FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Softwarový nástroj pro urychlení vývoje a nasazení nových průmyslových aplikací preparativní chromatografie
Student:	Bc. Adam Svoboda
Vedoucí:	Ing. Svatopluk Henke, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Preparativní chromatografie, zejména potom typu SMB (Simulated Moving Bed), je významná separační technologie tradičně využívaná v chemickém a farmaceutickém průmyslu. Pro nalezení vhodných operačních parametrů a efektivní řízení procesu je vhodné využít numerickou simulaci a optimalizaci procesu na základě matematického modelu. Parametrizaci matematického modelu je třeba provést na základě dat z laboratorních experimentů.
Vypracujte literární rešerši vystihující současný stav v následujících problematikách:
- Postupy při matematickém modelování chromatografického procesu
- Numerické metody pro řešení hyperbolických a parabolických parciálních diferenciálních rovnic
- Metody parametrizace modelů na základě experimentálních dat
- Využití matematických modelů jednotkových operací v procesním průmyslu
- Koncept „Edge Computing" v kontextu sběru a analýzy dat a integrace matematických modelů do průmyslových řídicích smyček
V praktické části proveďte následující tři kroky:
- V prvním kroku praktické části vytvořte program pro nalezení parametrů rovnovážného disperzního modelu a simulaci SMB procesu
- V druhém kroku praktické části navrhněte a za použití vytvořeného programu, již existujících softwarových komponent a platformy Industrial Edge (Siemens, DE) vytvořte multifukční webovou aplikaci integrující následující funkce: správa experimentálních dat, nalezení parametrů rovnovážného disperzního modelu, simulace a optimalizace SMB

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 9. ledna 2023 v Praze.

procesu, monitoring a úprava operačních parametrů SMB procesu, uživatelské rozhraní ve formě webových stránek.
- V třetím kroku praktické části otestujte systém s reálnými daty a na reálném hardware na Ústavu sacharidů a cereálií (VŠCHT Praha).
Diskutujte výhody či nevýhody integrace vývoje modelu a jeho průmyslového nasazení v jednom systému, zejména potom zhodnoťte vhodnost platformy Industrial Edge pro tyto účely.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 9. ledna 2023 v Praze.

Diplomová práce

SOFTWAROVÝ NÁSTROJ PRO URYCHLENÍ VÝVOJE A NASAZENÍ NOVÝCH PRŮMYSLOVÝCH APLIKACÍ PREPARATIVNÍ CHROMATOGRAFIE

Bc. Adam Svoboda

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Svatopluk Henke, Ph.D.
4. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat svému vedoucímu panu Ing. Svatopluku Henke, Ph.D. za jeho ochotu a odbornou pomoc při zpracování této práce. Dále bych chtěl poděkovat panu Ing. Tomáši Svobodovi, bez kterého by tato práce nebyla realizovatelná.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. května 2023 .....

Abstrakt

Modelově prediktivní řízení je důležitým nástrojem průmyslovích systémů, vyžaduje však velké množství dat a výpočetní síly. V této práci vytvoříme softwarové nástroje pro identifikaci matematického modelu chromatografie a implementaci modelového prediktivního řízení pro chromatografický proces za pomoci Siemens Industrial Edge, řešení pro edge computing od společnosti Siemens. Diskutujeme základy chromatografie, matematické modelování, inverzní problém, jeho řešení za pomoci optimalizačních algoritmů a průmyslové řídící systémy.

Klíčová slova Chromatografie, Siemens Industrial Edge, Optimalizace

Abstract

Model predictive control is important tool of industry systems, but it requires large amount of data and processing power. In this work, we will create a software tool for identification of mathematical model of chromatography and for implementing model predictive control for chromatography process with help of Siemens Industrial Edge, solution of edge computing by Siemens. We will discuss basics of chromatography, mathematical models, inverse problem and ways of solving it with optimization algorithms and industrial control systems.

Keywords Chromatography, Siemens Industrial Edge, Optimization

Seznam zkratek

BSON	Binary JSON
COTP	Connection Oriented Transport Protocol
CSS	Cascading Style Sheets
DB	Database
DCS	Distributed Control System
EDM	Equilibrium Dispersion Model
EIM	Equilibrium Ideal Model
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Industrial Edge
IEA	Industrial Edge App
IEAP	Industrial Edge App Publisher
IED	Industrial Edge Device
IEH	Industrial Edge Hub
IEM	Industrial Edge Management
ISO	International Organization for Standardization
IT	Information Technology
JSON	JavaScript Object Notation
MCP	Model Predictive Control
MQTT	Message Queue Telemetry Transport
OT	Operational Technology
PID	Proportional Integral Derivative
PLC	Programmable Logic Controller
REST	Representational State Transfer
SCADA	Supervisory Control and Data Acquisition
SHGO	Simplicial Homology Global Optimization
SMB	Simulated Moving Bed
TCP	Transmission Control Protocol
TPKT	Transport Packet

Introduction

Efektivita je klíčovým faktorem pro úspěch v průmyslové výrobě. Efektivita výroby se týká schopnosti vyrábět vysoce kvalitní produkty za co nejnižší náklady a s co nejmenším množstvím odpadu. V dnešním konkurenčním světě, aby se průmyslové podniky udržely konkurenceschopné, musí být schopny vyrábět produkty s vysokou kvalitou a za rozumnou cenu. Efektivita výroby také pomáhá snižovat náklady na energii a suroviny, což pomáhá snižovat celkové náklady na výrobu a zlepšuje udržitelnost výroby. Efektivita výroby se tedy týká nejen ekonomického hlediska, ale také ekologického a společenského hlediska.

Jeden ze způsobů zvýšení efektivity je použití matematických modelů. Tyto modely se využívají k simulaci výrobních procesů, což umožňuje výrobci optimalizovat různé veličiny, například teplota, tlak, rychlost a složení. Modely se také mohou použít k predikci chování procesu a k optimalizaci vstupů a výstupů. Tyto modely se také mohou využít pro řízení a kontrolu procesu v reálném čase, což umožňuje včasnou detekci a řešení problémů.

Průmyslové procesy však generují velké objemy dat, které je nutné rychle zpracovat, aby byly použitelné pro řízení v reálném čase. Tradiční cloud computing zde nemusí být vhodný, proto se zavedl koncept edge computing.

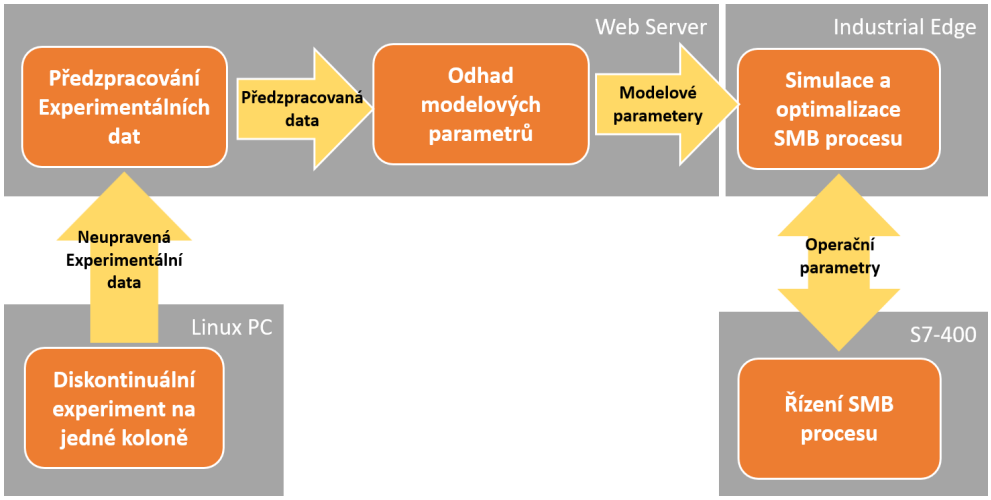
Edge computing se týká výpočetních kapacit, které jsou umístěny přímo v místě, kde se data vytvářejí nebo zpracovávají. Edge computing se liší od tradičního cloud computingu, kdy data jsou ukládána a zpracovávána na vzdálených serverech. Edge computing umožňuje rychlejší zpracování dat, snížení latence a nižší náklady na přenos dat. Edge computing také umožňuje zpracovávat data v místě, kde jsou k dispozici, což je důležité pro aplikace, které vyžadují offline nebo low-bandwidth připojení.

Kapitola 1
Cíl práce

Využití poznatků základního i aplikovaného výzkumu v průmyslové praxi je často zdlouhavý proces, který vyžaduje zapojení expertů z akademické i komerční sféry. Často je největší výzvou překlopení výzkumných nástrojů a softwarových komponent do robustní a ověřené aplikace, která může být přímo využita pro řízení a optimalizaci výrobního procesu, bez vystavení zúčastněných stran nadměrným rizikům, ať už technicky-funkčního charakteru, či z oblasti kyber bezpečnosti.

Cílem této práce je demonstrovat koncept kompletního pracovního postupu od identifikace modelu chemicko-inženýrského procesu až po jeho nasazení v pokročilé řídící smyčce průmyslového zařízení. Práce se zaměřuje na proces kontinuální preparativní chromatografie pro separaci chemicky čistých látek z nejrůznějších směsí. K tomuto účelu budou v rámci práce vyvinuty dva softwarové nástroje, první pro identifikaci matematického modelu daného chromatografického procesu a návazný druhý nástroj, který za využití průmyslově ověřené platformy Industrial Edge (Siemens AG, DE) umožní využití modelu pro řízení reálného procesu. V rámci analýzy a návrhu bylo třeba vypracovat rozsáhlou literární rešerži v oblasti matematického modelování preparativní chromatografie, numerického řešení parciálních diferenciálních rovnic a základům IT/OT integrace, Edge computingu a modelového prediktivního řízení v průmyslové praxi.

Nástroje budou využívat matematický popis chromatografického procesu zvaný Rovnovážný dispersní model, který je založený na zákonech zachování a rovnicích kontinua. Základní myšlenka celého projektu je popsána na diagramu 1.1.



Obrázek 1.1 Základní schéma projektu

První nástroj bude vyvíjen jako webová aplikace a bude zpracovávat data získaná z experimentů, pomocí kterých bude odhadovat modelové parametry. Odhad bude založen na porovnání zpracovaných dat s modelovými daty, které nastroj bude počítat z modelových parametrů za pomoci chromatografického modelu. Rozdíl těchto dvou datových sad bude nástroj minimalizovat za pomoci optimalizačních algoritmů, čímž nalezne optimální modelové parametry. Veškeré nahrané experimenty a výsledky optimalizací budou ukládány do databáze.

Druhý nástroj bude používat získané modelové parametry a matematický model k simulaci chromatografického procesu v režimu SMB, která bude použita k modelovému prediktivnímu řízení reálného procesu. Pro simulaci bude moci uživatel definovat libovolnou SMB stanici, což zahrnuje počet kolon v každé zóně, parametry jednotlivých kolon a velikosti mrtvých objemů dopravních cest mezi kolonami. Také bude umožněno vytvářet libovolnou směs k separaci, počet složek a jejich parmetry. Tento nástroj bude vyvíjen jako aplikace pro platformu Industrial Edge (Siemens AG, DE), která je založená na konceptu edge computing.

Kapitola 2
Analýza a návrh

2.1 Matematické modely chromatografického procesu

2.1.1 Základní princip chromatografie

Chromatografie je proces sloužící k separaci složek směsi. Za jejího objevitele je považován ruský botanik Michail Semenovich Tsvet, který metodu použil k separaci rostlinných pigmentů. [1]

Základní princip je založen na rozdílné rychlosti migrace jednotlivých složek směsi při pohybu médiem. Chromatografický systém se skládá z mobilní fáze, což je proud rozpuštěné směsi, a stacionární fáze, neboli pevně umístěné médium, po kterém se mobilní fáze pohybuje. Každá složka směsi interaguje se stacionární fází jinak, čímž je způsoben rozdíl v rychlosti migrace. [2]

Existuje několik typů chromatografie, jako například kapalinová chromatografie, plynová chromatografie a hmotnostní spektrometrie. V závislosti na typu chromatografie se používají různé typy kolon a různé metody detekce. Cílem chromatografie je získat co nejvyšší čistotu a kvalitu separovaných složek. [2]

Chromatografie se v praxi často realizuje pomocí chromatografických kolon, do kterých směs vstupuje, a ve které se rozděluje na jednotlivé složky pomocí interakce s výplňovým materiálem kolony a rozpouštědlem.

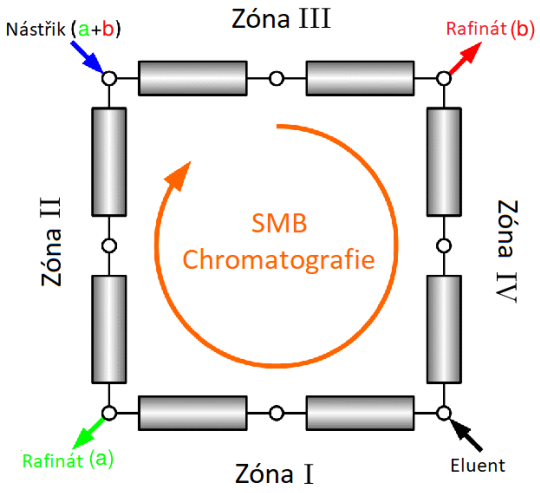
2.1.1.1 Kontinuální chromatografie

Kontinuální chromatografie se liší od běžné diskontinuální chromatografie tím, že směs vstupuje do kolony stále a výstup se odebírá stále, což umožňuje kontinuální průtok a separaci složek. Kontinuální chromatografie má řadu výhod, jako je vysoká účinnost separace, snížení spotřeby rozpouštědel a vysoká produktivita. Kontinuální chromatografie také umožňuje vysokou automatizaci a škálování procesu, což umožňuje vyšší efektivitu a nižší náklady.

Jeden ze způsobů realizace kontinuální chromatografie je tzv. moving-bed, kde se stacionární fáze pohybuje opačným směrem než mobilní fáze a dělená směs je vstřikována do středu kolony. To nám umožňuje nastavit sílu opačného proudu tak, že složky směsi pohybující se vyšší rychlostí opačný proud překonají, ale pomalejší složky budou unášeny, což nám umožňuje sbírat separované složky na každém konci kolony. [3]

Z praktického hlediska je však vytvoření skutečné moving-bed chromatografie obtížné, ve většině případů se proto používá SMB (simulated moving-bed) chromatografie.

Koncept SMB vysvětlíme na obrázku 2.1, kde můžeme vidět osm kolon napojených v kruhu s ventily mezi nimi, které slouží jako vstupy a výstupy systému. Vždy jen čtyři ventily jsou otevřené dva jako vstupy, Nástřik a Eluent, a dva jako výstupy, Extrakt a Rafinát. Opačný



Obrázek 2.1 Schéma SMB chromatografie [4]

proud je poté simulován přepínáním otevřených ventilů ve směru toku mobilní fáze, na obrázku značeno oranžovou šipkou, a jeho rychlost je úměrná intervalu přepínání. [5]

V SMB chromatografii musíme počítat s tím, že u vstupů a výstupů se mění objemové průtoky a u vstupů se míchají dvě směsi s různými koncentracemi. Výpočet průtoků je jednoduché sčítání.

Objemové průtoky se liší v každé zóně a jsou rovny součtu průtoků vstupujících do zóny.

$$\dot{V}_{l+1} = \dot{V}_l + \dot{V}_F$$

Je nutné podotknout, že výstupní průtoky lze reprezentovat zápornou hodnotou, případně je třeba hodnotu odečíst, jsou-li reprezentovány kladnou hodnotou.

U vstupů do SMB stanice dochází k míchání dvou různých roztoků s různými koncentracemi. Koncentrace vstupující do zóny se zpočítá ze vstupujících koncentrací a objemového průtoku.

$$c^{in}_{l+1} = \frac{(c^{out}_l * \dot{V}_l) + (c^{in}_F * \dot{V}_F)}{\dot{V}_{l+1}}$$

Při běhu SMB stanice můžeme pomocí výstupních koncentrací počítat několik parametrů, které jsou dobrými ukazateli vhodnosti operačních parametrů. Místo neustále se měnící aktuální koncetrace na výstupu je vhodné počítat s průměrnou koncentrací za jeden cyklus definovanou jako integrál koncentrací na výstupech přes jeden cyklus.

$$\langle c_A \rangle = \int_{t_s}^{t_s + N_c t^*} \frac{c_A}{N_c t^*} dt$$

t_s je čas, za který se SMB proces dostane do ustáleného stavu, což zajistí, že průměrná koncentrace přes jeden cyklus zůstane při neměnných operačních parametrech konstantní.

První parametr, který budeme sledovat, je čistota. Ta udává procentuální zastoupení složky ve výstupech. Vysoká čistota pomalejší složky v extraktu a rychlejší složky v rafinátu je dobrým ukazatelem vhodně nastavených operačních parametrů.

$$Pu_A = \frac{\langle c_A \rangle}{\langle c_A \rangle + \langle c_B \rangle} * 100\ [\%]$$

Čistota složky A se spočítá jako její průměrná koncentrace dělená součtem koncentrací složky A a B. Při větším počtů složek by se průměrná koncentrace hledané složky dělila součtem průměrných koncentrací všech složek.

Druhým parametrem je výtěžek. Jedná se o poměr hmotnosti složky ve výstupním proudu ku hmotnosti dané složky ve vstupním proudu.

$$Y_A = \frac{\dot{V}\langle c_A \rangle}{\dot{V}_F c_A^F} * 100\ [\%]$$

2.1.2 Přehled přístupů k matematickému popisu

Základem matematického popisu jsou zákony zachování, kdy se provádí bilance na pevném objemovém elementu nebo objemovém elementu pohybujícím se v kontinuu. Pokud vezmeme v potaz homogenitu vrstvy sorbentu s monosferickými částicemi, konstantní hustotu a viskozitu a ideální pístový tok v koloně, pak většina deterministických isotermních modelů bere v úvahu dva nebo více z následujících dějů: konvekce (kontinuální tok nebo kaskáda ideálně míchaných reaktorů), disperze, adsorbční rovnováha nebo adsorbční kinetika, přestup hmoty z mobilní fáze do stacionární fáze a další.

V rámci této práce byl zvolen popis chromatografického děje pomocí rovnovážného dispersního modelu. Tento model lze využít v kombinaci s lineární či nelineární isotermou charakterizující adsorbční rovnováhu separovaných látek a sorbentu. Při použití vhodných numerických metod dává řešení tohoto modelu dle dostupné literatury poměrně přesnou reprezentaci reálného chromatografického děje i v mezních podmínkách.[6]

2.1.2.1 Okrajové podmínky

Řešení rovnovážného dispersního modelu je okrajová úloha a je třeba definovat okrajové podmínky.

Pro modely chromatografie jsou v literatuře běžně uváděny dva typy okrajových podmínek. Tyto podmínky musí zahrnovat koncentraci vstupující na kolonu c^{in}.

Prvním typem jsou Dirichletovy okrajové podmínky, které jsou definovány jednoduchým vztahem, kde koncentrace na začátku kolony se rovná koncentraci vstupující do kolony v daném čase.

$$c_{l,i}(t,0) = c^{in}(t)$$

$$c_{l,i}(t,\infty) = 0$$

Druhým typem jsou Danckwertsovy okrajové podmínky, které berou v úvahu dispersní člen a jsou vhonější v případech, kdy disperse hraje významnou roli.

$$\left(\frac{\partial c_{l,i}(t,0)}{\partial x}\right) = \frac{u_m}{D_{ax,i}}(c_{l,i}(t,0)) - c_{l,i}^{in})$$

$$\left(\frac{\partial c_{l,i}(t,L)}{\partial x}\right) = 0$$

Dle dat získaných z experimentů provedených v rámci výzkumu na Ústavu sacharidů a cereálií (VŠCHT v Praze) bylo zjištěno, že při separaci na sorbentech s relativně malou velikostí částic (150-500 m) je zdánlivá disperze významným jevem ovlivňujícím chromatografický proces. Z toho důvodu bylo využito matematického popisu pomocí Danckwertsových okrajových podmínek.

2.1.3 Rovnovážný model

Rovnovážný model je založen na modelování rovnováhy mezi komponentami rozdělenými v koloně pomocí rovnic rovnováhy. Tyto rovnice se zpravidla týkají koncentrací jednotlivých složek ve vstupní směsi, koncentracích v eluentu a koncentracích komponent ve výstupu z kolony. Mezi často používané patří rovnovážný ideální model (EIM) a rovnovážný disperzní model (EDM).

2.1.3.1 Rovnovážný ideální model

Rovnovážný ideální model je založen na předpokladu, že všechny složky vzorku jsou zcela rozpuštěny v mobilní fázi a nevyskytuje se mezi nimi žádná interakce. V tomto modelu se předpokládá, že kolona má dokonalou homogenitu a nevyskytují se v ní žádné fyzikální jevy, jako je difúze a turbulence. Výsledkem tohoto modelu je ostrý pík pro každou složku, která je oddělena v koloně. Tento model je popsán následující parciální diferenciální rovnicí pro složku i. [7]

$$\frac{\partial c_{l,i}}{\partial t} = -u_m \frac{\partial c_{l,i}}{\partial x} - \frac{1-\varepsilon}{\varepsilon} \frac{\partial q^*_{i,s}}{\partial t}$$

Použijeme Danckwertsovy okrajové podmínky. Počáteční podmínka je všude rovna nule, protože reprezentuje prázdnou kolonu.

Počáteční podmínka:

$$c_{l,i}|_{t=0} = 0$$

Okrajová podmínka:

$$\left(\frac{\partial c_{l,i}}{\partial x}\right)\Bigg|_{x=0} = u_m(c_{l,i} - c^{in})$$

2.1.3.2 Rovnovážný disperzní model

Disperzní rovnovážný model zahrnuje vlivy, které jsou při ideálním modelu zanedbány. V tomto případě se předpokládá, že dochází k disperzi, difúzi a turbulentním jevům v koloně. V důsledku toho jsou píky výrazně širší než v případě ideálního modelu, což může zkomplikovat rozlišení mezi složkami vzorku. Tento model je popsán následující parciální diferenciální rovnicí pro složku i. Novým parametrem v této rovnici je disperzní koeficient $D_{ax,i}$. [6] [8]

$$\frac{\partial c_{l,i}}{\partial t} = -u_m \frac{\partial c_{l,i}}{\partial x} + D_{ax,i} \frac{\partial^2 c_{l,i}}{\partial x^2} - \frac{1-\varepsilon}{\varepsilon} \frac{\partial q^*_{s,i}}{\partial t}$$

Opět použijeme Danckwertsovy okrajové podmínky a počáteční podmínka je opět všude rovna nule, protože reprezentuje prázdnou kolonu.

Počáteční podmínka:

$$c_{l,i}|_{t=0} = 0$$

Okrajové podmínky:

$$\left(\frac{\partial c_{l,i}}{\partial x}\right)\Bigg|_{x=0} = \frac{u_m}{D_{ax,i}}(c_{l,i} - c^{in})$$

$$\left(\frac{\partial c_{l,i}}{\partial x}\right)\Bigg|_{x=L} = 0$$

2.1.3.3 Isotermy

V rovnovážném modelu chromatografie, isotherma popisuje vztah mezi koncentrací složek v pevné (stacionární) fázi a v mobilní fázi. Složky se rozdělují mezi pevnou a mobilní fázi v závislosti na jejich afinitě k oběma fázím. Čím větší je afinita složky k pevné fázi, tím delší je její retenční čas. [9]

Isoterma je grafický záznam tohoto vztahu, kde hodnoty retenčního času složek jsou vykresleny proti jejich koncentraci ve stacionární fázi. Tuto informaci lze využít k charakterizaci interakce mezi složkami a stacionární fází. Mezi nejčastěji používané patří lineární a Langmuirova isoterma.

Lineární isoterma tento vztah popisuje lineární funkcí. Jedná se o nejjednodušší isotermu, neuvažuje kapacitu sorbentu a není proto vhodná pro vyšší koncentrace. Jednoduchost je ale v mnoha případech i výhodou a lineární isoterma je popsána jedinou konstantou K_H zvanou Henryho konstanta, což značně zjednodušuje výpočet.

$$q^*_{i,s} = K_{H,i} c_{i,l}$$

$$0 < K_{H,i} < 1$$

Langmuirova isoterma kromě Langmuirovy konstanty K_L už uvažuje i maximální kapacitu sorbentu $Q_{i,max}$ zvanou saturační koeficient. Při nižších koncentracích má lineární charakter a při vyšších konverguje k limitě. [9]

$$q^*_{i,s} = \frac{Q_{i,max} K_{L,i} c_{i,l}}{1 + K_{L,i} c_{i,l}}$$

2.2 Numerické metody rovnovážného disperzního modelu

Matematickým popisem rovnovážného disperzního modelu chromatografie vzniknou parciální diferenciální rovnice. V této sekci popíšeme možné metody jejich řešení.

2.2.1 Přehled numerických metod

Obecné řešení parciálních diferenciálních rovnic je obtížné, proto se k jejich řešení používají numerické metody. Tyto metody jsou založeny na diskretizaci kontinuálního prostoru na konečný počet bodů a aproximaci hodnot v těchto bodech. Mezi nejčastější metody patří:

1. Metoda konečných prvků (finite element method - FEM)
2. Metoda konečných objemů (finite volume method - FVM)
3. Metoda konečných diferencí (finite difference method - FDM)

Tyto metody se liší způsobem, jakým diskretizují a řeší tyto diferenciální rovnice. FEM diskretizuje prostor na množinu elementů, které se dále aproximují za pomocí interpolace. FVM diskretizuje prostor na množinu objemů a zkoumá tok hodnoty funkce těmito objemy. FDM diskretizuje prostor na množinu bodů a zkoumá funkci v těchto bodech. Výběr správné metody závisí na charakteristikách konkrétního problému a na požadavcích na přesnost a účinnost řešení. [10]

Dalším důležitým prvkem pro řešení parciálních diferenciálních rovnic jsou okrajové a počáteční podmínky. Jedná se o sadu dalších omezení, bez kterých rovnice obecně nemají jednoznačné řešení. Tyto podmínky stanovují stav systému na počátku a v mezních bodech. Počáteční a okrajové podmínky výrazně ovlivňují výsledné řešení a je proto důležité, aby byly dobře určeny. [11]

2.2.2 Náhrady v metodě konečných diferencí

Při použití metody konečných diferencí, parciální diferenciální rovnice často nejde řešit ani pro diskretizovaný systém. Parciální diferenciální rovnice se proto často nahrazují rovnicemi, které výsledky aproximují z okolních bodů systému. Tyto náhrady mohou být explicitní nebo implicitní.

