VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

OPTIMALIZACE TEPLOTNÍHO POLE S FÁZOVOU PŘEMĚNOU

OPTIMIZATION OF THERMAL FIELD WITH PHASE CHANGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL PUSTĚJOVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. PAVEL POPELA, Ph.D.

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Michal Pustějovský

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Matematické inženýrství (3901T021)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace teplotního pole s fázovou přeměnou

v anglickém jazyce:

Optimization of Thermal Field with Phase Change

Stručná charakteristika problematiky úkolu:

Student si prohloubí znalosti problematiky modelů matematického programování se zaměřením na problematiku efektivní modifikace, transformace a implementace složených modelů. Téma práce navazuje na dlouhodobě rozvíjenou problematiku optimalizačních modelů zahrnujících omezení ve tvaru diferenciálních rovnic, řešených numerickými metodami ve spolupráci s energetickým ústavem.

Cíle diplomové práce:

1. Sestavení numerického modelu plynule odlévaného předlitku kruhového průřezu.
2. Následné vytvoření odvozeného zobecněného optimalizačního modelu.
3. Implementace a testování vybraných optimalizačních metod s důrazem na jejich efektivnost a možnou dekompozici.
4. Studium možností modifikace modelu s cílem jeho využití pro optimální řízení v reálných podmínkách.

Abstrakt
Práce se zabývá modelováním problému kontinuálního lití oceli. Tento proces výroby oceli získal nejen v České republice v posledních letech faktickou dominanci. V práci je řešen sochor s kruhovým průřezem, který bývá často neprávem opomíjen. Pro výpočet teplotního pole je nejprve vytvořen numerický model odlitku použitím diferenční metody při cylindrických souřadnicích. Následně jej využíváme při optimalizaci. Řešený optimalizační problém reprezentuje řízení soustavy při skokové změně licí rychlosti. Hlavním cílem práce je prozkoumat možnost paralelizace výpočtu jak numerického modelu, tak optimalizační úlohy pomocí prostorové dekompozice. K tomu byl použit nástroj stochastické optimalizace Progressive Hedging Algorithm.

Summary
This thesis deals with modelling of continuous casting of steel. This process of steel manufacturing has achieved dominant position not only in the Czech Republic but also worldwide. The solved casted bar cross-section shape is circular, because it is rarely studied in academical works nowadays. First part of thesis focuses on creating numerical model of thermal field, using finite difference method with cylindrical coordinates. This model is then employed in optimization part, which represents control problem of abrupt step change of casting speed. The main goal is to find out, whether the computation of numerical model and optimization both can be parallelized using spatial decomposition. To achieve that, Progressive Hedging Algorithm from the field of stochastic optimization has been used.

Klíčová slova
kontinuální lití oceli, diferenční metoda, cylindrická diskretizace, teplotní pole s fázovou přeměnou, prostorová dekompozice, paralelizace, stochastická optimalizace, progressive hedging

Keywords
continuous steel casting, difference method, cylindrical discretization, thermal field with phase change, spatial decomposition, paralellelization, stochastic optimization, progressive hedging

PUSTĚJOVSKÝ, M. *Optimalizace teplotního pole s fázovou přeměnou.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 62 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D..

Prohlašuji, že jsem diplomovou práci Optimalizace teplotního pole s fázovou přeměnou vypracoval samostatně pod vedením svého vedoucího RNDr. Pavla Popely, Ph.D. s použitím podkladů uvedených v seznamu použitých zdrojů.

Bc. Michal Pustějovský

Chtěl bych poděkovat svému vedoucímu RNDr. Pavlu Popelovi, Ph.D za neskutečně cenné a praktické rady při vypracovávání této práce. Dále bych rád poděkoval doc. Ing. Josefu Štetinovi, Ph.D. za pomoc při řešení problémů s technickou a technologickou částí práce, a také za čas, který mi věnoval.

Bc. Michal Pustějovský

Obsah

1 Úvod

Ocel patří již přes století mezi nejpoužívanější konstrukční materiály na světě. Její výroba prošla mnoha obměnami, od ruční ve středověku přes ingotování až po dnešní plně automatizovanou pomocí *kontinuálního odlévání*.

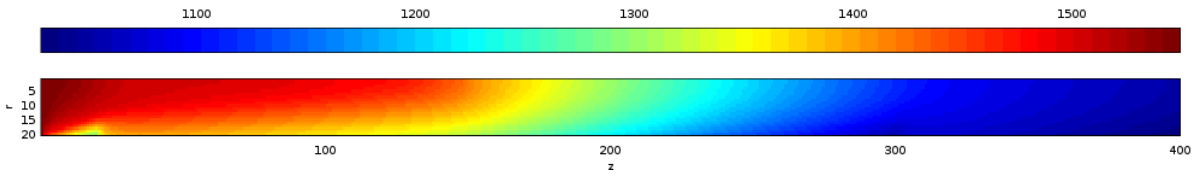
Tento proces v současnosti produkuje přes 90% světové oceli. Jedná se o výrobu polotovarů předem daného tvaru. Tekutá ocel vstupuje do tvarovacího zařízení (*krystalizátoru*), kde se vytvoří na okraji tenká kůra. Z tohoto zařízení se odlitek neustále vysouvá ven, do zóny chlazení. Zde je ostřikován chladicími tryskami, aby docházelo k optimálnímu tuhnutí stále tekutého středu. Schéma procesu lze nalézt na obrázku 2.6.

Na Energetickém ústavu FSI VUT v Brně se už po několik let touto problematikou zabývá skupina spolupracovníků doc. Ing. J. Štětiny, Ph.D. Tato skupina postupem času vyvinula řídicí systém pro odlitky obdélníkového průřezu [22, 19]. Ukázalo se, že pro použitelný model jsou nutné výpočetní sítě velkého rozsahu, které způsobují velkou výpočetní náročnost. Pro optimální řízení musí být systém ale schopen odpovídat nejlépe v reálném čase. Proto se nabízí otázka, jak tohoto docílit.

Jako nadějný přístup se ukázala paralelizace úlohy. Princip spočívá v rozdělení problému na několik dostatečně nezávislých částí, které se počítají simultánně na jednotlivých výpočetních jednotkách. Paralelizaci lze zavést několika způsoby. Ten, který byl vybrán pro tuto práci, se nazývá *prostorová dekompozice*. Stručně řečeno jde o rozdělení výpočetní oblasti na několik částí, které se počítají paralelně. Kvůli zajištění konvergence a zamezení skoků v řešení musíme dodat podmínky, které tyto části propojí.

Tato práce se zabývá aplikací prostorové dekompozice na problém skokové změny licí rychlosti využitím *Progressive Hedging Algorithmu (PHA)*, dekompozičního algoritmu stochastické optimalizace. Součástí zadání je vytvoření numerického modelu pro kruhový průřez odlitku, který na Energetickém ústavu řešen nebyl.

Nejprve se v kapitole 4 věnujeme numerickému řešení problému pomocí metody konečných diferencí. Přirozené se pro diskretizaci jeví použití cylindrických souřadnic. Sestavení modelu komplikuje složité chování materiálových vlastností. Ty jsou silně závislé na teplotě. Navíc tuto závislost nelze popsat analyticky či aproximovat křivkou dostatečně přesně a musíme používat experimentálně naměřená data. Další komplikace vznikly z důvodu přítomnosti fázových přeměn, které znemožňují použití implicitní metody (z důvodu výpočtu fázových přeměn *metodou entalpií*).



Obrázek 1.1: Vizualizace teplotního pole - spodní polovina podélného řezu

V další části (kapitole 6) se věnujeme problému optimálního řízení. Reprezentujeme jej úlohou nespojitého skoku parametru - licí rychlosti. Snažíme se najít takové rozložení intenzity chladicí soustavy, aby bylo nové teplotní pole co nejvíce podobné tomu před změnou. Rozdělíme odlitek na několik částí, z nichž každá má vlastní řiditelný okruh chlazení. Každou část počítáme nezávisle a snažíme se minimalizovat rozdíl mezi původní a novou teplotou na jednotlivých elementech.

Zde se ukázala velká slabina PHA, pomalá celková konvergence. Pro zrychlení výpočtu bylo nutné tento algoritmus upravit. Porovnání čisté paralelní implementace PHA, upraveného PHA a sériové implementace úlohy popsané tabulkou 6.1 je v následující tabulce 1.1.

Tabulka 1.1: Porovnání délky výpočtu modelů

	Paralelně	Počet iterací	Čas výpočtu
Upravený PHA	ano	16	68, 72 min
Sériový výpočet	ne	8	113, 95 min

Paralelizace výpočtů na více výpočetních jader pomocí *OpenMP* přinesla podstatné zrychlení celého algoritmu. Pomocí prostorové dekompozice naprogramované paralelně jsme poměrně rychle vyřešili výpočetně náročný problém optimálního řízení. Vedlejším produktem práce je zjištění možnosti použití prostorové dekompozice přímo pro výpočet numerického modelu. Nevýhodou prostorové dekompozice je vznik artefaktů na počátcích jednotlivých elementů zapříčiněný opatřeními vedoucími k urychlení konvergence.

2 Kontinuální odlévání oceli

Kontinuální lití oceli (zkráceně *kontilití*) je v současnosti nejpoužívanější technologický proces výroby oceli. Řadí se k poměrně mladým procesům. Jeho vznik datujeme někdy do 50. let 20. století, ovšem ve větší míře se začal využívat až v sedmdesátých a osmdesátých letech. V dnešní době se používá při výrobě zhruba 90% veškeré světové oceli [29].

Hlavním znakem kontilití je nepřetržitá výroba. Z nádoby s roztavenou ocelí se pomalu vysouvá odlévaný polotovar, který vlivem přirozeného ochlazování a také ostřikování vodními tryskami chladne. Po určité vzdálenosti už veškerá tekutá fáze ztuhne. Od této vzdálenosti můžeme odlitek uříznout a dále zpracovávat.

Ocel se vyrábí z železa, které je pro některé své vlastnosti (zejména křehkost) pro použití v průmyslu nevhodné. Při jeho průmyslové výrobě redukcí pomocí koksu a CO získáváme *surové železo*, bohužel stále velice křehké.

Tyto nevyhovující vlastnosti bývají způsobeny nečistotami, různými příměsemi a zejména příliš vysokým obsahem uhlíku. Ocel získáme, pokud koncentraci uhlíku v železe[1] snížíme *zkujňováním* pod $2,14\%$.

2.1 Historie výroby oceli

Prvním procesem, který dovoloval výrobu oceli v masovějším měřítku, je tzv. *bessemerování*. Tato technologie se jmenuje po svém zakladateli, Henrym Bessemerovi, který si ji nechal v roce 1856 patentovat. V principu jde o odstraňování uhlíku oxidaci na směs plynných oxidů. Toho se dosáhne v tzv. *konvertorech* profukováním taveniny vzduchem. Metoda se postupně vylepšovala přidáváním různých přísad, například vápence, nebo různorodou vyzdívkou konvertoru. Díky nim se dařilo zbavovat pořád většího množství příměsí a tedy zlepšovat vlastnosti oceli. Společně se zdokonalováním výrobního procesu se také snižovala cena oceli, která mezi lety 1890-1900 klesla až na sedminu původní hodnoty [8].

Na začátku poté byl tento proces dále vylepšen W. Siemensem použitím *Siemens-Martinovy* pece. Umožňuje snadnější regulaci složení oceli a její provoz (vytápění) je mnohem hospodárnější.

Kontinuální lití oceli nahrazuje přímé lití roztavené oceli z konvertoru (později krystalizátoru) do forem nebo ingotů.

Prvním zařízení, které připomínalo kontinuální lití, bylo zkonstruováno ve Spojených státech roce 1901 vlastníkem ocelárny Benjaminem Athou. Ovšem první opravdu rozšířené zařízení zkonstruoval v roce 1915 John. T. Rowley rovněž v USA. Podrobné popisy těchto strojů lze nalézt v [8].

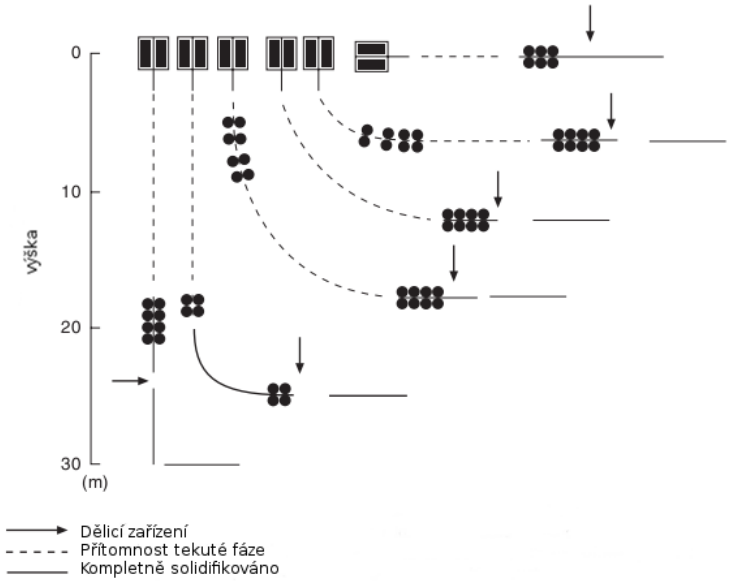
Až do začátku padesátých let bylo kontilití až na výjimky používáno pro neželezné kovy. U oceli totiž veškeré pokusy ztroskotávaly na příliš vysokém tření odlitku o stěny pece, které se dalo omezit pouze pomocí poměrně drahých technologií. Z těchto důvodů se v následujících letech vyvíjely technologie kontilití pouze pro speciální typy ocelí, kde potenciální větší výnos cenově vyvážil cenově náročnější zařízení.

[1] V technické praxi rozumíme pod pojmem *železo* pouze čisté železo bez obsahu uhlíku. Směs železa a uhlíku nazýváme podle koncentrace uhlíku vždy buď *litina* nebo ocel. Hraničním bodem je $2,14\%$. Nad touto hranicí se bavíme vždy o ocelích, pod ní o litinách.

2.2 Zařízení na kontinuální odlévání oceli

2.2.1 Typy

Podle své konstrukce se zařízení na kontinuální lití oceli (ZKO) dělí na vertikální, horizontální a radiální.



Obrázek 2.1: Jednotlivá konstrukční provedení ZKO, přeloženo [8]

Vertikální typ je nejstarší. K odlévání dochází samovolně vlivem gravitace, která také zaručuje symetričnost odlitku podle vertikální osy. Nicméně obrovskou nevýhodou je omezená délka odlévaných polotovarů daná výškou zařízení.

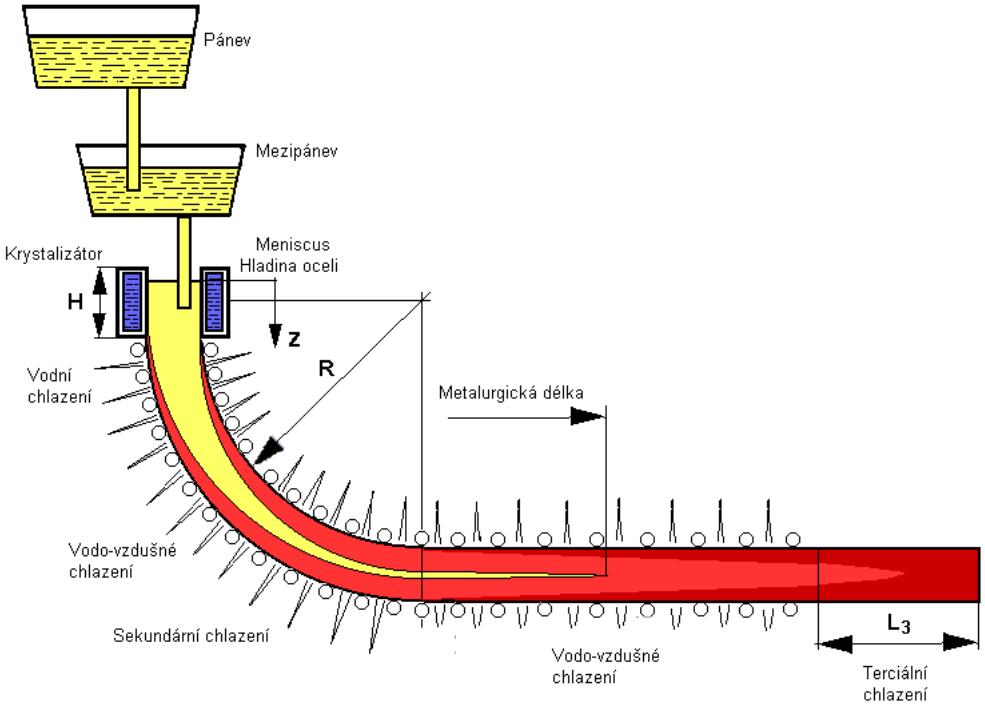
Horizontální typ se používá jen v omezené míře a zejména u neželezných kovů. Obrovským problémem je totiž potenciální vznik nerovnoměrného složení a tvarové asymetrie vlivem gravitace.

Nyní nejčastěji používaným typem je radiální zařízení. Jedná se o kombinaci předchozích dvou typů. Část u nádoby s roztaveným železem bývá ve vertikálním směru, což umožňuje rovnoměrné tuhnutí a symetrické složení. Od určité vzdálenosti se začíná postupně zakřivovat do horizontálního směru. Zde už se vyskytuje minimum tekuté fáze a tedy nehrozí nepravidelnosti ve složení a tvaru vlivem gravitace. V horizontální části už nejsme tak omezení v maximální délce odlitku, což zvyšuje produktivitu.

Dalším parametrem, který je nutné při rozlišování typů ZKO brát v úvahu, je tvar odlévané součásti. Odlitky s kruhovým a čtvercovým profilem nazýváme sochor (anglicky *billet*), s obdélníkovým profilem brama (*slab*). Existují další speciální typy, například kolejnicové či čtvercové profily.

2.2.2 Konstrukce

Konstrukce ZKO je nesmírně komplikovaná. Pro přehled si zde uvedeme pouze z hlediska funkce nejzásadnější části, které ilustrujeme na nejpoužívanějším radiálním ZKO.



Obrázek 2.2: Nejdůležitější části ZKO [29]

Vstupem do ZKO je roztavená ocel. Z tzv. kolébek se lije do *pánve (laddle)*, které slouží jako její zásobník. Často se zde také zbavuje nečistot, k čemuž slouží speciální vyzdívky [15].

Z pánve ocel pokračuje do *nálevky (mezipánev, angl. tundish)* Hlavním účelem mezipánve je zajistit rovnoměrný tok oceli do dalších částí zařízení např. v případě poruchy dávkování oceli z pánve. Obě tyto části ZKO bývají vyhřívány, z toho důvodu, aby se roztavená ocel držela na optimální teplotě. Konstrukce pánví má zásadní vliv na chemické složení oceli.

Následuje *krystalizátor (mold)*, jenž slouží k vytvarování odlévaného polotovaru. Jeho hlavní úkol spočívá v ochlazování svrchních části sochoru hluboko pod teplotu likvidu, aby se vytvořila *skořepina*[2]. Konstrukce bývá různá, zejména se ovšem jedná o materiály velice dobře odvádějící teplo (měď) [29]. Velice důležitou vlastností krystalizátoru je schopnost odvádět teplo a pomocí mazání (v nynější praxi zejména mazacími prášky) či jemných vibrací minimalizovat tření nebo vznik nárůstků. Zásadním způsobem ovlivňuje kvalitu povrchu odlitku. O krystalizátoru se také hovoří jako o oblasti primárního chlazení.

V krystalizátoru se nám začíná formovat předlitek. V dalším úseku už jej ale opouští a dostává se vně zařízení. Zde vstupuje do *sekundární zóny chlazení*. Ta je tvořena soustavou vodních chladicích trysek a vodicích válců. Jejich účelem je zchladit odlitek do takové míry, že v místě řezání už nebude jádro tekuté[3]. Na druhou stranu nesmí být chlazení příliš intenzivní, neboť v tom případě dochází ke vzniku prasklin a trhlin. Sestava trysek

[2]Skořepina (kůra)je nepravidelná tenká vrstva na povrchu předlitku, která vzniká v krystalizátoru. Jedná se vlastně o jakousi skořápku, která obklopuje stále tekutý vnitřek sochoru.

[3]Vzdálenost, do které je jádro tekuté, se nazývá *metalurgická délka*

a válců bývá velice komplikovaná. Typicky se v jednom ZKO vyskytuje více nezávislých okruhů s různými typy trysek [29].

Rozlišujeme také *terciální zónu chlazení*. Zde už odlitek chladne pouze přirozenou konvekcí a radiací.

Dále následuje *dělicí zařízení*, které oddělí již bezpečně ztuhlý odlitek. Ten je následně pomocí dopravníků přesunut k dochladnutí.

Dostatečně ochlazené odlitky poté putují k označení a do skladu, kde čekají na prodej nebo případné další zpracování.

2.3 Parametry kontinuálního lití

Z hlediska optimalizace procesu kontinuálního odlévání oceli je důležité znát parametry, kterými tento proces můžeme ovlivňovat. Mezi tyto patří

- Licí rychlost,
- Licí teplota,
- Chladicí systém.

Licí rychlostí rozumíme rychlost pohybu předlitku v oblasti sekundárního chlazení, nejčastěji měřenou přímo u krystalizátoru. Má zásadní vliv na *metalurgickou délku*, tedy vzdálenost, do které je jádro sochoru či bramy tekuté. Přirozeně chceme z hlediska produktivity co největší možnou licí rychlost. Vzhledem k metalurgické délce jsme omezeni konstrukcí (délkou) ZKO. Licí rychlost také zásadně ovlivňuje kvalitu předlitku, příliš rychlé tuhnutí vede ke vzniku prasklin. Pro různé typy předlitků a materiálů se výrazně liší, obecně se pro oceli pohybuje v rozmezí 2-8 m/min. U sochorů s kruhovým průřezem dosahujeme maximálně 2-3 m/min[4].

Licí teplota je teplota tekuté oceli, která vstupuje do krystalizátoru. Logicky je nutné, aby byla nad horní teplotou fázovou přeměny (teplota likvidu)[5].

