inverze pro jednotlivé hydromechanické parametry. Vodorovná osa představuje hodnotu parametru, svislá osa obsahuje pravděpodobnostní hustotu parametru. Navíc je tam zahrutá černá vodorovná čára, která znázorňuje region, ve kterém se nachází 94% dat.

U některých parametrů lze vidět, že výsledek inverze má výrazně nižší odchylku a tudíž vyšší pravděpodobností hustotu oproti původnímu odhadu. Toto chování lze vidět u parametrů `young_modulus`, `perm_k0` a `init_stress_x`. Jiné parametry se chovají jinak - u `perm_k0` lze vidět posun střední hodnoty oproti původnímu odhadu a u ostatních parametrů jsou viditelné menší odchylky od původního odhadu. Hodně parametrů se výrazně podobá původnímu odhadu, což může vypovídat o dobře zvoleném odhadu, nebo špatně zvolených parametrech inverze.



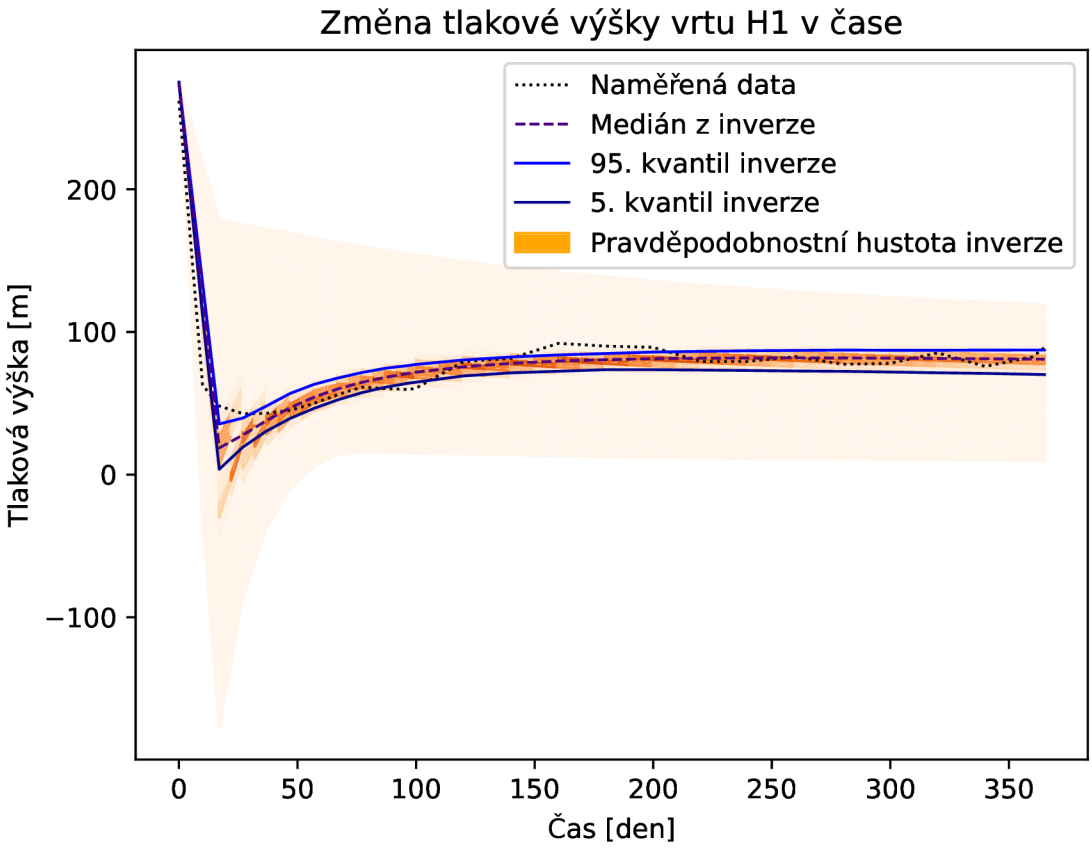
Obrázek 7.2: Vizualizace prvního datasetu - cesty řetězců

Druhá sada grafů 7.2 se vygenerovala přes `arviz.plot_trace()`. Levé grafy obsahují podobné informace jako minulá sada 7.1 - jedinou odlišností je, že zde jsou data mezi řetězci oddělené a nesloučené dohromady. Vodorovná osa je opět hodnota parametru, svislá je pravděpodobnostní hustota. Pravá část ukazuje způsob, jakým jednotlivé řetězce cestovali náhodným prostorem během inverze. Zde je vodorovná osa číslo iterace inverze v rámci řetězce a svislá osa je hodnota parametru.