Explicitní náhrada počítá aktuální stav přímo z předchozích stavů, zatímco implicitní hledá řešení rovnice, ve které vystupuje aktuální i předchozí stav a vede na soustavy rovnic. Implicitní metody jsou náročnější na výpočet a používají se především u tzv. tuhých rovnic, které jsou numericky nestálé a vyžadovaly by příliš jemnou diskretizaci, aby explicitní metody dávaly přijatelný výsledek.

2.2.2.1 Přehled explicitních metod

Existuje několik metod, které nahradí rovnici pro stav v daném bodě explicitní rovnicí. Mezi nejběžnější patří:

1. Eulerova metoda
2. Runge-Kuttovy metody
3. Adams-Bashforthova metoda
4. Taylorův rozvoj

Výběr závisí na požadavcích na stabilitu, přesnost a výpočetní náročnost. Zde budeme diskutovat pouze Runge–Kuttovu metodu 4. řádu.

Runge–Kuttova metoda 4. řádu pro krok h má následující předpis:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$t_{n+1} = t_n + h$$

$$k_1 = h * f(t_n, y_n)$$

$$k_2 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$

$$k_3 = h * f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$

$$k_4 = h * f(t_n + h, y_n + k_3)$$

Tato metoda se v jednom kroku dopouští chyby čtvrtého řádu, menší krok tak obvykle vede k přesnějším odhadům. [12]

2.2.2.2 Přehled implicitních metod

Stejně tak existuje několik metod, které aproximují stav implicitně. Mezi nejběžnější patří:

1. Eulerova metoda zpětné diference
2. Metody zpětné diference
3. Adams-Moultonova metoda
4. Cranck-Nicolsonova metoda

[13]

Výběr opět závisí na požadavcích na přesnost a výpočetní náročnost. Výhodou implicitních metod je často stabilita, která je zpravidla podstatně větší než u explicitních metod. V této práci budeme diskutovat pouze Cranck-Nicolsonovu metodu.

Cranck-Nicolsonova metoda aproximuje stav pro funkci F v čase $n+1$ a prostorovém bodu i následujícím předpisem:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}\left[F_i^{n+1}\left(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right) + F_i^n\left(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right)\right]$$

Počítá tedy průměr centrálních aproximací v předchozím a aktuálním časovém kroku.

Stejně jako ostatní implicitní metody, i tato metoda vede na maticové řešení rovnic. Výhodou Cranck-Nicolsonovy metody je, že tvar této matice je tridiagonální, což značně ulehčuje řešení. [14]

2.2.3 Řešení EDM s lineární isotermou metodou konečných diferencí

Po dosazení lineární isotermy do rovnice EDM modelu dostáváme následující rovnici:

$$\frac{\partial c}{\partial t} = \frac{D_{ax}}{\frac{(1-\varepsilon)K_H}{\varepsilon} + 1}\frac{\partial^2 c}{\partial x^2} - \frac{u_m}{\frac{(1-\varepsilon)K_H}{\varepsilon} + 1}\frac{\partial c}{\partial x}$$

Počáteční podmínky:

$$c|_{t=0} = 0$$

Okrajová podmínka:

$$\left(\frac{\partial c}{\partial x}\right)\Big|_{x=0} = u_m(c - c^{in})$$

Pro usnadnění zápisu si v rovnici provedeme substituci členů bez derivace.

$$a = \frac{D_{ax}}{\frac{(1-\varepsilon)K_H}{\varepsilon} + 1}$$

$$b = \frac{u_m}{\frac{(1-\varepsilon)K_H}{\varepsilon} + 1}$$

Výsledná rovnice má pak následující tvar:

$$\frac{\partial c}{\partial t} = a\frac{\partial^2 c}{\partial x^2} - b\frac{\partial c}{\partial x}$$

Pro řešení metodou konečných diferencí je třeba rozdělit stavový prostor času a délky kolony na prostor bodů. Počet diskretizačních bodů je potřeba nastavit na vhodnou hodnotu, kde výsledek má přijatelnou přesnost a výpočet trvá přijatelnou dobu.

Prostorovou dimenzi x v intervalu $\langle 0, L\rangle$ rozdělíme na n stejně velkých částí.

$$x_0 = 0, x_1 = h, x_2 = 2h, \ldots, x_{n-1} = L - h, x_n = L$$

$$h = \frac{L}{n},\ x_i = ih \text{ pro } i = 0, 1, \ldots, n$$

Časovou dimenzi t v intervalu $\langle 0, T\rangle$ rozdělíme na r stejně velkých částí.

$$t_0 = 0, t_1 = k, t_2 = 2k, \ldots, t_{r-1} = T - k, t_r = T$$

$$k = \frac{T}{r},\ t_j = jk \text{ pro } j = 0, 1, \ldots, r$$

Pro aproximaci výsledků můžeme použít explicitní nebo implicitní metodu.

2.2.3.1 Cranck-Nicolsonova metoda pro řešení EDM s lineární isotermou

Když aplikujeme Cranck-Nicolsonovu metodu, dostaneme následující aproximaci:

$$\frac{c_i^{j+1} - c_i^j}{k} = \frac{a}{2}\left[\frac{c_{i+1}^j - 2c_i^j + c_{i-1}^j}{h^2} + \frac{c_{i+1}^{j+1} - 2c_i^{j+1} + c_{i-1}^{j+1}}{h^2}\right] - \frac{b}{2}\left[\frac{c_{i+1}^j - c_{i-1}^j}{2h} + \frac{c_{i+1}^{j+1} - c_{i-1}^{j+1}}{2h}\right]$$

Levá okrajová podmínka:

$$\frac{1}{2}\left[\frac{c_1^j - c_0^j}{h} + \frac{c_1^{j+1} - c_0^{j+1}}{h}\right] = u_m(c_0^{j-1} - c_{in})$$

Pravá okrajová podmínka:

$$\frac{1}{2}\left[\frac{c_n^j - c_{n-1}^j}{h} + \frac{c_n^{j+1} - c_{n-1}^{j+1}}{h}\right] = 0$$

Zapsání těchto rovnic do matice vede na následující soustavu rovnic:

$$A\vec{c}^{j+1} = B\vec{c}^j + \vec{c}_{in}$$

Kde A a B jsou tridiagonální matice, $\vec{c}^j$ je známý vektor řešení v předchozím kroku a $\vec{c}^{j+1}$ je hledané řešení.

$$A = \begin{bmatrix} u_m + \frac{1}{2h} & -\frac{1}{2h} & 0 & 0 & \ldots & 0 \\ -\frac{ka}{2h^2} - \frac{kb}{4h} & 1 + \frac{ka}{h^2} & -\frac{ka}{2h^2} + \frac{kb}{4h} & 0 & \ldots & 0 \\ 0 & -\frac{ka}{2h^2} - \frac{kb}{4h} & 1 + \frac{ka}{h^2} & -\frac{ka}{2h^2} + \frac{kb}{4h} & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & -\frac{ka}{2h^2} - \frac{kb}{4h} & 1 + \frac{ka}{h^2} & -\frac{ka}{2h^2} + \frac{kb}{4h} \\ 0 & \ldots & 0 & 0 & -\frac{1}{2h} & \frac{1}{2h} \end{bmatrix}$$

$$B = \begin{bmatrix} -\frac{1}{2h} & \frac{1}{2h} & 0 & 0 & \ldots & 0 \\ \frac{ka}{2h^2} + \frac{kb}{4h} & 1 - \frac{ka}{h^2} & \frac{ka}{2h^2} - \frac{kb}{4h} & 0 & \ldots & 0 \\ 0 & \frac{ka}{2h^2} + \frac{kb}{4h} & 1 - \frac{ka}{h^2} & \frac{ka}{2h^2} - \frac{kb}{4h} & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \frac{ka}{2h^2} + \frac{kb}{4h} & 1 - \frac{ka}{h^2} & \frac{ka}{2h^2} - \frac{kb}{4h} \\ 0 & \ldots & 0 & 0 & \frac{1}{2h} & -\frac{1}{2h} \end{bmatrix}$$

$$\vec{c}^{j+1} = \begin{bmatrix} c_0^{j+1} \\ c_1^{j+1} \\ \vdots \\ c_i^{j+1} \\ \vdots \\ c_n^{j+1} \end{bmatrix}, \ \vec{c}^{j} = \begin{bmatrix} c_0^{j} \\ c_1^{j} \\ \vdots \\ c_i^{j} \\ \vdots \\ c_n^{j} \end{bmatrix}, \ \vec{c}_{in} = \begin{bmatrix} u_m c_{in} \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

2.2.4 Řešení EDM s nelineární isotermou metodou konečných diferencí

Po dosazení nelineární Langmuirovy isotermy do rovnice EDM modelu dostáváme tuto rovnici:

$$\frac{\partial c}{\partial t} = \frac{D_{ax}}{\frac{(1-\varepsilon)QK_L}{(-K_L c+1)^2\varepsilon}+1}\frac{\partial^2 c}{\partial x^2} - \frac{u_m}{\frac{(1-\varepsilon)QK_L}{(-K_L c+1)^2\varepsilon}+1}\frac{\partial c}{\partial x}$$

Počáteční podmínky:

$$c|_{t=0} = 0$$

Okrajová podmínka:

$$\left(\frac{\partial c}{\partial x}\right)\Big|_{x=0} = u_m(c - c^{in})$$

Stejný postup jako u lineární isotermy vede na substituci.

$$a = \frac{D_{ax}}{\frac{(1-\varepsilon)QK_{HL}}{(-K_L c+1)^2\varepsilon}+1}$$

$$b = \frac{u_m}{\frac{(1-\varepsilon)QK_L}{(-K_L c+1)^2\varepsilon}+1}$$

Zde nám však nastává problém. V těchto rovnicích nám vystupuje koncentrace, což je proměnná, kterou derivujeme. To znamená, že místo soustavy lineárních rovnic dostaneme soustavu nelineárních rovnic, což zkomplikuje výpočet.

Diskretizace je ekvivalentní s řešením s lineární isotermou.

2.2.4.1 Cranck-Nicolsonova metoda pro řešení EDM s nelineární isotermou

Kvůli výskytu koncentrace v členech bez derivaci nemůžeme použít stejný postup, který vedl na soustavu rovnic. Upravíme proto předpis následujícím způsobem:

$$\begin{aligned} \frac{c_i^{j+1} - c_i^j}{k} &= \frac{1}{2}\left[a^j \frac{c_{i+1}^j - 2c_i^j + c_{i-1}^j}{h^2} + a^{j+1}\frac{c_{i+1}^{j+1} - 2c_i^{j+1} + c_{i-1}^{j+1}}{h^2}\right] \\ &\quad -\frac{1}{2}\left[b^j \frac{c_{i+1}^j - c_{i-1}^j}{2h} + b^{j+1}\frac{c_{i+1}^{j+1} - c_{i-1}^{j+1}}{2h}\right] \end{aligned}$$

$$a^j = \frac{D_{ax}}{\frac{(1-\varepsilon)QK_{HL}}{(-K_L c_i^j+1)^2\varepsilon}+1}$$

$$a^{j+1} = \frac{D_{ax}}{\frac{(1-\varepsilon)QK_{HL}}{(-K_L c_i^{j+1}+1)^2\varepsilon}+1}$$

$$b^j = \frac{u_m}{\frac{(1-\varepsilon)QK_L}{(-K_L c_i^j+1)^2\varepsilon}+1}$$

$$b^{j+1} = \frac{u_m}{\frac{(1-\varepsilon)QK_L}{(-K_L c_i^{j+1}+1)^2\varepsilon}+1}$$

Tento předpis vede na soustavu nelineárních rovnic, které budeme řešit Newtonovou metodou. [12] Okrajové podmínky jsou stejné jako v případě lineární isotermy.

2.3 Metody parametrizace modelů na základě experimentálních dat

2.3.1 Inverzní problém

Inverzní problém je proces, ve kterém se z množiny měření či pozorování snažíme určit příčinu daného jevu. V případě existence matematického modelu, inverzní problém může znamenat snahu získat modelové parametry z naměřených dat. Důležitou vlastností inverzních problémů je fakt, že mívají více řešení, často nekonečné množství. Řešení inverzních problémů často zahrnuje použití matematických modelů, metod optimalizace a počítačových metod k odhadu neznámého vstupu z pozorovaného výstupu. [15]

Konkrétně pro chromatografii, inverzní problém znamená odhadnutí modelových parametrů tak, aby model co nejlépe odpovídal naměřeným datům. Tato data jsou ve formě naměřených koncentrací na výstupu z kolony v čase. Také známe rozměry kolony, čas a objem nástřiku a koncentrace složek v roztoku. Cílem bude co nejlépe odhadnout modelové parametry, jako jsou porozita, disperzní koeficient, Henryho nebo Lanqmuirova konstanta a saturační koeficient, což provedeme pomocí optimalizace.

2.3.2 Klasifikace optimalizačních problémů

Obecně optimalizační problém je úloha nalezení nejlepšího možného řešení. Které řešení je nejlepší rozhodneme podle optimalizačního kritéria, typicky například cena nebo čas. Pro dané kritérium můžeme hledat řešení s největší resp. nejmenší hodnotou kritéria. Tyto problémy pak nazýváme maximalizační resp. minimalizační.

Optimalizační problém můžeme také dělit podle výstupu. Problém, kde chceme rozhodnout, zda existuje řešení splňující nějaké kritérium, nazýváme rozhodovací problém, problém nalezení řešení s nejlepším možným kritériem, nazýváme konstruktivní problém, problém nalezení všech řešení splňující kritérium, nazýváme početní problém, atd. [16]

2.3.2.1 Řešení optimalizačních problémů

Existuje mnoho přístupů k řešení optimalizačních problémů, jejichž vhodnost nejčastěji závisí na druhu úlohy, požadavcích na dobu běhu a přesnosti výsledku.

Základním příkladem je tzv. „brute force“, což znamená prohledání celého stavového prostoru, nalezení všech řešení a výběr toho nejlepšího. Tímto přístupem vždy nalezneme pro diskrétní

konečný stavový prostor nejlepší řešení, ovšem prohledání celého stavového prostoru trvá příliš dlouho, aby byl tento přístup přijatelný.

Pro diskrétní stavový prostor je možné použít metodu větví a hranic, která stavový prostor prohledává jako strom a zanedbává větve, ve kterých se nemůže vyskytovat optimální řešení.

Další možností jsou simplexové metody. Tyto metody vyhodnocují funkci v několika bodech tvořící simplex. V každé iteraci se pak jeden z bodů, typicky ten s nejhorším vyhodnocením, nahradí novým bodem. Různé simplexové metody se od sebe liší způsobem výběru nového bodu.

Poslední zmíníme gradientové metody, které začínají v nějakém bodě a využívají vlastnosti gradientu k nalezení maxima nebo minima. Tyto metody nalézají lokální minimum a nenaleznou tak vždy nejlepší výsledek, pokud stavový prostor není uniformě konvexní nebo konkávní. [16]

2.3.2.2 Odhad parametrů EDM

V této práci budeme hledat nejlepší možné řešení. Optimalizační kritérium bude rozdíl mezi experimentálními a modelovými daty. Optimalizační parametry budou záviset na výběru modelu.

Protože prohledáváme kontinuální a potenciálně nekonečný stavový prostor, hledání globálního minima by za těchto podmínek bylo náročné a nepraktické. Zavedeme proto několik omezení, které nám hledání minima usnadní.

Zaprvé omezíme možné hodnoty optimalizačních parametrů do nějakého intervalu. Optimalizační parametry pro EDM reprezentují fyzikální vlastnosti modelu. Díky této znalosti je možné zvolit interval arbitrárně, tak aby parametr dával fyzikálně smysl.

Zadruhé po uživateli budeme vyžadovat počáteční odhad a budeme předpokládat, že je počáteční odhad dobrý, což je předpoklad pro správný výsledek několika metod, které použijeme. Dáme také uživateli nástroje pro to, aby dokázal najít dobrý počáteční odhad.

Zatřetí pro některé algoritmy bude nutné stavový prostor diskretizovat na konečnou množinu bodů.

Konkrétní algoritmy které použijeme:

1. „Brute force“ metoda
2. Nelder-Meadova metoda
3. Powellova metoda
4. SHGO

„Brute force“ metodu jsme popisovali výše a použijeme ji pro její jednoduchost a porovnání s ostatními metodami. Standardní „brute force“ metoda není pro kontinuální stavový prostor možná použít, prohledat nekonečné množství bodů by trvalo nekonečně dlouhou dobu, metoda si proto ve stavovém prostoru definuje síť bodů, které prohledá.

Nelder-Meadova metoda je simplexová metoda. Tato metoda v každém kroku seřadí hodnoty vrcholů $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$. Poté spočítá geometrický střed všech bodů kromě toho nejhoršího x_o a z něj spočítá odražený bod $x_r = x_o - \alpha(x_o - x_{n+1})$. Na základě odraženého bodu se algoritmus větví na několik případů.

Pokud je odražený bod lepší než druhý nejhorší bod ale horší než nejlepší bod, nahradí nejhorší bod v simplexu a pokračuje se novou iterací.

Pokud je odražený bod lepší než nejlepší bod, spočítá se nový expanzní bod $x_e = x_o - \gamma(x_r - x_o)$. Pokud je expanzní bod lepší než odražený bod, nahradí se nejhorší bod v simplexu expanzním bodem, pokud ne, nahradí se odraženým bodem.

Pokud je odražený bod mezi nejhorším a druhým nejhorším bodem, spočítá se zkrácený bod mino simplex $x_c = x_o - \rho(x_r - x_o)$. Pokud je vnější zkrácený bod lepší než odražený, nahradí zkrácený bod nejhorší bod simplexu a pokračuje se novou iterací.

Pokud je odražený bod horší než nejhorší bod, spočítá se zkrácený bod uvnitř simplexu $x_c = x_o - \rho(x_{n+1} - x_o)$. Pokud je vnitřní zkrácený bod lepší než nejhorší bod, nahradí zkrácený bod nejhorší bod simplexu a pokračuje se novou iterací.

Pokud v obou případech zkrácený bod není lepší, všechny body kromě nejlepšího se nahradí novým bodem $x_i = x_1 - \sigma(x_i - x_1)$.

α, γ, ρ a σ jsou koeficienty jednotlivých bodů. Standardní hodnoty jsou $\alpha = 1$, $\gamma = 2$, $\rho = 0.5$ a $\sigma = 0.5$. [17]

Powellova metoda je příkladem gradientové metody. Nejprve zadáme počáteční bod a v tomto bodě se vypočte gradient. Z gradientu se vypočtou směry, ve kterých se bude hledat. Metoda poté sérií jednodimenzionálních hledání podél těchto vektorů a vektorů z nich složených hledá minimum. Jednodimenzionální hledání může být provedeno libovolnou metodou. [18]

SHGO (simplicial homology global optimisation) využívá konceptu Simpliciální homologie pro zkoumání topologických vlastností stavového prostoru. Simpliciální homologie je způsob, jak převést geometrický objekt na objekt algebraický, konkrétně na řadu abelovských grup, které poté můžeme zkoumat.

Tento algoritmus umožňuje globální optimalizaci pro libovolný problém. Je také možné poskytnout algoritmu další informace o stavovém prostoru pro zrychlení, například definovat meze. [19]

2.4 Využití matematických modelů v procesním průmyslu

Matematické modely se v průmyslu používají k simulaci, optimalizaci a řízení výrobních procesů. Tyto modely lze vytvářet pro jednotlivé procesní jednotky nebo pro celý výrobní systém a zahrnují matematické popisy procesních vstupů, operačních podmínek a výstupů. Tyto modely mohou být využity k výpočtu parametrů jako jsou produktivita, kapacita, spotřeba energie, atd. a k následnému porovnání s měřenými daty. Modely také slouží k testování různých scénářů a řešení, což umožňuje výrobním podnikům nalézt nejlepší postupy pro maximalizaci efektivity a ekonomiky procesu. [20]

2.4.1 Řízení a regulace

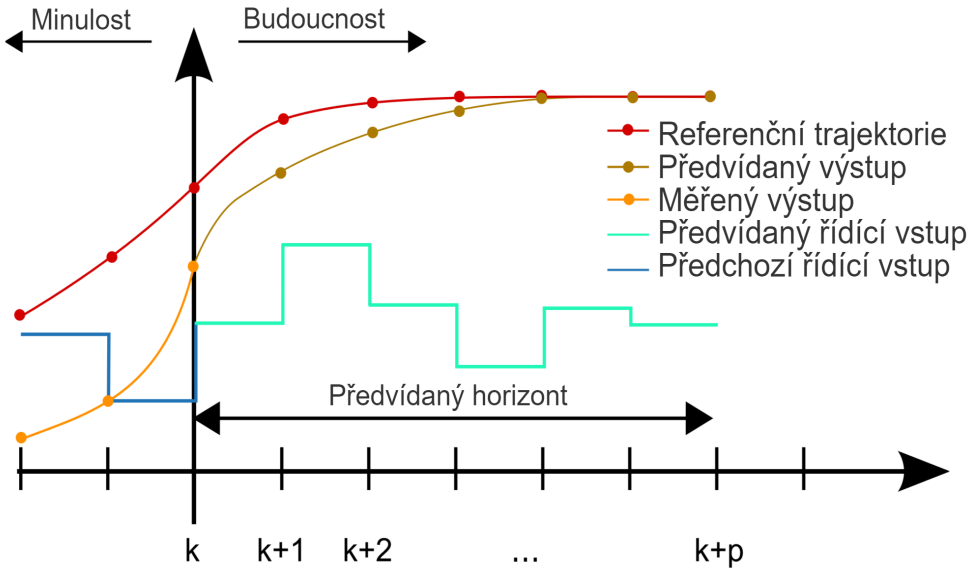
Řízení a regulace se v procesním průmyslu používají k udržení výrobních procesů v požadovaném stavu. Tyto systémy pomáhají udržovat procesy v rovnováze tím, že měří vstupy a výstupy a pomocí řídicích algoritmů upravují vstupy tak, aby výstupy odpovídaly požadovaným hodnotám.

Tradiční metody řízení a regulace jsou manuální nebo mechanické postupy. Tyto metody mohou zahrnovat manuální ovládání nebo nastavení parametrů, použití přímo řízených prvků jako jsou ventily a spínače, nebo použití jednoduchých analogových nebo digitálních regulačních zařízení, jako jsou regulátory teploty a tlaku. Tyto metody se již používají dlouhou dobu, stále se však používají v některých prostředích, kde jsou požadovány jednoduché a spolehlivé řešení.

Mezi jednoduchá a spolehlivá digitální regulační zařízení patří tzv. PID (proportional-integral-derivative) regulátor, který měří rozdíl mezi požadovanou hodnotou a skutečným výstupem a používá tento rozdíl k určení toho, jak má upravit vstup procesu. [21]

Pokročilé metody řízení používají sofistikovanější algoritmy pro řízení. Často jsou založeny na matematických modelech, fuzzy logice nebo umělé inteligenci. Jsou často používány v oblastech, kde je nutná vysoká míra přesnosti a stability.

Častou pokročilou metodou je modelové prediktivní řízení (Model Predictive Control - MPC). Tato metoda používá matematický model procesu k predikci budoucího výstupu a následnému řízení vstupu procesu tak, aby byl výstup co nejblíže požadované hodnotě 2.2. MPC se odlišuje od jednodušších metod řízení, jako je např. PID regulace, tím, že používá model procesu k predikci výstupu v budoucnu a optimalizuje vstup procesu vzhledem k požadované hodnotě. Toto optimalizační řešení se provádí pomocí výpočtů a algoritmů a umožňuje MPC reagovat na změny v procesu a zajistit, aby byl výstup co nejblíže požadované hodnotě. MPC se často používá v náročných procesech, jako jsou např. procesy chemické výroby, kde je nutné řídit mnoho vstupů a výstupů a zajistit co nejlepší kvalitu výstupu. [22]



Obrázek 2.2 Schéma modelového prediktivního řízení [23]

2.5 Edge Computing

2.5.1 Průmyslové řídicí systémy

Průmyslové řídicí systémy jsou speciální typy řídicích systémů, které se používají v průmyslu k řízení a automatizaci výrobních procesů. Automatizace umožňuje efektivnější a spolehlivější provoz, což zvyšuje produktivitu a snižuje náklady na provoz. Kladou velký důraz na efektivitu, spolehlivost a udržitelnost.