Chladicí systém tvoří soustava chladicích trysek. Doplňuje jej několik vodicích válců, jejichž primárním účelem je sice uchycení předlitku, nicméně vedením také významně odvádějí teplo. Chladicí soustava naprosto významně ovlivňuje celkovou kvalitu výrobku a proto je její správný návrh a zejména řízení snad tím nejdůležitějším v ZKO. Spolu s licí rychlostí tvoří také nejvýznamnější regulační prvek. Řízením průtoku chladicí kapaliny v tryskách, nebo alespoň jejich spínáním můžeme významně ovlivnit teplotní gradienty a metalurgickou délku odlitku. Jednotlivé trysky dělíme na vodní a vodovzdušné. Ty se liší dodatečným přívodem stlačeného vzduchu, který umožňuje rovnoměrnější rozložení chladicího účinku.

Činnost ZKO lze popsat spoustou dalších parametrů, které ale nejsou pro účely této práce důležité.

2.4 Vady předlitku

Jakost předlitku můžeme rozdělit na čtyři kategorie

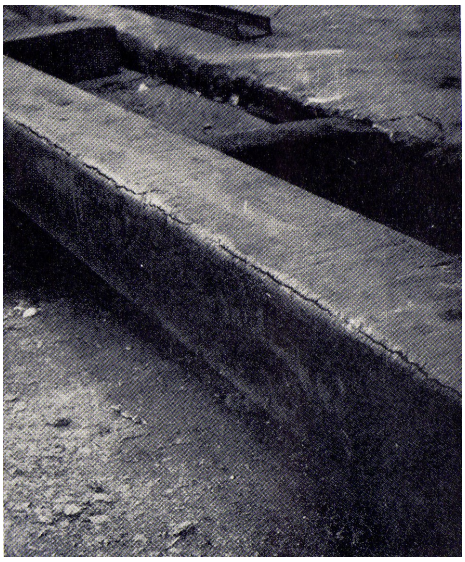
[4] U standardních ocelí

[5] Teplota tuhnutí a tání není u oceli stejná. Existuje jakési teplotní rozmezí krystalizace. Teplota začátku tuhnutí kapalné fáze se nazývá *teplota likvidu* a teplota tání pevné fáze *teplota solidu*

- Správné chemické složení,
- Jakost povrchu,
- Absence vad uvnitř předlitku,
- Geometrické vady.

Správné chemické složení ovlivňuje zejména tekutá ocel vstupující do pánve. V té lze ještě navázáním na její vyzdívku mírným způsobem vylepšit složení oceli. V případě přítomnosti nadlimitních koncentrací nežádoucích příměsí může mít ocel nevhodné vlastnosti (zejména zvýšená křehkost a únosnost). Krystalizací mohou v tomto případě vznikat také ložiska větší koncentrace dané příměsi a kvůli tomu můžeme skončit s výrazně nehomogenním materiálem.

Jakost povrchu určuje zejména kvalitní konstrukce krystalizátoru. Pokud není zajištěno dostatečné mazání mazacím práškem či popílkem, může docházet k ulpívání materiálu na stěně krystalizátoru. Důsledkem jsou příčné praskliny na povrchu odlitku. Dalším typem vady jsou propady (v řádech milimetrů).Jedná se o vady místní, většinou centimetrového rozsahu a nepravidelného tvaru, které způsobuje nedostatečné mazání či extrémní odchylka chladicího systému v oblasti sekundárního chlazení (např. vadná tryska). Spoustu rozličných vad bývá způsobena byť jen lokálním protrhnutím *skořepina*. Může dojít k vylití tekutého materiálu na povrch, které způsobí vznik nárustku. Závažnějším problémem ale je, pokud se ztuhlá část ulomí a zůstane v krystalizátoru, kde může způsobit rýhy po celém povrchu předlitku, což způsobí neopravitelnou chybu a často také zastavení celé linky. Při další variantě se ulomené kusy postupně z krystalizátoru samovolně uvolňují a ulpívají na povrchu předlitku.



Obrázek 2.3: Podélná trhlina v rohu předlitku způsobená chybným návrhem chladicí soustavy [5]

Obrázek 2.4: Podélná trhlina způsobená konkávností a chybnou geometrií krystalizátoru [5]

Vnitřní vady předlitku bývají způsobeny prakticky výhradně pouze buď nevhodným chemickým složením anebo příliš intenzivním nebo nehomogenním chlazením. V prvním případě se jedná pouze o nepravidelné drobné trhliny vyskytující se v celém objemu předlitku. V tom druhém mluvíme o příčných či podélných trhlinách (nebo jejich soustavách), vznikajících v oblastech s výskytem příliš vysokých teplotních gradientů.

Obrázek 2.5: Hvězdicové trhliny vzniklé z důvodu příliš intenzivního chlazení [5]

Tvarové vady jsou ze všech zde uvedených nejméně závažné. Třískovým obráběním bývají velice lehce napravitelné. Kritickými jsou pouze u tvarově složitějších průřezů, např. kolejnicových či nosníkových. Příčinou těchto defektů bývá nevhodná konstrukce ZKO (nevhodná rozteč či poloha vodicích válců) nebo nevhodné rozmístění prvků chladicího systému. Tyto chyby pak způsobují průhyby předlitku ve směru licí rychlosti, změnu průřezu (z kruhového na elipsovitý, z obdélníkového na lichoběžníkový). Vážnějšími projevy jsou konkávnost (konvexnost). Samy o sobě problémy nezpůsobují, nicméně může je doprovázet vznik trhlin uvnitř předlitku.

V reálném provozu se samozřejmě vyskytují zejména kombinace více druhů vad. Často se také stává, že jedna vada způsobuje druhou (například konkávnost povrchové trhliny, viz 2.4).

2.5 Shrnutí řešeného problému

Ústav energetiky Fakulty strojního inženýrství VUT v Brně se dlouhodobě zabývá problémem optimální řízení kontinuálního lití oceli. Na toto téma bylo vypsáno a úspěšně obhájeno několik diplomových a dizertačních prací [18, 19, 22]. V těchto pracích se řešila zejména úloha kontinuálního lití oceli pro čtvercový, případně obdélníkový profil odlévaného polotovar). Dalším zkoumaným parametrem byl typ lití oceli. Zkoumalo se, zdali je výhodnější použít přímou (vertikální) nebo radiální soustavu. Tímto se zabývala zejména dizertační práce Ing. Tomáše Maudera, Ph.D. [22].

Konečným cílem je získat řídicí software, který dokáže v reálném čase řídit rychlost posuvu sochoru tak, aby byla získaná ocel co nejkvalitnější. Tedy aby byly při jejím ochlazování teplotní gradienty co nejmenší. Na druhou stranu je také důležité, aby se sochor odléval dostatečně rychle z důvodu ekonomické efektivity provozu. Zásadními kritérii pro kvalitu řídícího softwaru jsou tedy

- Výpočet v reálném čase
- Schopnost optimalizovat licí rychlost vzhledem k teplotnímu poli
- Dostatečně přesný model kontilití

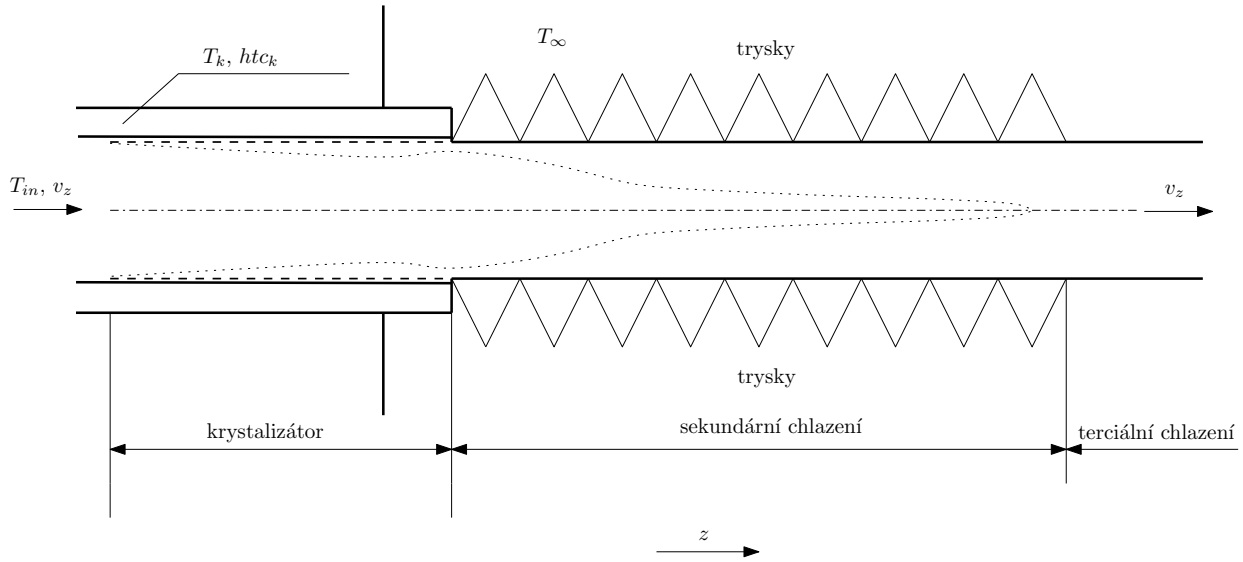
Mým úkolem je

- Sestavit numerický model pro sochor kruhového průřezu při horizontálním kontilití,
- Prozkoumat, zda by se dal pro numerický výpočet a následnou optimalizaci použít dekompoziční model z oblasti stochastické vícestupňové optimalizace,
- Provést analýzu výsledků a rozhodnout, zda je tento přístup v dané úloze z hlediska výpočetní rychlosti vhodný či nikoliv.

Řešenou úlohu lze nejlépe popsat na schématu 2.6.

Z nádoby s roztaveným železem (pánví) o teplotě T_{in}, se rychlostí v_z neustále vysunuje sochor o kruhovém průřezu (tyč). Jeho teplota se vlivem samovolného ochlazování vlivem nižší okolní teploty T_∞ a umělého vodního chlazení se vzrůstající vzdáleností od krystalizátoru snižuje. Nejvíce se sochor ochlazuje na povrchu, v ose tyče teplota klesá jen pomalu a materiál zůstává do určité vzdáleností tekutý (teplota solidu na obrázku 2.6 naznačena tečkovanou čarou). Tato oblast má za krystalizátorem v ideálním případě tvar kužele. Vodní chlazení je realizováno soustavou trysek s proměnlivou intenzitou chlazení. Pro jednoduchost uvažujme, že trysky mají rovnoměrné rozložení chladicího účinku a jsou rozmístěny symetricky podél osy válcového sochoru. V krystalizátoru se nacházejí desky chlazené vodou na teplotu T_k a koeficient přestupu tepla v krystalizátoru označme htc_k[6].

[6]Budeme předpokládat, že htc_k je konstantní. Vymodelovat reálné chování je velice technicky náročné a nad rámec této práce. Lze nalézt například v [29, 22]



Obrázek 2.6: Schéma kontilití

Vzhledem k tomu, že celé problém je úloha optimálního řízení, uvažujeme tzv. *výpočetní okno*. Nezajímá nás tedy jedna fyzická část sochoru, ale oblast v prostoru, do které materiál na jedné (zde levé) straně neustále vchází. Na protější straně zase vystupuje. Úlohu nám zjednoduší fakt, že ji lze chápat jako ustálený, *stacionární* stav, tedy že licí rychlost a průběh teploty v daném místě prostoru jsou konstantní. Velikost výpočetního okna ve směru licí rychlosti určíme na základě výsledků. Důležité je, aby se na jeho konci už teploty nijak dramaticky neměnily. V opačném případě bychom dostali nepřesné výsledky. Více na toto téma v kapitole 4 zabývající se numerickým modelem.

Na úloze nás nejvíce zajímají maximální hodnoty teplotních gradientů [8]. Je zřejmé, že ty budou přímo na povrchu, kde je ochlazování nejintenzivnější. Otázka polohy v podélném smyslu už tak jednoduchá není. Nabízejí se ovšem zjevná místa, kde lze tyto maximální gradienty nalézt. Mezi kandidáty patří určitě začátek sochoru, tedy místo přímo u krystalizátoru, nebo oblasti rozhraní vodního a samovolného chlazení. Přesná poloha záleží na nastavení systému trysek. Pokud budou chladit velice intenzivně, mohou svým efektem překonat velikost gradientů u počátku tyče.

3 Matematický model

Problém kontinuálního lití oceli popisuje tzv. *Kirchhoffova rovnice* [19, s. 32], [22]. Jedná se o parciální diferenciální rovnici druhého řádu. Úloha vedení tepla patří mezi parabolické úlohy [9]. Pro kartézské souřadnice (x, y, z) má tvar

$$\frac{\partial}{\partial \tau}(\rho c T) = k(T)\Delta T + \frac{\partial}{\partial \tau}(v_z \rho c T) + \dot{q}, \tag{3.1}$$

kde Δ značí Laplaceův operátor, funkci $T(x, y, z, \tau)$ chápeme jako teplotu závislou na poloze a čase, $\dot{q}$ označíme vývin tepla vlivem skupenských přeměn, ρ [kg m^{-3}] hustotu, c [Jkg^{-1}K^{-1}] měrnou tepelnou kapacitu a k [Wm^{-1}K^{-1}] tepelnou vodivost. Tyto materiálové vlastnosti musíme vzhledem k velkému rozpětí teplot a fázovým přeměnám uvažovat jako závislé na teplotě. Materiál navíc považujeme za homogenní, tedy materiálové vlastnosti můžeme „vytknout“ před derivace.

Rovnice nám říká, že změna teploty v čase je rovna součtu přeneseného tepla z okolí, tepla dodaného materiálem z mezipánve vlivem posuvné rychlosti v_z a tepla ze skupenských přeměn. Mějme na paměti znaménkovou konvenci. Pro dodané teplo používáme kladnou hodnotu a pro odebrané zápornou.

V naší úloze je nutné tuto rovnici transformovat do cylindrických souřadnic (r, φ, z). S využitím výše popsaných zjednodušení a [19] pak dostaneme

$$\rho c \frac{\partial T}{\partial \tau} = k\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\frac{\partial T}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\frac{\partial T}{\partial z}\right)\right] + v_z \rho c \frac{\partial T}{\partial z}. \tag{3.2}$$

3.1 Počáteční, okrajové podmínky

Počáteční a okrajové podmínky nám v prostoru (r, φ, z, τ) umožňuje popsat oblasti, které známe a díky kterým máme výpočet kde začít.

Pro vyřešení rovnice je nutná počáteční podmínka, která charakterizuje stav tělesa na začátku výpočtu, tedy pro $\tau = 0$. Úlohu kontinuálního lití charakterizuje počáteční podmínka

$$T(r, \varphi, z, 0) = T_{in}, \tag{3.3}$$

kde T_{in} je teplota tekuté oceli v mezipánvi.

Okrajové podmínky omezují chování popisovaných veličin na okrajích tělesa. V úloze kontinuálního lití připadají v úvahu zejména následující podmínky:

- I. druhu (Dirichletova) - předepisuje hodnotu teploty v bodě $T(\cdot, \tau)$,
- II. druhu (Neumannova) - předepisuje derivaci veličiny. V tomto případě se jedná o hustotu tepelného toku $\frac{\partial T}{\partial z} = -\frac{\dot{q}_z}{k}$. Pro speciální případ $\dot{q}_z = 0$ dostaneme $\frac{\partial T}{\partial z} = 0$.

Pro povrch sochoru $T(R, \cdot, \cdot)$ použijeme okrajovou podmínku druhého druhu, tedy předepíšeme

$$\frac{\partial T}{\partial r} = -\frac{\dot{q}_z}{k}.$$

Povrch tyče je ochlazován třemi typickými způsoby přenosu tepla: vedením (advekcí), prouděním (konvekcí) a zářením (radiací). Vzhledem k tomu, že ocel na povrchu sochoru

je už dostatečně tuhá, tak se zde proudění nevyskytuje. Intenzitu chlazení nám popisuje *koeficient přestupu tepla (htc; heat transfer coefficient)*, který je pro každý způsob ochlazování jiný. Označme htc_n a htc_r po řadě koeficienty přestupu tepla pro vedení a záření. Jejich hodnoty jsou dány vztahy [29, 15]

$$htc_n = 0.84(T(R,\varphi,z) - T_{inf})^{\frac{1}{3}} + htc_{cooling},$$

$$htc_r = \epsilon\sigma\left((T(R,\varphi,z) + 273,15)^2 + T_{inf}^2\right)\left((T(R,\varphi,z) + 273,15) + T_{inf}\right),$$

kde $htc_{cooling}$ je koeficient přestupu tepla reprezentující chladicí účinek vodních trysek, T_{inf} teplota okolí, $T(R,\varphi,z)$ povrchová teplota, ϵ relativní emisivita a σ Stefan-Boltzmannova konstanta ($5.87 \cdot 10^{-8}$ W $\cdot$ m^{-2} $\cdot$ K^{-1}). Emisivitu dostaneme pomocí regresního modelu [15][1]

$$\epsilon = \frac{0.85}{(1 + \exp(42.68 - 0.02682T_R))^{0.0115}},$$

kde T_R dosazujeme v Kelvinech. Tepelný tok ze sochoru do okolí je pak roven

$$\dot{q}_r = (htc_n + htc_r)(T_{inf} - T(R,\varphi,z)). \tag{3.4}$$

V případě části uvnitř krystalizátoru se bude situace trochu lišit. Přesná reprezentace tepla odebraného z bramy je poměrně složitá. Nicméně pro řešenou úlohu postačí aproximace pomocí

$$\dot{q}_r = (htc_k)(T_k - T(R,\varphi,z)), \tag{3.5}$$

kde $T_k = 80°C$ je teplota desek krystalizátoru a $htc_k = 2000$ Wm^{-2}K^{-1}.

Na konci sochoru uvažujeme volné navázání námi už nezkoumaného objemu (tzv. *volný konec*. Předpokládáme, že v těchto místech jsou už teplotní změny zanedbatelné. Tuto skutečnost nám reprezentuje okrajová podmínka

$$\frac{\partial T}{\partial z} = -\frac{\dot{q}}{k} = 0. \tag{3.6}$$

3.2 Metoda entalpií

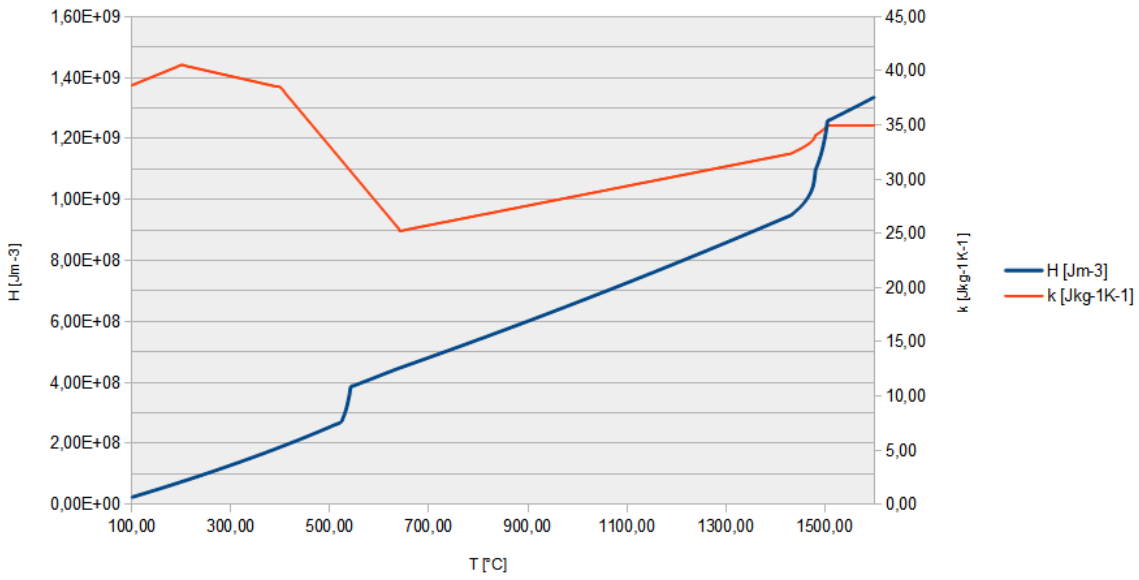
Nyní musíme v rovnici (3.1) popsat člen $\dot{q}$, který značí vývin tepla vlivem skupenských přeměn. K matematicko-fyzikálnímu popisu fázové přeměny se používá více metod [22, 19]. Pro úlohu kontinuálního lití se používá zejména tzv. *metoda entalpií*. Principem metody je převedení rovnice (3.1) tak, abychom místo s teplotou počítali s entalpií. To nám umožní velice zjednodušit výpočet. Entalpie v závislosti na teplotě nebo dodaném teple je totiž v kontextu této úlohy ryze rostoucí funkce. Pro teplotu toto naopak platit nemusí (3.1)[2].

Jednoduchým příkladem jsou veškeré skupenské přeměny. Například pokud dodáváme látce v pevném stavu blízko bodu tání teplo, začne se toto teplo spotřebovávat na změnu skupenství a teplota po nějakou dobu nebude růst (a v jistých případech může i klesat).

Tato metoda nám ale přináší jedno zásadní omezení. Prakticky nám znemožňuje použít jakoukoli implicitní numerickou metodu, protože by zásadním způsobem zkomplikovala sestavování matice soustavy. Jednotlivé prvky této matice prakticky nelze vyjádřit. Proto

[1] V citovaném textu se vzorec objevuje s chybou. Zde už je v pořádku.

[2] V grafu jsou znázorněny pouze závislosti $H(T)$ a $k(T)$. Je to z toho důvodu, že při výpočtu nebudeme c a ρ potřebovat.



Obrázek 3.1: Závislost entalpie H a tepelné vodivosti k na teplotě T

se k řešení takto složitých úloh pomocí metody konečných diferencí používá výhradně explicitních metod.

Aplikací metody entalpií dostaneme pro kartézský souřadný systém upravenou rovnici [22]

$$\frac{\partial H}{\partial \tau} = k(T)\Delta T + v_z \frac{\partial H}{\partial z}, \tag{3.7}$$

kde hlavní proměnou je objemová entalpie H [Jm^{-3}]. Teplotu zpětně dopočítáme pomocí vztahu

$$H = \rho c T \Rightarrow T = \frac{H}{\rho c},$$

kde vždy $\rho > 0$ a $c > 0$. Pro naši úlohu získáme přepisem rovnice (3.2) pomocí metody entalpií následující vztah

$$\frac{\partial H}{\partial \tau} = k\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\frac{\partial T}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\frac{\partial T}{\partial z}\right)\right] + v_z\frac{\partial H}{\partial z}. \tag{3.8}$$

3.3 Materiálové vlastnosti

Jak už bylo zmíněno výše, materiálové vlastnosti jsou silně závislé na teplotě. Tato vlastnost úlohy dosti komplikuje nalezení řešení úlohy. V rámci výzkumu na Ústavu energetiky FSI VUT se zkoumala možnost aproximace průběhů jednotlivých veličin pomocí polynomů. Naneštěstí uspokojivé aproximace byly nalezeny až u polynomů řádu vyššího než dvacet pět, což toto řešení činí poněkud nepraktickým.