Lze tu vidět způsob, jakým řetězce prozkoumávají náhodný prostor - každý řetězec začíná na jiném místě a prozkoumává jinou část náhodného prostoru. V určitých částech prostoru se budou řetězce držet dlouho, v jiných budou jen chvíli. Díky použití adaptivní návrhové hustoty by se s narůstajícím číslem iterace řetězec měl dostat do lepších částí náhodného prostoru, kde bude větší šance na přijmutí návrhu.

Parametr `perm_k0` má všechny řetězce s výjimkou jednoho, které se drží v rámci stejného regionu, kolem hodnoty -49. Na začátku lze vidět, že řetězce žačínají mimo tento region, postupně se k němu dostanou a pak se v něm drží. Podobné chování lze vidět u parametru `young_modulus`, kde tentokrát všechny řetězce se drží ve stejném regionu, ale opět se k němu museli ze začátku dostat.

Ostatní parametry nevykazují toto chování, alespoň ne do takové míry. Parametry `init_stress_x` a `perm_gamma` mají menší seskupení řetězců v určitých regionech, ale ostatní se vzájemně skoro nepřekrývají.



Obrázek 7.3: Vizalizace prvního datasetu - výsledky modelu

Graf 7.3 obsahuje vizualizaci výsledků simulace pro hodnoty z inverze. Tento graf je dobrým indikátorem správnosti výsledků inverze - ukazuje, jak blízko naměřeným hodnotám se inverze dostává. V grafu lze vidět několik křivek - černá tečkovaná křivka představuje naměřená data z ražby, ke kterým se chceme přiblížit. V regionu mezi plnou modrou a fialovou křivkou je 90% výsledků z inverze. Tmavě modrá čárkovaná křivka je medián z inverze. Část grafu obarvená odstíny oranžové představuje pravděpodobnostní hustotu výsledků inverze v dané části grafu. Sytější odstíny oranžové představují vyšší pravděpodobnostní hustotu. V určitých místech je hustota špatně vykreslená kvůli omezenému množství bodů. Vodorovná osa představuje čas simulace, svislá osa obsahuje hodnoty tlakové výšky v měřeném bodě horniny.

Lze vidět, že na základě naměřených dat je očekávané chování prudký pokles tlakové výšky během prvních 17 dní, po kterém následuje pozvolné zvyšování tlakové výšky. Většina výsledků z inverze má chování podobné - hodnoty výšky se liší maximálně o 25m, ale většina je v rámci 10m. Také lze vidět, že některé parametry z inverze produkují výrazně jiné hodnoty - některé mají větší pokles na začátku a některé mají velice malý pokles na začátku. Tyto výsledky představují okraje oranžově obarveného regionu.

Název parametru	Střední hodnota	Odchylka	ESS tail	R-hat
young_modulus	24.6	0.20	44	1.56
init_stress_x	17.3	0.19	14	2.23
init_stress_y	16.4	0.28	13	2.54
init_stress_z	17.9	0.24	18	2.65
perm_k0	-49.1	0.89	23	1.37
perm_eps	36.1	9.82	39	2.40
perm_delta	-36.8	0.10	36	2.15
perm_gamma	1.8	0.15	14	3.58

Tabulka 7.3: ArviZ summary pro první dataset

Tabulka 7.3 obsahuje část výpisu vygenerovanou z `arviz.summary()` a součástí jsou informace o střední hodnotě a rozptylu jednotlivých parametrů z inverze. Zároveň obsahuje dvě statistické metriky - R-hat pro vyčíslení úrovně korelace v datasetu a ESS-tail pro určení počtu kompletně nezávislých náhodných sad parametrů pro dosažení stejné přesnosti, jako dosáhla konečná část výsledků inverze.