Důležitým vývojovým krokem v moderních průmyslových systémech, ve kterých se stále více využívají technologie jako jsou senzory, řídicí systémy a analytické nástroje, byla jejich digitalizace a propojení s IT systémy, které přináší mnoho výhod, ale také mnoho překážek. [24]

Mezi nejběžnější typy systémů patří SCADA (Supervisory Control and Data Acquisition), která zajišťuje řízení a monitoring přež velké vzdálenosti z několika továren za pomoci centrálního řídícího systému. Dalším běžným typem systému je DCS (Distributed Control System), který řídí produkci na jedné lokalitě za pomoci centralizováné řídící smyčky. Reálné průmyslové řídící systémy jsou často kombinací těchto dvou typů. [25]

Důležitou součástí všech průmyslových řídících systémů jsou PLC (programmable logic controller), což jsou průmyslové počítače přizpůsobeny k ovládání výrobních procesů. Mohou obsahovat mnoho druhý digitálních a analogových vstupů a výstupů, být odolné vůči teplotě nebo vibracím atd. PLC byly poprvé použity v 60. letech v automobilovém průmyslu, od té doby se rozšířili do všech odvětví průmyslu. [26]

2.5.2 Základy IT/OT integrace

IT/OT integrace je proces spojování informačních technologií (IT) a operačních technologií (OT) do jednoho celku. Cílem je propojit IT systémy, které se starají o podnikové procesy a data, s OT systémy, které se starají o výrobní procesy a řízení technologií. Integrace umožňuje přenášet data mezi IT a OT a poskytuje jednotný pohled na výrobní procesy, což umožňuje efektivnější řízení a plánování výroby. IT/OT integrace také umožňuje využívat moderní technologie, jako jsou

analýza dat a umělá inteligence, k řízení výrobních procesů a zlepšování efektivity a produktivity. Tyto integrace jsou stále důležitější pro společnosti, které se snaží být konkurenceschopné v rychle se měnícím průmyslovém prostředí. [27]

Jednou z překážek je nutnost zajištění bezpečnosti průmyslových sítí a ochrany proti kybernetickým útokům. Při integraci IT a OT se otevírají nové přístupové cesty pro útočníky, kteří mohou využít bezpečnostních chyb v systémech a napadnout výrobní procesy, což může mít vážné dopady. Je proto nutné dbát na robustnost a odolnost bezpečnostních systémů. [25]

Dalším výzvou je omezená interoperabilita mezi IT a OT systémy. IT systémy jsou často navrženy pro běžné kancelářské účely a nejsou schopny pracovat s výrobními zařízeními a systémy. Na druhé straně OT systémy jsou často proprietární a navrženy pro specifické úkoly. Pokud není integrace provedena správně, může to vést k problémům s kompatibilitou a omezení funkcí a výkonu obou systémů. [27]

2.5.3 Cloud a Edge computing, výhody a nevýhody

Edge computing je koncept, který se zaměřuje na výpočetní funkce na okraji sítě, tj. přímo v místech, kde se data vytvářejí. Cílem je zpracovat data co nejblíže k zdroji, aby se minimalizovala latence a zvýšila spolehlivost.

Na druhé straně, cloud computing zahrnuje nasazení aplikací a služeb v cloudu, tj. ve vzdálených datových centrech, které jsou často spravovány třetími stranami. Tyto služby mohou být dostupné přes internet a mohou být snadno škálovány podle potřeby zákazníka.

Výhody edge computing zahrnují nižší latenci, vyšší spolehlivost, lepší bezpečnost a větší soukromí dat. Nevýhodou je vyšší náklady na hardware a složitost nasazení.

Výhody cloud computingu zahrnují nižší náklady na IT, snadnou dostupnost a škálovatelnost služeb, možnost úspory nákladů na hardware a snadnou správu a údržbu. Nevýhodami jsou závislost na kvalitě připojení k internetu, možnost omezení soukromí a bezpečnosti dat.

Edge computing je žádoucí pro výrobní průmyslové podniky, kde výpadek produkce může znamenat obrovskou ztrátu zisku a jsou zde proto kladeny velké nároky na spolehlivost, bezpečnost a robustnost a náklady na provoz edge systému jsou jen zlomek ostatních nákladů. [27]

2.6 Siemens Industrial Edge

Industrial Edge je řešení pro Edge Computing nabízené firmou Siemens.

Industrial Edge Hub (IEH) je výchozí bod pro získání a konfiguraci Industrial Edge Managmentu (IEM). Slouží také k nákupu a stažení Industrial Edge Aplikací a obsahuje důležité dokumenty s informacemi o použití Industrial Edge produktů.

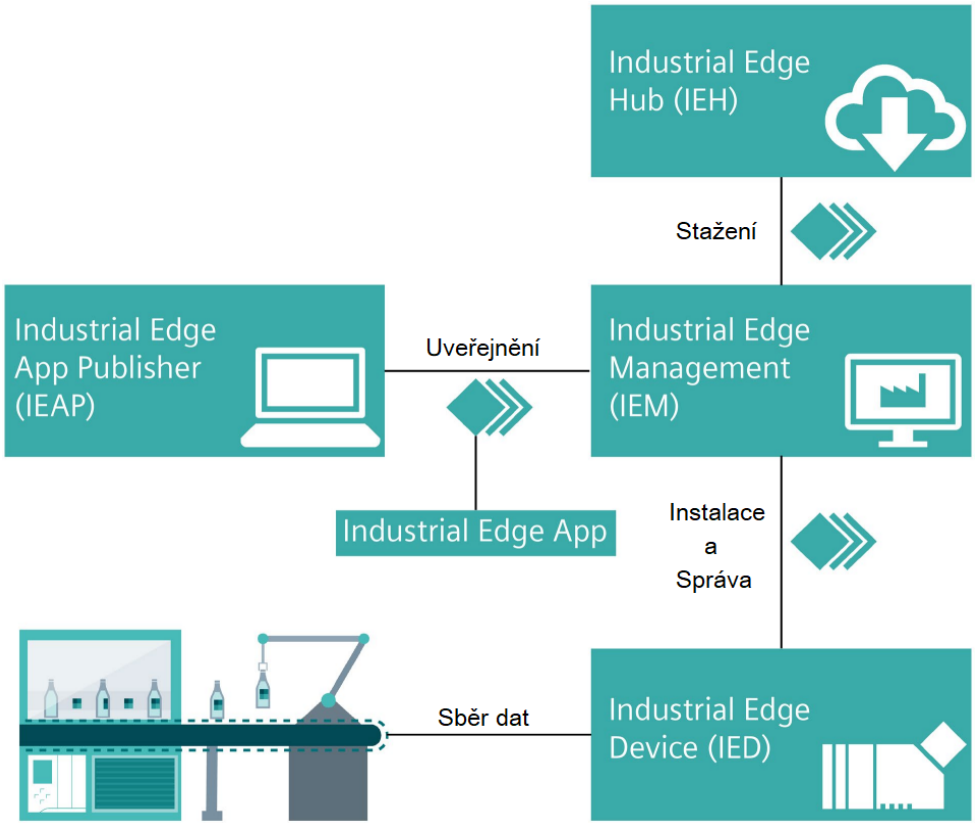
IEM je centrální bod Industrial Edge. Stará se o správu a řízení Industrial Edge Deviců (IED) a Industrial Edge Aplikací (IEA). Umožňuje vytvářet nové IED, instalovat na ně různé IEA, monitorovat jejich stav etc.

IED je zařízení, které stará o získávání, ukládání a zpracování dat z produkce a o její řízení, což dělá za pomocí vybraných IEA, které se zde spouští.

Industrial Edge App Publisher (IEAP) je aplikace, která usnadňuje publikaci vlastních aplikací do IEM.

IEA jsou aplikace spouštěné v IED a implementují potřebné úkoly a logiku, kterou má IED vykonávat. Jsou spouštěny ve formě Docker kontejnerů, což umožňuje snadné vytváření vlastních aplikací. Aplikace nabízené firmou Siemens které použijeme jsou SIMATIC S7 Connector, Databus a Data Service. [28]

Komunikace mezi aplikacemi uvnitř IED je převážně implementována pomocí MQTT protokolu, který je založen na modelu publish/subscribe. To znamená, že jednotlivé aplikace spolu nekomunikují přímo, ale přes prostředníka, běžně zvaného message broker. Uvnitř brokeru se nadefinují možná témata zpráv. Ostatní aplikace se na něj napojí a mohou poté publikovat zprávy



Obrázek 2.3 Schéma Industrial Edge

na nějaké téma, nebo nějaké téma odebírat. Message broker poté zajistí, aby každá aplikace, která odebírá nějaké téma, zprávu na dané téma obdržela. Databus je message broker pro IED.

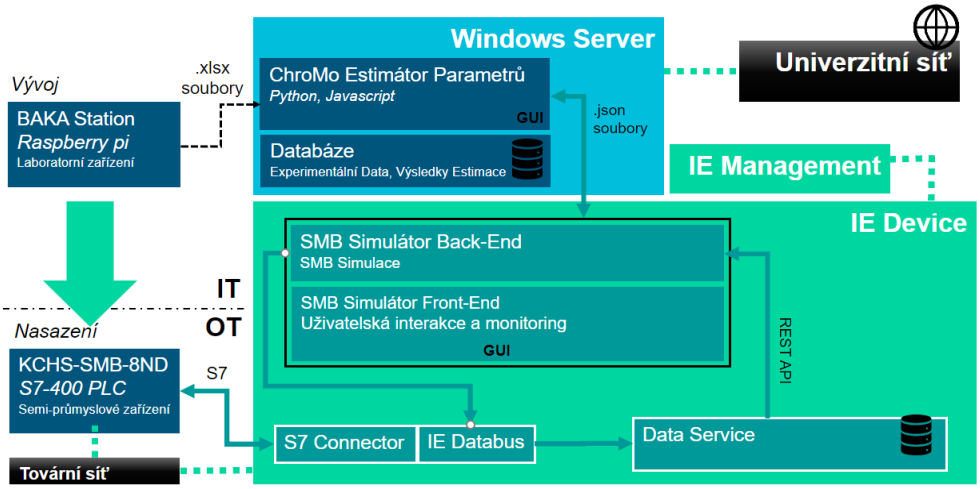
SIMATIC S7 Connector zajišťuje komunikaci se SIMATIC PLC, což je řada programovatelných logických automatů společnosti Siemens, které zajišťují čtení dat ze senzorů průmyslových zařízení. Přečtená data posílá s definovaným tématem na Databus a také poslouchá na definovaná témata, která umožňují na PLC zapisovat.

SIMATIC S7 Connector využívá S7 protokol, který byl vytvořen společností Siemens a je především používán pro komunikaci s PLC. Je založen na blokově orientovaném ISO přenosu. Využívá ISO-COTP (Connection Oriented Transport Protocol), který narozdíl od běžného TCP implementuje packetově orientovanou komunikaci, a TPKT (Transport Packet), který emuluje packetově orientované komunikace přes běžně používané TCP. [29]

Data Servis je aplikace, která umožňuje ukládání a agregaci dat přes libovolné časové úseky. Data Service se napojí na Databus a odebírá data různých konektorů, pro nás S7 konektoru, které pak ukládá. Pro získání dat vytváří REST rozhraní.

2.7 Návrh výsledné aplikace

Výsledný projekt se bude skládat ze dvou částí, IED a webového serveru. Pro IED vytvoříme aplikaci SMB simulátoru a použijeme několik dalších komponent vytvořených pro Industial Edge, jako je S7 konektor pro komunikaci s PLC zařízením měřící průmysloví proces, Databus pro zprostředkování komunikace mezi komponentami, Data Service pro ukládání naměřených dat. Na webovém serveru běží Parametr estimátor, protože vyžaduje větší množství výpočetní síly a není proto vhodné aby běžel v IED, a MongoDB databáze, kterou využívá Parametr estimátor pro persistenci dat.



■ **Obrázek 2.4** Schéma projektu

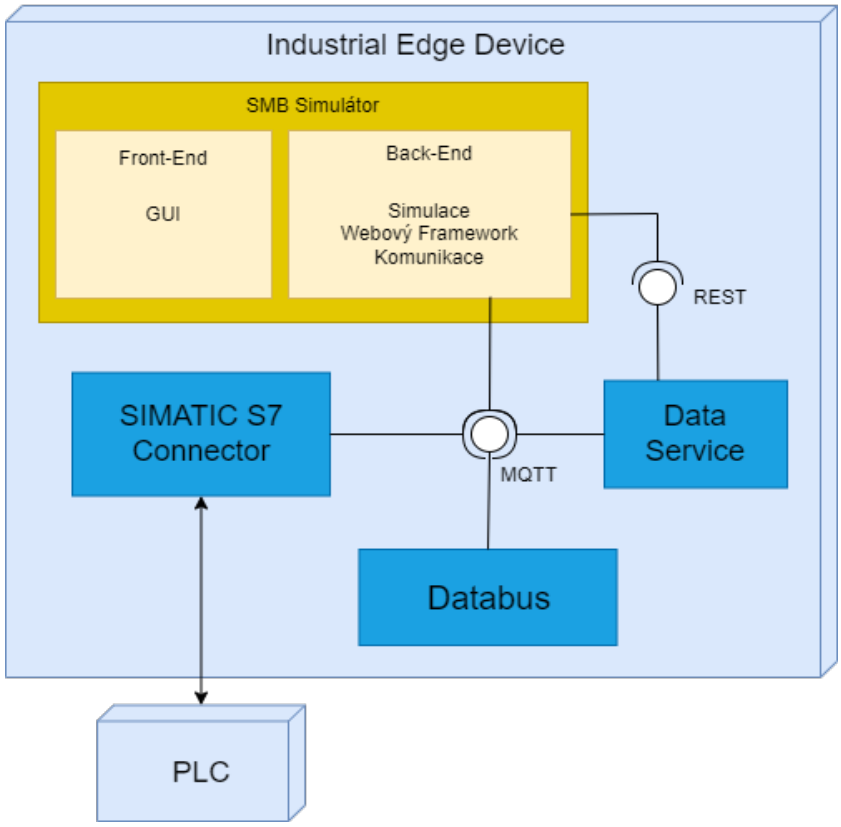
2.7.1 Industrial Edge Device

Pro prostředí IED bude vytvořena aplikaci pro simulaci SMB chromatografie. Aplikace v IED jsou spouštěny v Docker kontejnerech, proto i pro naši aplikaci vytvoříme Docker image a spustíme ji v kontejneru. Bude obsahovat implementovanou simulaci, webový framework pro zprostředkování uživatelského rozhraní v podobě webových stránek a rozhraní pro komunikaci s ostatními aplikacemi v IED.

SIMATIC S7 Connector bude zajišťovat komunikaci s PLC, které řídí reální proces. Data Service bude číst a ukládat naměřená data a vytváří REST rozhraní, ze kterého lze uložená data získávat. Databus zajišťuje MQTT komunikaci mezi aplikacemi. SMB simulátor bude číst data z REST rozhraní a bude umět publikovat zprávy pro SIMATIC S7 Connector, který je zapíše do PLC.

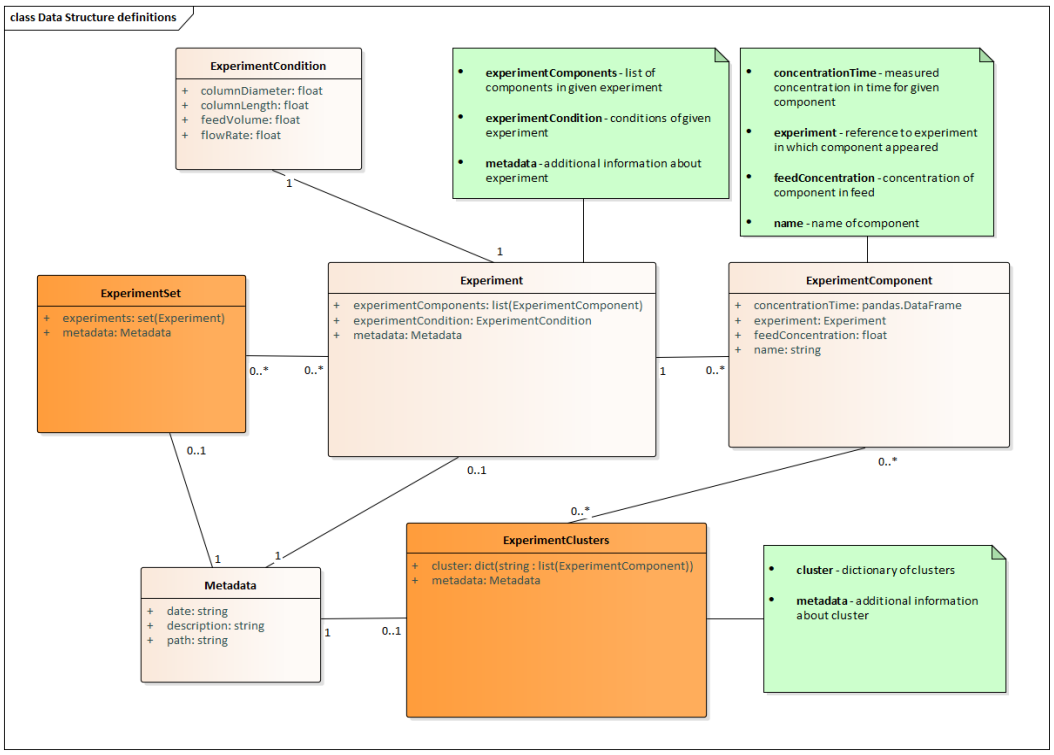
2.7.2 Estimátor parametrů

Průběh estimace parametrů se bude skládat z fáze preprocessingu, kde upravíme vstupní data tak, aby se eliminovala chyba v měření, a fáze optimalizace, kde budeme optimalizací hledat nejlepší parametry. Protože optimalizace se na základě vstupních dat a počátečního odhadu může volat mnohokrát a v každém kroku optimalizace se bude volat modelový řešič a porovnávat se modelová data s daty reálnými, bude potřeba větší množství výpočetního výkonu. Aplikace proto poběží na separovaném webovém serveru a bude možné data z nahraných experimentů a výsledné estimace přenést do SMB simulátoru.



Obrázek 2.5 Schéma Industrial Edge Device

Protože zpracováváme velké množství dat s velkým množstvím parametrů, vytvoříme datovou strukturu,kterou použijeme napříč celým programem a která nám ulehčí orientaci. Koncept datové struktury je popsán na diagramu 2.6.



Obrázek 2.6 Koncept datové struktury estimátoru parametrů

ExperimentSet bude množina experimentů, což bude naše hlavní pracovní sada, na které budeme provádět jednotlivé operace.

Experiment reprezentuje jeden experiment, který obsahuje objekt s parametry, za kterých byl experiment proveden *ExperimentCondition* a množinu složek reprezentující separovanou směs.

ExperimentComponent je složka, která obsahuje parametry týkající se dané složky a časovou posloupnost koncentrací, která pro ni byla v experimentu naměřena.

ExperimentCluster je pomocná struktura, do které budeme ukládat komponenty do množin na základě nějaké podobnosti. Tuto podobnost lze definovat libovolně, například stejné jméno, podobné podmínky experimentu etc.

Metadata jsou třída objektů, do kterých budeme ukládat data o objektu, ke kterému patří, například popis a datum experimentu nebo cesta k souboru.

Optimalizační část bude nejvíce výpočetně náročná a budeme proto chtít implementovat několik optimalizací, které povedou k přijatelné výpočetní době. Většina optimalizačních algoritmů umožňuje (některé dokonce vyžadují) zadání počátečního odhadu a mezí, které mají velký vliv na průběh optimalizace. Budeme proto chtít, aby uživatel tyto hodnoty zadal a dáme mu nástroje, které mu pomůžou k odhadu co nejlepších možných hodnot.

Druhou metodou, kterou se pokusíme snížit čas optimalizace, je rozdělení optimalizace do dvou úrovní. V první úrovni se bude optimalizovat pouze pro porozitu, což je parametr, který má každý model chromatografie, který implementujeme, a který je vlastností výplně kolony a je proto pro všechny složky stejný. V každém kroku optimalizace první úrovně se poté volá optimalizace druhé úrovně pro všechny zbylé parametry pro každou složku. Cílem tohoto přístupu je použití výsledků z optimalizace druhé úrovně jako odhad pro jejich další volání. Předpokládáme, že při

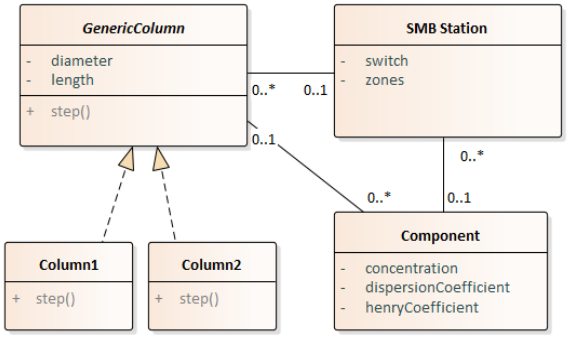
malé změně porozity se ostatní parametry také příliš měnit nebudou a výsledek předchozího běhu bude dobrým odhadem, což jak už jsme zmínili má velký vliv na průběh optimalizace.

I přesto optimalizace budou mít mnoho iterací a opakované kalkulování modelových křivek, porovnání s reálnými daty a počítání loss funkcí v každé iteraci jsou výpočetně náročné. Webová aplikace však musí umět obstarávat několik uživatelů najednou, budeme proto výpočetně náročné úlohy spouštět za pomocí vláken. Díky tomu hlavní vlákno bude moci stále obstarávat uživatele a lépe se tak využijí procesní zdroje.

Pro persistenci jsme se rozhodli použít databázi MongoDB, což je dokumentová databáze, která data ukládá jako dokumenty ve formě JSON (konkrétně její bitové formě BSON), protože aplikaci implementuje v pythonu a pracujeme převážně s datovými strukturami jako jsou listy a slovníky, které umožňují snadnou konverzi do formátu JSON. Také budeme ukládat pouze data o uživatelích, nahrané experimenty a výsledky optimalizace, kde vztahy mezi entitami jsou vcelku jednoznačné a nepotřebujeme vytvářet komplikované relace, pro které jsou vhodnější relační databáze.

2.7.3 SMB simulátor

Pro SMB simulátor je nutné, aby si uživatel mohl implementovat libovolnou SMB stanici a libovolný roztok, pro které se bude provádět simulace. Také aby mohl některé parametry měnit. Základní myšlenka, jak jej budeme implementovat je zachycena v diagramu 2.7.



Obrázek 2.7 Koncept SMB simulátoru

Hlavní stavební jednotkou budou třídy reprezentující chromatografickou kolonu, které budou obsahovat veškerou modelovou logiku pro simulaci. Budou obsahovat vlastnosti kolon jako délka a průměr, a budou si pamatovat svůj stav. Metoda *step()* na základě vstupních koncentrací spočítá jeden časový krok v simulaci a vrátí stav kolony. Protože chceme implementovat více modelů pro simulaci chromatografie, vytvoříme abstraktní třídu *GenericColumn*, pro kterou poté můžeme vytvářet více implementací.

Třída SMB Station bude obsahovat parametry a logiku týkající se SMB stanice. Bude obsahovat kolony a bude si pamatovat, ve které zóně je která kolona. Pro každý časový krok třída zavolá *step()* na každé koloně tak, že výstup jedné kolony bude vstupem do další kolony v pořadí a bude obsahovat logiku pro rotaci ventilů.

Obě třídy budou obsahovat množinu komponent, které reprezentují směs kterou separujeme. Tato třída nebude obsahovat žádnou modelovou logiku, pouze parametry vztahující se k jednotlivým složkám.

Simulaci bude možno spouštět ve dvou režimech, offline a online. Offline režim bude jednorázová simulace, pro kterou si uživatel definuje stanici, roztok, operační parametry a zadá čas, jak dlouho chce, aby simulace běžela. Po proběhnutí simulace se uživateli zobrazí grafy průběhu

koncentrace na výstupech, čistota a výtěžek jednotlivých složek a stav koncentrace v jednotlivých kolonách.

MPC bude implementován v online režimu simulace. Aplikace se dotáže na Data Service přes REST rozhraní a získá seznam parametrů. Uživatel poté kromě definice stanice a roztoku bude muset namapovat tyto parametry na operační parametry, které si aplikace poté přečte a použije pro simulaci.

Na stránce simulace bude možno pozorovat průběh simulace spolu s reálným měřením, které budou spolu časově synchronizované. Zároveň budou zobrazeny čistoty a výtěžky pro aktuální simulaci, budoucí horizont simulace a reálného měření, které budou periodicky aktualizovány. Bude také možné změnit operační parametry, které se nejprve projeví v budoucím horizontu. Bude potom možné se rozhodnout jestli změnu zachovat nebo vrátit, pokud se změna zachová, bude poslána zpráva do PLC, která změní parametry pro reálný proces.

2.7.4 Uživatelské a webové rozhraní

Pro vytvoření webového rozhraní u obou aplikací použijeme webový framework Flask, který poskytuje základní nástroje pro tvorbu webových aplikací jako správu routování, vytváření šablon, správa cookies etc. Zvolili jsme jej pro jeho jednoduchost a flexibilitu. Šablony webových stránek vytvoříme za pomoci šablonových funkcí a standardního HTML, CSS a Javascriptu.

Pro implementaci uživatelských účtů použijeme modul Flask_Login. Ten implementuje vytváření webových relací uložených v cookies na straně klienta a umožňuje přihlašování, odhlašování a omezení přístupu. Zároveň nevyžaduje žádné specifické způsoby přihlašování či ukládání uživatelských informací. Tato volnost nám umožňuje tento modul použít u obou aplikací, i když každá z nich implementuje uživatelské účty trochu jinak.

U estimátoru parametrů použijeme „standardní“ přístup k uživatelským účtům. Uživatel si může účet vytvořit a v něm si nahrávat experimenty a spouštět optimalizace. Každý uživatel bude mít svoje oddělené pracovní prostředí.

SMB simulátor bude mít pouze jeden administrátorský účet. Přihlášení k tomuto účtu bude potřeba pro vytvoření a spuštění simulace a bude třeba k řízení simulace. Uživatelé bez účtu budou moci tuto simulaci monitorovat, ale nebudou moci nic měnit.

2.8 Existující nástroje

Jediný veřejně dostupný software specializovaný na simulaci preparativní chromatografije Aspen Chromatography™. Jedná se o komerční software, který umožňuje provádět dynamické simulace a optimalizace kontinuální i diskontinuální chromatografie. Nabízí výběr modelu, diskretizační metody a způsobu řešení diferenciálních rovnic.

Nevýhodou je cena licencí znemožňující spolupráci, pokud druhá strana licence nevlastní. Další nevýhodou je nepřístupný zdrojový kód, který znemožňuje hlubší analýzu použitých algoritmů a vytváří „black box“ efekt.

Zásadní nevýhodou jsou nedostatky při odhadu parametrů a absence možnosti opravy experimentálních dat. Aspen sice umožňuje odhad parametrů z množiny experimentů, neumožňuje však definici rozdílných podmínek pro jednotlivé experimenty, jako jsou průtoky a koncentrace nástřiku. Zároveň nenabízí žádné možnosti pro předzpracování a opravu dat.

Kapitola 3
Realizace

3.1 Estimátor parametrů

Největší částí celého projektu je estimátor parametrů. Jedná se o aplikaci, která dokáže zpracovat data z diskontinuálních experimentů preparativní chromatografie a odhadnout nejlepší možné parametry EDM modelu. Proces estimace se skládá ze dvou částí, zpracování a preprocessing dat a optimalizace parametrů. Program také obsahuje pomocné funkce, které umožní uživateli lépe odhadnout počáteční podmínky a omezit tak dobu optimalizačního algoritmu.