Řešením se nakonec ukázala být experimentálně naměřená data. Vzorek osazený čidly pro měření jednotlivých vlastností se postupně zahříval v peci a po deseti stupních Celsia se odečítaly hodnoty měřicích přístrojů. Výsledky se zanesly do tabulky, která se nyní používá jednak k získání hodnot materiálových vlastností pro danou teplotu, tak k přepočtu entalpií na teploty a naopak. Postupem času se tyto experimenty pro různé typy

ocelí zanesly do databáze a začal vznikat software, který je schopen při zadání čísla oceli hodnoty příslušných materiálových vlastností vracet. Na Energetickém ústavu se k tomuto používá finský software IDS [22, 19] Mezi naměřenými hodnotami je nutno interpolovat. Vzhledem k nutnosti častého přístupu k hodnotám v tabulce je vhodné interpolovat veličiny pro teploty mezi jednotlivými kroky experimentu dopředu. Jako optimální varianta byla zvolena interpolace po jednom stupni Celsia. Vzhledem k malým změnám hodnot veličin v tak malém rozsahu se tento interval ukázal jako dostatečně malý.

Navíc nám tato interpolace umožní snížit výrazně počet operací. Pokud tabulku budeme chápat jako matici, kde sloupce tvoří jednotlivé veličiny, a interpolaci pro teploty provedeme od jednoho, příp. nula stupňů Celsia, můžeme hodnotu teploty chápat jako řádkový index a nemusíme tedy používat výpočetně náročné vyhledávací algoritmy.

4 Numerický model

Tuto úlohu nelze řešit analytickými metodami. Problémové je zejména chování materiálových charakteristik, které jsou silně závislé na teplotě. Pro spoustu z nich ale ani nemáme tuto závislost analyticky určenou. Veškeré pokusy o aproximaci nebo interpolaci těchto vlastností, a tím umožnění použití analytických metod, buď selhaly, nebo byly velice nepraktické. Další problém přestavují poměrně složité okrajové podmínky. Jejich nespojitost vylučuje získání klasického řešení. Ve spoustě případů ale můžeme nespojitost vyřešit pomoci tzv. *slabého řešení* [21, 9]. Nicméně ani v případě jeho použití nás kvůli silné závislosti materiálových charakteristik na teplotě nezbaví nutnosti řešení úlohy pomocí numerických metod.

4.1 Numerické metody

Pro řešení modelu lze použít několik numerických metod, které zde budou v krátkosti představeny.

Metoda konečných objemů

Anglicky známá jako *Finite Volumes Method (FVM)*. Tato metoda se používá nejčastěji v hydromechanice k vyřešení úloh proudění. Jedná se o síťovou metodu, objem řešeného tělesa se rozdělí na elementy. Prakticky výhradně se používají tzv. *primární* a *sekundární* sítě. Primární síť pokryje těleso nejčastěji soustavou trojúhelníků (resp. čtyřstěnů). Jejich vrcholy hrají roli středů sekundárních elementů (tzv. buněk). Vrcholy těchto buněk jsou tvořeny těžišti primárních elementů. Pro dvourozměrnou úlohu a tedy trojúhelníkovou primární síť dostaneme šestiúhelníkovou sekundární síť. Poté se požaduje, aby rovnice (3.7) platila pro každou tuto buňku.

Jednou z důležitých vlastností této metody je princip *slabých řešení*. Na rozdíl od těch klasických nám dovolují popsat i nespojité okrajové podmínky [21].

Dalším význačným konceptem metody jsou tzv. *numerické toky*, které nám umožnují vyjádřit v tomto případě tepelné toky v jednotlivých směrech. Existuje více typů numerických toků. Volba správného toku je velice důležitá a ne vždy triviální. Kvůli tomuto je v případě použití metody konečných objemů důležité mít k dispozici patřičnou testovací úlohu. Tato metoda je dopodrobna rozebírána v [21].

Galerkinovy metody

Nejznámějším zástupcem této skupiny metod je *Metoda konečných prvků (MKP; Finite Elements Method (FEM)*). Princip je podobný jako u metody konečných objemů. Zavádí se ale pouze primární diskretizace. Navíc také vynásobíme rovnici tzv. *testovací funkcí* a požadujeme, aby byla takto upravená rovnice splněna na každém elementu. Podobně jako konečné objemy pracují Galerkinovy metody se slabými řešeními.

K rozvoji FEM významnou měrou přispěl také brněnský rodák prof. RNDr. Miloš Zlámal, DrSc [13].

Oblast použití těchto metod je poměrně široká, viz [9]. Poměrně dominantní postavení si vybudovaly v mechanice, pevnostních a tepelných výpočtech nebo akustice. Často se také používají při výpočtech proudění kapalin a plynů.

Metoda kontrolních objemů

Tato síťová metoda je odvozena z diferenční metody, tedy derivace v rovnici (3.7) nahradíme konečnými diferencemi. Každému bodu diskretizace se ale navíc přiřazuje tzv. *kontrolní objem*, na který potom klademe podmínku, že změna teploty v daném objemu je úměrná tzv. tepelné bilanci, tedy (měrným) teplům podle jednotlivých souřadnic (φ, r, z) a změně vnitřního tepla (např. přeměna skupenství) [19, s. 38].

S touto metodou jsou na Energetickém ústavu bohaté zkušenosti [19]. Je ověřeno, že dává v poměru přesnosti a rychlosti výpočtu velice dobré výsledky. Nevýhodou je poměrně složité použití v případě komplikovanějších okrajových podmínek (různá velikost kontrolních objemů, více druhů okrajových podmínek na hranicích jednoho objemu apod.).

Metoda konečných diferencí

Známa také pod zkratkou *FDM (Finite Difference Method)* nebo jako *metoda sítí*. Už podle posledního názvu je jasné, že základem je dělení výpočetní oblasti nebo domény. Metoda pracuje s děleními ekvidistantními i obecnými.

Hlavní princip metody spočívá v náhradě derivací v jednotlivých bodech sítě tzv. *konečnými diferencemi* (nebo také *diferenčním podílem*) [9, s. 35-37]. Tato náhrada je odvozena z Taylorova rozvoje. Uvedeme zde příklad pro vyjádření druhé derivace.

Vezmeme-li funkci $u = u(x)$, interval $\langle a, b\rangle$, jeho rovnoměrné dělení označíme $x_1, \ldots, x_n$ s krokem $h = (x_i - x_{i-1})$, $i = 1, \ldots, n$ takové, že $x_1 = a$ a $x_n = b$. Dále $u(x_i)$ označíme pro přehlednost jako u_i , dostaneme pro první derivaci[1]

$$u'(x) = \frac{\mathrm{d}u(x)}{\mathrm{d}x} \approx \frac{(u_{i+1} - u_i)}{h}.$$

Dále zavedeme výraz $x_{i\pm\frac{1}{2}} = x_i \pm \frac{1}{2}h$. Pak zřejmě

$$u'_{i-\frac{1}{2}} = \frac{u_i - u_{i-1}}{h} \text{ a } u'_{i+\frac{1}{2}} = \frac{u_{i+1} - u_i}{h},$$

a nakonec dalším počítáním dostaneme tzv. *centrální diferenci*

$$u''_i \approx \frac{u'_{i+\frac{1}{2}} - u'_{i-\frac{1}{2}}}{h} = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}.$$

Uvedená aproximace je pro jednoduchost zavedena pro funkce jedné proměnné. U funkcí více proměnných a zejména parciálních derivací je odvození analogické.

Body na tzv. *hranici* nebo *okraji* ($i = 0$ a $i = n$) budou zřejmě činit potíže, protože bychom potřebovali i pro hodnoty $i = -1$ a $i = n + 1$, které nemáme k dispozici. Proto je nutné na okrajích zadat tzv. *okrajovou* podmínku.

[1] Je třeba si uvědomit, že můžeme tuto aproximaci zavádět zleva (tedy členy x_{i-1} a x_i), ale také zprava (x_{i+1} a x_i). Dostáváme pak po řadě *zpětnou* a *dopřednou diferenci*[10, s. 62-64].

Tato metoda byla na Ústavu energetiky FSI VUT rozpracována v rámci dizertační práce Ing. Tomáše Maudera, Ph.D. [22] pro čtvercový (resp. obdélníkový) průřez bramy včetně způsobu, jak řešit problémy v oblastech fázové přeměny. Její správnost byla také ověřena praktickými měřeními. Z těchto důvodů jsem si ji vybral jako základ pro numerické řešení úlohy. Na druhou stranu je třeba si uvědomit, že vzhledem k tomu, že v úloze řešíme válcový odlitek, dostaneme úplně jiné rovnice a odlišné okrajové podmínky.

4.2 Aplikace numerických metod na model kontilití

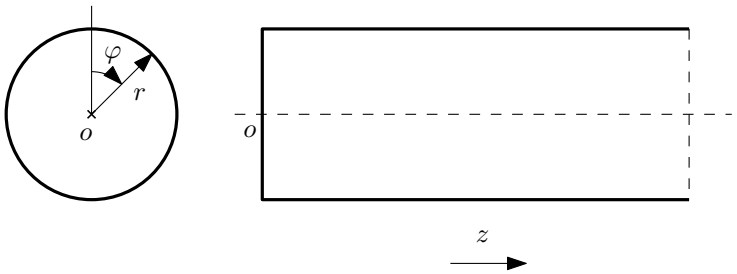
4.2.1 Diskretizace

Vzhledem k válcovému tvaru sochoru se jako nejpřirozenější volbou systému souřadnic jeví cylindrické (válcové) souřadnice (viz 4.1).

Při odlévání oceli je nutné zaručit co nejrovnoměrnější ochlazování. Pokud by bylo příliš rychlé, docházelo by ke vznikům nehomogenit a v nejhorším případě i prasklin v materiálu, které zásadním způsobem zhoršují kvalitu odlévané sochoru. Tento problém je nejpalčivější na povrchu odlitku, kde dochází k nejintenzivnějšímu chlazení a vyskytují se zde největší tepelné gradienty. Proto je nutné v blízkosti povrchu znát teplotní pole s co největší přesností. V řeči numerických metod to implikuje nutnost velice jemné sítě v okolí povrchu. Jemná síť ale zase znamená vyšší výpočetní náročnost. Rozumným kompromisem je v tomto případě použití nekonstantních intervalů diskretizace v radiální souřadnici. Ostatní souřadnice diskretizujeme rovnoměrným dělením.

Zavedeme tedy následující značení souřadnic:

1. r - vzdálenost bodu od podélné osy souměrnosti sochoru ve směru vnější normály,
2. ϕ - velikost úhlu bodu od svislé podélné řezné roviny,
3. z - vzdálenost bodu od začátku sochoru (konce krystalizátoru).
4. τ - časová diskretizace



Obrázek 4.1: Ilustrace diskretizace

Pro větší přehlednost také přiřadíme jednotlivým souřadnicím po řadě čítací indexy i, j, k, p a jejich množiny I, J, K, P. Označení $T^p_{i,j,k}$ pak značí $i, j, k-$tý element v čase $p \cdot \tau$. Dále označíme

$$\Delta r_i = r_i - r_{i-1}$$

jako i-tou diferenci vzhledem k souřadnici r a Δr jako vektor těchto diferencí.

Při numerické aproximaci nahrazujeme derivace tzv. *diferencemi*, tedy například pro T a souřadnici z aproximujeme první derivaci zpětnou diferencí

$$\frac{\partial T}{\partial z} \approx \frac{T_k - T_{k-1}}{\Delta z_{k-1}}.$$

Druhou derivaci aproximujeme centrální diferencí

$$\frac{\partial^2 T}{\partial z^2} \approx \frac{\frac{T_{k+1}-T_k}{\Delta z_k} - \frac{T_k - T_{k-1}}{\Delta z_{k-1}}}{\frac{\Delta z_k - \Delta z_{k+1}}{2}}$$

.

Rovnici (3.8) poté můžeme v souřadnicích r, φ, z aproximovat obdobně jako v [22, s. 45-47] následujícím způsobem:

$$Q_r = \frac{r_i \frac{T_{i+1,j,k}-T_{i,j,k}}{\Delta r_i} - r_i \frac{T_{i,j,k}-T_{i-1,j,k}}{\Delta r_{i-1}}}{r_i \frac{\Delta r_i + \Delta r_{i-1}}{2}},$$

$$Q_\varphi = \frac{\frac{T_{i,j+1,k}-T_{i,j,k}}{r_i \Delta \varphi_j} - \frac{T_{i,j,k}-T_{i,j-1,k}}{r_i \Delta \varphi_{j-1}}}{r_i \frac{\Delta \varphi_j + \Delta \varphi_{j-1}}{2}},$$

$$Q_z = \frac{\frac{T_{i,j,k+1}-T_{i,j,k}}{\Delta z_k} - \frac{T_{i,j,k}-T_{i,j,k-1}}{\Delta z_{k-1}}}{\frac{\Delta z_k + \Delta z_{k-1}}{2}},$$

Nakonec aproximujeme i entalpii a čas. Na rozdíl od prostorové diskretizace použijeme rovnoměrné dělení výpočetní domény:

$$H^{p+1}_{i,j,k} = H^p_{i,j,k} + \Delta\tau k_{i,j,k}(T^p_{i,j,k}) \left[Q_r + Q_\varphi + Q_z\right] + v_z \Delta\tau \frac{H^p_{i,j,k} - H^p_{i,j,k-1}}{\Delta z_{k-1}}. \tag{4.1}$$

Rovnice má zcela jasný význam. Vidíme, že entalpie v čase $p+1$ se rovná součtu entalpie v čase p, změny entalpie vlivem tepelných toků mezi sochorem a okolím a nakonec změnou entalpie ve výpočetním okně způsobenou přísunem materiálu z krystalizátoru o vysoké teplotě vlivem licí rychlosti v_z.

4.2.2 Počáteční a okrajové podmínky

Pro úspěšné řešení je samozřejmě nutné dodat k úloze počáteční a okrajové podmínky. V této části se budeme zabývat pouze numerickou reprezentací podmínek určených v odstavci 3.1.

Počáteční podmínku (3.3) reprezentujeme

$$T_{i,j,k} = T_{in},$$

kde T_{in} je teplota v krystalizátoru.

Pro následné výpočty je z numerického hlediska výhodné si úlohu předpočítat na hrubé síti. Tím dostaneme nepřesné řešení, které poté po vhodné interpolaci použijeme jako počáteční podmínku pro jemnější výpočetní síť. Velice se tím sníží výpočetní čas. Je ale nutné si uvědomit, že tímto krokem měníme povahu problému, a tedy tento postup

musíme aplikovat s rozvahou. V této úloze je to v pořádku, protože nás zajímá pouze rovnovážný (stacionární) stav a ne „cesta“ popř. rychlost, kterou se do něj postupným ochlazováním dostaneme.

Okrajové podmínky této úlohy patří k těm složitějším. Teoreticky musíme definovat až 26 různých okrajových podmínek. Pro jednoduchou ilustraci si lze představit výpočetní oblast souřadnic (r, φ, z) jako kvádr a uvědomit si, že pro každou stěnu, hranu i bod potřebujeme samostatnou okrajovou podmínku. V případě sochoru kruhového průřezu zůstává počet teoreticky nutných okrajových podmínek stejný. Nicméně díky použití diferenční metody lze spoustu z nich zanedbat anebo popř. zahrnout v jiné okrajové podmínce (na rozdíl od metody kontrolních objemů). Tímto krokem samozřejmě snižujeme přesnost numerického modelu. V naší zjednodušené úloze jsou ovšem tyto chyby naprosto zanedbatelné.

Vybrané okrajové podmínky reprezentuje obrázek 4.2. Z něj vyčteme následující podmínky:

1) Počátek sochoru. Teplota tedy odpovídá teplotě roztavené oceli, $T_{i,j,1} = T_{in}$.

2) Volný konec. Předpokládáme, že teplota už se dále výrazně nemění, tedy $\dot{q} = 0$, což je ekvivalentní s $T_{i,j,n_k} = T_{i,j,n_k-1}$.

3) Oblast středu sochoru. Diskretizací se „blížíme“ středu sochoru a předpokládáme, že zanedbaná oblast už na zbytek sochoru nemá vliv. Podmínku reprezentuje rovnost $T_{1,j,k} = T_{2,j,k}$.

4) Čistě numerická podmínka navázání prvního a posledního elementu v souřadnici φ. Nutno napsat z obou stran, tedy $T_{i,1,k} = T_{i,2,k},\ T_{i,n_j,k} = T_{i,n_j-1,k}$.

5) Vnější povrch sochoru. Body na povrchu budeme počítat stejným způsobem, jako body uvnitř, tedy dle rovnice (4.1). K tomu ale budeme potřebovat tzv. *fiktivní* body $T_{n_r+1,j,k}$. Teplota v těchto bodech se spočítá jako

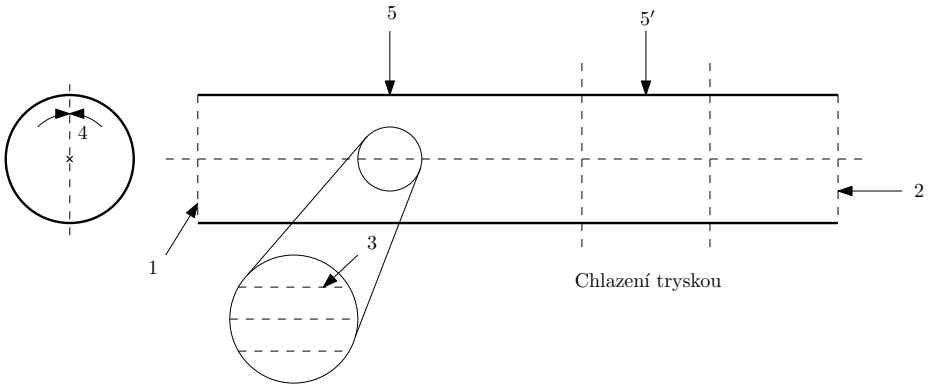
$$T_{n_r+1,j,k} = -\frac{2r_{n_r}\Delta r_{n_r}}{k} Q_r,$$

kde Q_r spočítáme podle (3.4). Povrch chlazený tryskou (v obrázku značený s čárkou) a povrch se samovolným chlazením se počítají stejně, změní se jen velikost koeficientu htc.

6) Krystalizátor. Zde dochází k silnému přestupu tepla mezi formujícím se sochorem a vodou chlazenými deskami. Postupujeme shodně se zbytkem povrchu, ale Q_r spočítáme dle (3.5).

Z hlediska numerického výpočtu je výhodnější reprezentovat podmínky 2), 3), 4) analogicky pomocí entalpií. Ve výpočtu si tak ušetříme spoustu přepočtů mezi entalpiemi a teplotami, které jsou časově velice náročné.

Hlavním tématem této práce je optimalizace teplotního pole. Matematický model v této kapitole je proto poměrně jednoduchý a lze vylepšit například přesnější reprezentaci oblasti okolo osy tyče. V přístupu výše předpokládáme, že diskretizace je dostatečně jemná na to, aby chyba způsobená zanedbáním objemu materiálu okolo středu tyče byla



Obrázek 4.2: Okrajové podmínky

nevýznamná. To samozřejmě nemusí platit vždy. Pokud bychom uvažovali silně nesymetrické chlazení, tímto přístupem bychom ztratili informaci o přenosu tepla přes střed sochoru.

V rámci testování byla naprogramována reprezentace singulární části diskretizace, tedy středového elementu, jako vektor M_k. Prvek M_k reprezentuje kruhovou část o poloměru r_0 v průřezu k (v souřadnici z). Zastupuje tedy body $T(0, j, k), \forall j \in J$. Jeho tepelnou bilanci pak spočítáme analogickým způsobem jako u středových bodů. Body $T(1, j, k)$ pak chápeme jako vnitřní.

Výsledkem porovnání původní implementace s implementací se středovými elemente bylo potvrzení zjištění, že jejich nezavedení nezpůsobí závažné změny řešení. Odchylky navíc s rostoucí jemností sítě klesaly. V počítačovém programu, konkrétně v metodě pro numerický výpočet, je možné přepínat mezi těmito modely pomocí logické proměnné *use_middle*.

4.2.3 Stabilita

Stabilita má v matematice obecně mnoho významů. V tomto případě nás zajímá *stabilita algoritmu*. Jedná se o schopnost iteračního algoritmu konvergovat ke správnému řešení. Nestabilní chování algoritmu se většinou velice snadno pozná, byť obecně bývá různé. Od určité iterace může řešení růst až nad všechny meze. Časté bývá také rozkmitání řešení mezi $-\infty$ a ∞, příp. nulou. Toto je také případ nestabilního chování řešené úlohy. Konkrétně začne kvůli chybě při výpočtu teplotního toku, která vede k jeho násobnému zvětšení, rychle růst teplota jeho sousedů. To zase vyvolá razantní snížení teploty další vrstvy bodů. Tyto chyby se v každém kroku prohlubují, až dosáhneme dokonce záporných hodnot. Je zřejmé, že největší vliv na stabilitu úlohy bude mít velikost kroků pro prostorovou (dr, $d\varphi$,dz) i časovou diskretizaci ($d\tau$). Z toho důvodu, že maximální možná velikost dr, $d\varphi$,dz je omezena kvůli požadované přesnosti modelu, můžeme stabilitu ovlivnit zejména volbou $d\tau$. Protože nám ale $d\tau$ určuje zásadním způsobem počet iterací (čas, po kterém se dostaneme do stacionárního stavu je vždy stejný, ale velikost $d\tau$ ovlivňuje počet iterací, které potřebujeme, abychom se do něj dostali) a tedy rozhodující měrou i čas výpočtu. Naštěstí lze pro naši úlohu odvodit *podmínky stability*, které zaručují stabilní chování algoritmu. Lze je najít v [22] (pro kartézský souřadný systém). V cylindrických souřadnicích musíme brát v úvahu, že radiální a úhlová délka elementů konverguje ve směru ke středu osy souměrnosti k nule.