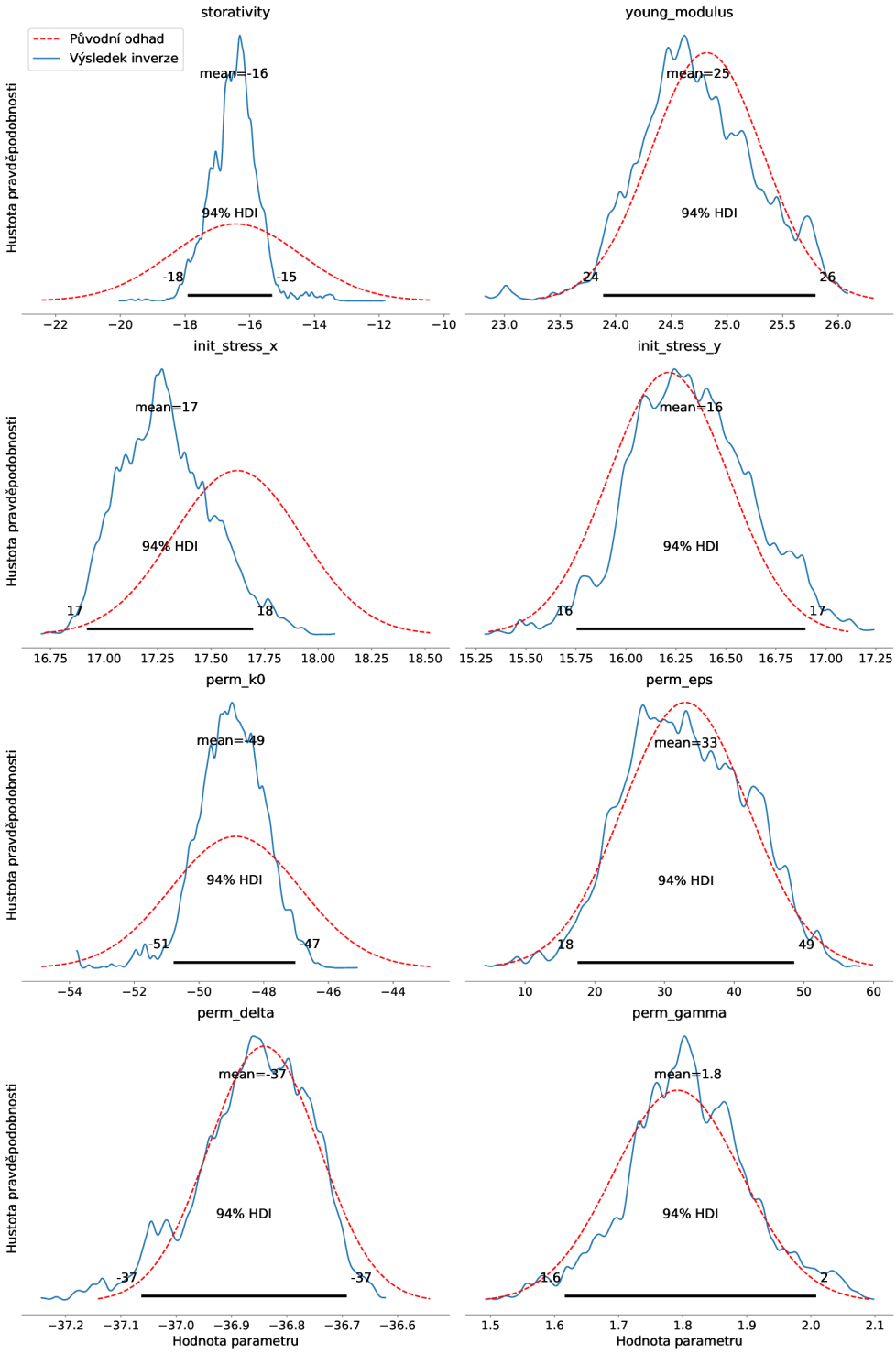
Údaje o střední hodnotě a odchylce odpovídají datům z přechozích grafů 7.1. Pro R-hat se doporučují hodnoty $1.00 < \hat{r} < 1.01$, takže z tabulkových hodnot lze předpokládat výraznou korelaci a autokorelaci ve výsledcích inverze. Pro ESS je ideální hodnota co nejblíže počtu iterací na řetězec. Pro první dataset se použilo 1000 iterací na řetězec, takže nejvyšší hodnota ESS je pouze 4.4%. Toto opět vypovídá o výrazné korelaci mezi řetězci i v rámci jednotlivých řetězců. Důvodem je pravděpodobně malá délka řetězce a malé množství zahozených návrhů.