Uživatelské rozhraní je implementováno ve formě webových stránek, které umožňují uživateli nahrát experimentální data, zadat potřebné parametry ve formě webových formulářů a získat výsledky. Uživatelská data, nahraná experimentální data a výsledky jsou ukládány do dokumentové databáze MongoDB.

Pro každé spuštění pomocné funkce a hlavní optimalizace se pro danou úlohu vytvoří vlákno, které celý proces obstará, včetně preprocessingu a vytvoření grafů. Díky tomu se zachová nepřerušený běh hlavního vlákna, které zabezpečuje obsluhu uživatelů.

3.1.1 Zpracování a preprocessing dat

Program přijímá data ve formátu xslx pro Microsoft Excel, což je běžný způsob ukládání dat naměřený při chromatografickém experimentu. Každý excel soubor reprezentuje jeden experiment a obsahuje experimentální parametry, jako rozměry kolony, objem a koncentrace jednotlivých složek nástřiku a průtok, a samotná data, což jsou koncentrace jednotlivých separovaných složek ve frakcích odebraných na výstupu z kolony ve známém čase 3.1.

Tato data se zpracují do datového formátu popsaného v sekci o návrhu této aplikace. Preprocessing dále postupuje podle diagramu 3.2.

První krok „Fit Gauss“ nahradí experimentální data asymetrickou Gaussovou křivkou, která nejlépe odpovídá naměřeným datům. Tím se data „vyhladí“ a redukují se anomálie při měření a lidská chyba. Asymetrická Gausova křivka je dána následující funkcí:

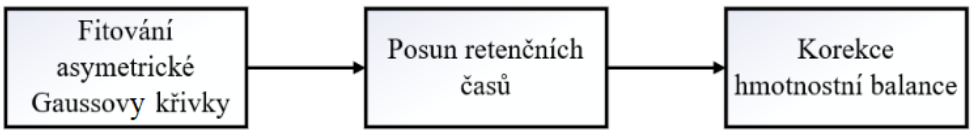
$$c_{i,j}(t) = \frac{A}{C\sqrt{2\pi}} * e^{\left(-\frac{(t-B)^2}{2C^2}\right)} * \left(1 + erf\left(\frac{D(t-B)}{C\sqrt{2}}\right)\right)$$

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} dy,\ y \in \langle 0, x \rangle$$

A, B, C a D jsou parametry získané regresní analýzou metodou nejmenších čtverců. Implementace také obsahuje kód, který škáluje velikost vstupních dat založených na maximální hodnotě koncentrace.

	A	B	C	D	E
1	Experiment conditions			Experiment description	
2	Column lenght	235	mm	Separace směsi Sac, Glc, Fru	
3	Column diameter	16	mm	Date and Time	
4	Flowrate	150	mL/h	10/27/2021	
5	Feed volume	0,85	mL		
6	deadVolume	0,75	mL		
7	Feed conc	Sac	Glc	Fru	ManOH
8		9,09	18,78	20,48	24,38
9	Time	**Sac**	**Glc**	**Fru**	**ManOH**
10	0	0	0	0	0
11	2	0	0	0	0
12	4	0	0	0	0,0017
13	6	0	0	0	0,003
14	8	0,0026	0	0	0,0602
15	10	0,0098	0	0	0,3863
16	12	0,0273	0	0	0,7898
17	14	0,0555	0	0	1
18	16	0,0861	0	0	1
19	18	0,1098	0	0	2
20	20	0,137	0	0	2

Obrázek 3.1 ukázka souboru experimentu



Obrázek 3.2 Postup předzpracování

Výpis kódu 3.1 Použití ztrátové funkce v „Retention Time Correction“

```
def Shift_Loss_Function(shifts, avgPeakTimes, cluster):
  sum = 0
  for idx, exp in enumerate(cluster):
    for comp in exp.experimentComponents:
      column = pd.to_numeric(comp.concentrationTime[comp.name])
      peakIndex = column.idxmax()
      peakTime = comp.concentrationTime.iloc[peakIndex, 0]
      sum += abs(peakTime + shifts[idx] - avgPeakTimes[comp.name])
  return sum
...
initalGuess = list()
bounds = list()
for exp in value[0]:
  initalGuess.append(0)
  bounds.append((exp.maxShift, None))
res = minimize(Shift_Loss_Function,
               initalGuess,
               args=(avgPeakTimes,
               value[0]),bounds=bounds,
               method='Nelder-Mead')
```

Druhý krok „Retention Time Correction“ posune retenční časy (vrcholy píku) všech složek tak, aby byl jejich rozdíl mezi experimenty se stejnými podmínkami co nejmenší, čímž se eliminují chyby v měření času při startu experimentu a během odebírání frakcí. Pro tuto operaci nejprve vytvoříme clustery experimentů, které mají stejné podmínky. Pro každou složku v clusteru pak spočítáme průměrný čas píku.

Dále definujeme funkci pro optimalizační algoritmus, která má jako argument časový posun každého experimentu v clusteru a vrací sumu rozdílů píků od průměru. Časový posun je definován pro experiment, což znamená, že všechny složky v jednom experimentu se musí posunou o stejný čas. Posun je omezen tak, aby koncentrace nemohly být posunuty do záporných čísel. Nalezneme první hodnotu, která přesahuje „nulový práh“ a s jeho pomocí definujeme maximální posun záporným směrem. Nulový práh je definován uživatelem.

Nakonec pro tuto funkci spustíme optimalizační algoritmus, který pro každý experiment najde nejlepší časový posun. Pro optimalizaci jsme použili Nelder-Meadův algoritmus implementovaný v knihovně *scipy* 3.1.

Formulace optimalizace, kde σ_j^r je posun experimentu j, $T_{j,i}^p$ je čas píku komponenty i v experimentu j a $\bar{T}_i^p$ je průměr časů píků složky i ze všech experimentů, je následující:

$$min\ \ \xi(\sigma_1^r, \sigma_2^r, \dots \sigma_n^r) = \sum_{x=1}^{n} \sum_{i=1}^{m} \left| T_{x,i}^p(\sigma_x^r) - \bar{T}_i^p \right|;$$

$$i = 1, 2, \dots m,\ x = 1, 2, \dots n$$

$$c_{i,x,k} > c_{tol} \rightarrow t_{i,x,k} > 0;\ k = 1, 2, \dots K$$

Poslední krok ”Mass Balance Correction”upraví čas nástřiku tak, aby hmotnost na vstupu odpovídala hmotnosti na výstupu a nebyly porušeny zákony zachování hmoty. Pro tuto operaci byla opět nadefinována funkce, která je použita pro optimalizační algoritmus. Tato funkce bude mít jako argument čas nástřiku $\tau_{i,j,feed}$, pomocí kterého spočítá teoretickou hmotnost dané složky v nástřiku 3.2.

$$m_{i,x,feed} = c_{i,x,feed} * \tau_{i,x,feed} * u_m * \frac{\pi d^2 \varepsilon}{4}$$

Výpis kódu 3.2 Použití ztrátové funkce v „Mass Balance Correction“

```
def Loss_Func(feedTime):
outputSum = 0.0
for comp2, comp3 in
    zip(exp2.experimentComponents,exp3.experimentComponents):
  df2 = comp2.concentrationTime
  peakVal = float(df2[comp2.name].loc[df2[comp2.name].idxmax()])
  # Skip the component if the final concentrations are greater
  # than 1/10 of the maximum concentration, indicating incomplete
  # reaction or adsorption
  if float(df2[comp2.name].iat[-1]) > peakVal / 10 or
     float(df2[comp2.name].iat[-2]) > peakVal / 10 or
     float(df2[comp2.name].iat[-3]) > peakVal / 10:
    continue
  trapzRes = np.trapz(x=df2.iloc[:, 0].to_numpy(),
                      y=df2.iloc[:, 1].to_numpy())
  comp_output_mass = trapzRes *
                     exp2.experimentCondition.flowRate /
                     3600  # [mg]
  comp_feed_mass = feedTime *
                   comp2.feedConcentration *
                   exp2.experimentCondition.flowRate
                   # feedTime in [h], result in [mg]
  outputSum += abs(comp_output_mass - comp_feed_mass)
return outputSum

newFeedTime = scipy.optimize.minimize_scalar(
                  Loss_Func,
                  bounds=(
                      initialFeedTime - (initialFeedTime / 2),
                      initialFeedTime + (initialFeedTime / 2)),
                  method='bounded')
```

Kde $c_{i,x,feed}$ je koncentrace komponenty i v nástřiku, u_m je objemový průtok kolony a d je průměr kolony. Poté získáme celkovou hmotnost na výstupu z kolony.

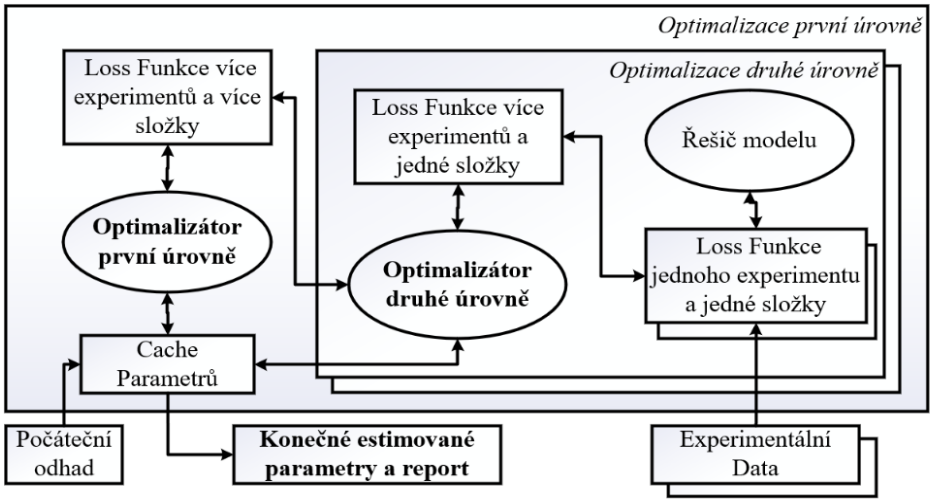
$$m_{i,x,L} = u_m * \frac{\pi d^2 \epsilon}{4} \int_0^{\tau_x} c_{i,x,feed} dt$$

Kde τ_x je čas experimentu. Rozdíl těchto dvou hodnot minimalizujeme. Na základě optimalizovaného času nástřiku poté měníme i objem nástřiku. Protože minimalizujeme pro pouze jeden parametr, použijeme Brentovu metodu. Hranice jsme arbitrárně definovali jako ± 12 času nástřiku. Předpokládáme, že kdyby optimalizace nalezla výsledek mimo tyto hranice, jednalo by se o příliš velkou opravu a můžeme usoudit, že experimentální data jsou poškozené a nepoužitelné.

Každý z těchto kroků je volitelný, nicméně výsledky optimalizace mohou být výrazně ovlivněny nepřesnostmi a chybami v měření a je proto vhodné je co nejvíce eliminovat.

3.1.2 Estimace modelových parametrů

Předzpracovaná data poté vstupují do optimalizátoru, ve kterém se hledají optimální modelové parametry. Optimalizace probíhají na dvou vnořených úrovních. Na první úrovni se optimalizuje porozita, což je vlastnost sorbentu a je stejná pro všechny složky, zbylé parametry se optimalizují na druhé úrovni pro každou složku zvlášť a liší se pro různé modely.



Obrázek 3.3 Diagram architektury algoritmu pro dvouúrovňovou optimalizaci

Rozdělení do dvou úrovní je založeno na předpokladu, že při malé změně porozity se optimalizované parametry druhé úrovně nebudou příliš lišit a můžeme proto výsledky předchozího běhu druhé úrovně použít jako dobrý počáteční odhad, což značně urychlí další běh.

Proces estimace parametrů popíšeme na obrázku 3.3. Do optimalizace vchází počáteční odhad, který uložíme do cache parametrů, která je globální a je přístupná ze všech funkcí na všech úrovních. Použité optimalizační funkce vyžadují, aby výstupem z optimalizované funkce byla skalární hodnota. Globální cache parametrů nám umožňuje ukládat různé parametry ve všech úrovních optimalizace bez nutnosti předávání těchto parametrů jako návratové hodnoty.

Poté se je volána optimalizace první úrovně s úvodním odhadem porozity, která minimalizuje výstup z loss funkce první úrovně. Ta volá optimalizaci druhé úrovně pro jednotlivé složky a jejím výstupem je suma jejich výstupů. Optimalizace druhé úrovně minimalizuje hodnotu loss funkce druhé úrovně 3.3. Poté co nalezne minimum zapíše nalezené parametry do cache parametrů a vrátí minimální hodnotu. Loss funkce druhé úrovně volá loss funkci jedné složky pro jeden experiment a vrací jejich sumu.

Loss funkce jedné složky pro jeden experiment volá solver funkci, která zajišťuje výpočet modelů pro zadané parametry. Výslednou modelovou křivku poté porovná s experimentálními daty a vrátí jejich rozdíl, který se spočítá z rozdílu jednotlivých datových bodů.

Aby bylo možné porovnávat body ve stejném časovém okamžiku, je použita interpolace *scipy.interpolate.interp1d*, která umožňuje i exptrapolaci v případech, ve kterých se neshodují délky experimentálních a modelových dat 3.4.

Byly implementovány čtyři způsoby, kterými lze počítat rozdíl datových bodů. „Simple“ počítá absolutní hodnotu rozdílu, „Squares“ počítá druhé mocniny rozdílu, což odpovídá metodě nejmenších čtverců, „LogSimple“ počítá přirozený logaritmus absolutních hodnot rozdílu, a „LogSquares“ počítá přirozený logaritmus druhých mocnin rozdílu. Metoda nejmenších čtverců je používána nejčastěji.

Poté co proběhne optimalizace první úrovně, zapíše se výsledná porosita do cache parametrů, ve které jsou nyní veškeré výsledky, ze kterých se vytvoří objekt výsledku optimalizace.

Pro každou úroveň si uživatel mohl vybrat jednu ze čtyř různých optimalizačních algoritmů z knihovny *scipy.optimize*, které jsou brute-force, Nelder-Mead, Powell a SHGO. Také si může změnit některá nastavení pro tyto funkce, jako maximální počet iterací, počet bodů diskretizace stavového prostoru etc.

Při použití Powellova algoritmu bylo zjištěno, že algoritmus může ve svých iteracích použít

Výpis kódu 3.3 Ukázka zanoření funkcí při dvouúrovňové optimalizaci

```
def Lev1_Optim( ... ):
  res = handle_Optim_Settings(Lev1_Loss_Function, ... )
  return res.fun

def Lev1_Loss_Function( ... ):
  sum = 0
  for key in experimentClustersComp.clusters:
    res = Lev2_Optim(experimentClustersComp.clusters[key], ... )
    sum += res
  return sum

def Lev2_Optim( ... ):
  res = handle_Optim_Settings(Lev2_Loss_Function, ... )
  return res.fun

def Lev2_Loss_Function( ... ):
  sum = 0
  for comp in experimentCluster:
    res = Single_Loss_Function_Choice(comp, ... )
    sum += res
  return sum
```

Výpis kódu 3.4 Vytvoření modelové křivky pro porovnání s reálnými daty

```
realCurve = experimentComp.concentrationTime
modelCurve = Solver_Choice(solver, params, ...)[:, -1]
modelCurve = Dead_Volume_Adjustment(modelCurve, ...)
timeVec = np.linspace(0, time, modelCurve.size)
f = interp1d(timeVec, modelCurve, fill_value="extrapolate")
modelCurveInterpolated = f(realCurve.iloc[:, 0].to_numpy())
```

Výpis kódu 3.5 Hlavní výpočet v řešiči EDM s lineární isotermou

```
for i in range(1, Nt):  # Advance in time
  b = B.dot(c[i - 1, :])
  b[0] = b[0] + flowSpeed * feed[i]  # From left boundary
  c[i, :] = linalg.solve_banded((1, 1), A_diag, b)
```

hodnoty mimo zadané hranice. To vedlo k tomu, že algoritmus mohl použít zápornou hodnotu, což vedlo k nesmyslným výsledkům nebo chybě. Protože nebylo nalezeno vhodné řešení, bylo rozhodnuto Powellův algoritmus momentálně nepoužívat.

V našem programu implementujeme rovnovážný disperzní model (EDM) pro lineární a Langmuirovu isotermu. Parametry pro EDM s lineární isotermou jsou Henryho konstanta a disperzní koeficient. Parametry EDM s Langmuirovou isotermou jsou Langmuirova konstanta, disperzní koeficient a saturační koeficient.

Parametr	Značení	Jednotka
Celková porosita stacionární fáze	ε	-
Konstanta lineární isotermy	$K_{H,i}$	-
Konstanta Langmuirovy isotermy	$K_{L,i}$	-
Saturační konstanta stacionární fáze	$q_{m,i}$	$g * L^{-1}$
Disperzní koeficient	$D_{ax,i}$	$mm^2 * s^{-1}$

Jak již bylo zmíněno, pro každý z těchto modelů byla implementována *solver* funkce, která daný model řeší. Pro řešení EDM s lineání isotermou byla použita Crank-Nicolsonova implicitní náhrada, vytvoří se matice A a B a pro řešení soustavy rovnic byla použita knihovnu *scipy.linalg*. Konkrétně byla použita funkce *solve_banded*, která efektivně řeší soustavy s maticemi, které jsou nenulové jen na několika diagonálách 3.5.

Crank-Nicolsonova implicitní náhrada byla použita také pro řešení EDM s Langmuirovou isotermou. Potřebné matice však nemůžou být stejně jednoduše vytvořeny, byla proto použita funkce *scipy.optimize.root*. Pro ni byla definována funkce, která z vektoru koncentrací v prostoru v předchozím a aktuálním časovém kroku vypočítá vektor koncentrací pomocí dříve popsané rovnice. Pro řešení této rovnice byla použita funkce *scipy.optimize.root*, která nalezne hledaný vektor koncentrací v daném čase. Optimalizaci je volána v každém časovém kroku, kde koncentrace v předchozím kroku jsou výsledky optimalizace předchozího kroku. Jako úvodní odhad také použijeme koncentrace předchozího kroku 3.6.

Aby bylo možné porovnávat hodnoty loss funkcí jednotlivých složek, je třeba hodnoty faktorizovat. Způsob faktorizace si volí uživatel z následujících možností:

1. Žádná faktorizace
2. Faktorizace maximální hodnotou koncentrace
3. Faktorizace druhou mocninou maximální hodnoty koncentrace
4. Faktorizace koncentrací v nástřiku
5. Faktorizace druhou mocninou koncentrace v nástřiku
6. Faktorizace hmotností v nástřiku
7. Faktorizace druhou mocninou hmotnosti v nástřiku

Výpis kódu 3.6 Hlavní výpočet v řešiči EDM s Langmuirovou isotermou

```
def function(c1,
             c0,
             feedCur,
             porosity,
             langmuirConst,
             saturCoef,
             disperCoef,
             flowSpeed):
  f = np.zeros(len(c0))  # Preparation of solution vector
  for i in range(0, len(c0)):  # Main loop
    if i == 0:  # Left boundary
      f[0] = ((((c0[1] - c0[0]) / dx) + ((c1[1] - c1[0]) / dx)) / 2) -
             (flowSpeed * (c1[0] - feedCur))
    elif i > 0 and i < Nx - 1:
      denominator0 = \
        ((1 - porosity) * saturCoef * langmuirConst) /
        ((((-langmuirConst * c0[i] + 1) ** 2) * porosity) + 1)
      denominator1 = \
        ((1 - porosity) * saturCoef * langmuirConst) /
        ((((-langmuirConst * c1[i] + 1) ** 2) * porosity) + 1)
      secondDer0 = (c0[i - 1] - 2 * c0[i] + c0[i + 1]) / (dx ** 2)
      secondDer1 = (c1[i - 1] - 2 * c1[i] + c1[i + 1]) / (dx ** 2)
      firstDer0 = (c0[i + 1] - c0[i - 1]) / (dx * 2)
      firstDer1 = (c1[i + 1] - c1[i - 1]) / (dx * 2)
      timeDer = (c1[i] - c0[i]) / dt
      disperElem = ((disperCoef / denominator0 * secondDer0) +
                    (disperCoef / denominator1 * secondDer1)) / 2
      convElem = ((flowSpeed / denominator0 * firstDer0) +
                  (flowSpeed / denominator1 * firstDer1)) / 2
      f[i] = disperElem - convElem - timeDer
    elif i == Nx - 1:  # Right boundary
      f[Nx - 1] = (((c0[Nx - 1] - c0[Nx - 2]) / dx) +
                   ((c1[Nx - 1] - c1[Nx - 2]) / dx)) / 2
    return f
...
for i in range(1, Nt):
  sol = optimize.root(fun=function,
                      x0=c[i - 1, :],
                      method='hybr',
                      args=(c[i - 1, :],
                            feed[i],
                            porosity,
                            langmuirConst,
                            saturCoef,
                            disperCoef,
                            flowSpeed))
  c[i, :] = sol.x
  # Save the L2-norm of the residuals at each time step
  residuals[i] = np.linalg.norm(sol.fun)
```

3.1.3 Manuální estimace a analýza loss funkce

Manuální estimace je pomocná funkce, který umožní uživateli zobrazit si výslednou křivku modelu pro zadané parametry spolu s experimentálními body před preprocessingem a po preprocessingu. Nejprve se zavolá modelový solver na data bez preprocessingu, poté se provede preprocessing a solver se volá znova. To umožní uživateli zhodnotit, jaký preprocessing chce použít. Také to umožní najít parametry tak, aby modelová data zhruba odpovídala reálným datům, a použít je pro úvodní odhad pro následnou analýzu loss funkce, případně i pro hlavní optimalizaci.

Analýza loss funkce je pomocná funkce, která umožňuje uživateli zobrazit graf hodnot Loss funkce pro řadu parametrů. Uživatel si vybere hodnotu porosity a v případě modelu s Langmuirovou isotermou hodnotu saturačního koeficientu. Dále si definuje interval a krok pro Henryho či Langmuirovu konstantu a disperzní koeficient. Funkce poté zobrazí povrchový graf pro všechny kombinace hodnot v zadaných intervalech s daným krokem. Tato funkce umožní uživateli dobře odhadnout intervaly, ve kterých se nachází minimum a omezit tak optimalizaci pro rychlejší běh.

3.1.4 Rozhraní

Webové rozhraní je vytvořené za použití webového frameworku Flask. Uživatelské rozhraní je vytvořeno ve formě webových stránek a umožňuje vytvářet uživatelské účty, přidávat experimentální data a na nich spouštět optimalizace a získávat výsledné estimace. Tyto stránky jsou vytvořeny ve formě šablon, které Flask umí vykreslovat pomocí šablonového enginu Jinja2, a Javascriptu, který dělá stránky dynamické a příjemnější k použití.

Protože umožňujeme optimalizaci pro relativně obecnou množinu experimentů, mohou se různé optimalizace od sebe výrazně lišit, například počtem experimentů, složkami v roztocích, počtem složek etc, čímž se výrazně liší i potřebné parametry pro optimalizaci. Jinja2 umožňuje stánky nadefinovat šablony tak, aby se dynamicky měnily pro různé experimenty.

Komunikaci s MongoDB jsme použili knihovnu *mongoengine*, která implementuje mapování mezi python objektem a mongo dokumentem. Knihovna *mongoengine* je nástavba knihovny *PyMongo*, která implementuje samotnou komunikaci. V databázi vytváříme tři druhy dokumentů 3.4:

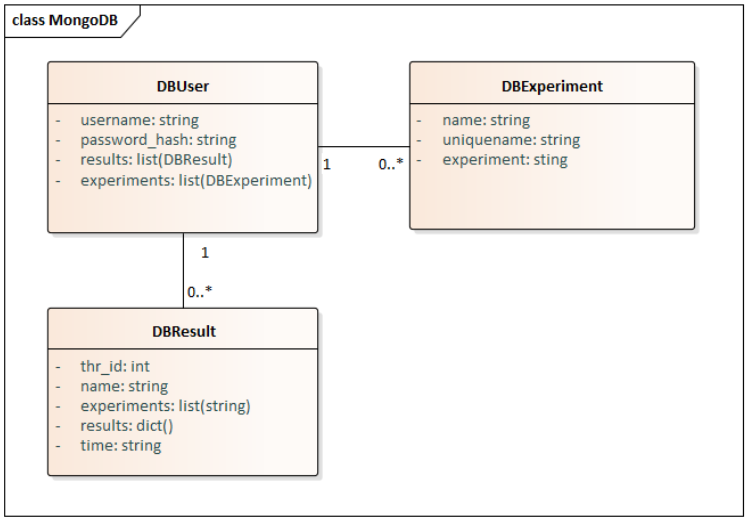
1. DBUser, obsahující přihlašovací údaje a seznam odkazů na experimenty a výsledky optimalizace.
2. DBExperiment, obsahující data o experimentu.
3. DBResult, obsahující data o výsledcích optimalizace.

Aplikace běží na webovém serveru a může k ní proto přistupovat několik uživatelů najednou. Je proto implementováno přihlašování a vytváření účtů za pomoci knihovny *flask_login*. Každý uživatel má svojí množinu experimentů a výsledků optimalizace. Při zadání nějaké výpočetně náročné úlohy se v aplikaci vytvoří dedikované vlákno, díky čemuž je možné obsloužit několik uživatelů najednou bez omezení přístupu k aplikaci. Uživatel také může spustit několik optimalizací najednou, které se po doběhnutí automaticky uloží do databáze a zobrazí v seznamu výsledků.