Bohužel, vzhledem k tomu, že spoustu materiálových konstant aproximujeme či interpolujeme, jsou tyto podmínky pouze orientační a přesné nastavení parametrů musíme odzkoušet. Pro náš model vychází časový krok $d\tau$ v závislostech na jemnosti sítě $10^0 - 10^{-5}$. Bohužel, u velkých sítí takto malý krok způsobuje, že k dosažení stacionárního stavu je nutný extrémně velký počet iterací. Bez předpočítané počáteční podmínky bývá u těchto sítí z vlastních zkušeností délka výpočtu v řádech dní.

4.3 Implementace modelu

Numerický model byl nejdříve naprogramován v softwaru *Octave* [11]. Jedná se o open-source variantu známého numerického softwaru *Matlab*. Obě prostředí mají společný programovací jazyk (liší se pouze v maličkostech, Octave má větší podporu standardní syntaxe „C“ jazyků, liší se také názvy některých funkcí). Výhodou Matlabu je oproti Octave jeho rychlost (zhruba dvakrát až desetkrát). Jejich obrovskou výhodou je velice jednoduchá syntaxe jazyka, kterou se lze velice snadno naučit. Také mají rozsáhlé knihovny již naprogramovaných funkcí, výborné prostředky pro vizualizaci výsledků a zejména opravdu extenzivní manuály. V případě Matlabu existuje široká komerční podpora. Octave má zase na druhou stranu velice ochotnou komunitu. Z hlediska výkonu ve své specializaci, tedy maticových operacích, patří tyto programy ke špičce.

Najdou se ale i nevýhody. Mezi ně patří zejména relativní pomalost při práci se soubory a také nízký výkon při cyklických operacích, které nejsou volány vnitřně (cykly typu *for*, *while*, *repeat* apod.).

Bohužel, pro implementaci explicitního numerického solveru se použití cyklů nedá vyhnout. Pro řešení úlohy diferenční metodou už na síti s řádově 10^5 elementů trval výpočet v prostředí Octave přes hodinu. Vzhledem ke snaze získat software pro výpočet v reálném čase je jeho použití absolutně nemožné.

Po prozkoumání dalších alternativ se jako nejvhodnější prostředek ukázal jazyk C++. Jedná se o nízkoúrovňový, velice efektivní programovací jazyk. Patří k dnes mezi nejpoužívanější programovací prostředky vůbec. K jeho výhodám patří rychlost, všestrannost, nezávislost na platformě (s výjimkou MacOS, kde musíme používat pouze Objective C) a existence mnoha různorodých knihoven. Z nízkoúrovňovosti ale vyplývají i největší nevýhody. To je zejména složitá syntaxe, nutnost chápat principy fungování hardwaru (memory management, ukazatele - adresy proměnných, prvků polí, destruktory), složitý debugging a také neexistence nástrojů pro vizualizaci (ve velice omezené míře lze potlačit využitím různých knihoven). I přes tyto nevýhody je ve své oblasti tento jazyk naprosto bezkonkurenční, co se týká výkonu, tak i rozšířenosti.

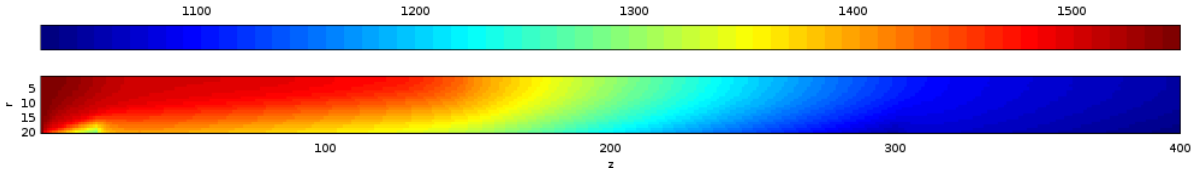
Nejdříve bylo odzkoušeno naprogramovat úlohu pomocí vlastní implementace matic a vektorů v prostředí Microsoft Visual Studio. Bohužel, její výkon byl také poměrně nízký, i když se dosáhlo zhruba pětinásobného zrychlení. Dále následovala implementace maticových struktur pomocí open-source C++ knihovny pro lineární algebru *Armadillo* [26]. Tato knihovna byla inspirována prostředími typu Matlab a snaží se převést syntaxi a způsob práce s prvky lineární algebry do C++, a tím spojit výhody obou prostředí - rychlost C++ a jednoduchost a širokou škálu metod Matlabu. A opravdu, naprostá většina základních matlabovských funkcí, včetně jednoduchého načítání a zapisování souborů je

přítomna[2] [27]. Nevýhoda horšího debuggingu byla sice v této knihovně zmírněna, ale stále je na míle vzdálena výše zmíněným prostředím. Nicméně použitím této implementace se dosáhlo až stonásobného (!) zrychlení oproti Octave. Některé příklady použití jsou v příloze B.

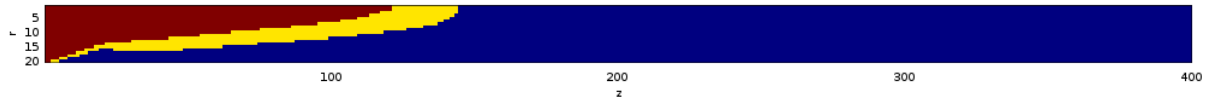
4.4 Výsledky

Výsledky numerického modelu znázorníme vizualizací řezů v rovině (r, z) (tedy jako podélný průřez sochorem). Parametry modelu jsou uvedeny v tabulce 4.1.

Vzhledem k tomu, že se jedná o model symetrický podél podélné osy, je kvůli zrychlení výpočtu volen dostatečně velký krok v úhlu φ. Model je připraven i na nekonstantní dělení výpočetní oblasti.



Obrázek 4.3: Vizualizace teplotního pole



Obrázek 4.4: Vizualizace fází v sochoru

Na obrázku 4.3 vidíme průběh teploty při stacionárním stavu ve spodní polovině podélného řezu. Osy obrázku mají význam indexu elementu diskretizace v dané souřadnici. Velice dobře lze vidět formování skořepiny v krystalizátoru, teplota na okraji sochoru zde klesá až k 1300°C. Méně zřetelně můžeme identifikovat rozhraní sekundárního a terciálního chlazení v okolí 300. elementu - teplota v terciální části po omezenou vzdálenost v ose z mírně stoupá, protože díky náhle absenci umělého chlazení převažuje teplo dodané licí rychlostí nad teplem odebraným přirozenou radiací a konvekcí. Na obrázku 4.4 je zobrazena reprezentace fází v sochoru, tekutá ocel červenou barvou, ocel v pevném stavu modrou a ta nacházející se v přechodovém stavu (mezi teplotou solidu a likvidu) je znázorněna oranžově.

Vidíme, že veškerá ocel krystalizuje v oblasti okolo 6 m od začátku soustavy. To je poměrně nízká vzdálenost, ideální bývá v praxi kolem 10 m. Toto je způsobeno zejména jednoduchostí modelu, v reálném případě není chlazení tryskami kontinuální, ale střídají se krátké úseky tryska - vodicí válec. V místech styku s vodicími válci navíc nevyzařujeme teplo radiací, ale pouze konvekcí. To vede k dalšímu snížení odvodu tepla a zvýšení rozdílu mezi realitou a tímto modelem.

[2]Poměrně elegantně lze vyřešit i nedostatek nástrojů pro vizualizaci. Stačí uložit získaná řešení ve formě *ASCII* souboru, které je čitelný Matlabem či Octave. Pak pouze stačí tento software spustit, načíst soubor a provést vizualizaci. Tyto kroky byly nakonec také zautomatizovány.

Tabulka 4.1: Parametry vizualizace numerického modelu

Technické parametry		Numerické parametry	
Poloměr sochoru	0,1 m	Délka výpočetní oblasti	20 m
Délka krystalizátoru	1 m	dr_0	0,005 m
Délka terciálního chlazení	5 m	$d\varphi_0$	$\frac{\pi}{2}$
Licí teplota	1550°C	dz_0	0,05 m
Teplota okolí	25°C	$d\tau$	0,1 s
Licí rychlost	1,2 mmin^{-1}	Počet iterací	12000
Htc trysek	250 Wm^{-2}K^{-1}	Čas výpočtu	7,62 min

Zavedení vodicích válců do modelu samozřejmě není vůbec složité. Nicméně nemá cenu se tím zabývat, pokud netvoříme model na míru reálné soustavy, což ale není úkolem této práce.

Dalším důvodem je to, že konstrukce reálného chlazení tryskami nebývá kolem podélné osy symetrická. Většinou se chladí pouze jedna strana sochoru. Toto způsobuje další odchylky v modelu.

Numerická implementace byla srovnávána se solvery vyvinutými na Ústavu energetiky v rámci prací [19] a [22]. Jedná se o modely řešené pro obdélníkový příp. čtvercový tvar tělesa, takže veškerá srovnání jsou pouze přibližná.

Vzhledem k výše zmíněným poznámkám došlo k předpokládanému „příliš intenzivnímu“ chlazení. Na druhou stranu, i přes svou jednoduchost se tato implementace chová v souladu s očekáváními a rozdíly v teplotním poli nejsou tak velké, aby implikovaly závažné chyby.

5 Matematické programování

Matematické programování (nebo také optimalizace) je matematickým oborem, který se zabývá hledáním extrémů (lokálních či globálních) funkcí. Optimalizace se v praxi využívá ve všech myslitelných souvislostech, od ekonomie přes vědu a průmysl až po lékařství, viz např. [6].

V této kapitole se zaměříme zejména na definování pojmů a principů nutných k tomu, abychom mohli úspěšně používat a zejména interpretovat výsledky různých optimalizačních algoritmů. Celá kapitola vyvrcholí tzv. *Progressive Hedging Algortihm (PHA)*, který poté v následující kapitole aplikujeme na řešenou úlohu. I když v této úloze náhodu neuvažujeme, budeme se také zabývat stochastickou optimalizací. To je nutné z toho důvodu, že PHA byl jako dekompoziční algoritmus původně odvozen pro úlohy stochastické optimalizace. K jeho pochopení jsou alespoň základní znalosti z této oblasti naprosto nezbytné.

5.1 Deterministická optimalizace

Pojmem deterministická optimalizace rozumíme úlohy, ve kterých se nevyskytují náhodné veličiny. Na rozdíl od *stochastické* optimalizace nabývají všechny proměnné a parametry pouze jedné hodnoty (resp. posloupnosti hodnot). Přídavné jméno *deterministická* se zpravidla vynechává.

Zjednodušeně lze optimalizaci popsat jako hledání příslušného extrému účelové funkce na množině dané jistým počtem omezení. Jedním z jednoduchých příkladů optimalizace je například problém vázaných extrémů funkce z oblasti matematické analýzy [20].

Vzhledem k tomu, že problémy, které musí optimalizace řešit, jsou zřídkakdy tak jednoduché jako školní příklady, bývá zvykem zapisovat je v tzv. *základním (standardním) tvaru*, který vypadá následovně:

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{p}), \tag{5.1}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{p}) \leq 0, \tag{5.2}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{p}) = 0, \tag{5.3}$$

$$\mathbf{x} \in \mathbf{X}. \tag{5.4}$$

Funkci $f(\mathbf{x}, \mathbf{p})$ vektorové proměnné $\mathbf{x}$ a vektoru parametrů $\mathbf{p}$ nazýváme *účelovou funkcí (objective function)*. Výraz $\min_{\mathbf{x}} \cdot$ chápeme ve smyslu "minimalizuj účelovou funkci $\cdot$ přes vektor proměnných $\mathbf{x}$". Vektorové funkce $\mathbf{g}(\mathbf{x}, \mathbf{p})$ a $\mathbf{h}(\mathbf{x}, \mathbf{p})$ značí omezení (*constraints*)[1]. Je zvykem značit funkci omezení "$\leq$"písmenem g (v anglické literatuře se nicméně často vyskytuje f, případně f_c) a omezení rovnosti h. Omezení týkající se samotného vektoru $\mathbf{x}$ se zapisují obecně $\mathbf{x} \in X$, kde X je množina "vhodných"$\mathbf{x}$ (v anglické literatuře často jako *bounds*). Často se takto vymezuje oblast, ve které hodnoty vektoru $\mathbf{x}$ hledáme, tedy se určuje maximální a minimální možná hodnota (např. $\mathbf{x} \geq \mathbf{0}$).

Množinu

$$C(\mathbf{x}) = \{\mathbf{x} \in \mathbf{X}, \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \mathbf{h}(\mathbf{x}) = \mathbf{0}\},$$

[1] Omezení uvedená v 5.2 a 5.3 píšeme pro obecný případ jako vektorové funkce. Pro danou úlohu může totiž existovat více omezení daného typu.

tedy všech $\mathbf{x}$ splňujících podmínky (5.2)-(5.4) nazýváme *oblastí přípustných řešení (set of feasible solutions, feasible region*). Pokud je tato množina prázdná, úloha nemá přípustné (optimální) řešení.

V optimalizaci často řešíme úlohy, ve kterých chceme účelovou funkci maximalizovat. Maximalizaci $\max_{\mathbf{x}} f(\mathbf{x})$ přepíšeme na minimalizaci $\min_{\mathbf{x}} -f(\mathbf{x})$. Podobně mezi sebou převádíme omezení kladnosti a zápornosti u omezení danými vektorovou funkcí $\mathbf{g}$.

Velkou výhodou standardního tvaru je skutečnost, že spousta úloh z různých aplikačních oblastí lze převést na podobný tvar, což ale na první pohled nebývá zřejmé. Díky tomu se můžeme při řešení spousty úloh opřít již o existující přístupy a ušetřit si tak velké množství práce.

Při hledání řešení nelineární úlohy se opíráme zejména o *Karush-Kuhn-Tuckerovy podmínky (KKT)*[2, s. 188]. Zobecňují za určitých podmínek metodu Lagrangeových multiplikátorů i na problémy s omezeními ve tvaru nerovností. Jedná se o nutné podmínky optimality (ve speciálních případech i postačující). Na KKT je založena velká část nelineárních optimalizačních solverů[2].

Speciální typem deterministické optimalizace je tzv. *lineární optimalizace*. Účelová funkce a omezení jsou lineární a lze je tedy vyjádřit následujícím způsobem:

$$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x}, \tag{5.5}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{5.6}$$

$$\mathbf{x} \in \mathbf{X}, \tag{5.7}$$

kde $\mathbf{c}$ je sloupcový vektor $m \times 1$ a $\mathbf{x}$ sloupcový vektor $n \times 1$. Matice A má rozměry $k \times n$. $\mathbf{X}$ značí polyedrickou množinu [1]. Ze standardního tvaru vidíme, že omezení (5.6) nám dávají soustavu vzájemně se protínajících poloprostorů. Jednoduše lze ukázat, že neprázdná množina přípustných řešení je v tomto případě konvexní mnohostěn (obecně neomezený). Možná nás zarazí, že v účelové funkci neřešíme posunutí o konstantu. Má to však jednoduché vysvětlení. Konstanta pouze zvýší či sníží hodnotu funkce, ale nezmění polohu extrému vzhledem k $\mathbf{x}$. Protože primárně nás zajímá optimální hodnota $\mathbf{x}$, nemusíme se tímto posunutím vůbec zabývat. Omezení (5.7) bývá v úlohách lineární optimalizace nejčastěji ve tvaru $\mathbf{x} \geq \mathbf{0}$. V tomto případě tedy požadujeme pouze nezáporná řešení.

Pro lineární optimalizaci platí spousta vět, které velice zjednodušují hledání optimálního řešení. Jednou z nejdůležitějších je *věta o optimalitě pro lineární úlohu*, která nám říká, že optimální řešení, pokud existuje, nastává vždy ve vrcholu (vrcholech) konvexního mnohostěnu, který tvoří oblast přípustných řešení [1, s. 81-85].

Nejznámějším způsobem řešení je *simplexová metoda* [1, s. 81, příp. s. 108], [6, s. 370]. Jedná se o iterační algoritmus, který využívá maticový tvar úlohy. Tu pomocí volných proměnných převádí na řešení systému lineárních rovnic pomocí elementárních sloupcových a řádkových úprav (používá se *Gaussova eliminační metoda*).

[2]KKT podmínky jsou založeny na analýze stacionárních bodů. Ovšem pokud jsou omezení $\mathbf{g}$ či $\mathbf{h}$ nediferencovatelné nebo nemají derivace, nemůžeme KKT použít. Tento fakt značně komplikuje řešení takovýchto úloh, protože numerické řešení KKT je základem nejpoužívanějších algoritmů na řešení optimalizačních úloh

K lineární optimalizaci se často přidává tzv. *kvadratická optimalizace*. Podobně jako lineární úloha má svůj vlastní specifický tvar:

$$\min_{\mathbf{x}} \mathbf{x}^T \mathbf{H} \mathbf{x}, \tag{5.8}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{5.9}$$

$$\mathbf{x} \in \mathbf{X}, \tag{5.10}$$

kde H je symetrická a pozitivně semidefinitní matice. Pro řešení těchto úloh existují speciální algoritmy. Lze je také řešit algoritmy pro nelineární optimalizaci.

Na závěr tohoto odstavce si uvedeme jednoduchý příklad z nelineární optimalizace.

Příklad (Nelineární optimalizace)

Uvažujme nyní pro ilustraci případ nelineární optimalizace v rovině, tedy $n = 2$, v následujícím tvaru:

$$\min_{x_1,x_2} (x_1 - 2)^2 + (x_2 - 3)^2, \tag{5.11}$$

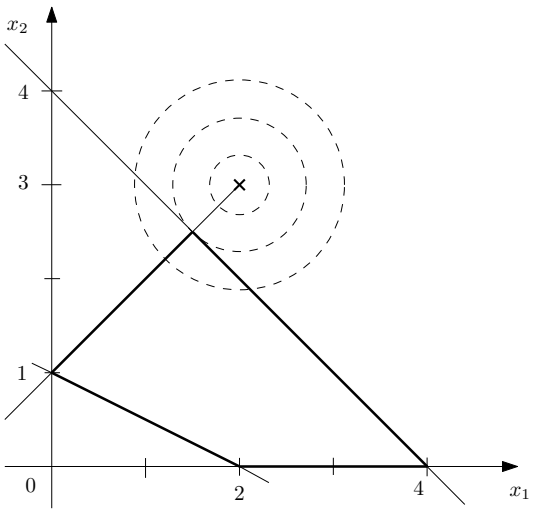
$$\frac{1}{2}x_1 + x_2 \geq 1, \tag{5.12}$$

$$x_1 + x_2 \leq 4, \tag{5.13}$$

$$x_1 - x_2 \leq -1, \tag{5.14}$$

$$x_1, x_2 \geq 0. \tag{5.15}$$

Při pohledu na účelovou funkci vidíme, že se jedná o rotační paraboloid s minimem v bodě $[2, 3]$. Lineární omezení (5.12)-(5.15) nám v rovině x_1, x_2 určují následující oblast přípustných řešení:



Z obrázku je zřejmé, že minimem bude hraniční bod oblasti, který je "nejblíže"průmětu vrcholu paraboloidu $[2, 3]$. Elementárními výpočty dostaneme souřadnice bodu optima. Taky vidíme, že oblast přípustných řešení odpovídá konvexnímu mnohoúhelníku. Toto ovšem platí pouze pro lineární omezení. Tuto úlohu lze také chápat jako problém nalezení vázaného extrému funkce (5.11) s omezeními (5.12)-(5.15).

5.1.1 Optimalizace s více účelovými funkcemi

V mnoha úlohách optimalizace se objevují problémy, ve kterých nám pouze jedna účelová funkce nestačí. Jako jednoduchý příklad můžeme uvést situaci, kdy chceme například na výrobním stroji maximalizovat rychlost výroby a zároveň minimalizovat počet neshodných kusů. Úlohy tohoto typu se převádějí do standardního tvaru poměrně jednoduše použitím *vah*. Váhami rozumíme posloupnost nezáporných (reálných) čísel $\{w_i\}_{i=1}^n$ takovou, že $\sum_{i=1}^n w_i = 1$. Pro náš příklad platí $n = 2$. Pokud označíme účelové funkce úlohy po řadě $f(\mathbf{x})$ a $g(\mathbf{x})$, dostaneme

$$\min_{\mathbf{x}} -w_1 f(\mathbf{x}) + w_2 g(\mathbf{x}).$$

Funkce $f(\mathbf{x})$ je brána se znaménkem mínus, protože převádíme maximalizační úlohu na minimalizační. V obecném tvaru má celková účelová funkce tvar

$$\min_{\mathbf{x}} \sum_{i=1}^{n} w_i f_i(\mathbf{x}).$$

Návod jak určit hodnoty vah obecně neexistuje. V lepším případě bývají zřejmé ze zadání úlohy nebo technicko-fyzikálních zákonitostí. Často je ale musíme určit buď testováním, nebo musíme mít k dispozici odhad experta (např. [6, s. 73-74])

Často se využívají při řešení složitějších optimalizačních úloh, např. s penalizací, kdy nám umožňují ovlivňovat rychlost konvergence úlohy. Velký význam mají také ve stochastické optimalizaci, např. u úloh, kdy chceme v určitém poměru maximalizovat střední hodnotu a minimalizovat rozptyl náhodné veličiny [6, s. 73-74].

5.2 Stochastická optimalizace

Velkou částí úloh matematického programování tvoří problémy, ve kterých hraje určitou roli náhoda. Jednoduchým a snadno srozumitelným motivačním příkladem je například úloha optimálního investičního portfolia [6, s. 530].

Úlohou stochastického programování ve standardním tvaru rozumíme:

$$\begin{aligned} \min_{\mathbf{x}} & f(\mathbf{x}, \xi), \\ & g(\mathbf{x}, \xi) \leq 0, \\ & h(\mathbf{x}, \xi) = 0, \\ & \mathbf{x} \in \mathbf{X}, \end{aligned}$$

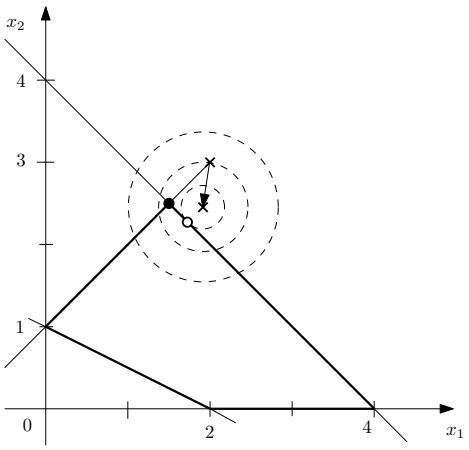
kde $\xi \in \Xi$ je náhodná veličina z množiny Ξ. Ostatní veličiny byly definovány v předchozí části. Tato úloha je definována obecně. Ve většině aplikací se náhodná veličina vyskytuje pouze v některých omezeních nebo nemusí být přítomna v účelové funkci.