Počet iterací	Přijatých	Zamítnutých	Poměr přijatých
10000	2248	7752	22.48%

Tabulka 7.4: Dodatečné informace k prvnímu datasetu

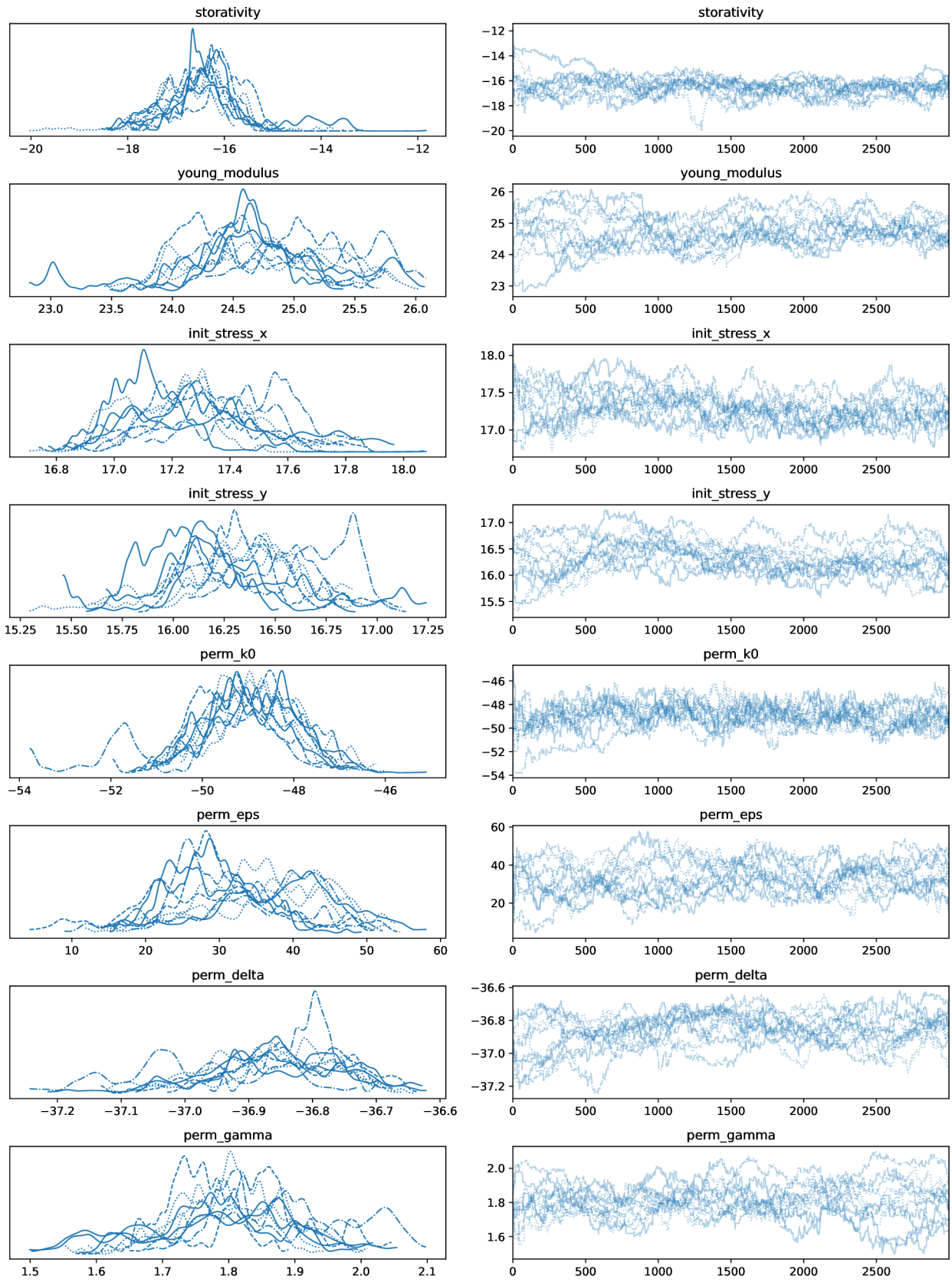
Posledním výsledkem k prvnímu datasetu jsou informace o počtu přijatých a zatmítnutých návrhů během inverze v tabulce 7.4. Celkem se provedlo 10000 iterací inverze a 2248 se z toho přijalo. Z toho vychází poměr přijatých návrhů 22.48%. Tento poměr je relativně blízko cílené hodnotě 24%, ke které se tato návrhová hustota snaží přiblížit. Celý proces inverze pro tento dataset trval necelých 23 hodin a 217 hodin procesorového času.