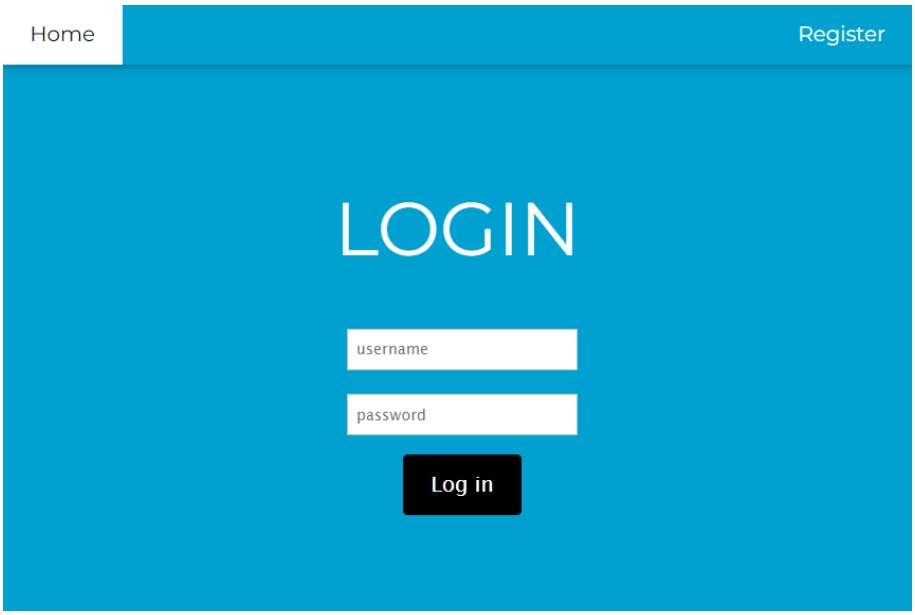
3.1.5 Popis uživatelské interakce

Na úvodní stránce 3.5 je nejprve třeba se přihlásit, případně registrovat účet pokud jej uživatel ještě nemá.

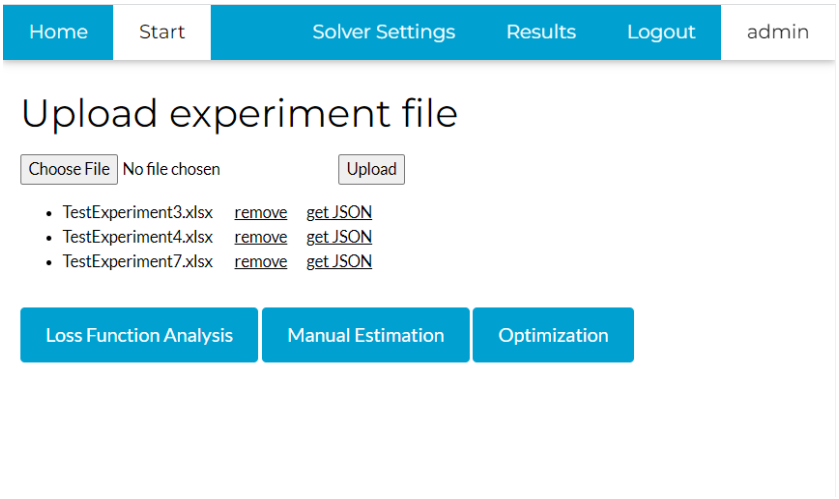
Po přihlášení je uživatel přesměrován na stránku 3.6, na které nahraje experimenty se kterými chce pracovat. Zde je také možné experimenty mazat a získat experiment serializovaný ve formátu JSON, který pak lze použít v SMB Simulátoru.



Obrázek 3.4 Diagram tříd MongoDB



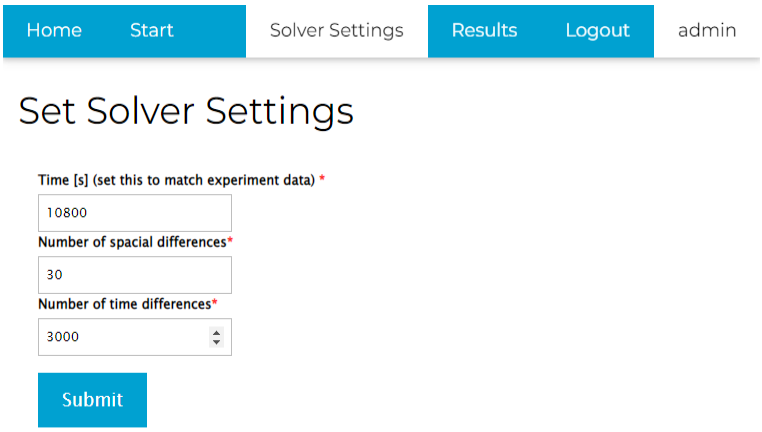
Obrázek 3.5 Parametr Estimator - přihlašovací stránka



Obrázek 3.6 Parametr Estimator - stránka pro nahrání experimentů

Poté co uživatel nahraje požadované experimenty, může pokračovat do Loss Function Analysis nebo Manual Estimation, což jsou dříve zmíněné pomocné funkce, nebo může rovnou pokračovat na samotnou optimalizaci. Také může jít na stránku Solver Settings.

Na stránce Solver Settings je možné nastavit diskretizační parametry a čas pro řešiče modelů. Konkrétně lze nastavit počet prostorových diferencí, počet časových diferencí a čas, po který modelový řešič počítá. Čas je vhodné nastavit tak, aby co nejlépe odpovídal času reálných experimentů, čímž se omezí nutnost extrapomace při porovnávání dat.

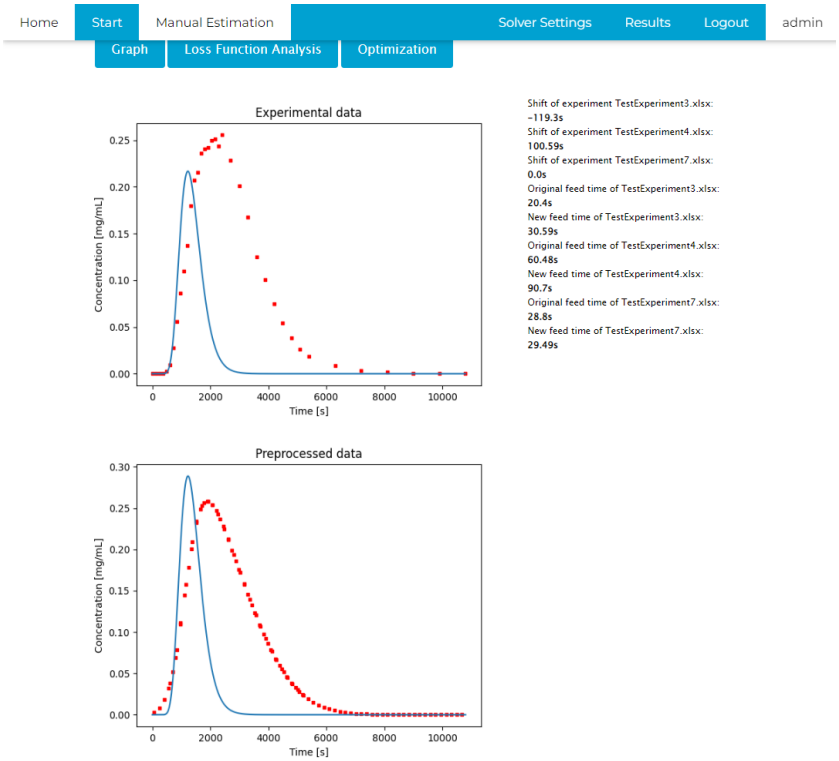


Obrázek 3.7 Parametr Estimator - stránka Solver Settings

Na stránce Manual Estimation uživatel vyplní formulář, ve kterém specifikuje žádaný preprocessing a model, vybere komponentu a zadá modelové parametry. Aplikace poté zobrazí modelovou křivku s experimentálními body před a po preprocessingu a další informace o průběhu výpočtu modelové křivky 3.8. Uživatel pak může pozorovat, jaký vliv má preprocessing a může zkoušet různé parametry a nalézt vhodný počáteční odhad.

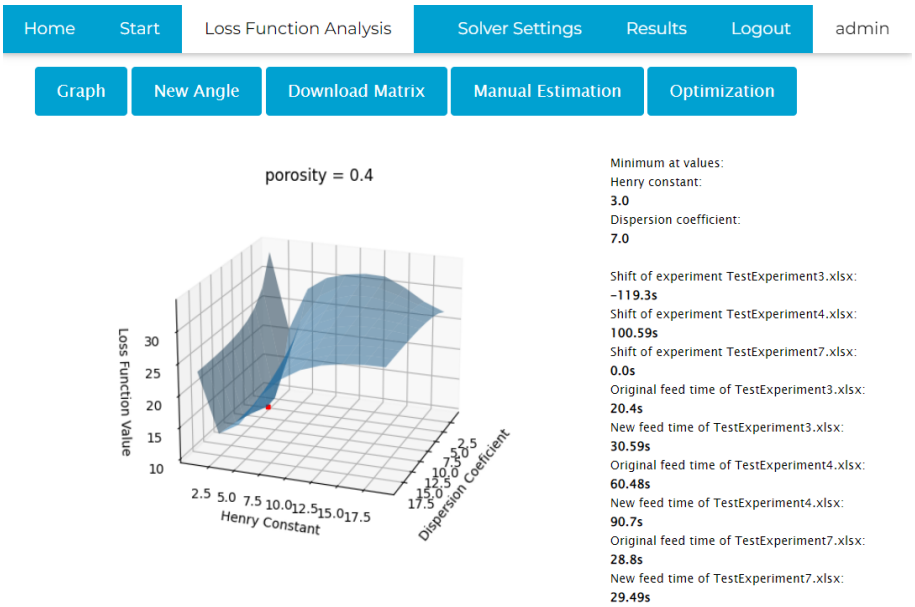
Na stránce Loss Function Analysis opět uživatel vyplní žádaný preprocessing, loss funkci, model a faktor, poté vybere složku, pro kterou chce analýzu provést a specifikuje interval modelových parametrů, ve kterém chce analýzu provést.

Aplikace mu poté ukáže graf hodnot loss funkce v závislosti na modelových parametrech 3.9.



■ **Obrázek 3.8** Parametr Estimator - stránka Manual Estimation

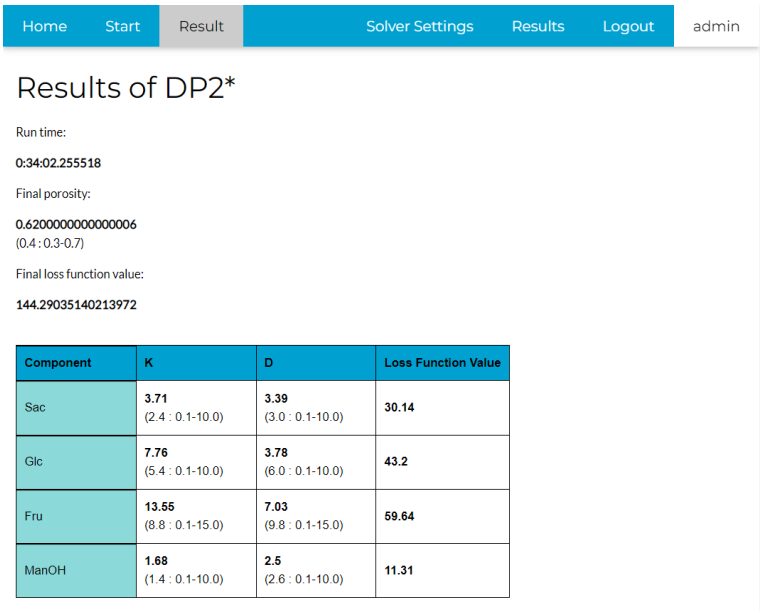
Díky tomu může uživatel odhadnout v jaké oblasti se nachází minimum a použít tuto informaci pro finální optimalizaci. Napravo od grafu se nachází informace o preprocessingu. Uživatel si také může výslednou matici stáhnout pro externí analýzu.



■ **Obrázek 3.9** Parametr Estimator - stránka Loss Function Analysis

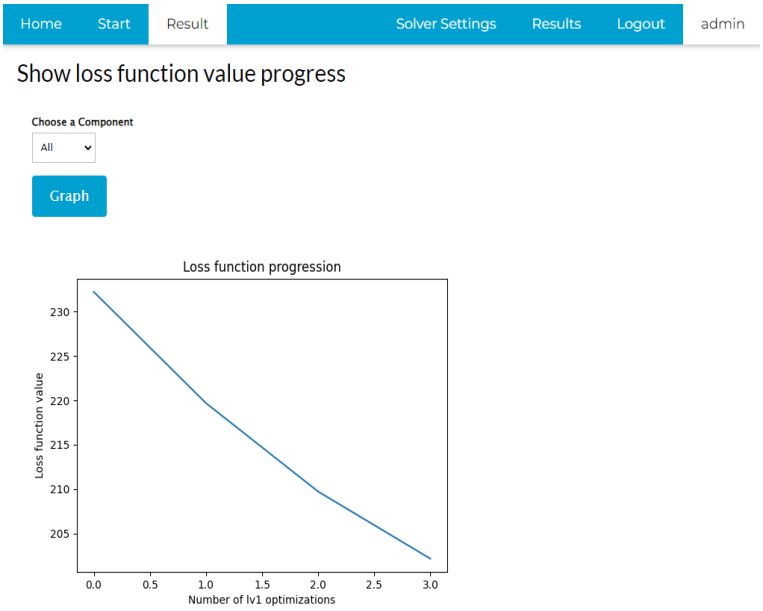
Na stránce Optimization uživatel naposledy vyplní žádaný preprocessing, loss funkci, faktor a model. Poté vyplní hranice a úvodní odhad pro porositu a modelové parametry každé složky. Nakonec si ještě vybere optimalizační algoritmy pro první a druhou úroveň, nastaví jim požadované parametry a spustí optimalizaci.

Jakmile proběhne optimalizace, uživateli se zobrazí stránka s výsledky 3.10, na které jsou výsledné hodnoty optimalizovaných parametrů a výsledné hodnoty loss funkcí. Také se zde nachází zadané parametry optimalizaci, intervaly a úvodní odhady, doba běhu a použité optimalizační algoritmy. Uživatel si také může zobrazit průběh hodnot loss funkce 3.11 a porovnat reálná data s modelovými daty získaná použitím výsledných modelových parametrů.

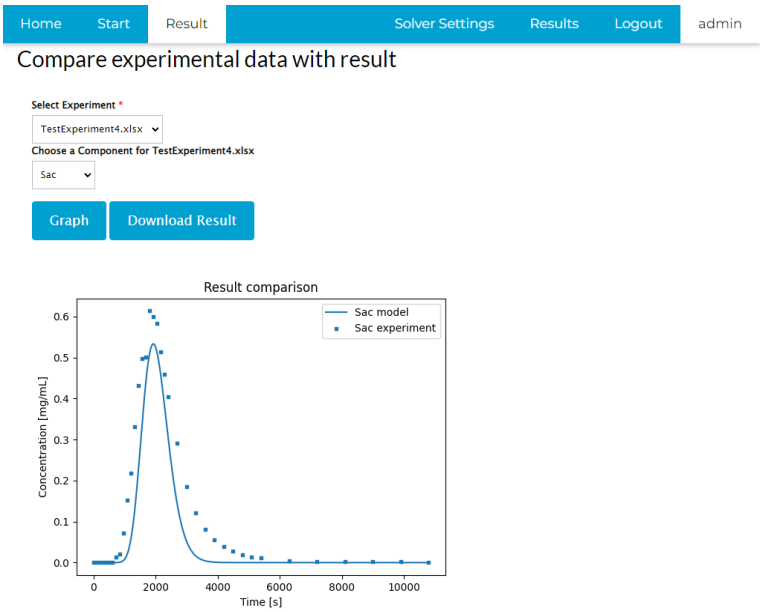


Obrázek 3.10 Parametr Estimator - stránka Result

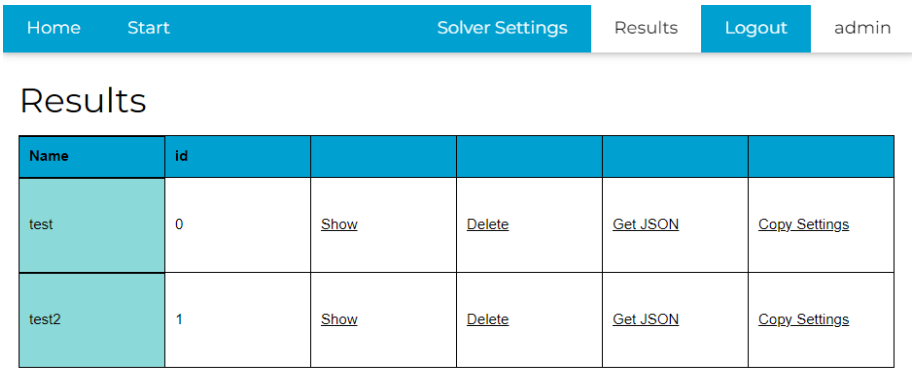
Zobrazit si výsledky předchozích optimalizací je možné na stránce Results 3.13, kde je také možné výsledky mazat, exportovat JSON, který lze opět použít v SMB simulátoru, nebo zkopírovat nastavení, což uživatele přesměruje na stránku optimalizace s předvyplněným formulářem podle předchozího běhu.



Obrázek 3.11 Parametr Estimator - stránka Result - průběh hodnoty loss funkce v optimalizaci



Obrázek 3.12 Parametr Estimator - stránka Result - porovnání modelových dat s reálnými daty



Obrázek 3.13 Parametr Estimator - stránka Results

3.2 SMB simulátor

Druhou částí projektu je simulátor SMB chromatografie. Cílem tohoto programu simulovat chromatografický proces v SMB módu ze zadaných parametrů stanice a roztoku. Pro simulátor je vytvořeno uživatelské rozhraní, které umožňuje uživateli nadefinovat libovolnou SMB stanici a libovolný roztok a pro něj spustit simulaci.

Hlavní funkcionalitou, kromě samotné simulace, je umožnit uživateli porovnávat simulaci s reálnými daty a na jejich základě provádět změny v simulaci a měnit parametry reálného procesu. Program byl vytvořen pro Industrial Edge Device (IED), na kterém zároveň běží i sběrnice dat z výroby. Tento program umí tato data získat, zpracovat a zobrazit uživateli spolu s užitečnými statistikami jako je čistota a výtěžek.

3.2.1 Implementace SMB simulace

SMB simulace je implementována podle diagramu 3.14.

Hlavní stavební jednotkou SMB stanice jsou kolony. Abstraktní třída GenericColumn reprezentuje obecnou strukturu objektu kolony. Všechny kolony mají rozměry a porozitu jako základní parametry a columnType jako popisek typu kolony. První metoda implementována kolonou je *init()*, která v argumentech dostane průtok, délku časového kroku a počet prostorových diferencí, z nichž inicializuje výpočet. Další metodou je *step()*, která vypočítá jeden časový krok a vrátí výsledky 3.7. *getInfo()* je metoda pro získání informací o dané koloně.

LinColumn a NonLinColumn jsou implementace třídy kolony. LinColumn implementuje kolonu podle EDM modelu s lineární isotermou a NonLinColumn podle EDM modelu s nekompetetivní Langmuirovou isotermou. Mají také navíc list komponent a metodu *add()*, která umožňuje komponentu přidat. Kolony poté počítají časové kroky pro každou složku.

Třída Component je jednoduchá třída, která pouze obsahuje parametry komponenty a umožňuje vytvořit svoji hlubokou kopii metodou *copy()*.

Třída Tube reprezentuje spojovací trubice a je podobná třídám reprezentující kolony, nicméně neobsahuje rozměry ani porozitu, ale pouze mrtvý objem a při volání *init()* se nezadává počet prostorových diferencí. To protože při inicializaci se z těchto parametrů spočítá počet časových kroků, kolik trvá roztoku protéct danou trubicí, vytvoří se řada této délky a každý krok se pouze řada posune o jednu pozici, vstupní hodnota jde na začátek a výstupem je poslední hodnota. Při změně parametrů a novém volání *init()* se může změnit délka řady a je třeba z hodnot ve staré frontě vyplnit novou. K tomu použijeme interpolaci a následné vyrovnání množství hmoty pomocí poměru integrálů staré a nové řady 3.8.

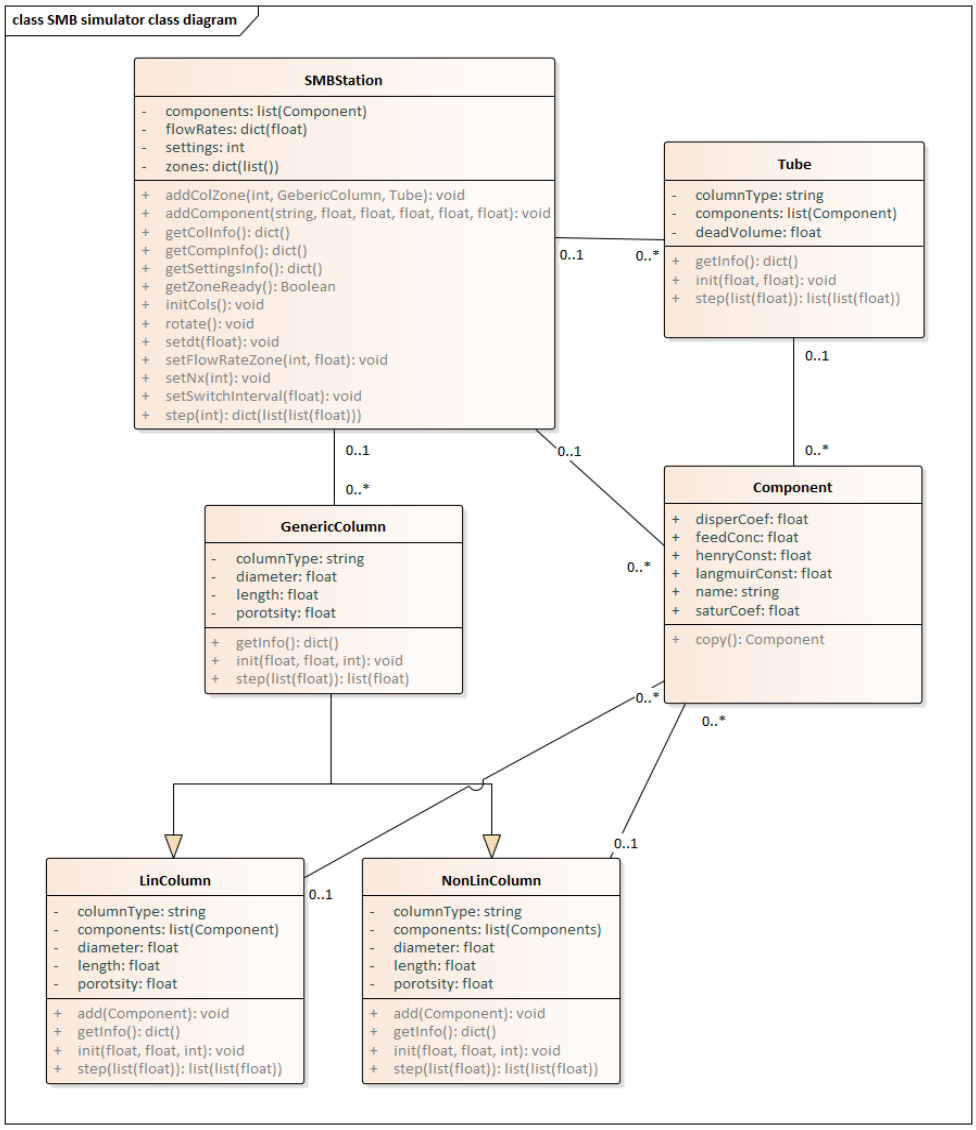
SMBStation je pak hlavní třída, která nám umožňuje vkládat kolony do jednotlivých zón *addColZone()*, definovat roztok *addComponent()*, nastavení diskretizace *setdt()* a *setNx()*, nastavení průtoků v zónách *setFlowRateZone()*, nastavení přepínacího intervalu *setSwitchInterval()* a pak inicializace všech kolon a trubiček *initCols()* a krok v čase *step()*. Metoda *rotate()* provede rotaci ventilů simulující protisměrný proud a volá se podle nastavení přepínacího intervalu. Také obsahuje metody pro získání informací o kolonách, složkách v roztoku a nastavení stanice.

3.2.2 Rozhraní

3.2.2.1 Webové rozhraní

Uživatelské rozhraní je opět vytvořené za použití webového frameworku Flask. Stejně jako u estimátoru, uživatelské rozhraní je vytvořeno ve formě webových stránek za pomocí šablon enginu Jinja2 a Javascriptu. Umožňuje vytvořit SMB stanici, definovat roztok a spustit simulaci, která má dva módy, offline a online.

Offline simulace je jednorázová simulace SMB procesu. Uživatel si nadefinuje SMB stanici, roztok a další parametry simulace jako prostorovou diskretizaci a časový krok. Uživatel pak zadá



Obrázek 3.14 Diagram tříd SMB simulátoru

Výpis kódu 3.7 Definice metody *step()* ve třídách LinColumn a NonLinColumn

```
class LinColumn(GenericColumn):
  def step(self, cins):
    output = []
    for comp, cin in zip(self.components, cins):
      b = comp.B.dot(comp.c)
      b[0] = b[0] - (self.flowSpeed * 2 * self.dx * cin)
      comp.c = linalg.solve_banded((1, 1), comp.A_diag, b)
      output.append(comp.c.tolist())
    return output

class NonLinColumn(GenericColumn):
  def step(self, cins):
    output = []
    for comp, cin in zip(self.components, cins):
      sol = optimize.root(fun=self.function,
                          x0=comp.c,
                          method='hybr',
                          args=(comp.c,
                                cin,
                                self.porosity,
                                comp.langmuirConst,
                                comp.saturCoef,
                                comp.disperCoef,
                                self.flowSpeed))
      comp.c = sol.x
      output.append(comp.c.tolist())
    return output
```

Výpis kódu 3.8 Definice metod *init()* a *step()* ve třídě Tube

```
class Tube:
  def init(self, flowRate, dt, dummyVal = 0):
    self.flowRate = flowRate
    self.dt = dt
    self.t = (self.deadVolume/self.flowRate)*3600
    self.deadSteps = int(self.t//self.dt)
    remainder = self.t%self.dt
    if remainder >= dt/2:
      self.deadSteps += 1
    for comp in self.components:
      if not hasattr(comp, 'c'):
        # first init, creates array
        comp.c = np.zeros(self.deadSteps)
      else:
        # not first init, adjusts array
        x = np.linspace(0, len(comp.c), len(comp.c))
        # interpolation function
        f = spi.CubicSpline(x, comp.c)
        xnew = np.linspace(0, len(comp.c), self.deadSteps)
        # adjusted array with correct number of points
        cnew = f(xnew)
        # calculate masses
        intgOld = np.trapz(comp.c, x)
        intgNew = np.trapz(cnew, xnew)
        # calculate difference in mass
        massDiff = 1
        if intgNew != 0:
          massDiff = intgOld/intgNew
        # adjust for mass difference
        cnew = np.multiply(cnew, massDiff)
        comp.c = cnew

  def step(self, cins):
    output = []
    for comp, cin in zip(self.components, cins):
      comp.c = np.roll(comp.c, 1)
      comp.c[0] = cin
      output.append(comp.c.tolist())
    return output
```

čas, jak dlouho chce, aby simulace probíhala. Aplikace pak spočítá simulaci po zadaný čas a zobrazí uživateli výsledný stav stanice a výstupní koncentrace složek přes celý časový úsek.