V úlohách s náhodností v účelové funkci lze také rozšířit možnosti, jak definovat účelovou funkci. Můžeme tedy zejména maximalizovat střední hodnotu náhodné veličiny nebo minimalizovat její rozptyl. V souvislosti s příkladem optimalizace investičního portfolia si to lze představit tak, že maximalizací střední hodnoty se snažíme dosáhnout maximálního zisku (ale už nespecifikujeme rozptyl!), zatímco minimalizací rozptylu dostaneme nějakou hodnotu (neřešíme její výši), která se ale objeví s vysokou pravděpodobností.

Náhodnost si lze velice jednoduše představit na příkladu (5.11). Uvažujme nyní náhodnost v účelové funkci:

$$\min_{x_1,x_2}(x_1-2)^2+(x_2-3-\xi)^2.$$

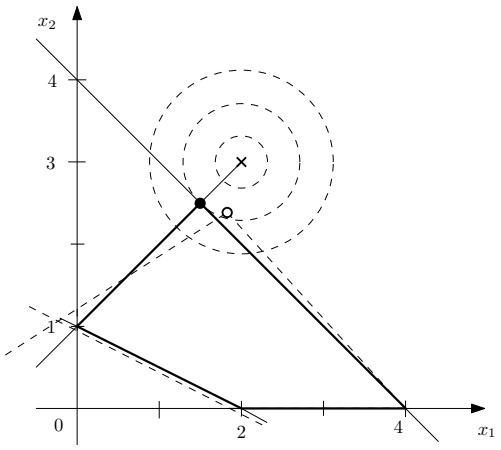
Tato změna se projeví v náhodně velkém posuvu středu paraboloidu. Všimněme si, že můžeme, ale nemusíme dostat ve srovnáním s výsledkem z příkladu (5.11) stejné výsledky. Posun je znázorněn šipkou. Černou tečkou je označeno původní řešení a bílou tečkou nové.



Obrázek 5.1: Náhodnost v účelové funkci

Průměty vrstevnic neposunutého paraboloidu nejsou pro přehlednost zakresleny.

Náhodnost se ale podobně může projevit také v omezeních, kde zapříčiní náhodnou deformaci oblasti přípustných řešení. Zase můžeme, ale nemusíme dostat rozdílné výsledky ve srovnání s (5.11) Náhodné veličiny zřejmě představují poměrně velkou komplikaci.



Obrázek 5.2: Náhodnost v omezeních

Existují dva základní způsoby, jakými můžeme náhodnost odstranit. Obecně se nahrazení úlohy s náhodou říká *deterministický přepis (Underlaying Programme, Deterministic Equivalent)*. Existuje více způsobů, jak toho docílit. Dopodrobna jsou popsány například zde [24]. Zaměříme se zde v krátkosti pouze na pár z nich, které jsou pro nás důležité.

5.2.1 Wait-and-See

Tento přístup funguje přesně podle svého názvu. Do češtiny ho můžeme přeložit jako „Počkej a uvidiš“. Funguje na podobném principu jako regresní analýza. Tedy na základě předchozích realizací ξ vypočteme pro každou z nich optimální řešení podobně jako když pro měření v regresi určujeme metodou nejmenších čtverců bodové odhady regresních koeficientů. Z tohoto ale vyplývá omezení použití Wait-and-See (WS). V některých problémech jednoduše nemůžeme odhadnout hodnotu ξ nebo nemáme k dispozici historii jejího chování, případně je nemožné zkoumat chování této veličiny pomocí pokusů. Důležité je to, že o hodnotě ξ můžeme rozhodnout už před aplikací optimalizace a je na předchozím chování ξ závislé. Toto rozhodnutí tedy označíme jako $\mathbf{x}(\xi)$. Účelová funkce pak má tvar $z = f(\mathbf{x}(\xi), \xi)$.

5.2.2 Here-and-Now

Zde se musíme rozhodnout dříve než o realizaci náhodné veličiny ξ můžeme vůbec něco zjistit. Na rozdíl od WS nemáme k dispozici historii jejího chování ani nemůžeme provádět experimenty. Navíc rozhodnutí musí pro každou realizaci ξ být naprosto stejné, tedy rozhodnutí $\mathbf{x} \neq \mathbf{x}(\xi)$. Z těchto vlastností vyplývá nutnost složitějších algoritmů na vyřešení. Většina z nich je navíc iterační. Proto způsobují velké a komplexní Here-and-Now (HN) úlohy spoustu problémů a jejich řešení trvá řádově delší dobu. Proto, je-li to možné, upřednostňujeme WS přístup. Obecně lze účelovou funkci HN modelu popsat jako

$$z = \varepsilon_\xi f(\mathbf{x}, \xi),$$

kde ε je vhodný funkcionál „odstraňující“ náhodnost z funkce f.

5.2.3 Expected Value

Do češtiny se dá přeložit jako *očekávaná hodnota*. Nicméně i v české literatuře se většinou používá anglický název. Jedná se o bezesporu nejčastější způsob určení funkcionálu ε. Očekávanou hodnotu určujeme různými způsoby, nejčastěji ze zkušeností nebo expertním odhadem. Pokud víme, že veličina ξ má normální rozdělení, můžeme také použít střední hodnotu. Očekávanou hodnotu ξ označme ξ^{EV}. Účelová funkce pak má obecně tvar

$$z^{EV} = f^{EV}(\mathbf{x}) = f(\mathbf{x}, \mathrm{E}_\xi(\xi^{EV})).$$

Rozlišujeme také *Expected Value Objective (EO)*, který popisuje očekávanou hodnotu účelové funkce:

$$z^{EO} = f^{EO}(\mathbf{x}) = \mathrm{E}_\xi f(\mathbf{x}, \xi^{EO}).$$

.

5.2.4 Scénářový přístup

Jedním z dalších přístupů, jak daný problém řešit, jsou tzv. *scénářové úlohy (Scenario Programmes)*. Určíme si z nějakého důvodu důležité realizace náhodné veličiny ξ a tyto realizace očíslujeme. Jejich dosazením do problému dostaneme skupinu scénářů. Budeme je označovat indexem s z indexové množiny S. Budeme chtít, aby získané celkové optimální řešení bylo splněno na všech scénářích.

Úlohu stochastického programování nejdříve přeformulujeme následujícím způsobem

$$\min_{\mathbf{x}} \sum_{s=1}^{|S|} p_s f(\mathbf{x}, \xi^{\mathbf{s}}), \tag{5.16}$$

$$\mathbf{g}(\mathbf{x}, \xi^s) \leq 0 \ \forall s \in S, \tag{5.17}$$

$$\mathbf{h}(\mathbf{x}, \xi^{\mathbf{s}}) = 0 \ \forall s \in S, \tag{5.18}$$

$$\mathbf{x} \in C_s \ \forall s \in S, \tag{5.19}$$

kde nezáporná reálná čísla $p_s \in \langle 0, 1 \rangle$ jsou pravděpodobnosti nastání jednotlivých scénářů, tedy $p_s = P(\xi = \xi^s)$. Budeme také potřebovat, aby $\sum_{s=1}^{|S|} p_s = 1$. Pak p_s označujeme jako váhy.

Jednotlivé *individuální scénáře (Individual Scenarios Approach, Individual Scenario Programme)* pak mají následující tvar

$$\min_{\mathbf{x}} f(\mathbf{x}, \xi^{\mathbf{s}}), \tag{5.20}$$

$$\mathbf{g}(\mathbf{x}, \xi^s) \leq 0, \tag{5.21}$$

$$\mathbf{h}(\mathbf{x}, \xi^{\mathbf{s}}) = 0, \tag{5.22}$$

$$\mathbf{x} \in C_s, \tag{5.23}$$

kde ξ^s značí realizaci náhodné veličiny ξ ve scénáři s. Vidíme, že se nám úloha rozpadne na sérii deterministických problémů, které se liší hodnotou ξ^s. Jejich vyřešením dostaneme s optimálních řešení x^s. Všimněme si, že oblasti přípustných řešení mohou být pro různé scénáře odlišné. Tedy optimální řešení scénáře s_1 nemusí být v scénáři s_2 vůbec přípustné! Zřejmě se chceme těmto situacím vyhnout a proto zavádíme *přípustné řešení úlohy scénářové úlohy* $\mathbf{x}^*$ takové, že

1) $\mathbf{x}^* = \hat{\mathbf{x}} = \sum_{s \in S} p_s x^s$,

2) $\mathbf{x}^* \in \bigcap_{s \in S} C_s$.

První podmínka nám říká, že $\mathbf{x}^*$ musí být v jistém smyslu „průměrným“ řešením úlohy *(implementable)*. Druhá říká, že musí ležet v průniku oblastí přípustných řešení jednotlivých scénářů *(admissable)*. Nastává ovšem otázka, jestli nás podmínka číslo dvě příliš neomezuje. Nebylo by možné v některých méně pravděpodobných nebo nedůležitých scénářích dovolit jistou nepřípustnost celkového řešení a tedy příslušné omezení *relaxovat*? Ukázalo se, že toto opravdu lze udělat. Jako nástroj nám poslouží *penalizace*, kterou popíšeme v následující kapitole.

5.3 Dvoustupňové stochastické programování

V angličtině pod názvem *Twostage Stochastic Programming*. Hlavní myšlenkou je následující princip. Rozhodnutí, které při volbě realizace učiníme, je vždy do určité míry chybné.

Pokusíme se jej tedy v kroku číslo dvě, poté, co se tato veličina už jistým způsobem realizovala, nějakým způsobem opravit. Jednotlivé proměnné a funkce ve stupni $i = 1, 2$ budeme značit horním indexem. Rozhodnutí v prvním stupni označíme $\mathbf{x}$ a v druhém $\mathbf{y}$. V této části uvedeme stručně pouze pojmy nezbytné pro zavedení *Progressive Hedging algoritmu*. Problematika je podstatně širší a lze ji najít např. v [4, 18, 23].

5.3.1 Penalizační úlohy založené na scénářích

V angličtině pod pojmem *Scenario-based programmes with recourse*.

V každém scénáři určujeme „důležitou“ nebo „pravděpodobnou“ realizaci ξ^s náhodné veličiny ξ. Ovšem tímto náhodné chování zcela nepostihneme a může dojít k chybám či dokonce porušení oblasti přípustných řešení. Pro nápravu zavádíme *penalizaci (recourse)* $\mathcal{Q}(\mathbf{x})$. Je zřejmé, že penalizace musí záviset také na náhodné veličině ξ a tedy pouhé $\mathcal{Q}(\mathbf{x})$ nestačí

$$Q(\mathbf{x}, \xi) : \ \mathbb{R} \times \Xi \mapsto \mathbb{R} \cup \{\infty\}.$$

Funkce $Q(\mathbf{x}, \xi)$ se nazývá *cena (míra) penalizace (recourse cost function)*. V optimalizační úloze bychom ale znovu zavedli náhodnost, které jsme se použitím *Here-and-Now* zbavili. Proto musíme zavést deterministický ekvivalent penalizační funkce. Nejpoužívanějším způsobem je použití Expected Value objective (EO) (viz odstavec 5.2.3)

$$\mathcal{Q}(\mathbf{x}) = \mathrm{E}_\xi Q(\mathbf{x}, \xi).$$

Vracíme se tedy ke značení $\mathcal{Q}(\mathbf{x})$ a tuto funkci nyní pojmenujeme *funkcí průměrné ceny penalizace (average cost recourse function)*. Penalizaci lze dále v mnohém rozšířit a zobecnit, viz [23, 2, 18].

Penalizace nám reprezentuje chybu, které jsme se dopustili „potlačením“ náhodnosti ξ. Logicky tedy chceme její velikost minimalizovat. Penalizační funkce také může mít mnoho různých tvarů (zejména lineární, kvadratická). Také ji lze aplikovat různými způsoby (nejčastěji aditivně, multiplikativně). V našem případě individuálního scénáře nás bude zejména zajímat aditivní použití.

Součástí penalizační funkce bývají často rozličné váhy, které nám umožňují rozhodnout, jak velkou nepřípustnost řešení budeme chtít tolerovat. Jejich nastavení bývá obvykle předmětem zkoušení a ladění.

Penalizace se používá nejčastěji jako lineární funkce. V úlohách, kde by měla smysl kvadratická, se penalizace nahrazuje pomocí *rozšířeného langrangiánu (augmented lagrangian)*. Principem tohoto přístupu je, že účelovou funkci nahradíme *Lagrangeovu funkcí (langrangiánem)* v rozšířeném tvaru. Úloha se pak řeší iteračně, odhad v následující iteraci je vždy lepší než v té předchozí. Více o tomto přístupu v [2, s. 485].

Velkou výhodou rozšířeného lagrangiánu oproti obyčejné penalizaci je rychlost konvergence [2]. Připomeňme, že penalizace může nabývat až hodnoty nekonečno, což značí, že daná úloha za použití dané realizace ξ nemá řešení). Další velkou výhodou je diferencovatelnost lagrangiánu, což nám umožňuje využít pro řešení úloh velice široké spektrum algoritmů, které ke správné funkci informaci o derivaci účelové funkce vyžadují.

Hlavní myšlenkou scénářového přístupu je uvažovat ξ jako náhodnou veličinu konečným diskrétním rozdělením, tedy $|\xi| < \infty$. Chování veličiny ξ reprezentujeme *scénáři*,

které značíme indexem s, $s \in S$. Každý scénář nám reprezentuje realizace ξ^s. Dále označme rozhodnutí v druhém stupni $\mathbf{y}_s = \mathbf{y}(\xi^s)$. Pak lze penalizaci pomocí EO přepsat jako

$$\mathrm{E}_\xi Q(x, \xi) = \sum_{s=1}^{S} p_s Q(x, \xi^s).$$

Příslušný deterministický ekvivalent scénářové úlohy s penalizací pak dostaneme v následujícím tvaru

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) + \sum_{s=1}^{S} p_s Q(x, \xi^s), \\ \mathbf{g}_1(\mathbf{x}) \leq 0 \ \forall s \in S, \\ \mathbf{h}_1(\mathbf{x}) = 0 \ \forall s \in S, \\ \mathbf{x} \in C_s \ \forall s \in S, \end{aligned}$$

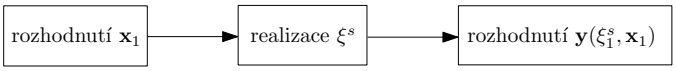
kde $Q(x, \xi^s)$ získáme vyřešením

$$\begin{aligned} Q(\mathbf{x}, \xi^s) = \min_{\mathbf{y}_s} \mathbf{q}(\mathbf{x}, \mathbf{y}_s, \xi^s), \\ \mathbf{t}_1(\mathbf{x}, \xi^s) + \mathbf{g}_2(\mathbf{y}(\xi^s), \xi^s) \leq 0, \\ \mathbf{t}_2(\mathbf{x}, \xi^s) + \mathbf{h}_2(\mathbf{y}(\xi^s), \xi^{\mathbf{s}}) = 0. \end{aligned}$$

Vektor $\mathbf{q}$ popisuje „cenu“ penalizace v druhém stupni (podrobněji v [18, 23]).

5.3.2 Struktura dvoustupňových úloh

Dvoustupňové úlohy, jak byly zavedené výše, mají přesně danou strukturu.

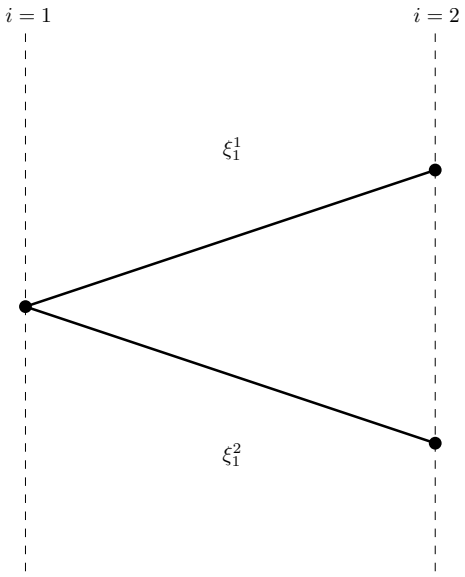


Všimněme si, že rozhodnutí $\mathbf{x}$ a $\mathbf{y}$ jsou závislé pouze na realizaci veličiny ξ v předchozím stupni. Tato velice důležitá vlastnost se v angličtině nazývá *nonanticipativity*, což lze do češtiny přeložit jako *nepředvídatelnost*. Říká nám, že rozhodnutí ve stupni i provádíme vždy dříve, než můžeme pozorovat realizaci náhodné veličiny ξ_i^s.

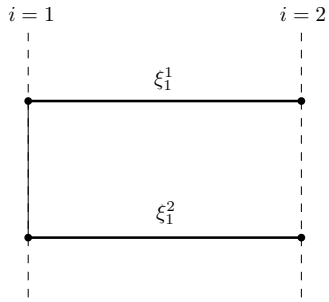
Jednotlivá rozhodnutí nám také vytvářejí specifickou strukturu. Nazýváme ji *strom rozhodnutí (decision tree)*. Například pro $|S| = 2$ získáme strom na obrázku 5.3.2.

Pokud použijeme podmínku neočekávatelnosti jako omezení (*explicit nonanticipativity constraints*), můžeme strom rozhodnutí přepsat na následující tvar *split-variable formulation* znázorněný obrázkem 5.3.2.

Tato vlastnost je velice důležitá, protože nám dovoluje úplnou separaci scénářů a tím pádem umožňuje jejich paralelní výpočet. Podmínku neočekávatelnosti zde reprezentuje tenká svislá čára, která „svazuje“ navzájem si odpovídající proměnné.



Obrázek 5.3: Dvoustupňová úloha - strom rozhodnutí



Obrázek 5.4: Dvoustupňová úloha - strom rozhodnutí (split-variable)

5.3.3 Progressive Hedging Algorithm

Progressive Hedging Algorithm (PHA) je algoritmus pro vícestupňovou stochastickou optimalizaci se scénářovou dekompozicí. Používá se k řešení zejména nelineárních úloh (pro lineární problémy lze využít také, nicméně pro ně je z důvodu rychlejší konvergence vhodnější použít *L-Shaped metodu* [6, s. 555]). Tento algoritmus byl navržen v roce 1989 (resp. 1991) R. Tyrrellem Rockafellarem a R. J-B Wetsem [25]. Algoritmus pracuje na základě dekompozice úlohy na scénáře. V každé iteraci se nezávisle na sobě spočítají jednotlivé scénáře (viz 5.2.4)a jejich řešení se z hlediska pravděpodobnosti nastání daných scénářů zprůměrují. Do účelové funkce scénářů se přidává penalizace, která zvyšuje její hodnotu, pokud se dané lokální řešení liší od průměrného řešení (5.2.4). Chová se tedy jako metrika. Dále se každému lokálnímu řešení přiřazují v každé iteraci váhy, které říkají, jak moc je toto řešení „hodnotově“ blízko průměrného řešení. Algoritmus iteruje až do té doby, než je splněna ukončovací podmínka. V té je pomocí konstant zahrnuto, do jaké míry se liší průměrná řešení mezi dvěma iteracemi a rozdíl mezi průměrným řešením a lokálním řešením ve scénářích.

PHA se díky své robustnosti poměrně intenzivně používá v rozličných aplikacích. Dovoluje nám totiž použít nejen nelineární účelovou funkci, ale zejména nelineární omezení ve formě nerovností. Pokud také máme k dispozici optimalizační solver, který nevyužívá

informaci o gradientech účelové funkce a omezení, nemusí být dokonce tyto ani diferencovatelné! To nám dává obrovskou svobodu při formulaci úloh.

Pro jednoduchost uvedeme algoritmus ve formě pro dvoustupňovou úlohu. Obecně lze nalézt v [4, 25, 23].

Dvoustupňový PHA

Z důvodu větší přehlednosti a názornosti provedeme následující přeznačení. Označme jednotlivé scénáře indexem s, $s \in S$, iterace algoritmu značíme j. Dále $\mathbf{x}$ označme rozhodnutí v prvním stupni a $\mathbf{y}$ rozhodnutí v druhé stupni. Vektorem $w^j(s)$ označme váhy. Matice $\hat{\mathbf{X}}$ je $n \times 2$ matice průměrných řešení, p_s, $s \in S$ váhy scénářů z hlediska jejich pravděpodobnosti.

Inicializace

Zvolme parametr penalizace $\rho > 0$ a ukončovací kritérium $\epsilon > 0$. Označme $|S| = L$. Nastavíme $\hat{x}^0(s) = \mathbf{w}^0(s) = \mathbf{0}_T \ \forall s \in S$ a iterační index $j = 1$.

Hlavní algoritmus

1. Pro každé $s \in S$ spočítejme

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}, \mathbf{y}, s) + \mathbf{w}^{j-1}(s)^T \cdot \mathbf{x} + \frac{1}{2}\rho\|\mathbf{x} - \hat{\mathbf{x}}^{j-1}(s)\|^2, \ \mathbf{x}, \mathbf{y} \in C_s,$$

   a označme řešení jako $\mathbf{X}^j(s) = (\mathbf{x}^j(s), \mathbf{y}^j(s))$.

2. Pro každé $s \in S$ spočítejme $\hat{\mathbf{X}}^j(s) = (\hat{\mathbf{x}}^j, \hat{\mathbf{y}}^j(s))$:

$$\hat{\mathbf{x}}^j(s) = \hat{\mathbf{x}}^j = \sum_{s \in S} p_s \mathbf{x}^j(s),$$
$$\hat{\mathbf{y}}(s) = \mathbf{y}(s).$$

   Pokud ukončovací podmínka

$$\delta = \left(L\|\hat{\mathbf{x}}^{j-1} - \hat{\mathbf{x}}^j\|^2 + \sum_{s \in S} p_s \|\hat{\mathbf{y}}^{j-1}(s) - \hat{\mathbf{y}}^j(s)\|^2 + \sum_{s \in S} p_s \|\mathbf{x}^j(s) - \hat{\mathbf{x}}^j\|^2 \right) \leq \varepsilon,$$

   pak algoritmus zastavíme a $\hat{\mathbf{X}}^j(s) = (\hat{\mathbf{x}}^j, \hat{\mathbf{y}}^j(s))$ je řešením problému s chybou menší než ε. Jinak $\forall s \in S$ spočítejme

$$\mathbf{w}^j(s) = \mathbf{w}^{j-1}(s) + \rho(\mathbf{x}^j(s) - \hat{\mathbf{x}}^j(s)),$$

   nastavíme $j = j + 1$ a přejdeme na další iteraci.