7.2.2 Dataset 2



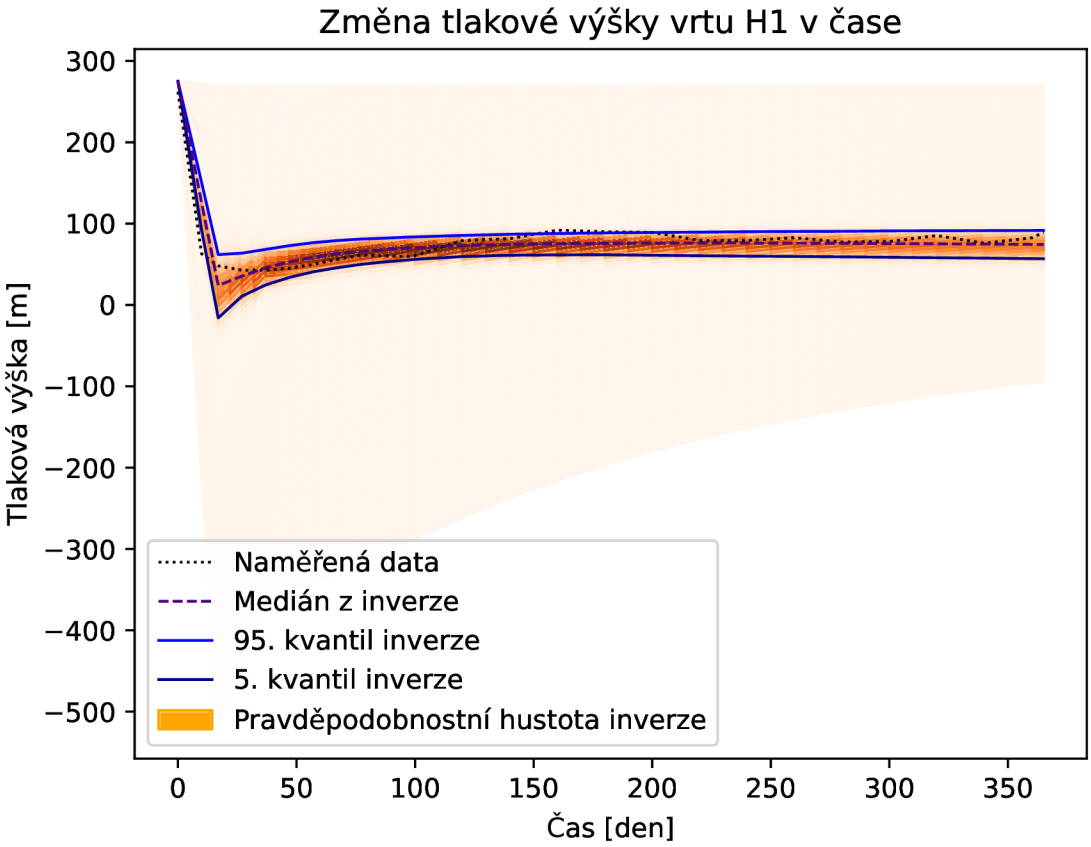
Obrázek 7.4: Vizualizace druhého datasetu - marginální rozdělění

U parametrů `init_stress_x` a `perm_k0` lze v grafu 7.4 vidět podobné výsledky, jako u předchozího datasetu v 7.1. `init_stress_x` má výsledné rozdělění posunuté doleva oproti původnímu odhadu a má zvýšenou špičku. `perm_k0` má opět špičku kolem střední hodnoty původního odhadu, ale tentokrát menší. Novým parametrem v tomto datasetu je `storativity`, u které lze vidět špičku kolem středu původního odhadu a zmenšení rozptylu. Parametr `young_modulus` se tentokrát chová jinak –neobsahuje už vysokou špičku, ale spíše kopíruje původní odhad. U ostatních parametrů lze vidět rozdělění, které jsou podobnější původnímu odhadu oproti prvnímu datasetu. Toto je pravděpodobně kvůli výrazně delším řetězcům.



Obrázek 7.5: Vizualizace druhého datasetu - cesty řetězců

V grafu 7.5 jsou opět vidět cesty řetězců v náhodném prostoru. U několika parametrů lze vidět podobné chování jako v 7.2 u prvního datasetu – ze začátku řetězce prozkoumávají velkou část náhodného prostoru, ale postupně se dostanou do určitého regionu a tam se drží. Takto se tu chová `storativity`, `young_modulus` a `perm_k0`. U ostatních nelze vidět nějakou výraznou konvergenci do určitého regionu. Naopak u `perm_gamma` řetězce s rostoucí iterací inverze prozkoumávají větší část prostoru oproti začátku.



Obrázek 7.6: Vizalizace druhého datasetu - výsledky modelu

Výsledky dopředného modelu pro druhý dataset jsou vidět v grafu 7.6. Výrazným rozdílem zde je rozsah svislé osy, kde nejnižší hodnoty dosahují až pod -400. V prvním datasetu 7.3 bylo minimum pouhých 150. Odůvodnění této změny bude pravděpodobně zahrnutí nového parametru `storativity` do inverze, který je asi schopný výsledek simulace výrazně měnit. Jinak medián iverze a většina dat jsou seskupeny poblíž naměřených dat, takže inverze pořád funguje správně. Pravděpodobnostní hustota inverze dosahuje maxima kolem naměřených dat, což je viditelné díky sytější oranžové barvě kolem křivky naměřených dat. Hrubé obarvení oranžovou v určitých mísetch je opět důsledek špatné interpolace při tvorbě grafu.