Online simulace je kontinuální proces, který spolu se simulací čte naměřená data a zobrazuje je uživateli k porovnání. Stejně jako u offline simulace si uživatel nadefinuje stanici, roztok a parametry simulace. Kromě toho uživatel ještě zadá, jak se některé parametry namapují na tagy z Data Service.

Každá ukládaná proměnná v Data Service je identifikována tagem. Protože tyto tagy mohou mít libovolný název a aplikace neumí rozhodnout, které proměnné znamenají které parametry, musí je uživatel namapovat manuálně. Některá data z těchto tagů jsou pak přímo použita jako parametry pro simulaci, například průtoky v jednotlivých zónách nebo přepínací interval.

Po namapování tagů uživatel ještě zadá časové údaje, jak dlouho proces již běží kvůli synchronizaci a časový horizont, jak do budoucnosti simulace běží, jak dlouho do minulosti chce data zobrazovat, a spustí simulaci.

Stránka s online simulací zobrazuje uživateli celkově dvě sady výstupů, Naměřená data přečtená z Data Service a simulovaná data, která běží napřed o budoucí časový horizont, Pro každý z nich je zobrazena čistota a výtěžek složek a stav stanice, pro simulovaná data jsou zobrazeny tyto hodnoty dvě, jedna pro budoucí horizont a druhá pro aktuální čas simulace.

Vykreslování a průběžné dotazování na server je provedeno za pomoci Javascriptu. Stránka se pravidelně dotazuje na server, zda má nová data. Tento dotaz je HTTP Post požadavek, který v těle zprávy obsahuje informaci o čase posledních obdržených dat. Server poté odešle data, která jsou novější než čas v dotazu.

Uživatel může provádět změny parametrů jak v simulaci tak v reálném procesu. Běžně se změna provede nejprve v simulaci a zobrazí se pouze v budoucím časovém horizontu, pomocí čehož se uživatel může rozhodnout, jestli změna měla chtěný efekt.

Jakmile uběhne čas daný časovým horizontem, aplikace se zeptá uživatele, jestli chce změnu ponechat nebo vrátit. Když se uživatel rozhodne změnu ponechat, odešle se na server požadavek na ponechání změn. Server se poté postará o to, aby se změny odeslali na Databus, ze kterého se zapíší do PLC.

Pokud se uživatel rozhodne změny vrátit, aplikace zapomene nový stav stanice a použije starý stav, který si uložila před provedením změn, pro který zpětně dopočítá časový horizont. Starý stav je uložen tak, že se vytvoří hluboká kopie stanice.

Je také možné provést "tvrdou"změnu parametrů, kde se změny propíšou jak pro simulaci tak pro reálný proces a není možné pak změny zpětně vrátit.

Všechny tyto operace jsou možné provádět pouze, pokud je uživatel přihlášen k administrativnímu účtu. Pro nepřihlášené uživatele umožňuje aplikace monitorovat běžící online simulaci, ale neumožňuje provádět změny. Server zajišťuje to, aby se při přihlášení k administrativního účtu provedlo odhlášení na ostatních zařízení, a je tedy možné být přihlášen pouze z jednoho zařízení zároveň. Tím se eliminuje možnost konfliktu, kdyby se snažili simulaci řídit dva uživatelé zároveň.

Na straně serveru se při spuštění online simulace vytvoří tři vlákna, které se starají o periodické čtení dat z Data Service, výpočet simulovaných dat, výpočet čistoty a výtěžku a ukládání dat. Protože tyto vlákna pracují paralelně a sdílejí data, bylo třeba implementovat synchronizaci za pomoci zámků.

Reálná data obsahují časovou známku, pomocí které zajišťujeme jejich synchronizaci se simulací. Pokaždé když server obdrží nová data z Data Service, dopočítá se simulace do stejného času.

3.2.2.2 Komunikace s Data Service

Data Service implementuje REST rozhraní, které je použito pro přístup k datům. To vyžaduje přihlášení pro které v IED existuje webové API. Aplikace pošle POST žádost na adresu IED

s cestou */login/direct* s přihlašovacími údaji v těle žádosti. V odpovědi pak dostane token, který je pak nutný použít v cookies pro komunikaci s Data Servicem.

Pro získávání seznamu proměnných je použita žádost GET */DataService/Variables*. Odpověď na tento požadavek obsahuje seznam všech proměnných s metadaty a identifikátory, které jsou potřeba pro přístup k samotným datům.

Data jsou získávána dvěma způsoby. První způsob je pro data, u kterých jsou zajímavé pouze poslední naměřené hodnoty. Typicky jsou to data o stavu stanice, průtoky, pozice, přepínací perioda atd. Pro tato data je použita žádosti GET */DataService/Data/<id>*, která jako query parametr vyžaduje časový interval, dále umožňuje zadat pořadí dat a počet datových záznamů. Pro získání posledního záznamu je použit interval od aktuálního času po stejný čas den zpátky, sestupné pořadí a je vzata pouze první hodnotu.

Druhý způsob čtení je pro data, u kterých jsou zajímavé všechny hodnoty. Typicky to jsou naměřené koncentrace na výstupech. Zde je použita žádost POST */DataService/Data/Delta*. V těle žádosti je zadán identifikátor proměnné a časová známka, od kdy jsou hodnoty zajímavé. Pro tyto žádosti je uložena polední časová známka, pro kterou již data byla obdržena a následující žádosti se dotazují pouze na novější data.

Při spuštění online simulace je spuštěno vlákno s procesem, který periodicky čte všechny relevantní hodnoty, v prvním případě je aktualizuje a v druhém případě je ukládá.

Pro HTTP komunikaci je použit modul v pythonu *requests*.

3.2.2.3 Zápis hodnot

Data Service pouze čte, ukládá a zprostředkovává naměřená data a neumožňuje tak zápis. Aplikace je proto přímo připojena k Databusu MQTT klientem implementovaným v knihovně *paho-mqtt*. Zápis pak probíhá tak, že publishneme zprávu na topic *ie/d/j/simatic/v1/s7c1/dp/w/PLC0*. Struktura topicu znamená následující:

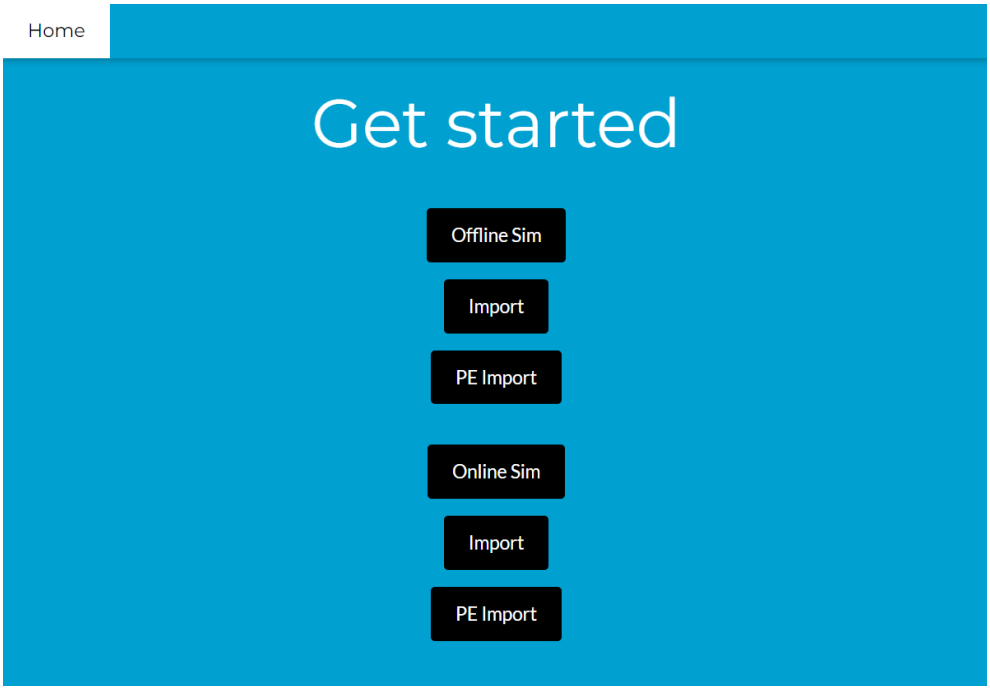
- **ie** - Industrial edge
- **d** - druh zasílaných dat, „d“ pro data, „m“ pro metadata
- **j** - kódování zaslaných dat. „j“ pro JSON
- **simatic** - schéma zaslaných dat
- **v1** - verze schématu
- **s7c1** - identifikátor aplikace poskytovatele - „s7c1“ pro SIMATIC S7 Connector instance 1
- **dp** - indikátor obsahu dat - „dp“ pro DataPoints
- **w** - účel zprávy - „w“ pro write
- **PLC0** - název kolekce definované v S7 Connectoru

Obsah zprávy ve formátu JSON obsahuje identifikátor tagu, hodnotu proměnné a časovou známku.

Obrázek 3.15 Ukázka MQTT zprávy

3.2.3 Popis uživatelské interakce

Na úvodní obrazovce si uživatel může vybrat mezi online a offline simulací 3.16. Pro obě tyto možnosti může také importovat nastavení z estimátoru parametrů nebo přímo z předchozích simulací.



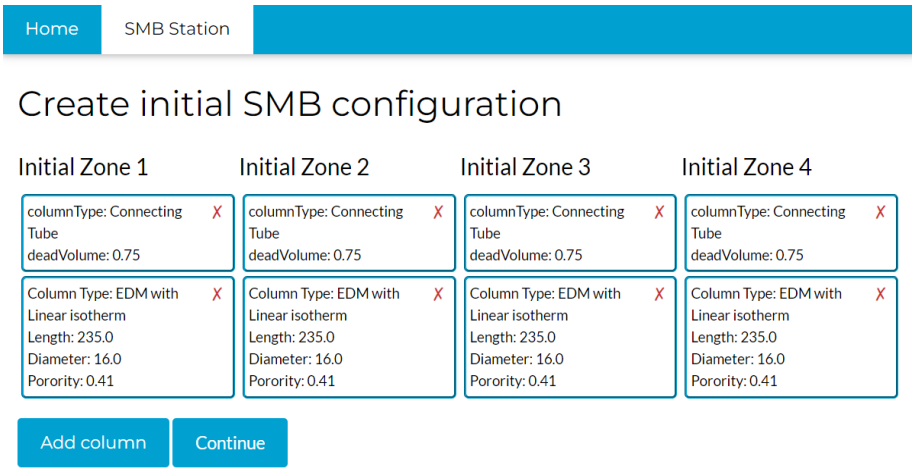
Obrázek 3.16 SMB simulátor - úvodní stránka

Při výběru offline simulace se nejprve definuje SMB stanice 3.17. Uživatel zadá parametry a vybere, do které zóny kolonu vložit. Jakmile je v každé zóně alespoň jedna kolona, je možné pokračovat dále.

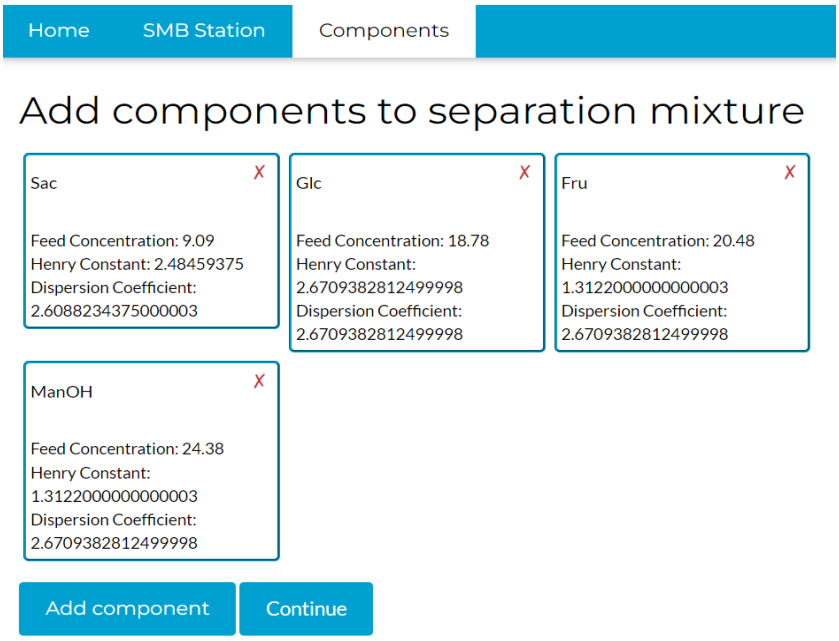
Následuje definice roztoku 3.18. Uživatel může přidat libovolné množství složek roztoku s různými modelovými parametry.

Posledním krokem před spuštěním samotné simulace je zadat průtoky v jednotlivých zónách, definovat diskretizaci a zadat čas simulace 3.19.

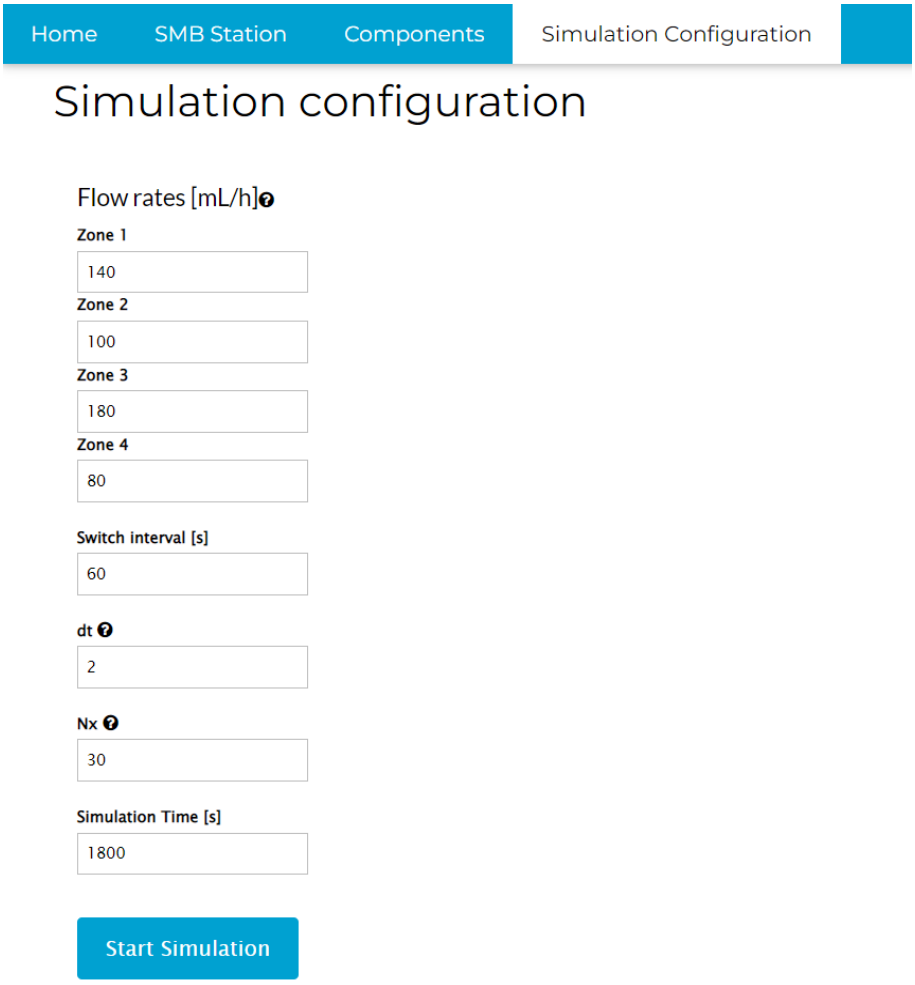
Výsledek offline simulace je rozdělen do dvou částí. V první části jsou informace o výstupech



Obrázek 3.17 SMB simulátor - definice SMB stanice

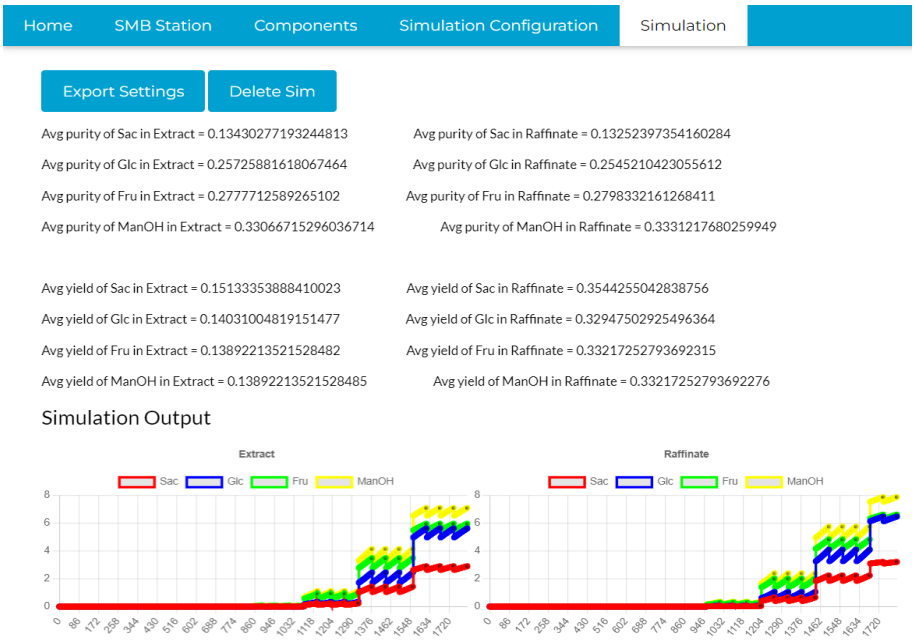


Obrázek 3.18 SMB simulátor - definice roztoku

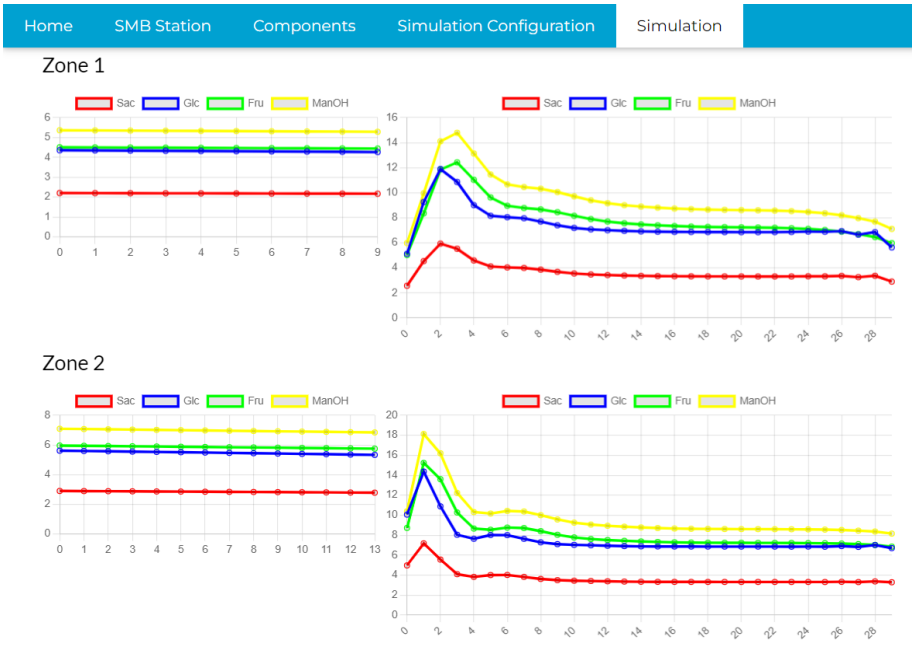


Obrázek 3.19 SMB simulátor - konfigurace simulace

ze stanice, tzn. koncentrace v extraktu a rafinátu a spočítané čistoty a výtěžky pro všechny složky 3.20. Druhá část pak ukazuje výsledný stav kolon 3.21.

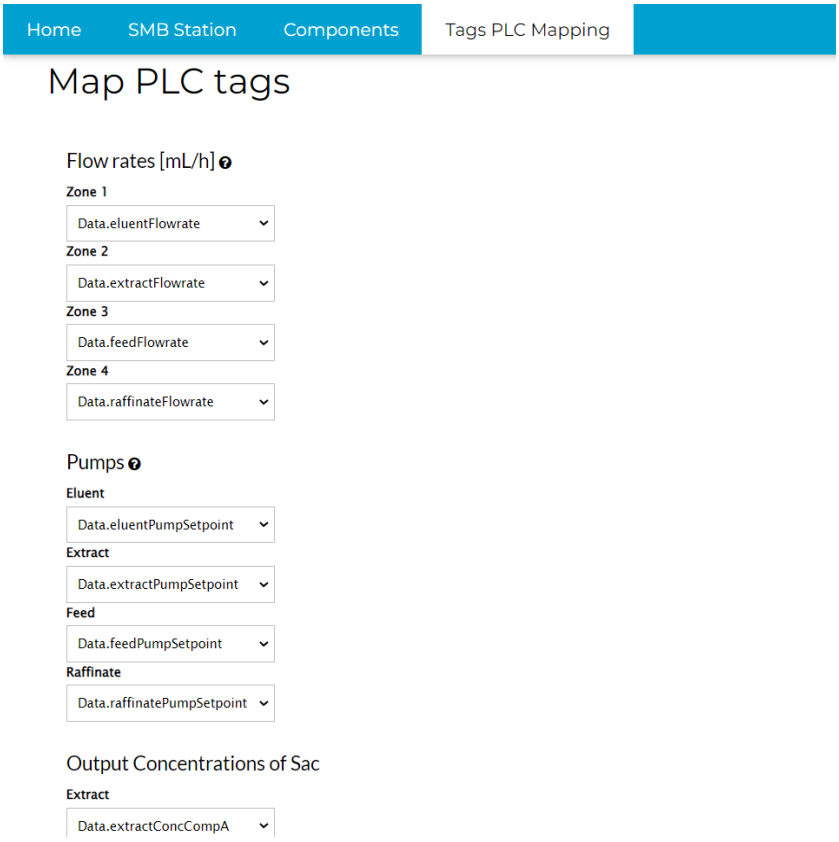


■ **Obrázek 3.20** SMB simulátor - offline simulace - výstup



■ **Obrázek 3.21** SMB simulátor - offline simulace - stav kolon

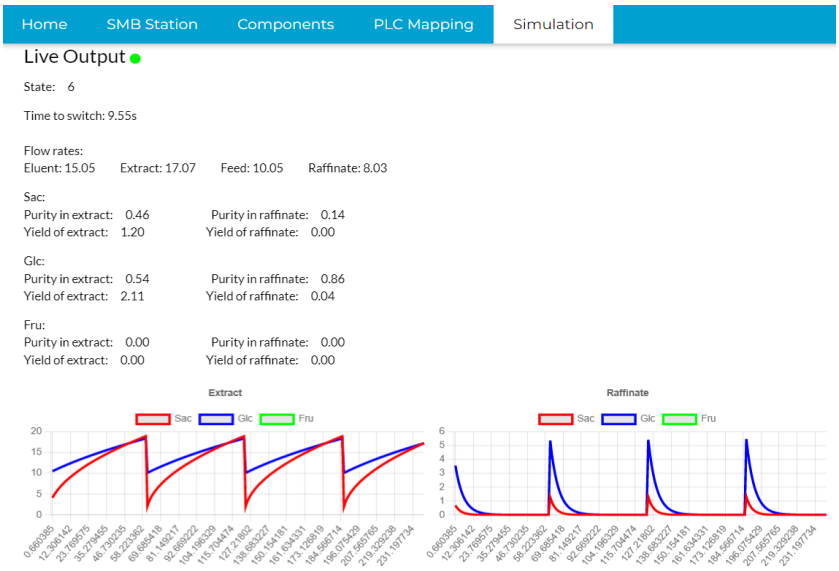
V online simulaci dva první kroky jsou stejné. Třetí krok se skládá z nastavení simulace a mapování tagů z řídícího PLC na simulační parametry 3.22. Navíc si uživatel zadá budoucí horizont, ve kterém je simulace počítána v předstihu ku reálnému času.



Obrázek 3.22 SMB simulátor - online simulace - mapování PLC tagů

Stránka online simulace je rozdělena do dvou částí. Výstup simulace 3.23, ve které se promítne předchozí a budoucí horizont, a výstup reálných dat 3.24. Každá část obsahuje informace o čistotě a výtěžku výstupů. Je také zobrazena pozice reálné a simulované stanice, čas do přepnutí ventilů, průtok v reálné stanici a stav připojení k Data Service.