K algoritmu uvedeme pár komentářů. Pro jeho chod je nesmírně důležité vhodně zvolit nastavení parametrů ρ, které ovlivňuje velikost penalizace, a ε, zásadním způsobem rozhodující o přesnosti výsledku a počtu iterací. Všimněme si také, že penalizace má tvar

rozšířeného langrangiánu zmíněného v kapitole 5.3.1. To je velice důležité, neboť právě tento langrangián je důvodem, proč lze přistoupit k úplné dekompozici scénářů. Dalším důležitým poznatkem je, že optimalizační algoritmus voláme nezávisle pouze na jednotlivé scénáře. Díky tomu můžeme, pokud jsou funkce $f(\mathbf{x}, \mathbf{y}, s)$ a jednotlivá omezení diferencovatelná, používat k jejich vyřešení širokou škálu (rychle konvergujících) optimalizačních algoritmů. Nevýhodou algoritmu je velice pomalá celková konvergence. Díky tomu se prakticky nepoužívá pro lineární problémy. Zde upřednostňujeme zejména na lineární modely specializující se *L-Shaped metodu*, která poté vede na *Bendersovu dekompozici* [23].

Hlavním cílem práce je aplikace *PHA* na deterministickou úlohu, ve které se jistým způsobem objevuje analogie náhodnosti. Nicméně stále bude tato analogie deterministická, a tedy nebude důvod „opravovat“ řešení v dalším stupni úlohy z hlediska náhodnosti. Proto si bohatě vystačíme pouze s jednostupňovým PHA algoritmem, který má následující tvar.

Jednostupňový PHA

Inicializace

Zvolme parametr penalizace $\rho > 0$ a ukončovací kritérium $\epsilon > 0$. Nastavíme počáteční hodnoty $\hat{x}^0(s) = \mathbf{w}^0(s) = \mathbf{0}\ \forall s \in S$ a iterační index $j = 1$.

Hlavní algoritmus

1. Pro každé $s \in S$ spočítejme

$$\min_{\mathbf{x}} f(\mathbf{x}, s) + \mathbf{w}^{j-1}(s)^T \cdot \mathbf{x} + \frac{1}{2}\rho\|\mathbf{x} - \hat{\mathbf{x}}^{j-1}\|^2,\ \mathbf{x} \in C_s,$$

   a označme řešení jako $\mathbf{x}^j(s)$.

2. Spočítejme $\hat{\mathbf{x}}^j = \sum_{s \in S} p_s \mathbf{x}^j(s)$.
   Pokud ukončovací podmínka

$$\delta = \left(\|\hat{\mathbf{x}}^{j-1} - \hat{\mathbf{x}}^j\|^2 + \sum_{s \in S} p_s \|\hat{\mathbf{x}}^j(s) - \hat{\mathbf{x}}^j\|^2 \right) \leq \epsilon,$$

   pak algoritmus zastavíme a $\hat{\mathbf{x}}^j$ je řešením problému s chybou menší než ϵ. Jinak $\forall s \in S$ spočítejme

$$\mathbf{w}^j(s) = \mathbf{w}^{j-1}(s) + \rho(\mathbf{x}^j(s) - \hat{\mathbf{x}}^j),$$

   nastavíme $j = j + 1$ a přejdeme na další iteraci.

Pro ilustraci toho, jak *PHA* funguje, uvažujme následující jednoduchý příklad se dvěma scénáři

Příklad (Jednostupňový PHA)

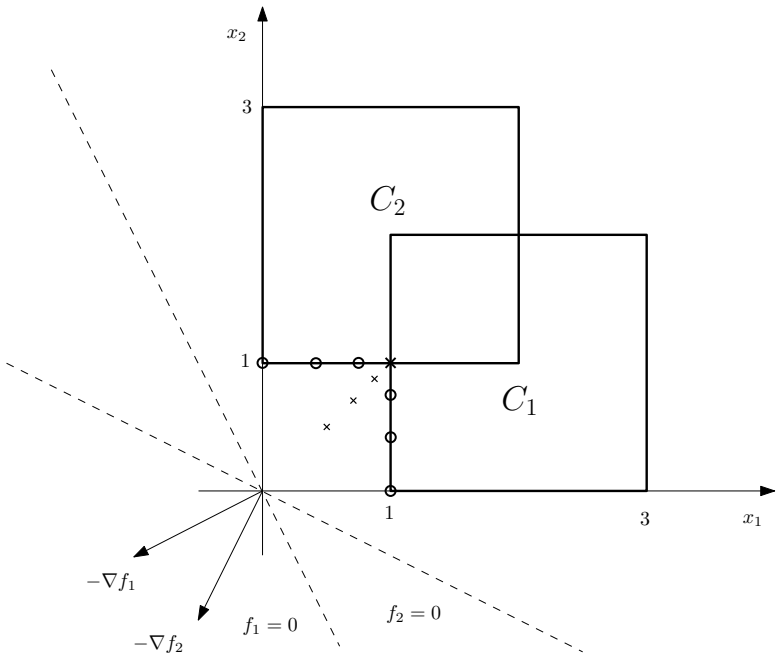
Vyřešte pomocí jednostupňového *PHA* algoritmu úlohu

$$\begin{aligned}\min_{x_1,x_2}\ & p_1 f_1(x_1,x_2) + p_2 f_2(x_1,x_2),\\ f_1(x_1,x_2) =& 2x_1 + x_2,\ 0 \le x_1 \le 2,\ 1 \le x_2 \le 3,\\ f_2(x_1,x_2) =& x_1 + 2x_2,\ 1 \le x_1 \le 3,\ 0 \le x_2 \le 2,\end{aligned}$$

kde p_s jsou váhy ve scénáři s. Uvažujte $p_1 = p_2 = 0.5$.

Řešení. Nejprve označíme oblasti přípustných řešení na scénáři s jako C_s. Dále aplikujeme algoritmus pro jednostupňový PHA, takže máme pouze rozhodnutí prvního stupně, které ovšem uvažujeme různé pro každý scénář s a dle předchozích odstavců značíme $\mathbf{x}(s)$. Máme dva scénáře $s \in S = \{1, 2\}$. Grafická reprezentace je na následujícím obrázku. Oblasti příp. řešení C_s zde tvoří čtverce. Funkce f_1 a f_2 jsou vyznačeny jako řezy rovinou (x_1, x_2). Dále $-\nabla f(s)$, tedy spádové směry, naznačují šipky. Řešení na scénářích ve významných iteracích jsou vyznačena kolečky. Posloupnost odpovídajících řešení $\hat{\mathbf{x}}$ označují křížky. Tato průměrná řešení $\hat{\mathbf{x}}$ jsou vzhledem k $p_1 = p_2 = 0.5$ na ose mezi prázdnými kolečky znázorněnými řešeními $\mathbf{x}(s)^j$.

Vidíme, že algoritmus se snaží minimalizovat účelovou funkci nad jednotlivými C_s nezávisle na sobě. Algoritmus skončí v místě k průniku oblastí C_s, protože zde dostáváme pro oba scénáře nulovou penalizaci. Také je naznačeno, že u $\mathbf{x}(s)^j$ vzdálenějších od získaného opt. řešení je rozdíl $|\mathbf{x}(s)^j - \mathbf{x}(s)^{j-1}|$ větší. Zde se právě projevuje vliv penalizace, která hodnotu účelové funkce pro nepřípustná řešení úměrně s mírou této nepřípustnosti zvyšuje. Čím blíže jsme přípustnému optimálnímu řešení, tím je vliv penalizace na hodnotu účelové funkce menší a blížíme se k němu pomaleji.



Obrázek 5.5: Vizualizace úlohy jednostupňového PHA se dvěma scénáři

6 Optimalizace kontinuálního lití pomocí dekompozice

Hlavním úkolem této práce je prozkoumat, zda lze *Progressive Hedging Algorithm* použít pro zefektivnění výpočtů nutných pro optimální řízení kontinuálního odlévání oceli. Optimální řízení zde budeme reprezentovat úlohou reakce systému chlazení na nespojitý skok licí rychlosti v_z. Tento problém nám umožňuje velice přirozeně navázat na předchozí kapitoly. Samozřejmě se nabízí spousta variantních úloh. Některé z nich, včetně postupů k jejich řešení, jsou k nalezení v [3].

6.1 Formulace úlohy

Jednou z optimalizačních úloh, kterou potřebujeme při řízení procesu kontilití řešit, je náhlá změna licí rychlosti v_{z0}. Díky této změně se začne teplotní pole uvnitř sochoru měnit, až se ustálí v novém stacionárním stavu, odpovídajícímu nové licí rychlosti v_{z1}. Tato událost může mít katastrofální následky. Mohou nastat dva scénáře:

1) $v_{z1} > v_{z0}$, tedy dojde ke zvýšení licí rychlosti. V tomto případě může dojít vlivem příliš rychlého chlazení ke vzniku prasklin uvnitř i na povrchu sochoru. Tyto vady jsou neopravitelné a vedou tedy ke značné ekonomické ztrátě (2.4).

2) $v_{z1} < v_{z0}$, kdy sochor zpomalí. Zde hrozí hlavně prodloužení metalurgické délky sochoru až za hranici oblasti sekundárního (příp. terciálního) chlazení. Tekuté jádro se může dostat až k dělicímu zařízení, což má za následek jednak jeho poničení ulpívajícím materiálem, tak také znehodnocení celého úseku sochoru. Po uříznutí totiž dojde k extrémně rychlému ochlazení jádra, což má za následek vznik prasklin a dokonce vede k změně geometrie sochoru. Znovu se jedná o neopravitelné vady, i když nyní lze alespoň části sochoru, které jsou dostatečně vzdáleny od konců rozděleného úseku, zachránit.

Naštěstí lze v omezené míře výchylky v licí rychlosti kompenzovat. Jedním z nejjednodušších řešení je reagovat změnou intenzity vodního chlazení. V prvním případě, tedy $v_{z1} > v_{z0}$, bychom chlazení zintenzivnili. V druhém případě přichází v úvahu samozřejmě snížení intenzity. Otázkou zbývá, jak zjistit přesné (optimální) hodnoty chladicího systému po změně soustavy. Na ni se pokusíme dát v následujících odstavcích odpověď.

Logicky tedy budeme chtít, aby se teploty v místech snímání před a po změně rychlosti rovnaly. Očíslujme-li jednotlivá čidla indexy $n,\ n = 1, \ldots, N$ a zavedeme funkci

$$f(v_{z0}, v_{z1}) = \sum_{n=1}^{N} |T^n_{v_{z0}} - T^n_{v_{z1}}|,$$

kterou chápeme jako součet rozdílů ve všech kontrolních bodech v absolutní hodnotě. Zřejmě požadujeme, aby tento součet byl co nejmenší. Přirozeně nám vzniká minimalizační úloha s účelovou funkcí $f(v_{z0}, v_{z1})$. Dále víme, že teplotní pole závisí nejen na licí rychlosti, ale také intenzitě chlazení. Jednotlivé samostatně regulovatelné celky tohoto chlazení

můžeme reprezentovat parametrem přestupu tepla. Označme jej $htc_m,\ m = 1, \ldots, M$, kde M je celkový počet chladicích celků. Tedy můžeme zapsat

$$f = f(v_{z0}, htc_{10}, \ldots, htc_{M0}; v_{z1}, htc_{11}, \ldots, htc_{M1}).$$

Tento zápis je ale velice dlouhý a nepraktický. Uvažujme tedy výchozí stav jako pevný a omezme se prozatím pro jednoduchost na případ $M = N = 1$ (úsek s jedním čidlem a jednou soustavou trysek). Dále zrušme nyní zbytečnou indexaci dle m a n. Dostaneme účelovou funkci $f(v_{z1}, htc)$, kde v_{z1} chápeme jako parametr a htc jako proměnnou.

K formulaci úlohy nám chybí omezení. První jejich část tvoří fyzikální a technická omezení. V tomto případě se jedná zejména o intenzitu chladicího účinku trysek, kterou ohraničíme

$$htc_{\min} \leq htc \leq htc_{\max}.$$

Pro dolní omezení zpravidla volíme $htc_{\min} = 0\ \mathrm{Wm^{-2}K^{-1}}$, jenž popisuje vypnutou trysku. Shora htc omezíme $htc_{\max} = 1000\ \mathrm{Wm^{-2}K^{-1}}$.

Další část omezení tvoří závislosti teplot na dvojici $(v_{z\cdot}, htc.)$. Bohužel, tuto závislost nelze vyjádřit funkcí, protože T dostáváme jako výsledek numerického (iteračního) algoritmu 4.2. To působí značné komplikace zejména při výběru algoritmů pro řešení úlohy (5.3.3).

Celou úlohu lze tedy zapsat v následujícím tvaru:

$$\min_{htc} |T_{v_{z0}} - T_{v_{z1}}(v_{z1}, htc)|, \tag{6.1}$$

$$T_{v_{z1}} \leftarrow \mathrm{NUM.MODEL}(v_{z1}, htc), \tag{6.2}$$

$$htc_{\min} \leq htc \leq htc_{\max}, \tag{6.3}$$

kde teplotu v čidle T_{vz1} získáme spočítáním numerického modelu 4 pro konstantu v_{z1} a parametr htc. Jedná se o omezení ve tvaru rovnosti. Tato úloha vypadá na první pohled velice jednoduše. Nicméně musíme si uvědomit, že teplota v čidle je obecně závislá na teplotách v **celém** okolí. Pro získání její hodnoty musíme vždy přepočítat **celý** model, což vede k vysoké výpočetní náročnosti. Navíc optimalizační algoritmus řešící úlohu musí „zkoušet“ jednotlivá htc a sledovat, jak se změní velikost účelové funkce. Při každé nové hodnotě htc se musí celý výpočet znovu přepočítat.

Rozšiřme si nyní tuto úlohu na problém popsaný obrázkem 2.6. Nejprve vytvořme *primární dělení*, stejné jako v numerickém modelu. Dále rozdělme sochor podélně na n_{elem} částí a toto dělení nazvěme *sekundárním dělením*. Z důvodů, které jsou vysvětleny v odstavci 6.2, použijme *přesazenou síť* - ve směru osy z se budou elementy překrývat. Počet průřezů, které elementy s a $s+1$, $s = 1, \ldots, n_{elem}$ sdílejí, označme n_{offset}.

Předpokládejme, že každý element s má svůj vlastní chladicí okruh reprezentovaný koeficientem přenosu tepla htc_s[1]. Předpokládejme, že uvnitř každého elementu bylo umístěno teplotní čidlo.

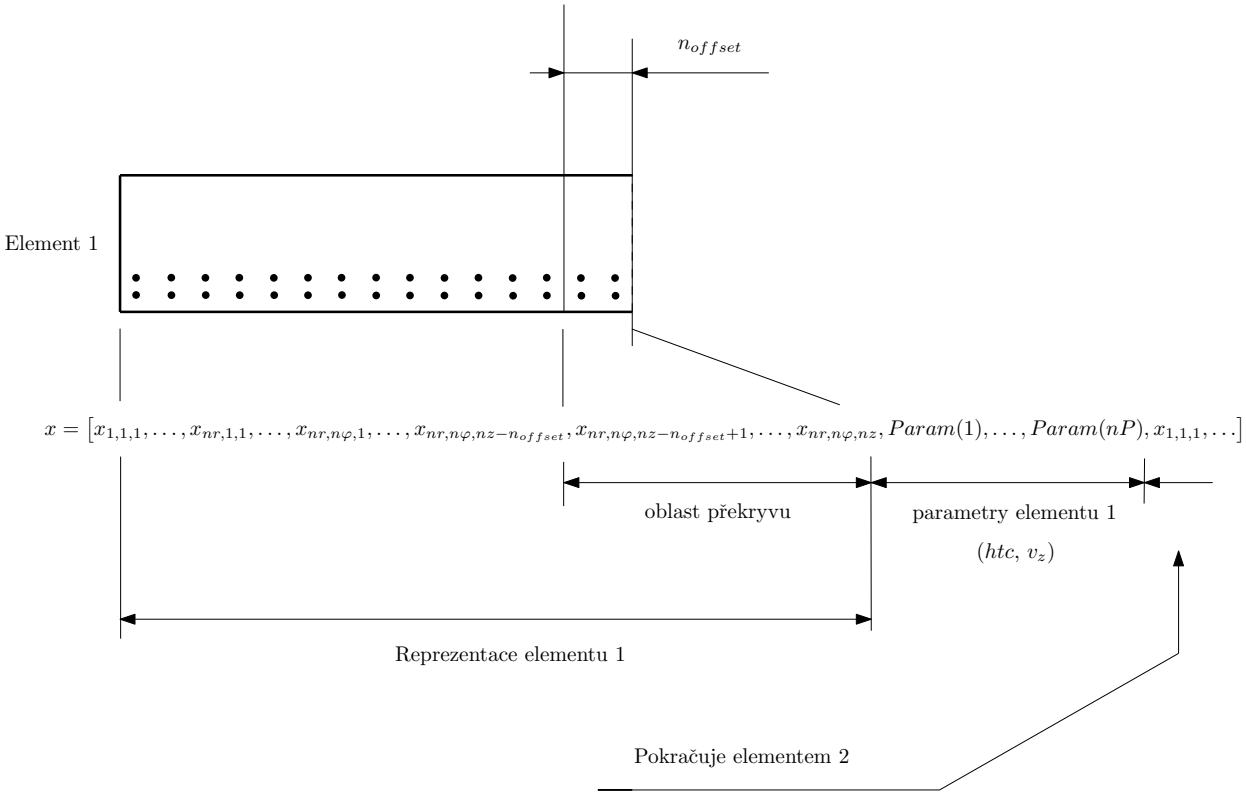
Každý z nich pak můžeme reprezentovat sloupcovým vektorem $\mathbf{x}_s$, jehož prvky jsou

- vektorizované prvky primárního dělení $T_s(r, \varphi, z)$ elementu s,

[1]Zde ovšem nastává problém v oblasti překryvu, která patří doc dvou elementů. Situací vyřešíme tak, že v přesahu prvků s a $s+1$ použijeme htc_s. Za touto volbou je hlubší význam. Úloha znázorněná 2.6 patří mezi tzv. *úlohy s volným pravým koncem*. Na levém okraji ve smyslu souřadnice z známe teplotu přesně. Vlivem OP na plášti sochoru dochází k jejímu snížení, což ovlivňuje teplotu na pravém konci. Proto má větší smysl ztotožňovat htc v přesahu s levým elementem než naopak.

- vektor parametrů $Param = (htc_s, v_z)$.

Symbolem $\mathbf{x}$ pak označme sloupcový vektor vzniklý složením všech $\mathbf{x}_s$. Strukturu dělení a $\mathbf{x}$ popisuje obrázek 6.1.



Obrázek 6.1: PHA struktura vektoru $\mathbf{x}^T$

Zadání problému tedy také rozšíříme. Hledáme vektor $\mathbf{htc} = (htc_1, \ldots, htc_{n_{elem}})$ reprezentujících chlazení takový, že teplotní pole po náhlém výkyvu z ideální licí rychlosti v_{z0} na v_z1 se změní co nejméně.

6.2 Prostorová dekompozice užitím PHA

Hlavním cílem práce je prozkoumat možnost aplikace prostorové dekompozice na úlohu optimalizace teplotního pole při kontinuálním odlévání sochoru kruhového průřezu. Budeme se zabývat zejména otázkami, zda tato dekompozice má z hlediska výpočetní efektivity smysl. Prakticky to znamená, jestli časová úspora, kterou paralelizací získáme, převáží dodatečné režijní náklady, které jejím použitím vzniknou. Práce navazuje na prvotní snahu aplikovat prostorovou dekompozici při výpočtu napětí v nosníku, kterou ve své diplomové práci úspěšně obhájila Ing. Šabartová [28]. Aplikací *Progressive Hedgingu* dokázala výhodnost paralelizace u tohoto problému. Na tuto diplomovou práci navazujeme a využíváme získané výsledky. V určitém směru ji také rozšiřujeme, především rozšířením úlohy na více dimenzí a také aplikací na vyloženě deterministický problém. Na rozdíl od Ing. Šabartové se zabýváme explicitním numerickým algoritmem (na rozdíl od metody konečných prvků použité v [28]), což má pro aplikaci PHA veliký význam.

Zopakujeme zde některé definice z minulého odstavce. Je to čistě z toho důvodu, abychom je nyní uvedli do souvislosti s PHA.

Jak už bylo řečeno, hlavní myšlenkou prostorové dekompozice je rozdělení výpočetní domény na několik částí (*sekundární dělení*). Primárním dělením označujeme diskretizaci použitou v numerickém modelu. Z výsledků [28] ale vyplývá, že to samo o sobě nestačí. Kvůli konvergenci PHA je nutné vytvořit *sekundární síť s překryvem (přesazenou síť)*. Každý ze sekundárních elementů poté považujeme za *scénář*. V této úloze se objevuje jedna specifická vlastnost. Počet scénářů není pro všechny body stejný. U elementů mimo oblast přesazení existuje scénář pouze jeden, naopak pro prvek z oblasti překryvu existují scénáře dva (reprezentace levým a pravým elementem).

Pokud bychom se na problém podívali z pohledu stochastické optimalizace, máme vlastně několik tyčí o velikosti jednoho sekundárního elementu, na které máme náhodné teplotní pole. Jednotlivá data na elementech (teploty v bodech primárního dělení) jsou realizacemi této náhodné veličiny.

Pro naši úlohu je zdaleka nejvýhodnějším způsobem sekundární diskretizace dělení ve směru souřadnice z. Každý element, až na první, bude mít stejné okrajové podmínky v ose z. Víme, že se jedná o úlohu s volným pravým koncem, takže i získáváme výpočetní směr, ze kterého bude úloha nejrychleji konvergovat. Navíc každý element, s výjimkou prvního či posledního, bude mít stejnou velikost. Výrazně nestejný rozsah elementů by způsoboval zpomalení výpočtu.