Název parametru	Střední hodnota	Odchylka	ESS tail	R-hat
storativity	-16.5	0.74	56	1.15
young_modulus	24.7	0.53	77	1.27
init_stress_x	17.3	0.21	106	1.29
init_stress_y	16.3	0.30	60	1.12
perm_k0	-49.0	1.04	53	1.12
perm_eps	33.0	8.7	163	1.24
perm_delta	-36.8	0.10	47	1.15
perm_gamma	1.8	0.1	28	1.37

Tabulka 7.5: ArviZ summary pro druhý dataset

Schrnutí druhého datasetu je dostupné v tabulce 7.5. U společných parametrů s 7.3 lze vidět u středních hodnot a odchylek převážně podobné hodnoty. Pouze u `young_modulus` je výrazně větší odchylka. Velké rozdíly jsou vidět ve sloupcích ESS tail a R-hat. Druhý dataset má hodnoty R-hat výrazně menší, což vypovídá o nižší korelaci a autokorelaci v datasetu. ESS tail má výrazně vyšší hodnoty a dataset tedy nese větší množství informací. Toto je možné očekávat, jelikož dataset obsahuje 3x více dat, ale pro hodně parametrů je nárůst ESS tail vyšší než 3x. Například pro `init_stress_x` je nárůst ESS oproti prvnímu datasetu tail 7.5x.

Počet iterací	Přijatých	Zamítnutých	Poměr přijatých
30000	7322	22678	24.40%

Tabulka 7.6: Dodatečné informace k druhému datasetu

Nakonec v 7.6 jsou informace o počtu přijatých a zamítnutých návrhů. Celkem se provedlo 30000 iterací a 7322 se přijalo, z čehož vychází poměr přijatých návrhů 24.4%. Tento poměr je tentokrát ještě blíže k cílové hodnotě 24% než první dataset 7.4. Inverze trvala 31 hodin a 295 hodin procesorového času, ale obsahuje 3x víc dat, takže použití MLDA udělá inverzi přibližně 2x rychlejší.

7.3 Problémy s inverzí

Hlavní výzvou s inverzí je výpočetní náročnost – otestování jedné sady parametrů trvalo několik desítek hodin, u posledních pokusů to bylo až několik dní. Po získání výsledků se pak muselo určit, proč se inverze chová jak se chová, jaké parametry jsou pravděpodobní pachatelé a jaké upravy se musí udělat pro možné dosažení lepšího výsledku. Přechod na MLDA toto udělal ještě obtížnější - přidalo se víc konfigurovatelných parametrů a inverze se rozšířila o jeden model.

Další problémy se vyskytovaly ojediněle během inverze, kde inverze prostě spadla. Chybové hlášky pádů moc informací v sobě neměli, převážně velice obecné chyby se kterými nelze dohledat zdroj problému. Tyto pády nastávali v zřejmě náhodných bodech inverze. Někdy to bylo 30 minut od začátku, někdy 15 hodin od začátku. Asi nejčastější z těchto chyb souvisela s pracovním adresářem Ray, kde podle chybové hlášky adresář prostě přestal být dostupný. Důvod těchto chyb je pořád neznámý.

8 Porovnání s existujím řešení

Existující řešení *surrDAHM* (surrogate Delayed Acceptance Metropolis-Hastings) je aplikace, která obsahuje implementaci Bayesovské inverze a byla v minulosti použita společně s Flow123d pro provedení Bayesovské inverze u hydromechanického modelu [18]. Podporuje MLDA a náhradní modely ve formě meta modelů. SurrDAHM vzniklo v rámci dizertační práce [5]. Shrnutí porovnání lze vidět v tabulce 8.

SurrDAHM je pokročilý a více experimentální nástroj, který ale také má své nedostatky. Konfigurace inverze je v surrDAHM náročný proces, množství laditelných parametrů inverze je hodně. Zároveň jsou problémy s nasazením aplikace na výpočetní cluster s použitím kontejneru, což omezuje integraci s Flow123d.

Tato práce má současně podporu pro 15 parametrů na ladění inverze (hodně z nich jsou binární) a je možné přidat podporu pro další. Zároveň je inverze plně paralelní na výpočetním clusteru i na více uzlech. Všechny použité knihovny pro vytvoření aplikace jsou relativně nové, udržované a dobře zdokumentované.

SurrDAHM má podporu pro náhradní meta modely, které jsou sice výpočetně levné, ale špatně aproximují dopředný model. V této práci se používá hrubá konečno-prvková síť, která je výpočetně náročnější, ale zároveň efektivně aproximuje jemný model.