Obrázek 3.23 SMB simulátor - online simulace - simulované hodnoty



Obrázek 3.24 SMB simulátor - online simulace - reálná data

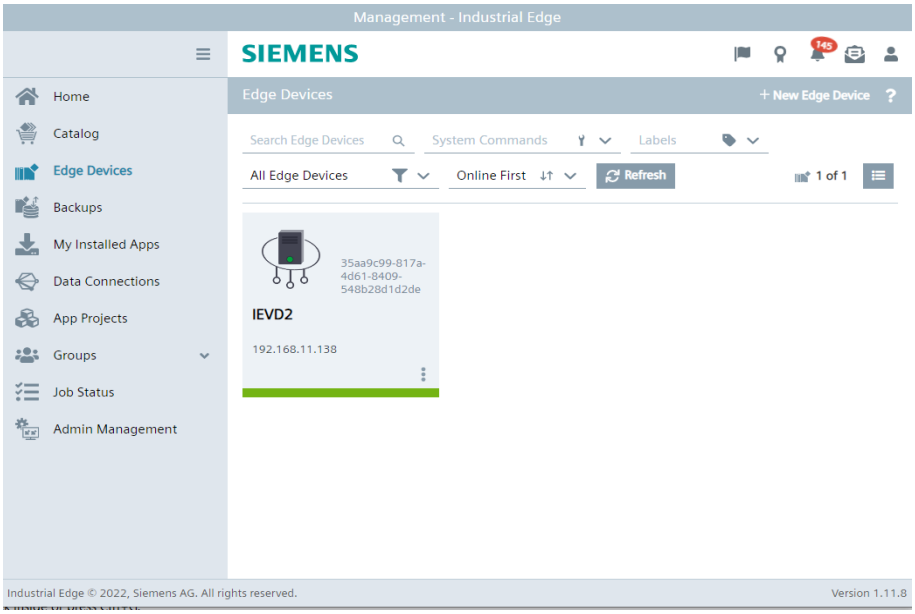
3.3 Industrial Edge

V této sekci bude popsán postup pro vytvoření IED. Bude ukázáno vytvoření instance IEM, registraci IED do IEM, instalaci aplikací a jejich konfiguraci a spuštění. Také bude ukázán postup pro vytvoření vlastní aplikace a způsob integrace s ostatními aplikacemi v IED.

3.3.1 Vytvoření IEM a registrace IED

Pro vytvoření IEM je potřeba přístup do IEH, ve kterém je k dispozici disk s obrazem IEM, který je za pomocí virtualizačního nástroje VMWare nakonfigurován a spuštěn. V IEH je vytvořena IEM instance a je stažen konfigurační soubor, pomocí kterého je IEM aktivován. Při aktivaci je vytvořen administrativní účet, pomocí kterého se do IEM přihlašuje.

Dále je vytvořen IED 3.25. Pro naše účely je použit virtuální IED, který je opět spuštěn za pomoci VMWare. V IEM je vytvořen konfigurační soubor, pojmenuje se nový IED, vybere se jeho typ a definuje se administrativní účet. Za pomocí konfiguračního souboru je IED aktivován.



Obrázek 3.25 IEM - list připojených IED

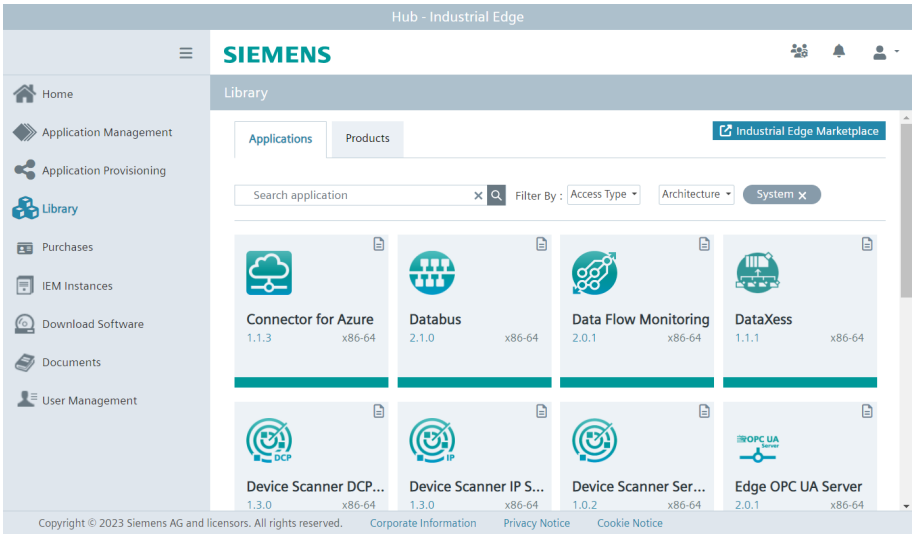
3.3.2 Instalace a konfigurace aplikací

Potřebné aplikace je nejprve třeba nahrát do IEM. Databus, Data Service a SIMATIC S7 Connector je získán z nabídky aplikací v IEH 3.26. Ty se nachází v knihovně aplikací, pouze se vybere instance našeho IEM a aplikace se automaticky nahraje 3.27.

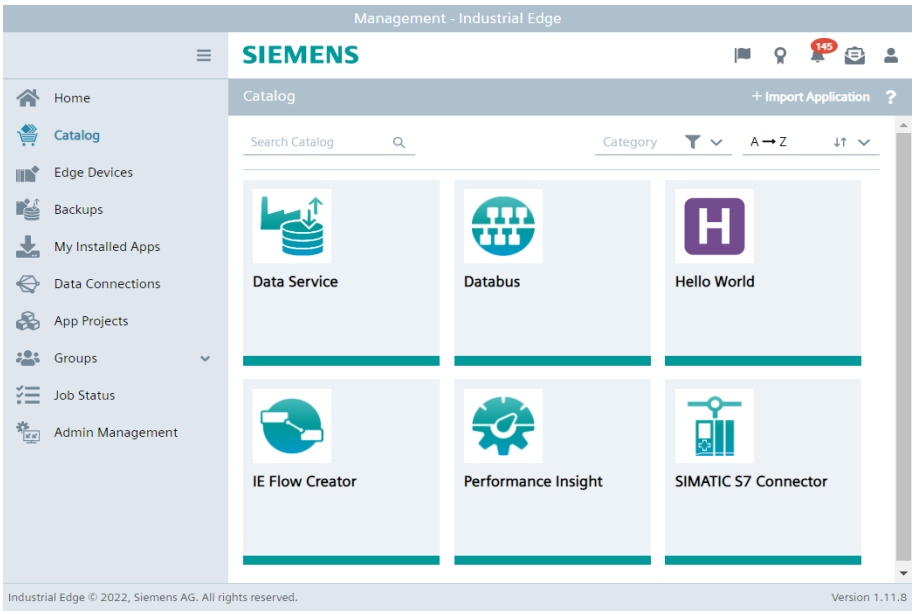
Pro vlastní aplikaci SMB simulátoru je třeba Docker engine, Docker compose a aplikaci IEAP.

Nejprve je pro naši aplikaci vytvořen Dockerfile, pomocí kterého je z aplikace vytvořen Docker image. Je vybrán libovolný podkladový python image, aplikaci je do něj zkopírována a jsou nainstalovány závislosti.

Dále je vytvořen docker-compose.yml soubor, ve kterém je nadefinováno, jak se má image v kontejneru spustit. Je důležité publikovat port, na kterém aplikace uvnitř kontejneru poslouchá



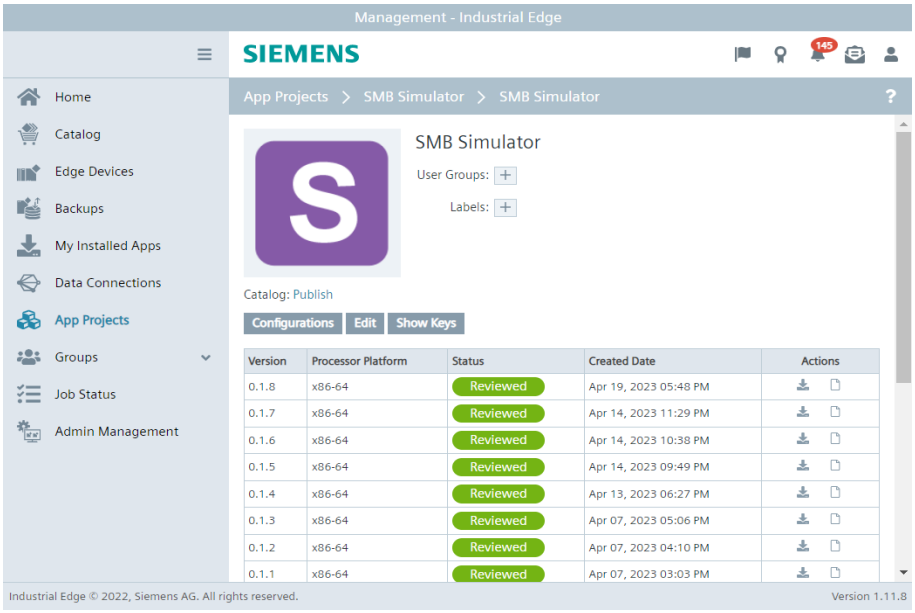
Obrázek 3.26 IEH - knihovna aplikací



Obrázek 3.27 IEM - seznam stažených aplikací

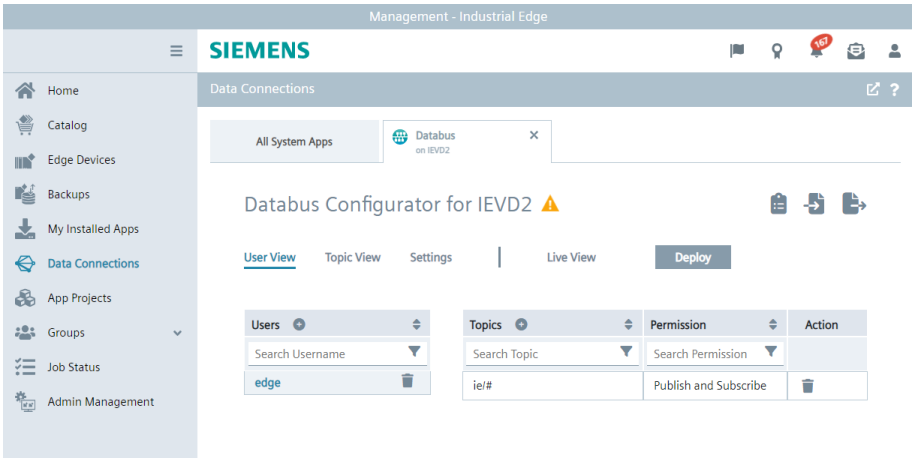
a napojit kontejner na network proxy-redirect, přes který probíhá komunikace s Databusem uvnitř IED.

IEAP je napojen na Docker engine, ze kterého stáhne Docker image, a na IEM, na který bude aplikace nahrána. Za pomoci souboru docker-compose.yml a Docker image je vytvořena nová verze aplikace a je nahrána na IEM 3.28. Jakmile jsou všechny aplikace v IEM, jsou následně nainstalovány do IED 3.31.



■ **Obrázek 3.28** IEH - seznam verzí nahrané aplikace

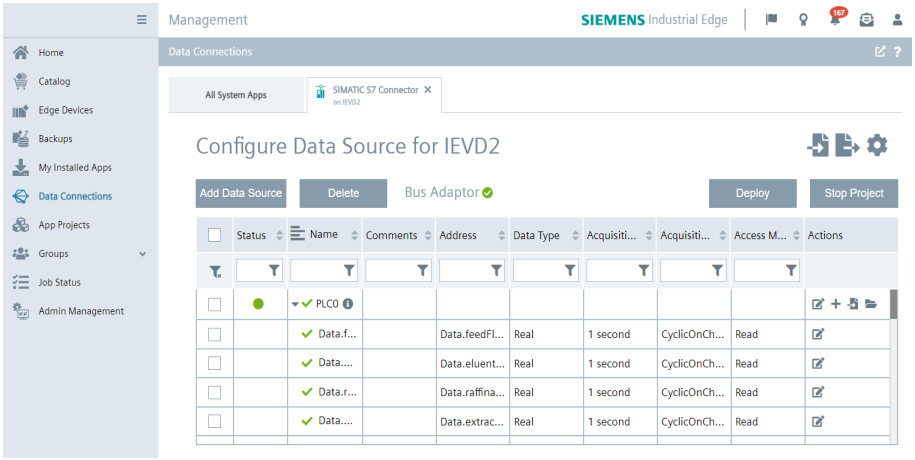
Databus a SIMATIC S7 Connector je potřeba nakonfigurovat pomocí konfiguračních aplikací v IEM. Pro Databus je vytvořen uživatel, pomocí kterého se všechny aplikace k Databusu přihlašují. Dále je vytvořeno téma *ie/#*, obecné téma obsahující všechna témata začínající na *ie/*, která se standardně používají pro komunikaci mezi IE aplikacemi 3.29.



■ **Obrázek 3.29** IEM - konfigurace pro Databus

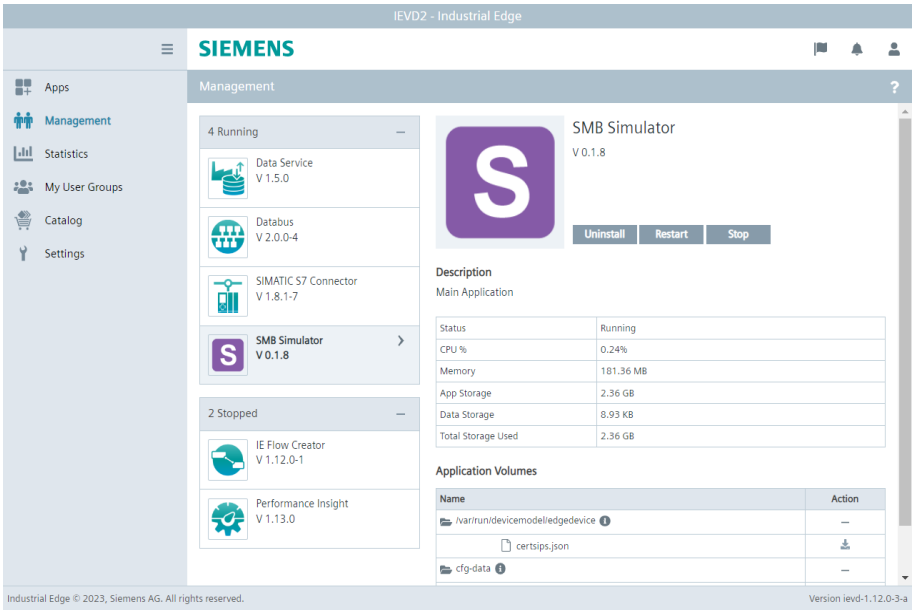
V konfiguraci SIMATIC S7 Connectoru je třeba přihlásit se k Databusu a definovat PLC, ze kterého budeme číst. Je třeba zadat jeho ip adresu, typ a vybrat veškeré tagy, pro které jsme

poté nadefinovali jejich typy a nastavit, zda je možné z nich jen číst, nebo do nich i zapisovat 3.30.



■ **Obrázek 3.30** IEM - konfigurace pro SIMATIC S7 Connector

Data Service má vlastní uživatelské rozhraní, ke kterému je možné přistoupit přímo z IED. Protože čtení a ukládání dat ze SIMATIC S7 Connectoru je jeden z běžných způsobů použití této aplikace, má už přednastavenou konfiguraci pro tuto komunikaci a stačí se pouze přihlásit k Databusu. Poté už jsou jen vybrány tagy, které chceme ukládat, a způsob jejich uložení. Data Service nabízí mnoho způsobů jak data ukládat, třídit a agregovat. V této aplikaci jsme však tyto funkce nepoužili.



■ **Obrázek 3.31** IED - seznam nainstalovaných a běžících aplikací

Kapitola 4
Výsledky

Aplikace estimátoru parametrů je nasazena na Vysoké škole chemicko-technologické v Praze. Webové rozhraní je přístupné přes školní síť a používá se pro estimace modelových parametrů v rámci dlouhodobého výzkumu na Ústavu sacharidů a cereálií zaměřenému na vývoj aplikací preparativní chromatografie pro separaci sacharidů a příbuzných látek.

Jedním z běžných problémů, ke kterému dochází při výpočtu modelu, je skok koncentrace na začátku kolony, když se přestává vpouštět nástřik. Kvůli diskretizaci času a prostoru, koncentrace na začátku kolony klesne na nulu během jednoho časového kroku, což často vede k nestabilitě výpočtu, která se projeví „vlněním“ na grafu koncentrace. Tento jev byl jeden z hlavních důvodů, proč byla zvolena Crank-Nicolsonova implicitní metoda, která by měla zlepšit stabilitu. Stále však má velikost časového a prostorového kroku velký vliv na stabilitu výpočtu. Větší počet diferenčních náhrad použitých při diskretizaci modelu ovšem vede k delšímu času výpočtu, je proto nutné najít vhodný počet, při kterém je výpočet stabilní a počet diferencí není zbytečně vysoký. [30]

Byly provedeny jednoduché test řešiče EDM s lineární isotermou, pro stejné operační parametry byly měněny počty časových a prostorových diferencí a byly sledovány průběhy koncentrace v čase v několika prostorových bodech v koloně. Také byla sledována hmotnost složky na vstupu a na výstupu, což se osvědčilo jako vhodný ukazatel úspěšného výpočtu, tedy byla splněna bilance složky.

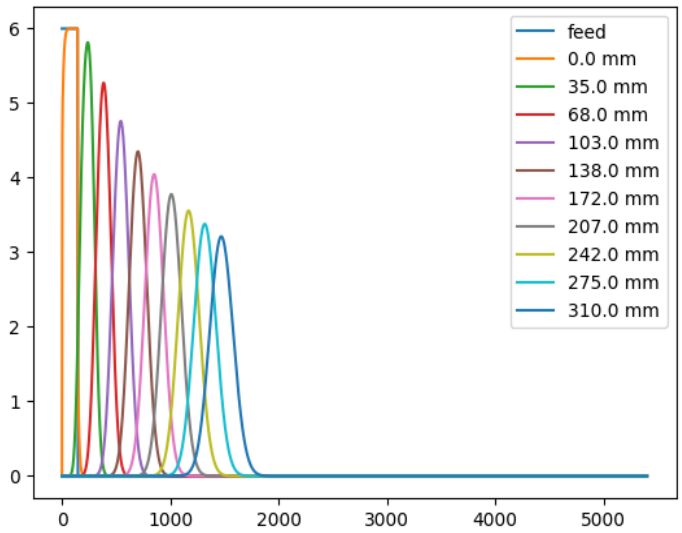
Hmotnost na vstupu	36.0 mg	-
Hmotnost na výstupu	36.0 mg	-
Rozdíl	0.0 mg	0.0%

Tabulka 4.1 Rozdíl hmotnosti mezi vstupem a výstupem – 1500 časových diferencí, 150 prostorových diferencí

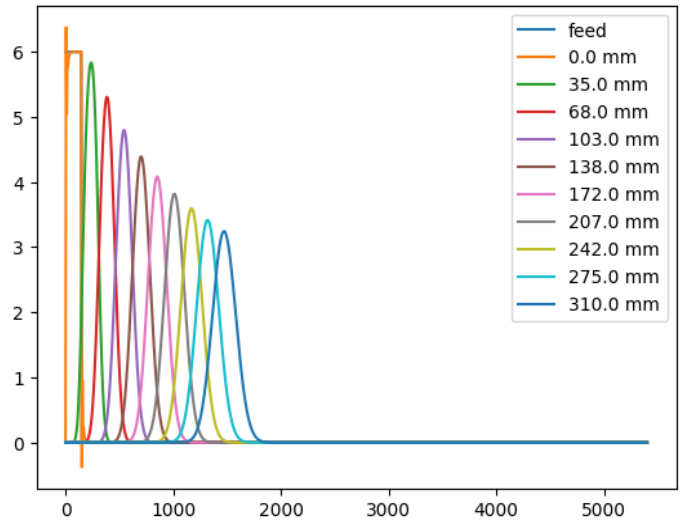
Na grafu 4.1 je vidět, že při dostatečném množství časových a prostorových diferencí byl výpočet stabilní, nedocházelo k žádnému patrnému vlnění a v tabulce 4.1 je vidět, hmotnost na výstupu odpovídala hmotnosti na vstupu 4.1. Dále byl snižován počet časových diferencí.

Z grafu 4.2 je patrné, že už při počtu 1000 časových kroků docházelo k nestabilitě na začátku výpočtu, což se dále projevilo v neshodě mezi vstupní a výstupní hmotností, což je vidět v tabulce 4.2. Dále bylo použito opět 1500 časových kroků a zmenšil se počet souřadnicových kroků.

V tabulce 4.3 je vidět, že snížení počtu souřadnicových kroků se sice neprojevilo na neshodě hmotností, nicméně je vidět patrné vlnění na grafu 4.3, které by zaneslo nepřesnost do následného porovnání s reálnými daty.



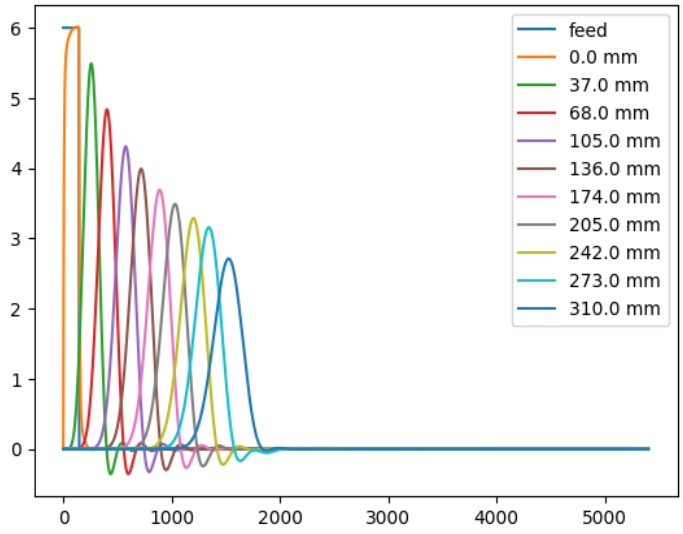
■ **Obrázek 4.1** Grafy koncentrací napříč kolonou v čase – 1500 časových diferencí, 150 prostorových diferencí



■ **Obrázek 4.2** Grafy koncentrací napříč kolonou v čase – 1000 časových diferencí, 150 prostorových diferencí

Hmotnost na vstupu	36.0 mg	-
Hmotnost na výstupu	36.45 mg	-
Rozdíl	0.45 mg	-1.25%

■ **Tabulka 4.2** Rozdíl hmotnosti mezi vstupem a výstupem – 1000 časových kroků, 150 prostorových kroků



Obrázek 4.3 Grafy koncentrací napříč kolonou v čase – 1500 časových diferencí, 50 prostorových diferencí

Hmotnost na vstupu	36.0 mg	-
Hmotnost na výstupu	36.0 mg	-
Rozdíl	0.0 mg	0.0%

Tabulka 4.3 Rozdíl hmotnosti mezi vstupem a výstupem – 1500 časových diferencí, 50 prostorových diferencí

Důležitým rozhodnutím při implementaci estimátoru parametrů byl výběr optimalizačního algoritmu. Ve výsledné aplikaci bylo rozhodnuto dát uživateli na výběr ze tří možností, které zde budou porovnány. Byly sledovány dva parametry – doba běhu a výsledná hodnota ztrátové funkce. Byly použity všechny možnosti preprocessingu, ztrátová funkce byla použita „Squares“, ve které se sčítají druhé mocniny hodnoty rozdílu, model bych použit EDM s lineární isotermou a jako faktor pro normalizaci byla použita maximální hodnota koncentrace na výstupu. Před optimalizací byla provedena krátká analýza experimentu, pomocí které byl proveden počáteční odhad parametrů. Testy byly provedeny na procesoru Intel(R) Core(TM) i7-7820HQ CPU 2.90GHz. Pro všechny optimalizační algoritmy byly použity výchozí nastavení.

Algoritmus na 1. úrovni	Algoritmus na 2. úrovni	Čas	Hodnota ztrátové funkce
Brute force	Brute force	13:40:02.79	57.4987
Nelder-Mead	Nelder-Mead	10:43:36.67	56.7190
SHGO	SHGO	12:14:16.12	56.7461

Tabulka 4.4 Porovnání doby běhu a hodnoty ztrátové funkce jednotlivých algoritmů na obou úrovních

„Brute force“ trval nejdelší dobu a měl největší hodnotu ztrátové funkce. To není překvapivé, předpokládali jsme, že „brute force“ není vhodný algoritmus pro úlohu s kontinuálním stavovým prostorem. Z dalších měření byl proto z důvodů ušetření času vynechán.

Nelder-Mead v porovnání se Simplicial Homology Global Optimization (SHGO) nalezl lepší řešení a za kratší dobu.

Poté byly testovány kombinace algoritmů, čímž jsme se snažili zjistit, zda je některý algoritmus lepší v první nebo druhé úrovni. Použili jsme stejný výpočet jako v prvním měření.

Algoritmus na 1. úrovni	Algoritmus na 2. úrovni	Čas	Hodnota ztrátové funkce
SHGO	Nelder-Mead	3:45:36.31	58.5525
Nelder-Mead	SHGO	10:40:27.69	56.7460

Tabulka 4.5 Porovnání doby běhu a hodnoty ztrátové funkce kombinací algoritmů

Žádná z kombinací nevykazuje lepší výsledky nebo čas než algoritmy jednotlivě, naopak je vidět, že některé algoritmy příliš dobře nespolupracují. Nelder-Mead na první úrovni a SHGO na úrovni druhé vykazovaly srovnatelné výsledky, jako jednotlivé algoritmy na obou úrovních, nicméně čas výpočtu se oproti Nelder-Mead na obou úrovních zlepšil jen nepatrně, zatímco výsledek se mírně zhoršil.

Nelder-Mead nalezl nejnižší hodnotu ztrátové funkce, zkusili jsme pro něj omezit počet iterací a sledovali jsme, jak efektivně je možné snížit čas běhu na úkor hodnoty ztrátové funkce. Na obou úrovních byl nastaven počet iterací na stejnou hodnotu.

Počet iterací	Čas	Hodnota ztrátové funkce
Neomezeno	10:43:36.67	56.7190
20	8:26:53.29	56.7190
10	3:41:52.00	56.7192

Tabulka 4.6 Porovnání doby běhu a hodnoty ztrátové funkce Nelder-Mead algoritmu při omezení počtu iterací

Snížení počtu iterací mělo velký vliv na snížení času běhu a relativně malý vliv na hodnotu ztrátové funkce. Při deseti iteracích byl čas výrazně menší oproti většině ostatních měření a

zároveň byl výsledek lepší než u všech ostatních měření, ve kterých byl použit jiný algoritmus než Nelder-Mead.