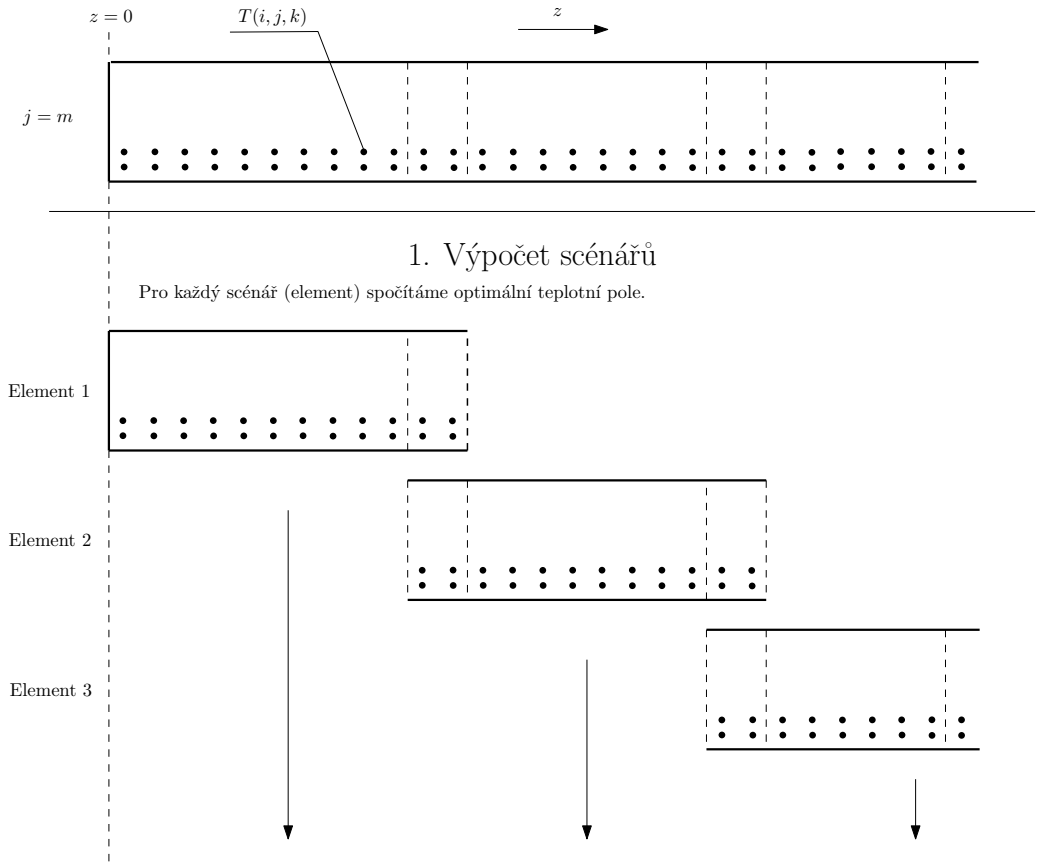
Při aplikaci jednostupňového PHA je nutné pracovat s vektorem proměnných (obrázek 6.1). To je celkem nepříjemná komplikace, protože pro teplotní výpočet zase potřebuje reprezentaci problému jako třídimenzionálního pole. Nedá se tedy mezi numerickým výpočtem a PHA vyhnout převádění proměnných z jedné formy do druhé. Budeme se držet značení z odstavce 5.3.3 s malými úpravami. Vzhledem k tomu, že se jedná o jednostupňovou úlohu, nebudeme používat dolní index ke značení stupně, ale ke značení scénáře. Toto řešení umožní značení zpřehlednit. Elementy sekundárního dělení budeme indexovat s, $s = 1, \ldots, n_{elem}$. Optimalizační úloha bude mít tvar velice podobný tvar jako (6.1). Díky penalizaci v účelové funkci je ale nutné pracovat s vektorizovanou primární diskretizací. Kontrolní body elementu s označíme pro přehlednost $x_{T,s,vz\cdot}$:

$$\min_{htc} |x_{T,s,vz0} - x_{T,s,vz1}| + \mathbf{w}_s^T \mathbf{x}_s + \frac{\rho}{2} \|\mathbf{x}_s - \hat{\mathbf{x}}_s\|^2, \tag{6.4}$$

$$x_{T,s,vz1} \leftarrow \text{NUM.MODEL}(v_{z1}, htc_s), \tag{6.5}$$

$$htc_{s,\min} \leq htc_s \leq htc_{s,\max}, \tag{6.6}$$

kde $\mathbf{w}_s^T$ značí váhovou funkci PHA pro element s, $\mathbf{x}_s$ je vektor proměnných odpovídající elementu s (popsán na obrázku 6.1) a $\hat{\mathbf{x}}_s$ vektor průměrných řešení. Parametr ρ určuje „míru“ penalizace. Algoritmus sestává ze dvou částí. Jako první musíme v každé iteraci spočítat nové optimální numerické pole, založené na datech získaných z předchozích iterací (obrázek 6.2).



Obrázek 6.2: První výpočetní fáze PHA - diskretizace, scénáře

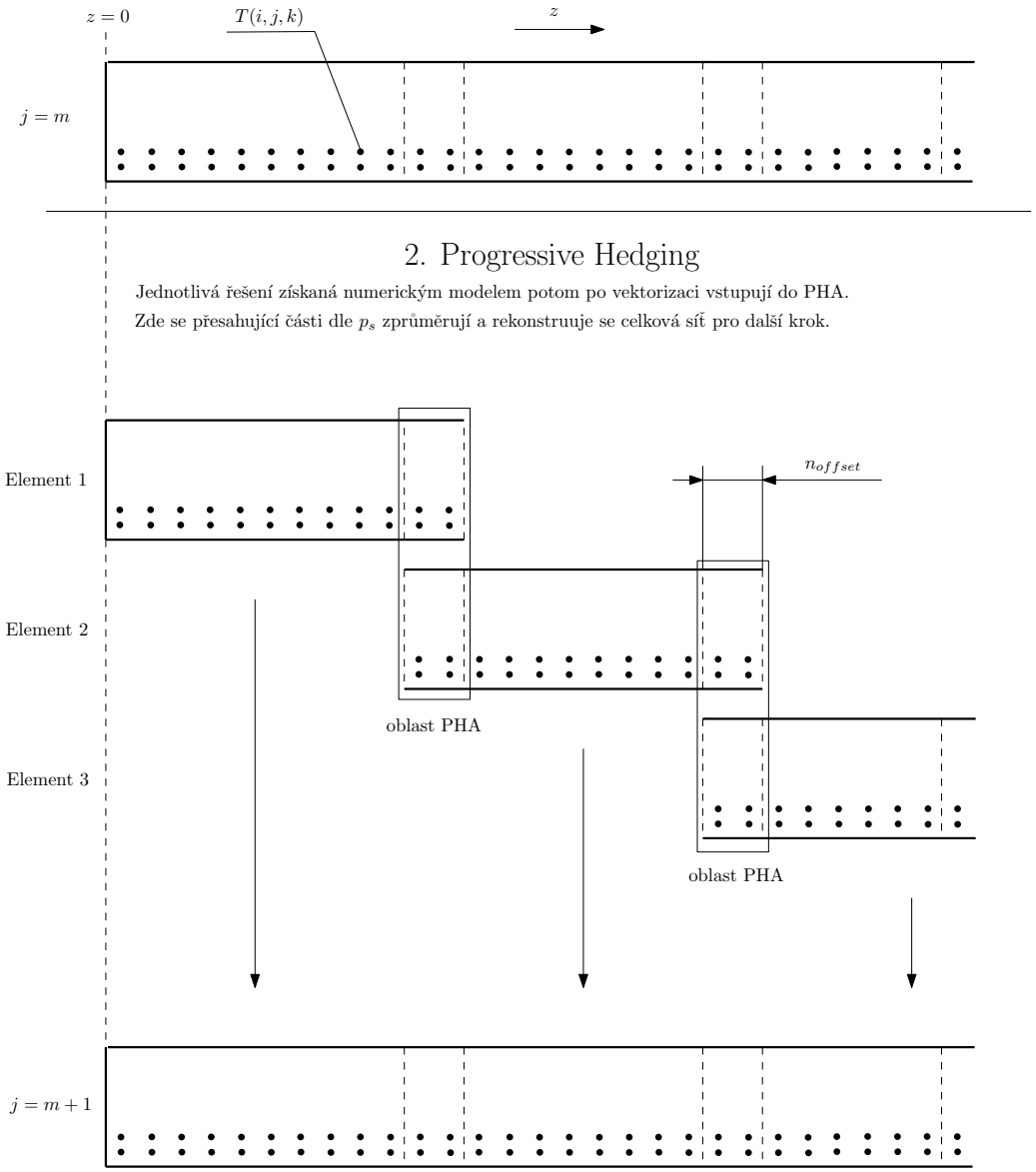
Dále po výpočtu jednotlivých scénářů postupujeme dle algoritmu 5.3.3. Spočítáme pro každou oblast překryvu $\hat{\mathbf{x}}_s$[2] a dosadíme je na příslušná místa vektoru $\mathbf{x}$. Upravíme váhy, vyhodnotíme ukončovací kritérium a provedeme aktualizaci vektorů předchozí iterace. Musíme uvědomit, že algoritmus počítá numerický výpočet a zároveň optimalizační úlohu. Proto budou ze začátku teplotní rozdíly v čidlech v řádu desítek až stovek stupňů, než se dostaneme na odpovídající hodnoty teplotního pole pro rychlost v_{z1} a optimalizace začne dávat smysluplné výsledky..

6.3 Paralelizace

Jak už bylo řečeno, tato optimalizační úloha je z důvodu přepočtů teplot při optimalizaci velice výpočetně náročná. Přirozeně vyvstává otázka, zda by nešlo výpočet nějak zefektivnit. Zde se nabízí dvě cesty:

- Zvýšení efektivity numerického výpočtu.
- Zvýšení efektivity optimalizace.

[2]Musíme si uvědomit, že licí rychlost ve všech elementech je stejná. V PHA to musíme zaručit vynětím z optimalizace na scénářích a poté sdružením všech rychlostí z jednotlivých scénářů do jednoho místa vektoru $\mathbf{x}$. Nejlogičtější volbou je jeho konec. Pokud bychom to neudělali, dostaneme naprosto odlišné a samozřejmě chybné výsledky.



Obrázek 6.3: Druhá výpočetní fáze PHA

Efektivitě numerického výpočtu se věnujeme v 4.2. Zvýšení rychlosti optimalizace lze dosáhnout zejména dobrou volbou optimalizačního algoritmu. Důležitá není pouze rychlost výpočtu (u iteračních solverů konvergence), ale i vhodnost pro danou úlohu (specializované algoritmy).

Při zvyšování efektivity výpočtů ale existuje i další cesta. Jde o myšlenku *paralelizace*. Stručně řešeno, se jedná o přístup, kdy se snažíme rozdělit celkový objem výpočtů na několik částí, které poté počítáme zároveň. Teoreticky lze tedy dosáhnout pro n výpočetních jednotek n-násobného snížení časové náročnosti.

Tento přístup je ale ze své podstaty z hlediska aplikace významně omezen. Má smysl jej použít pouze pro úlohy, ve kterých se vyskytují svou strukturou navzájem identické podúlohy (*dílčí* problémy), které se liší pouze daty. Nutným předpokladem je také určitá izolovanost, dílčí problémy nesmí být provázané.

Ideálním příkladem, na kterém můžeme demonstrovat potřebné vlastnosti, je násobení matic Mějme následující matice $A(m \times n)$, $B(n \times p)$, $C(m \times p)$.

Naším úkolem je spočítat součin $C = A \cdot B$. Prvek součinové matice c_{jk} dostaneme dle

$$c_{jk} = \sum_{i=1}^{n} a_{ji} b_{ik},$$

Vidíme tedy, že pro prvek c_{jk} potřebujeme znát pouze j-tý řádek matice A a k-tý sloupec matice B. Nepotřebujeme pro jeho výpočet znát celou matici. Tím pádem lze celý výpočet C rozložit na několik subproblémů, které po svém vyřešení složí matici C. Toto má obrovský význam, protože prakticky veškerá numerická matematika využívá ve formulaci úloh násobení velkých matic. Paralelizace maticového násobení je implementována prakticky ve všech softwarech, ať už implicitně (Matlab) či pomocí různých knihoven, často založených na Fortranu (LAPACK, BLAS, OpenBlas...).

Toto je jeden z důvodů, proč se také snažíme při numerických výpočtech upřednostňovat implicitní řešení či skládání lineárních zobrazení. Tyto úlohy se totiž dají převést na násobení matic.

Lze ale paralelizovat i vybrané úlohy, které nejsou úplně dekomponované. Pokud taková provázanost existuje, pak ji musíme nějakým způsobem obejít. Zde se ukazuje veliká síla PHA, který nezávislosti scénářů dosahuje poměrně jednoduchým způsobem.

Předpokládáme, že nastanou scénáře 1 a 2. Každý z těchto scénářů má pravděpodobnost realizace. Předpokládejme, že vyřešením optimalizační úlohy na scénářích dostáváme různé vektory $\mathbf{x}_1$ a $\mathbf{x}_2$. Vypočítáme průměrné řešení $\hat{\mathbf{x}}$. Spočítáme pro každý scénář váhovou funkci, která nám říká, jak moc se liší od průměrného řešení. Ty x_s, které jsou hodnotami k $\hat{\mathbf{x}}$ nejblíže, mají hodnoty vah nejvyšší. Tímto způsobme iterujeme pořád do té doby, než je splněna ukončovací podmínka, ve které se vyskytuje velikost změny průměrného řešení mezi iteracemi a také rozdíl řešení průměrného a těch na scénářích. Provázanost úloh se tedy vyskytuje mimo optimalizační úlohu, která je na výpočet nejsložitější a tím způsobem můžeme jednotlivé scénáře paralelizovat. Výpočty PHA algoritmu jsou v porovnání s optimalizací triviální.

Násobení matic je typickým příkladem *prostorové dekompozice (spatial decomposition)*, jejímž hlavním principem je dělení oblasti, se kterou počítáme. V optimalizaci se často používá *scénářová dekompozice (scenario decomposition)* [18].

6.3.1 Možnosti paralelizace

Z hlediska možnosti volby typu hardwaru, který budeme chtít použít, se nabízí zejména *grafické karty (GPU)* a *procesory (CPU)*. Grafické karty se skládají ze stovek jednoduchých výpočetních jednotek. Díky tomu se hodí zejména pro výpočty, ve kterých se vyskytuje spousta jednodušších dílčích úloh. Oproti tomu procesory mají pouze jednotky až desítky jader. Jejich výkon je oproti výpočetním jednotkám GPU řádově vyšší. Hlavním rozhodovacím parametrem je tedy zřejmě náročnost dílčí úlohy.

K praktické implementaci paralelizace si lze vybrat několik druhů softwaru. Pro grafické karty (dnes už i procesory) se nejčastěji používá programovací jazyk *OpenCL*. Jedná se o obdobu jazyka C, resp. C++, která je přizpůsobena architektuře grafických karet (speciální typy proměnných, struktura programování, extenzivní podpora zpracování obrazu apod.). Velkou jeho výhodou je nezávislost na platformě a hardwaru. Jeho variantou je *CUDA* od společnosti NVidia. V mnohém je tento jazyk lepší, nicméně jeho podpora je omezena na grafické karty od této společnosti.

Pro procesory se nabízí hlavně protokol *MPI (Message Parsing Interface)*, resp. jehož open-source implementace *OpenMPI*. Strukturou je také podobné jazyku C++. Nabízí opravdu široké možnosti synchronních i asynchronních výpočtů na více jádrech či CPU. Další známá platforma se nazývá *OpenMP* [14], jejíž výhodou je velice jednoduché používání. Za to se ale platí omezením na výpočty pouze na jednom počítači (pracuje totiž se sdílenou pamětí (*shared memory*). Pro demonstrační příklad v této práci ale OpenMP plně dostačuje a proto byl vybrán jako prostředek, kterým budeme úlohu paralelizovat.

6.4 Výsledky

Řešená úloha byla naprogramována v jazyce C++ využitím knihovny Armadillo (B). Z důvodu snazší kompilace knihoven jsme jako vhodnější systém vybrali linuxové prostředí Linux Mint založené na Ubuntu[3]. Program je podrobněji popsán v příloze D.

Tabulka 6.1: Parametry dekompoziční úlohy

Technické parametry		Numerické parametry	
Poloměr sochoru	0,1 m	Délka výpočetní oblasti	20 m
Délka krystalizátoru	1 m	dr_0	0,01 m
Délka terciálního chlazení	5 m	$d\varphi_0$	$\frac{\pi}{2}$
Licí teplota	1550°C	dz_0	0,1 m
Teplota okolí	25°C	$d\tau$	0,5 s
Rozsah *htc* trysek	$250 - 1000\,\mathrm{Wm^{-2}K^{-1}}$	Počet iterací	5000
Parametry dekompozice		**Optimalizační parametry**	
Licí rychlost v_{z0}	$1{,}0\,\mathrm{mmin^{-1}}$	Krok htc_{step} opt. alg.	50
Licí rychlost v_{z1}	$1{,}2\,\mathrm{mmin^{-1}}$	Počet iterací PHA	dle typu
Pův. *htc* trysek	$500\,\mathrm{Wm^{-2}K^{-1}}$	Penalizace ρ	0
Počet scénářů n_{elem}	4	Ukončovací parametr ε	$\emptyset$
Počet řezů z v elementu	60		
Překryv elem. s a $s+1$	25%		

Pro výpočet optimalizační úlohy na elementech (konkrétně omezení daných diferenciální rovnicí vedení tepla) se využívá třída pro výpočet numerického modelu, který byl odvozen v kapitole 4. Pro optimalizaci potřebujeme pouze mírně odlišné parametry metod.

Parametry optimalizační úlohy shrnuje tabulka 6.1. Nejprve si pro parametry „ideálního“ nastavení systému vygenerujeme n_{elem} kontrolních bodů umístěných po jednom

[3]Defaultně je přítomen kompilátor gcc. U Windows je buď nutné nainstalovat MinGw, nebo použít zpoplatněnou verzi Microsoft Visual Studio.

v každém elementu těsně pod povrchem[4]. Během výpočtu PHA se snažíme na jednotlivých scénářích minimalizovat odchylky teplot vzniklé po skokové změně $v_{z0} \to v_{z1}$.

Po spočtení úlohy dostáváme optimální parametr chlazení, které jsou společně s rozdíly teplot v čidlech shrnuty v tabulce 6.2.

Tabulka 6.2: Výsledky pro úlohu 6.1

Model	Paralelní				Sériový			
Čidlo	1	2	3	4	1	2	3	4
htc	600	700	600	250	600	700	600	250
ΔT	-3	4	-1	4	-3	3	-2	3

Vidíme, že pro parametry 6.1 dosahujeme v 6.2 rozdílu teplot mezi stavy 0 a 1 v řádu jednotek. Výjimkou může být poslední element, jež se nachází většinou v zóně terciálního chlazení, a tedy nemůžeme jeho chlazení příliš regulovat. Musíme si navíc uvědomit, že zvýšením licí rychlosti nedochází pouze k posunu teplotního pole, ale spíše k deformaci. Proto i nulový rozdíl teplot v místě čidel nebude znamenat, že je metalurgická délka stejná. Vždy dojde k její změně, která ovšem stále závisí na teplotě v okrajích. Ke zjištění této závislost by bylo možné využít například regresního modelu založeného na numerických simulacích.

Jednou z otázek, které má tato práce za cíl zodpovědět, je možnost aplikace PHA na deterministický problém prostorové dekompozice. Na tuto otázku nyní můžeme odpovědět kladně. Algoritmus bylo ale nutné kvůli přítomnosti více oblastí dekompozice mírně upravit (6.2).

V případě použití PHA se projevila velice pomalá celková konvergence řešení. Ani po 40 iteracích jsme nedosáhli požadovaného stavu. OpenMP nám sice umožňuje pro $nElem = 4$ počítat na počítači s čtyřjádrovým procesorem jednotlivé scénáře zcela paralelně, nicméně stále jsme výpočet nedokončili. Velkým problémem byl vznik oscilací v posloupnosti spočítaných řešení v jednotlivých iteracích. Dále se objevilo velice problematické chování vah PHA. Způsobovaly nestability v numerickém řešení.

Vzhledem k tomu, že víme, že námi řešená úloha je *úlohou s volným pravým koncem*, nabízí se otázka, zda by nešlo PHA upravit tak, aby tuto vlastnost bral v úvahu. Ukázalo se, že pokud předepíšeme na levou OP elementů hodnoty získané PHA (jako Dirichletovu podmínku) místo podmínky nulového toku, konvergenci algoritmu podstatně urychlíme. Řešení jsme dostali už po 10 až 20 iteracích. Výše zmíněné oscilace prakticky vymizely. Stále byl program počítán paralelně. Nevýhodou je vznik nepřesností na hranicích elementů (viz obrázek 6.4).

Poslední testovanou variantou se stala sériová verze, kdy jsme pro levou OP elementu s použili příslušnou hodnotu z elementu $s-1$ z té stejné iterace PHA. Nepoužíváme už tedy průměrných řešení, ale přímo hodnoty „zleva“. Takto upravený algoritmus už přestává být PHA. Nicméně testování ukázalo, že algoritmus konverguje velice blízko k řešení už ve 3. iteraci, bez jakýchkoliv oscilací. konečný výsledek dostáváme v 8. iteraci. Na druhou

[4]V praxi se čidla neumísťují na povrch, protože by velice silně ovlivnila průběh chlazení, docházelo by zde ke skokové změně mat. vlastností. Navíc by díky omývání chladicí kapalinou docházelo ke zkreslování výsledků. V reálných provozech se pak k měření používají odlišné metody, které jsou například popsány v [19].

stranu výpočty musí probíhat sériově, a proto zabere každá iterace mnohem více času než v předchozích případech. Tato vlastnost nám také umožňuje snížit pro tuto úlohu překryv elementů až na singulární (jeden řez v souřadnici z)[5].

Porovnání jednotlivých metod lze nalézt v tabulce 6.3.