Další omezení v surrDAHM je omezenost návrhové hustoty - obsahuje pouze jeden typ návrhové hustoty. Tato práce obsahuje návrhové hustoty dvě a lze jí rozšířit o další typy, které už jsou naimplementované v rámci tinyDA.

Výsledky inverze se v surrDAHM ukládají do souboru .csv, kde pro vizualizaci je nutné vytvořit vlastní grafy. Výsledky inverze v této práci se ukládají do formátu ArviZ InferenceData, ve kterém lze jednouše získat informace o výsledcích včetně mnoha různých grafů pro vizualizaci.

Název	surrDAHM + Flow123d	tato práce
Počet parametrů	?	15
Paralelizace (1 uzel)	Ano*	Ano
Paralelizace (2+ uzly)	Ano*	Ano
Počet návrhových hustot	1	2**
Formát výstupu	CSV	InferenceData
Dokumentace	Omezená	Plná***

*: Je možná, ale při použití kontejneru problematická

**: Dvě jsou již integrované, ale v tinyDA je jich dostupných k použití více

***: Plná dokumentace na straně použitých knihoven, částečná na straně aplikace

Tabulka 8.1: Porovnání se surrDAHM

9 Závěr

V rámci této práce proběhlo seznámení s Bayesovskou inverzí, jejím matematickým podkladem a překlad vědomostí do jednoduché vlastní implementace inverze. Dále následovala rešerše existujících implementací, kde se pro dvě vybrané knihovny tinyDA a PyMC porovnalo několik dostupných algoritmů inverze a diskutovalo se nad výsledky a rozdíly.

Vzniklá aplikace je ve formě knihovny v jazyce Python. Aplikace úspěšně integruje dopředný model z Flow123d pro hydromechanickou simulaci. Klíčové body aplikace jsou paralelnost i na výpočetním clusteru, použití MLDA pro dosažení lepších výsledků inverze, jednoduchá konfigurace přes konfigurační soubor a nativní podpora vizualizace výsledků a vyčíslení kvality inverze.

Inverzi je možné ladit pomocí konfiguračního souboru, kde se nachází veškeré informace o původních odhadech a parametrech inverze, včetně parametrů návrhové hustoty, různých typů původních odhadů a počtu iterací inverze. Formát souboru není správně ošetřený a není úplně zdokumentovaný a normalizovaný.

Existuje možnost použít MLDA pro efektivnější inverzi. Při správném nastavení konfiguračního souboru má aplikace podporu pro jedno-úrovňovou DA i víceúrovňovou DA. Náhradní modely jsou ve formě hrubších konečno-prvkových sítí. Na základě výsledků práce MLDA pro tuto úlohu představuje méně výpočetně náročnou alternativu, která dosahuje porovnatelných výsledků. Chybí doladit inverzi s použitím MLDA, aby výsledky byly méně korelované.

Celá aplikace je zabalená do jednotného prostředí ve formě Docker kontejneru, který obsahuje již nainstalovaný Flow123d a všechny závislosti aplikace. Kontejnerizace zároveň umožňuje efektivní nasazení na výpočetní cluster.

Aplikace je plně paralelní, každý řetězec je schopný vytížit jedno procesorové jádro. Kvůli výpočetní náročnosti inverze aplikace podporuje také nasazení na výpočetní cluster. Lze použít libovolný cluster, ale aplikace má přímo podporu nasazení přes službu Metacentrum. Nejsou vyřešené náhodné pády aplikace, které možná souvisí s vícevláknovou implementací.

Výsledky inverze se ukládají do formátu `ArviZ InferenceData`, který má nativní podporu pro vizualizaci výsledků inverze a výpočet statistických metrik pro určení kvality výsledků. Tyto grafy se automaticky vygenerují po dokončení inverze. Dostupné jsou také textové soubory s diagnostickými informacemi. Grafy lze dále modifikova v případě, že je nutné přidat další informace.

Možným vylepšením aplikace je přidat podporu pro jiné původní odhady a návrhové hustoty pro efektivnější inverzi. Dále je možné najít lepší parametry inverze k řešené úloze pro dosažení lepších výsledků a případnou kalibraci aplikace. U ML-

DA by bylo možné přidat podporu pro jiné modely než hrubší konečno-prvkové sítě.

Začátek práce obsahuje úvod Bayesovské inverze a úloh použití hydromechanických modelů. Následuje rešerše existujících implementací Bayesovské inverze a jejich porovnání. Praktická část popisuje proces tvorby aplikace, včetně její kontejnerizace a paralelizace. Následovalo nasazení na výpočetní cluster, kde bylo provedeno mnoho iterací a ladění Bayesovské inverze a nejlepší výsledky se použily pro dokázání funkční inverze a aplikace. Aplikace nakonec byla porovnána s existujícím řešení.