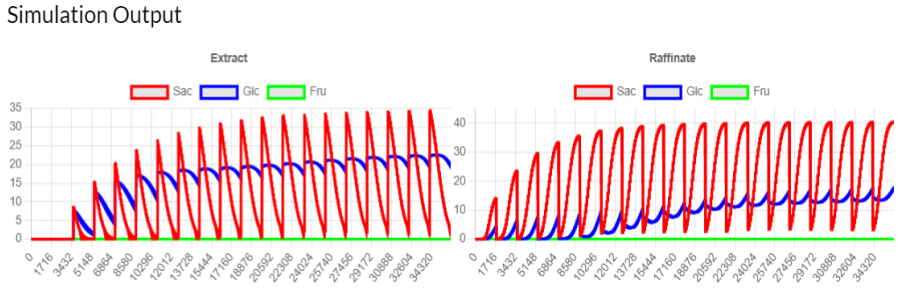
Na základě provedených měření můžeme usoudit, že Nelder-Mead je nejvhodnější algoritmus z těchto možností a omezením počtu iterací se dá efektivně kontrolovat čas na úkor kvality výsledku. Je však nutné podotknout, že počáteční odhad má velký vliv na tento algoritmus. [31] Aplikace estimátoru parametrů proto obsahuje nástroje, které ulehčují nalezení vhodného počátečního odhadu, což je dalším důvodem, proč je Nelder-Mead algoritmus vhodný.

Dále budou hodnoceny výsledky a nasazení aplikace SMB simulátoru. Ta byla za pomoci Docker Engine v20.10.21 a Industrial Edge App Publisher (IEAP) verze 1.9.5 nahrána do Industrial Edge Management (IEM) verze 1.11.8 a nainstalována na virtuální Industrial Edge Device (IED) verze 1.12.0-3-a. Byla provedena úspěšná instalace aplikací Data Service V 1.5.0, Databus V 2.0.0-4 a SIMATIC S7 Connector V 1.8.1-7. Byla proveda úspěšná integrace těchto komponent a za pomoci aplikace SMB simulátoru bylo provedeno úspěšné čtení a zápisování tagů připojeného PLC, které zajišťuje primární řídící proces, a umožnilo se tak uživateli daný proces monitorovat a řídit.

Takto připravené IED lze napojit k libovolnému PLC, které řídí SMB proces, a umožnit tak uživateli monitorovat a řídit daný proces.

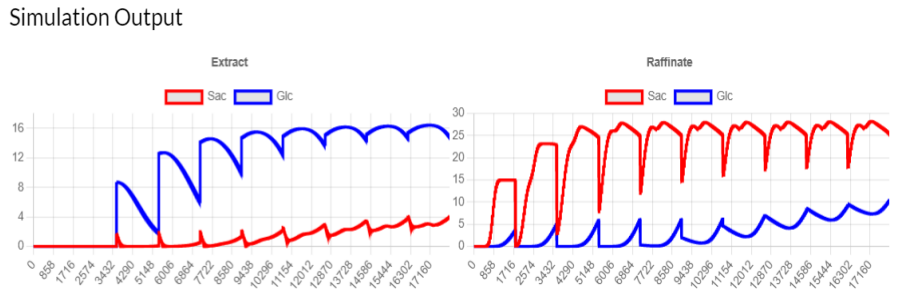
SMB chromatografie a její simulace je obecně citlivá na nastavení procesních parametrů. Pro přepínací interval je třeba nalézt optimální hodnota, při které je pomalejší složka dostatečně unášena simulovaným opačným proudem, zatímco rychlejší složka simulovaný opačný proud překoná. Při špatně nastavených průtocích může docházet k zahlcení kolony, kdy se složky dostanou přes hranici čtvrté a první zóny. [32]

Graf 4.4 je příkladem nevhodně nastavených procesních parametrů. V obou výstupech jsou obě složky, což znamená, že ani jedna složka nedosáhne dostatečné čistoty.



■ **Obrázek 4.4** Ukázka výstupních koncentrací [g/l] v čase [s] – Simulace s nevhodně nastavenými parametry

Graf 4.5 je příkladem dobře nastavených parametrů. V každém výstupu vystupuje převážně jedna složka.



■ **Obrázek 4.5** Ukázka výstupních koncentrací [g/l] v čase [s] – Simulace s dobře nastavenými parametry

Pro otestování aplikace SMB simulátoru v online režimu, tj. při paralelním běhu simulace se čtením reálného procesu, bylo vytvořeno virtuální PLC řady S7-1500 za pomoci programu S7-PLCSIM Advanced V4.0 SP1. Pro toto virtuální PLC byl za pomoci programu Totally Integrated Automation Portal V17 (zkráceně TIA Portal), který umožňuje programovat PLC, vytvořen simulovaný SMB proces, který obsahuje tagy a simuluje chování tagů teoretického reálného procesu.

Pro komunikaci s virtuálním PLC byl nakonfigurován SIMATIC S7 Connector, který virtuální PLC propojil s Databusem. Byl použit Optimalizovaný S7 protokol, nicméně je možné použít Legacy S7 protokol, který je běžně používán staršími typy Siemens PLC, jako jsou S7-1200 nebo S7-300/400. Kdyby bylo třeba jiného typu komunikace, v Industrial Edge Hub (IEH) knihovně aplikací je možné získat další konektory implementující další druhy komunikace, jako jsou například protokoly založené na ethernet komunikaci, jako je OPC UA nebo TCP/IP, nebo protokoly založené na RS-232 používané průmyslovými sběrnicemi, jako je Profibus. Protože komunikace mezi konektorem a SMB simulátorem probíhá přes Databus, je možné použít jiný konektor bez většího úsilí.

V tomto simulovaném prostředí bylo úspěšně otestováno ukládání tagů do aplikace Data Service, čtení dat z Data Service a jejich vykreslování v aplikaci SMB simulátoru. Dále byl úspěšně sledován vliv nových operačních parametrů v budoucím horizontu simulace a při rozhodnutí nové operační parametry použít bylo otestováno zapsání nových parametrů do PLC simulující reálný separační proces, čímž je umožněno manuální modelové prediktivní řízení procesu.

Kapitola 5
Závěr

Cíle práce vytyčené v úvodní kapitole byly bez výhrad splněny. Byly navrženy a vyvinuty dva rozsáhlé nástroje určené pro propojení výzkumné a průmyslové činnosti na poli preparativní chromatografie. Nástroje se zabývají matematickým modelováním chromatografického procesu, zejména potom matematickým popisem založeným na zákonech zachování hmoty nazvaným Rovnovážný disperzní model chromatografie. [8] Tento model je oblíbený zejména z důvodu, že dokáže popsat asymetrický charakter koncentračních píků systému zahrnutím členu pro zdánlivou axiální disperzi, zároveň ovšem zůstává relativně jednoduchým modelem pro výpočet. Model lze efektivně využít pro systémy prediktivního řízení v průmyslové praxi, což bylo demonstrováno za pomoci vyvinuté aplikace pro platformu Industrial Edge (Siemens AG, DE). Tato platforma slouží pro komerční průmyslové účely a nabízí veškerou infrastrukturu pro bezpečné připojení k automatizačním a řídícím systémům založených na robustních průmyslových kontrolérech a systémech vyššího procesního řízení.

Během vývoje softwarových nástrojů byl použit distribuovaný systém správy verzí Git hostovaný na webu GitHub.com. Zdrojový kód pro back-end obou hlavních nástrojů byl napsán ve vyšším programovacím jazyku Python 3.10 za použití dostupných open-source knihoven. Front-end obou nástrojů a grafické uživatelské rozhraní je ve formě webových stránek a byl naprogramován ve vyšším jazyce JavaScript, také s použitím dostupných open-source knihoven.

5.1 Estimátor parametrů

První část vyvinutého softwarového balíčku byla nazvána Parameter Estimator a slouží k nalezení parametrů zmíněného modelu na základě dat zjištěných při laboratorních experimentech preparativní chromatografie s pulsním nástřikem. Software byl vyvíjen v úzké spolupráci s výzkumníky z Ústavu sacharidů a cereálií (VŠCHT, Praha).

Architektura programu je objektově orientovaná a byly implementovány procedury pro správu a klasterizaci experimentálních dat a výsledků, řešiče modelu, algoritmy pro výpočet loss funkcí a dvojúrovňovou optimalizaci, což je úloha, na kterou odhad modelových parametrů vede. Pro ukládání dat byla použita NoSQL databáze MongoDB. Nástroj Parameter Estimator byl testován za použití reálných dat z experimentů provedených na Ústavu sacharidů a cereálií (VŠCHT, Praha) a debugován ve spolupráci s výzkumníky ústavu.

Současná verze nástroje implementuje veškeré požadavky na funkce a možnosti užívání. Nástroj bude v budoucnu dále rozšiřován, upraven design uživatelského rozhraní a intenzivně využíván pro výzkumnou a pedagogickou činnost. Nástroj může být v budoucnu snadno zpřístupněn široké odborné veřejnosti na veřejné webové doméně jako open-source služba, popř. je v budoucnu možná i jeho komercionalizace.

5.2 SMB simulátor

Druhou zásadní částí této práce byl návrh a vývoj aplikace pro platformu Industrial Edge (Siemens AG, DE), která umožňuje simulaci kontinuálního chromatografického procesu v SMB režimu. Tato aplikace využívá modelových parametrů zjištěných za pomoci programu Parameter Estimator (které je možné exportovat a importovat ve formě JSON souboru) a také využívá stejných řešičů rovnovážného disperzního modelu.

SMB Simulator umožňuje uživateli charakterizovat simulovanou SMB stanici, včetně počtu a parametrů jednotlivých kolon a mrtvého objemu dopravních cest mezi nimi. Také je možné definovat počet složek v separované směsi a parametry modelu specifické pro každou z nich. Uživatel má možnost provést dynamickou simulaci SMB procesu v offline módu, nebo připojit PLC v online módu. Pro připojení signálů z reálného SMB procesu je nutné namapovat důležité tagy, jako jsou hodnoty průtoků na vstupních a výstupních proudech, koncentrace složek v těchto proudech a další. Poté je zpřístupněn runtime, ve které je simulace procesu prováděna v synchronizaci s monitorováním reálného procesu. Uživatel může také nadefinovat horizont budoucnosti, do které bude proces simulován. Při změně operačních parametrů jsou simulovány budoucí stavy systému a na základě vyhodnocení predikované změny v čistotách separovaných složek a jejich výtěžku se může uživatel rozhodnout, zda navrhované změny ponechá, nebo zachová stávající počáteční parametry.

Aplikace byla vyvinuta pro Docker kontejner, což je standardní způsob integrace nových aplikací v prostředí Industrial Edge. Kromě přípravy samotného Docker image, bylo třeba nakonfigurovat další softwarové komponenty pro Industrial Edge, jako např S7-Connector pro připojení PLC, IE Databus pro komunikaci mezi aplikacemi na bázi MQTT a Data Service pro zprávu a ukládání dat získávaných z PLC a komunikací s ostatními aplikacemi pomocí REST API. Bylo také připraveno virtuální PLC z rodiny S7-1500, které simuluje reálný hardwarový komponent a odezvu procesních signálů odezvu SMB stanice.

Nástroj byl vyvíjen a testován ve spolupráci s experty společnosti Siemens. Po otestovaní aplikace ve virtualizovaném prostředí je možné konstatovat, že je systém připraven na nasazení na semi-průmyslové SMB stanici KCHS-SMB-8ND (VŠCHT, Praha), která je řízena pomocí kontroléru S7-400 (Siemens AG, DE).

5.3 Možnosti budoucího vývoje nástrojů

Hlubší analýza přesnosti a rychlosti optimalizačních algoritmů a implementace nových metod je zajímavým námětem pro práci do budoucna. Knihovna *scipy* nabízí několik dalších algoritmů, které jsou dobře implementované a snadno použitelné.

Paralelizace výpočtu je dalším zajímavým tématem. Implementovaná aplikace vytváří vlákno pro každý výpočet, což znamená, že při běhu malého množství výpočtů nejsou plně využity výpočetní zdroje. Použití více vláken a případně i více procesů pro jeden výpočet je vhodné v případě nutnosti výpočtu jedné optimalizace za co nejkratší dobu.

Vzhledem k tomu, že jsou používany optimalizační algoritmy, nabízí se přístup rozdělení stavového prostoru na stejně velké části a každému vláknu přiřadit jeden z výsledků každého vlákna vybrat ten nejlepší. Tento přístup však bude silně záviset na výběru optimalizačního algoritmu. Pro „brute force“ by toto řešení nejspíše fungovalo dobře, sofistikovanější algoritmy by však mohly dlouho hledat minimum na části stavového prostoru, která by se brzo vyloučila při výpočtu na jednom vlákně, což by mohlo vést ke zvýšení počtu iterací a snížení efektivity. Je možné rozdělit vláknům jednotlivé optimalizace druhé úrovně, což by mohlo vést k lepší efektivitě, náročnost výpočtu a doba běhu je však stále velmi nepředvídatelná a rozložení práce by nebylo optimální.

Důležitým tématem pro další vývoj je automatizace modelového prediktivního řízení. V této práci byla připravena možná platforma pro modelové prediktivní řízení, simulace procesu za po-

moci modelu a její porovnání s reálnými výstupy. Automatizované modelové prediktivní řízení však nebylo implementováno a je tedy na uživateli, aby na základě poskytnutých informací proces řídil. Při dalším vývoji nástroje by bylo vhodné implementovat algorimy, které by na základě získaných informací automaticky řídily daný proces. Přirozeným přístupem je periodicky provádět optimalizaci operačních parametrů v budoucím horizontu s cílem maximalizovat výtěžek a čistotu. Tento výpočet by musel být v závislosti na délce budoucího horizontu relativně rychlý, což jde ruku v ruce s hlubší analýzou optimalizačních algoritmů.

Pro obě aplikace byl implementován rovnovážný disperzní model chromatografie s lineární a nekompetetivní Langmuirovou isotermou. Tyto modely jsou široce využívané, nicméně mají svoje výhody a nevýhody. Lineární isoterma je jednoduchá a výpočet je proto velmi rychlý, ale zanedbává nasycení sorbentu a nedá se proto použít pro vyšší koncentrace. Langmuirova isoterma už lépe popisuje proces při vyšších koncentracích, komplikuje však výpočet a stále zanedbává například kompetici o aktivní místa sorbentu mezi složkami, která může být v některých případech důležitá. Je tedy vhodné implementovat více modelů. Při vývoji byl kladen velký důraz na modularitu, díky čemuž je aplikace připravená na rozšíření o další modely.

Platforma Siemens Industrial Edge umožnila snadné nasazení simulátoru. Nabízí řešení pro většinu běžných úkolů, které by uživatel mohl potřebovat, jako jsou konektory pro mnoho druhů komunikace nebo sběr a monitoring dat. Také umožňuje vytváření vlastních řešení pro specifické úkoly, jako je například modelové prediktivní řízení. Další výhodou je, že nabízená řešení používají široce používané technologie jako je Docker, REST nebo MQTT, což velmi usnadňuje integraci vlastních aplikací.

Industrial Edge je průmyslem ověřená technologie, která má velkou akceptaci mezi provozovateli průmyslových závodů. V této práci je na příkladu modelového prediktivního řízení dokázáno, že IE je vhodná technologie pro IT/OT integraci a propojení výzkumných a výrobních systémů. V budoucnu bude vznikat mnoho nových aplikací pro Industrial Edge platformu, což má potenciál vytvořit převrat v průmyslové automatizaci.

Příloha A

Seznam použitých symbolů

a, b	Substituované členy parciální diferenciální rovnice bez derivace
c	Koncentrace, g/ml
$\langle c \rangle$	Průměrná koncentrace za jeden cyklus, g/ml
h	Délka prostorového kroku, mm
k	Délka časového kroku, s
m	Hmotnost, g
q_m	Saturační konstanta sorbentu, g/L
q^*	Rovnovážná koncentrace v sorbentu, g/ml
t	Čas, s
t^*	Čas přepnutí, s
t_s	Čas nutný k dosažení ustáleného stavu, s
u	Aproximované řešení parciální diferenciální rovnice Crank-Nicolsonovou metodou
u_m	Rychlost toku, mm/s
x	Axiální koordináta, mm
x_r	Odražený bod Nelder-Meadova algoritmu
x_e	Expanzní bod Nelder-Meadova algoritmu
x_c	Zkrácený bod Nelder-Meadova algoritmu
y	Aproximované řešení parciální diferenciální rovnice Runge-Kuttovou metodou
A, B, C, D	Koeficienty asymetrické Gausovy křivky
D_{ax}	Axiální disperzní koeficient, mm^2/s
K_H, K_L	Adsorbční koeficienty
L	Délka kolony, mm

N_c Počet kolon

Pu Čistota,

Q Kapacita sorbentu, g/ml

T Čas experimentu, s

T^p Čas píku, s

$\bar{T}^p$ Průměr časů píku, s

$\dot{V}$ Objemový průtok, ml/h

Y Výtěžek,

$\alpha, \gamma, \rho, \sigma$ Koeficienty bodů Nelder-Meadova algoritmu

ε Celková porozita sorbentu

σ^r Časový posun „Retention Time Correction“

ξ Objektivní funkce „Retention Time Correction“

τ Čas nástřiku, s

Bibliografie

1. RODRIGUES, A. *Simulated Moving Bed Technology: Principles, Design and Process Applications*. Elsevier Science, 2015. ISBN 9780128020517. Dostupné také z: https://books.google.cz/books?id=qPGcBAAAQBAJ.
2. VOGEL, H.C.; TODARO, C.M. *Fermentation and Biochemical Engineering Handbook, 2nd Ed.: Principles, Process Design and Equipment*. Elsevier Science, 1996. ISBN 9780815517139. Dostupné také z: https://books.google.cz/books?id=qBfk8keNDbAC.
3. RAJENDRAN, Arvind; PAREDES, Galatea; MAZZOTTI, Marco. Simulated moving bed chromatography for the separation of enantiomers. *Journal of Chromatography A*. 2009, roč. 1216, č. 4, s. 709–738. ISSN 0021-9673. Dostupné z DOI: https://doi.org/10.1016/j.chroma.2008.10.075. Editors' Choice III.
4. ZHANG, Yongjin; FENG, Lihong; SEIDEL-MORGENSTERN, Andreas; BENNER, Peter. Accelerating optimization and uncertainty quantification of nonlinear SMB chromatography using reduced-order models. *Computers & Chemical Engineering*. 2016, roč. 96. Dostupné z DOI: 10.1016/j.compchemeng.2016.09.017.
5. CHARTON, Frédéric; NICOUD, Roger-Marc. Complete design of a simulated moving bed. *Journal of Chromatography A*. 1995, roč. 702, č. 1, s. 97–112. ISSN 0021-9673. Dostupné z DOI: https://doi.org/10.1016/0021-9673(94)01026-B. 1994 International Symposium on Preparative Chromatography.
6. ZHONG, Guoming; GUIOCHON, Georges. Simulated moving bed chromatography. Effects of axial dispersion and mass transfer under linear conditions. *Chemical Engineering Science*. 1997, roč. 52, č. 18, s. 3117–3132. ISSN 0009-2509. Dostupné z DOI: https://doi.org/10.1016/S0009-2509(97)00133-4.
7. DEVAULT, Don. The Theory of Chromatography. *Journal of the American Chemical Society*. 1943, roč. 65, č. 4, s. 532–540. Dostupné z DOI: 10.1021/ja01244a011.
8. GOLSHAN-SHIRAZI, S.; GUIOCHON, G. The Equilibrium-Dispersive Model of Chromatography. In: *Theoretical Advancement in Chromatography and Related Separation Techniques*. Ed. DONDI, Francesco; GUIOCHON, Georges. Dordrecht: Springer Netherlands, 1992, s. 35–59. ISBN 978-94-011-2686-1. Dostupné z DOI: 10.1007/978-94-011-2686-1_2.
9. BELLOT, J.C.; CONDORET, J.S. Modelling of liquid chromatography equilibria. *Process Biochemistry*. 1993, roč. 28, č. 6, s. 365–376. ISSN 1359-5113. Dostupné z DOI: https://doi.org/10.1016/0032-9592(93)80023-A.
10. PEIRÓ, Joaquim; SHERWIN, Spencer. *Handbook of Materials Modeling: Methods*. Dordrecht: Springer Netherlands, 2005. ISBN 978-1-4020-3286-8. Dostupné z DOI: 10.1007/978-1-4020-3286-8_127.

11. DRÁBEK, P.; HOLUBOVÁ, G. *Parciální diferenciální rovnice: úvod do klasické teorie*. Západočeská univerzita, Fakulta aplikovaných věd, 2001. ISBN 9788070827666. Dostupné také z: https://books.google.cz/books?id=7PEvAAAACAAJ.
12. KUBÍČEK, M.; DUBCOVÁ, M.; JANOVSKÁ, D. *Numerické metody a algoritmy*. Vysoká škola chemicko-technologická, 2005. ISBN 9788070805589. Dostupné také z: https://books.google.cz/books?id=wVJRAAAACAAJ.
13. ASCHER, Uri M.; RUUTH, Steven J.; WETTON, Brian T. R. Implicit-Explicit Methods for Time-Dependent Partial Differential Equations. *SIAM Journal on Numerical Analysis*. 1995, roč. 32, č. 3, s. 797–823. Dostupné z DOI: 10.1137/0732037.
14. MÍKA, S.; PŘIKRYL, P.; BRANDNER, M. *Speciální numerické metody: numerické metody řešení okrajovỳch úloh pro diferenciální rovnice*. Vydavatelskỳ servis, 2006. Texty z aplikované matematiky. ISBN 9788086843131. Dostupné také z: https://books.google.cz/books?id=nTt6tgAACAAJ.
15. TARANTOLA, A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial a Applied Mathematics, 2005. Other titles in applied mathematics. ISBN 9780898715729. Dostupné také z: https://books.google.cz/books?id=jKeHEWtzdKgC.
16. NAYAK, S. *Fundamentals of Optimization Techniques with Algorithms*. Elsevier Science, 2020. ISBN 9780128211267. Dostupné také z: https://books.google.cz/books?id=PU1BzQEACAAJ.
17. NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. *The Computer Journal*. 1965, roč. 7, č. 4, s. 308–313. ISSN 0010-4620. Dostupné z DOI: 10.1093/comjnl/7.4.308.
18. LEWIS, Robert Michael; TORCZON, Virginia; TROSSET, Michael W. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*. 2000, roč. 124, č. 1, s. 191–207. ISSN 0377-0427. Dostupné z DOI: https://doi.org/10.1016/S0377-0427(00)00423-4. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.
19. ENDRES, Stefan C; SANDROCK, Carl; FOCKE, Walter W. A simplicial homology algorithm for Lipschitz optimisation. *Journal of Global Optimization*. 2018, roč. 72, s. 181–217.
20. CROSS, Mark. The Role and Practice of Mathematical Modelling in Industry Today. In: 1994.
21. BENNETT, Stuart. The past of PID controllers. *Annual Reviews in Control*. 2001, roč. 25, s. 43–53.
22. NIKOLAOU, Michael. Model predictive controllers: A critical synthesis of theory and industrial needs. *Advances in Chemical Engineering*. 2001, roč. 26, s. 131–204.
23. BEHRENDT, Martin. *A basic working principle of Model Predictive Control*. 2009. Dostupné také z: https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg. [Online; accessed 03-May-2023].
24. TABAK, D. Digitalisation of control systems. *Computer-Aided Design*. 1971, roč. 3, č. 2, s. 13–18. ISSN 0010-4485. Dostupné z DOI: https://doi.org/10.1016/0010-4485(71)90063-7.
25. STOUFFER, Keith; FALCO, Joe; SCARFONE, Karen et al. Guide to industrial control systems (ICS) security. *NIST special publication*. 2011, roč. 800, č. 82, s. 16–16.

26. ALPHONSUS, Ephrem Ryan; ABDULLAH, Mohammad Omar. A review on the applications of programmable logic controllers (PLCs). *Renewable and Sustainable Energy Reviews*. 2016, roč. 60, s. 1185–1205. ISSN 1364-0321. Dostupné z DOI: https://doi.org/10.1016/j.rser.2016.01.025.

27. HANSSEN, Geir K; ONSHUS, Tor; JAATUN, Martin Gilje; MYKLEBUST, Thor; OTTERMO, Maria; LUNDTEIGEN, Mary Ann; NORWAY, Petroleum Safety Authority. Principles of digitalisation and IT-OT integration. 2021. Dostupné také z: https://www.ptil.no/globalassets/fagstoff/prosjektrapporter/ikt-sikkerhet/sintef---report---principles-of-digitalisation-and-it-ot-integration.pdf.

28. TEAM, Industrial Edge Documentation. *Industrial Edge Documentation* [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://docs.eu1.edge.siemens.cloud/.

29. MIRU, Gyorgy. *The siemens S7 communication - part 1 general structure*. 30-1-2016. Dostupné také z: http://gmiru.com/article/s7comm/.

30. BIENIASZ, Lesław K.; ØSTERBY, Ole; BRITZ, Dieter. Numerical stability of finite difference algorithms for electrochemical kinetic simulations: Matrix stability analysis of the classic explicit, fully implicit and Crank-Nicolson methods and typical problems involving mixed boundary conditions. *Computers & Chemistry*. 1995, roč. 19, č. 2, s. 121–136. ISSN 0097-8485. Dostupné z DOI: https://doi.org/10.1016/0097-8485(94)00054-I.

31. SHINTARO, Takenaga; YOSHIHIKO, Ozaki; MASAKI, Onishi. Practical initialization of the Nelder–Mead method for computationally expensive optimization problems. *Optimization Letters*. 2023, roč. 17, č. 2, s. 283–297. ISSN 1862-4480. Dostupné z DOI: 10.1007/s11590-022-01953-y.

32. ZOBEL, Steffen; HELLING, Christoph; DITZ, Reinhard; STRUBE, Jochen. Design and Operation of Continuous Countercurrent Chromatography in Biotechnological Production. *Industrial & Engineering Chemistry Research*. 2014, roč. 53, č. 22, s. 9169–9185. ISSN 0888-5885. Dostupné z DOI: 10.1021/ie403103c.

Obsah přílohy