Seznam použité literatury

1. VEHTARI, Aki; GELMAN, Andrew; SIMPSON, Daniel; CARPENTER, Bob; BÜRKNER, Paul-Christian. Rank-Normalization, Folding, and Localization: An Improved R^ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*. 2021, roč. 16, č. 2. ISSN 1936-0975. Dostupné z DOI: 10.1214/20-ba1221.
2. DODWELL, T. J.; KETELSEN, C.; SCHEICHL, R.; TECKENTRUP, A. L. Multilevel Markov Chain Monte Carlo. *SIAM Review*. 2019, roč. 61, č. 3, s. 509–545. Dostupné z DOI: 10.1137/19M126966X.
3. COMMUNITY, PyMC. *The MLDA sampler - PyMC example gallery*. [B.r.]. Dostupné také z: https://www.pymc.io/projects/examples/en/latest/samplers/MLDA_simple_linear_regression.html.
4. BUE, Mikkel. *Delayed Acceptance with Adaptive Error Model*. [B.r.]. Dostupné také z: https://github.com/mikkelbue/tinyDA/blob/main/examples/Delayed%20Acceptance%20with%20Adaptive%20Error%20Model.ipynb.
5. BÉREŠOVÁ, Simona. *Bayesian approach to the identification of parameters of differential equations*. 2015. Dis. pr. Technical University of Ostrava.
6. BŘEZINA, Jan; STEBEL, Jan; EXNER, Pavel; HYBŠ, Jan. *Flow123d* [https://flow123d.github.io, repository: https://github.com/flow123d/flow123d]. 2011–2023.
7. RUTQVIST, Jonny et al. Modeling of Damage, Permeability Changes and Pressure Responses during Excavation of the TSX Tunnel in Granitic Rock at URL, Canada. *Environmental Geology*. 2009, roč. 57, č. 6, s. 1263–1274. ISSN 1432-0495. Dostupné z DOI: 10.1007/s00254-008-1515-6.
8. EXNER, Pavel; BÉREŠOVÁ, Simona; BŘEZINA, Jan; ŠTEBEL, Jan. *Application of Bayesian Inversion to Characterization of EDZ*. 2022.
9. WIECKI, Thomas et al. *pymc-devs/pymc: v5.13.1*. Zenodo, 2024. Ver. v5.13.1. Dostupné z DOI: 10.5281/zenodo.10973000.
10. HOFFMAN, Matthew D.; GELMAN, Andrew. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research 15*. 2014.
11. OSTHEGE, Micheal; BRUNKHORST, Greg. DEMetropolis(Z) Sampler Tuning. In: TEAM, PyMC (ed.). *PyMC examples*. [B.r.]. Dostupné z DOI: 10.5281/zenodo.5654871.

12. OSTHEGE, Micheal; BRUNKHORST, Greg. DEMetropolis and DEMetropolis(Z) Algorithm Comparisons. In: TEAM, PyMC (ed.). *PyMC examples*. [B.r.]. Dostupné z DOI: 10.5281/zenodo.5654871.

13. BUE, Mikkel. *tinyDA*. [B.r.]. Dostupné také z: https://github.com/mikkelbue/tinyDA/tree/main/examples.

14. KUMAR, Ravin; CARROLL, Colin; HARTIKAINEN, Ari; MARTIN, Osvaldo. ArviZ a unified library for exploratory analysis of Bayesian models in Python. *Journal of Open Source Software*. 2019, roč. 4, č. 33, s. 1143. Dostupné z DOI: 10.21105/joss.01143.

15. DEVS, ArviZ. *ArviZ InferenceData schema specifications*. [B.r.]. Dostupné také z: https://python.arviz.org/en/latest/schema/schema.html.

16. MORITZ, Philipp et al. *Ray: A Distributed Framework for Emerging AI Applications*. 2018. Dostupné z arXiv: 1712.05889 [cs.DC].

17. *Metacentrum*. [B.r.]. Dostupné také z: https://metavo.metacentrum.cz/cs/about/index.html.

18. BŘEZINA, Jan et al. *Posouzení vlivu zóny ovlivněné ražbou na bezpečnost hlubinného úložiště pomocí výpočetních metod*. 2023. Dostupné také z: https://geomop.nti.tul.cz/endorse/metodika/Endorse_metodika_29_01_2023_final.pdf.

A Odkazy

- GitHub repozitář se zdrojovým kódem:
  https://github.com/bagr-sus/BayesHMI/tree/snapshot