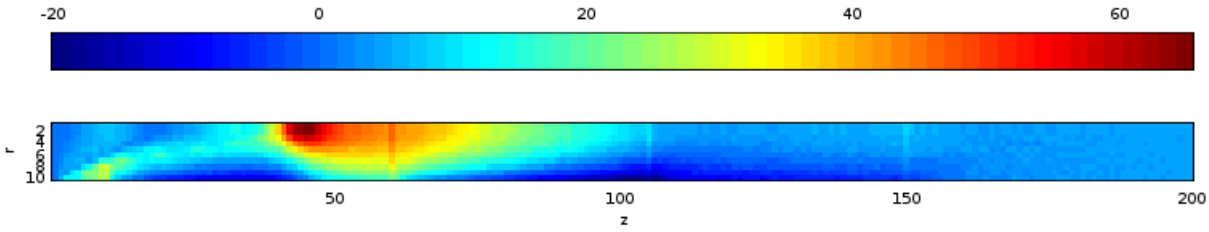
Tabulka 6.3: Porovnání délky výpočtu modelů

	Paralelně	Počet iterací	Čas výpočtu
Upravený PHA	ano	16	68, 72 min
Sériový výpočet	ne	8	113, 95 min

V případě takto rychlé konvergence se ukázalo, že penalizace formou rozšířeného lagrangiánu je v úloze vlastně naprosto zbytečná, protože konvergenci nijak neurychlí. Naopak, může se stát, že tento člen ovlivní výpočet negativním způsobem (rozkmitání). Proto volíme parametr $\rho = 0$.

Na obrázku 6.4 je vykreslen rozdíl průřezů $T_{v_{z1}} - T_{v_{z0}}$. Vidíme, že naše domněnky o tom, že změnou rychlosti nedojde pouze k posunutí, ale i změně tvaru teplotního pole, se potvrdily. I přes velice dobrou shodu v kontrolních bodech se nám znatelně prodlouží metalurgická délka odlitku. Na obrázku si lze všimnout i „přechodů“ v oblastech navázání elementů. To je způsobeno aplikací Dirichletovy okrajové podmínky na začátku elementů. Tato chyba všem nastává pouze pro jeden z řez v elementu a proto celkový výsledek příliš neovlivní. Při výpočtu detailnějších modelů s jemnější sítí by pak byla naprosto zanedbatelná. Chybu lze velice snadno odstranit interpolací z okolních teplot.

Omezení htc na rozpětí $250 - 1000$ $\mathrm{Wm^{-2}K^{-1}}$ se ukázalo být jako dobrou volbou v zamezení získání extrémních htc v navazujících elementech (místo situace $0 - 1000 - 0$ dostaneme například $250 - 700 - 250$). Pokud bychom toto neudělali, může na rozhraní elementů dojít ke vzniku míst s vysokými gradienty teplot a tím numerickým chybám (lokální ztráty stability numerického řešení).



Obrázek 6.4: Rozdíl teplotních polí stavů v_{z0} a v_{z1} (bez postprocessingu)

Výpočet probíhal na stolním počítači s čtyřjádrovým procesorem Intel Core i5-3470, 3.20 GHz.

[5] V [28] se prostorová dekompozice používá pro metodu konečných prvků a zejména pro model s oběma konci pevnými. V tomto modelu je nutný dosti veliký přesah, protože řešení v jednom bodě závisí na okolních hodnotách ve všech směrech prakticky stejně silně. V úloze s volným koncem toto neplatí. U implicitních metod je tato závislost ještě silnější.

7 Závěr

V této práci byl ukázán příklad využití prostorové dekompozice na úlohu optimálního řízení procesu plynulého odlévání oceli. Za tímto účelem byly propojeny rozličné části aplikované matematiky: numerické metody, optimalizace a v menší míře statistika.

První část práce se zabývá modelováním teplotního pole kontinuálního odlévání oceli a příslušným numerickým řešením pomocí metody konečných diferencí. Pro mnoho aplikací může být zajímavé použití cylindrických souřadnic pro nekonstantní dělení výpočetní domény. Velkou hodnotu má také naprogramovaný software pro výpočet modelu, ve kterém je několik zajímavých technických řešení zvyšujících výpočetní efektivitu algoritmu. Software zvládá simulaci plně trojrozměrného nesymetrického problému.

Ukázali jsme, že pro popis daného problému lze diferenční metodu s velice dobrými výsledky použít. Efektivní implementací v jazyce C++ se dosahuje výsledků srovnatelných s pokročilým softwarem.

V druhé části práci se věnujeme prostorové dekompozici. K jejímu provedení používáme nástroj stochastické vícestupňové optimalizace, Progressive Hedging Algorithm. Jeho aplikací jako heuristiky na deterministický problém jsme ukázali možnost rozdělení výpočetní domény na více části a následné paralelizace výpočtů, která umožňuje časové zefektivnění celé úlohy. Podařilo se nám danou úlohu úspěšně vyřešit a navíc také ukázat, že do jisté míry upravený PHA lze použít.

Vedlejším výsledkem je možnost použití prostorové dekompozice přímo pro výpočet numerického modelu.

Problematika kontinuálního odlévání oceli se stále intenzivně studuje na Energetickém ústavu Fakulty strojního inženýrství VUT v Brně. V současnosti je předmětem dalších vypsaných diplomových a dizertačních prací.

Na tuto práci lze navázat více způsoby. V numerickém modelu se jedná zejména o zapracovaní složitějších podmínek chladicího systému - rozšíření o posunovací válce, reprezentaci chladicího účinku trysek náhodným rozdělením či rozšířením programu pro výpočet o grafické rozhraní. V případě dekompozice se jedná o naprogramování paralelizace na výpočetních clusterech či grafických kartách.

Seznam použitých zdrojů

[1] BAZARAA, M. a John J. JARVIS. *Linear programming and network flows*. New York: Wiley, 1977, 565 s. ISBN 04-710-6015-1.

[2] BAZARAA, M., Hanif D. SHERALI a C. SHETTY. *Nonlinear programming: Theory and algorithms*. 3. vydání. Hoboken: John Wiley, 2006, 853 s. ISBN 04-714-8600-0.

[3] BETTS, John T. *Practical methods for optimal control using nonlinear programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2001, 190 s. ISBN 08-987-1488-5.

[4] BIRGE, John R. a François LOUVEAUX. *Introduction to stochastic programming*. 2. vydání. New York: Springer, 2011, 485 s. Series in operations research and financial engineering (Springer). ISBN 978-1-4614-0236-7.

[5] BRITISH IRON AND STEEL RESEARCH ASSOCIATION. NOMENCLATURE OF CONTINUOUS CASTING DEFECTS GROUP. *Definitions and causes of continuous casting defects*. London: The Iron and Steel Institute, 1967, 39 s.

[6] BRANDIMARTE, Paolo. *Numerical methods in finance and economics: A MATLAB-based introduction*. 2 vydání. Hoboken, N.J.: Wiley Interscience, 2006, 669 s. ISBN 04-717-4503-0.

[7] ÇENGEL, Yunus A. a Afshin J. GHAJAR. *Heat and mass transfer: Fundamentals and applications*. 4. vydání. New York: McGraw-Hill, 2011, 902 s. ISBN 978-0-07-131112-0.

[8] CRAMB, Alan W. Making, *Shaping and Treating of Steel: Casting volume*. 11. vydání. Pittsburgh, PA: AIST, 2003. ISBN 09-307-6704-7.

[9] ČERMÁK, Libor. *Numerické metody pro řešení diferenciálních rovnic* [online]. Brno: CERM, 2015, 81 s. [cit. 2015-05-06]. Dostupné z: http://www.math.fme.vutbr.cz/download.aspx?id_file=3934. Skriptum. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[10] ČERMÁK, Libor a Rudolf HLAVIČKA. Numerické metody. Brno, [2015]. Dostupné také z: http://www.math.fme.vutbr.cz/download.aspx?id_file=3940. Skriptum. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[11] EATON, John W. *Octave* [online]. [1998] [cit. 2015-05-08]. Dostupné z: http://www.gnu.org/software/octave/

[12] Eclipse CDT. *Eclipse* [online]. [2001] [cit. 2015-05-06]. Dostupné z: http://eclipse.org/cdt/

[13] Franců, J.: K nedožitým osmdesátinám profesora Zlámala. *Pokroky matematiky, fyziky a astronomie*. 49 Brno, 2004, č. 4, str. 345-347.

[14] GATLIN, Kang Su a Petr ISENSEE. OpenMP and C++: Reap the Benefits of Multithreading without All the Work. *MSDN Magazine* [online]. 2005, **20**(no. 5) [cit. 2015-05-26]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/cc163717.aspx

[15] HARDIN, Richard A., Kai LIU, Atil KAPOOR a Christoph BECKERMANN. A Transient Simulation and Dynamic Spray Cooling Control Model for Continuous Steel Casting. *Metallurgical and materials transactions B*. Warrendale, PA: Minerals, Metals, 2003, **34B**(3). ISSN 1073-5615. Dostupné také z: http://user.engineering.uiowa.edu/~becker/documents.dir/hardin_cctransient.pdf

[16] INCROPERA, Frank P. *Fundamentals of heat and mass transfer*. 6. vydání. New York: John Wiley, 2007, 997 s. ISBN 04-714-5728-0.

[17] JOHNSON, Steven G. ABINITIO. *The NLopt nonlinear-optimization package* [online]. 2007 [cit. 2015-05-06]. Dostupné z: http://ab-initio.mit.edu/wiki/index.php/NLopt

[18] KLIMEŠ, Lubomír. *Stochastic Programming Algorithms*. Brno, 2010, 95 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[19] KLIMEŠ, Lubomír. *Optimalizace parametrů sekundárního chlazení plynulého odlévání oceli*. Brno, 2014, 192 s. Dizertační práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[20] KUREŠ, Miroslav. *Matematická analýza II: Vázané extrémy* [online]. Brno, 2008 [cit. 2015-05-09]. Dostupné také z: http://mathonline.fme.vutbr.cz/download.aspx?id_file=936. Přednáška. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[21] LUKÁČOVÁ-MEDVIĎOVÁ, Mária. *Numerical modelling in computational fluid dynamics*. Hamburg, 2003. Skriptum. Technical university Hamburg-Harburg.

[22] MAUDER, Tomáš. *Optimalizace bramového plynulého odlévání oceli za pomoci numerického modelu teplotního pole*. Brno, 2012, 150 s. Dizertační práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[23] POPELA, Pavel. *An Objected-Oriented Approach to Multistage Stochastic Programming*. Praha, 1998. Dizertační práce. Karlova univerzita.

[24] POPELA, Pavel. *Stochastic programming*. Brno, 2004, 72 s. Přednáška. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[25] ROCKAFELLAR, R. T. a Roger J.-B. WETS. Scenarios and Policy Aggregation in Optimization Under Uncertainty. *Mathematics of Operations Research*. 1991, **sv. 16.**(vydání 1.): s.119-147. DOI: 10.1287/moor.16.1.119.

[26] SANDERSON, Conrad. *Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments*. Technical report, NICTA [online]. 2010 [cit. 2015-05-06]. Dostupné také z: http://arma.sourceforge.net/armadillo_nicta_2010.pdf

[27] API Reference for Armadillo 5.100. SANDERSON, Conrad a Ryan CURTIN. *Armadillo C++ linear algebra library* [online]. 2008 [cit. 2015-05-06]. Dostupné z: http://arma.sourceforge.net/docs.html

[28] ŠABARTOVÁ, Zuzana. *Spatial decomposition for differential equation constrained stochastic programs*. Brno, 2012, 92 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství.

[29] ŠTĚTINA, Josef. *Dynamický model teplotního pole plynule odlévané bramy*. Ostrava, 2007. Dostupné také z: http://ottp.fme.vutbr.cz/users/stetina/disertace. Dizertační práce. Vysoká škola báňská - Technická univerzita Ostrava, Fakulta metalurgie a materiálového inženýrství.

Seznam použitých zkratek a symbolů

Symbol	Jednotka	Popis
$\Delta x, \Delta\varphi, \Delta z$	m	Vektor prvků dělení v souřadnici x, φ, z
ϵ	–	Poměrná zářivost (emisivita)
H	Jm^{-3}	Měrná entalpie
htc	$\mathrm{Wm}^{-2}\mathrm{K}^{-1}$	Koeficient přestupu tepla
htc_n, htc_r	$\mathrm{Wm}^{-2}\mathrm{K}^{-1}$	Koeficient přestupu tepla konvekce, radiace
htc_k, htc_s	$\mathrm{Wm}^{-2}\mathrm{K}^{-1}$	Koeficient přestupu tepla v krystalizátoru, ve scénáři
k	$\mathrm{Wm}^{-1}\mathrm{K}^{-1}$	Tepelná vodivost
L	m	Délka výpočetní oblasti
n_r, n_φ, n_z	–	Počet prvků výpočetní sítě
σ	$\mathrm{Wm}^{-2}\mathrm{K}^{-1}$	Stefan-Boltzmannova konstanta
φ	rad	Úhlová souřadnice
$\dot{q}$	Wm^{-2}	Hustota tepelného toku
$\dot{Q}$	W	Tepelný tok
r	m	Souřadnice vzdálenosti od osy sochoru
R	m	Poloměr sochoru
T	$°C, K$	Teplota
T_∞	$°C, K$	Teplota okolí
T_{in}, T_k	$°C, K$	Teplota v mezipánvi, v krystalizátoru
τ	s	Časový krok
v_z	ms^{-1}	Licí rychlost
z	m	Souřadnice vzdálenosti od začátku krystalizátoru
$C.$	–	Množina přípustných řešení
$f(\cdot)$	–	Účelová funkce proměnných $\cdot$
$\mathbf{g}$	–	Vektor omezení ve tvaru nerovnosti
$\mathbf{h}$	–	Vektor omezení ve tvaru rovnosti
$\mathbf{p}$	–	Vektor parametrů optimalizace
p_s	–	Váhy ve scénáři s
$\mathcal{Q}(\mathbf{x})$	–	Penalizace (průměrná cena penalizace)
$Q(\mathbf{x}, \xi)$	–	Cena penalizace
ρ	–	Penalizační parametr
$\mathbf{w}$	–	Vektor vah
$\mathbf{x}$	–	Vektorová proměnná opt. úlohy
$\mathbf{x}$	–	Rozhodnutí v první fázi dvoustupňové opt.

Symbol	Jednotka	Popis
$\hat{\mathbf{x}}$	–	Průměrné řešení
$\mathbf{x}^*$	–	Přípustné řešení scénářové úlohy
ξ	–	Náhodná veličina
$\mathbf{y}$	–	Rozhodnutí v druhé fázi dvoustupňové opt.
$T^n_{v_{z\cdot}}$	$°C, K$	Teplota v n-tém čidle
$\mathbf{x}_s$	(různé)	Vektor proměnných odpovídající s-tému elementu
$x_{T,s,v_{zi}}$	$°C, K$	Teplota v čidle v elementu s

Zkratka	**Popis**
ZKO	Zařízení na kontinuální odlévání
FDM	Metoda konečných diferencí
FEM, MKP	Metoda konečných prvků
FVM	Metoda konečných objemů
OP	Okrajová podmínka
EO	Expected Value Objective
EV	Expected Value
HN	Here and Now
KKT	Karush-Kuhn-Tuckrovy podmínky
PHA	Progressive Hedging Algorithm
WS	Wait-and-See
MPI	Message Parsing Interface

Seznam obrázků

Seznam tabulek

A Optimalizační algoritmus

Optimalizační úloha (6.1) resp. (6.4) je z hlediska řešení velice problematická. Na první pohled je patrná zjevná nelinearita omezení a účelové funkce. Musíme se tedy omezit na použití nelineárních solverů.

Jedná se o tzv. *opt. problém s omezeními danými diferenciální rovnicí*. V případě, že by tuto rovnici bylo možné řešit analyticky, nenastává žádný problém a můžeme použít standardní solvery. Ovšem ve spoustě případů analytické řešení z různých důvodů není možné (nekonstatní materiálové vlastnosti, složité okrajové podmínky apod.) Musíme tedy použít numerické metody. Tím pádem ale ztrácíme velice důležitou vlastnost, na které stojí naprostá většina solverů - diferencovatelnost, a tedy znalost spádových směrů. Tento fakt vylučuje použití naprosté většiny opt. algoritmů. Světlými výjimkami jsou například algoritmy CONOPT z prostředí GAMS či COBYLA z balíčku NLOpt [17]. Bohužel, ztrátou informace o spádových směrech ztrácíme podstatnou informaci o poloze opt. řešení. Výše zmíněné algoritmy různými způsoby tento deficit obcházejí. Nelze se ale vyhnout alespoň částečné „full-enumeraci“ - výpočtu na určité množině bodů uvnitř oblasti přípustných řešení, ze které se poté snažíme interpolací či jinými, sofistikovanějšími, způsoby zjistit polohu opt. řešení. Tyto solvery ale budou vždy pomalejší než ty standardní, využívající spádové směry.

Nejvážnější problém ale nastává v tom, že všechny řešiče předpokládají úlohu v standardním tvaru. To naše úloha bezesporu je, ale má také jednu zvláštní vlastnost. Ne všechny proměnné, které se vyskytují v omezeních a účelové funkci, jsou nezávislé. Všechny teploty uvnitř modelu totiž závisí čistě pouze na měnících se parametrech kontinuálního lití - licí rychlosti a intenzitě chlazení, přičemž licí rychlost uvažujeme po nespojitém skoku za konstantní. Tedy pro každý element máme pouze **jednu** nezávislou proměnnou. Nicméně v účelové funkci se musí z důvodu penalizace objevit celý vektor $\mathbf{x}_s$. V omezeních se analogicky vyskytují veškeré teploty (i když z nich pak vybíráme pouze teplotu u čidla). Optimalizační algoritmus tedy pracuje s celým vektorem x_s, který má rozsah tisíců prvků. Zřejmě, metody založené na výpočtu v několika bodů v prostoru tak vysoké dimenze, budou neskutečně pomalé a navíc s velice nejistou konvergencí. Při testovacích výpočtech s algoritmem COBYLA [17] se tyto obavy potvrdily a reálnou úlohu nebyl vůbec schopen vyřešit.

Z těchto důvodů byl autor práce přinucen naprogramovat si vlastní opt. algoritmus, který by byl schopen úlohu vyřešit. Ve své podstatě pracuje metodou „full enumerace“. Tedy v rámci zvoleného kroku vypočítá pro všechny varianty nezávislé proměnné (htc_s reprezentující chlazení) účelovou funkci a vrátí index a hodnotu toho htc, u kterého se dosáhlo minima. Jedná se o poměrně jednoduchou metodu. Nicméně v této aplikaci je nesmírně účinná a poráží s přehledem obecné solvery. Jednou z možností navázání na tuto práci může být také nalezení sofistikovanějšího algoritmu.

B Knihovna Armadillo

Jedná se o open-source knihovnu pro lineární algebru [26]. Její výhodou je snaha o přenesení programovacích postupů z prostředí Matlab do C++, Pythonu, R a dalších. Příklady některých typických operací:

- Vytvoření matice $C_{m\times n}$: `mat C(m,n);`
- Počet řádků a sloupců C: `C.n_ rows;C.n_ cols;`
- Naplnění matice číslem k: `C.fill(k);`
- Vybrání prvního řádku C: `C.rows(0);` nebo `C(0,span::all);`

Armadillo také podporuje třídimenzionální pole (`cube`) a „cell arrays“:

- Vytvoř třídimenzionální pole $T_{m\times n\times k}$: `cube T(m,n,k);`
- Zobraz poslední řez T: `T.slice(T.n_ slices);`
- Vytvoř pole F pro n vektorů: `field<vec> F(n,1);`

Knihovna podporuje spoustu maticových operací, (pseudo)inverze, rozklady, řídké matice, základy statistiky nebo řešení soustav. Více informací lze nalézt v dokumentaci [27].

C Program pro výpočet teplotního pole

Program je napsán v C++ jako třída *HeatComp*. Obsahuje následující (vybrané) proměnné a metody

- `cube T`: Třídimenzionální pole pro ukládání dat.
- `mat ConstT`: Matice reprezentující tabulku závislosti mat, vlastností a entalpie na teplotě.
- `mat CoolM`: Matice mapující rozložení *htc* na povrchu sochoru.
- `HeatComp(·)`: Konstruktory.
- ∼`HeatComp(void)`: Destruktor.
- `void SetConstM(char *filename)`: Preprocessing matice konstant.
- `void SetDisc(double dr0, double dzo, double phi)`: Nastavení dělení domény.
- `void SetSqR(double dr0 ,double pR)`: Možné nastavení nekonstantního dělení podle funkce $\sqrt{dr}$.
- `double getConstT(double` $\&T$`, char c)`: Najde hodnotu mat. vlastnosti c pro teplotu T.
- `double getConstH(double` $\&H$`, char)`: Analogicky v závislosti na entalpii H.
- `cube TempToEn(cube` $\&T$`)`: Přepočítej pole teplot na entalpie.
- `cube EnToTemp(cube` $\&H$`)`: Přepočítej pole entalpií na teploty.
- `Compute(int pmax, char swtch)`: Spočítej teplotní pole s $pmax$ iteracemi.
- `ComputeOpt(int pmax, char swtch, cube` T_f`)`: Spočítej teplotní pole s levou OP T_f. Využívá se pro optimalizaci.

Výpočetní metoda se nazývá *Compute*. Její hlavní části je cyklus `while` $p < pmax$ reprezentující iterace řešení v časovém kroku $d\tau$. Uvnitř máme hlavní výpočetní trojnásobný cyklus `for`, který probíhá celé pole $T(i, j, k)$. Pokud se nacházíme uvnitř sochoru (tedy $i, j, k \neq 0$ a zároveň $i < ndr, j < ndphi, k < ndz$), přepočítáváme změny entalpií podle (4.1). Pro body s $k = 0$, máme $T(\cdot, \cdot, 0) = T_{in}$. U ostatních okrajů postupujeme podle příslušných OP popsaných v 4.2.2. Ke konci iterace p přepočítáváme matici H zpět na T. Pro ušetření paměti nepoužíváme indexaci dle iterace, ale pamatujeme si pouze matice té předchozí, které pak značíme indexem $\cdot\_old$.

Lze také zapnout výpočet pomocí reprezentace osy sochoru vektoru nastavením boolovské proměnné *use_middle* na `true`.

D Program pro výpočet PHA

Jako třída `Optimise`. Je mnohem složitější než `HeatComp` a obsahuje spoustu proměnných, proto zde bude popsán velice stručně.

- `Optimise(·)`: Konstruktor nastavující počet elementů, parametry úlohy apod.
- Metody `ToVect, DeVect, DeVectM,`: Převádějí vektorový zápis elementů na maticový resp. „cube“ a naopak.
- `SetElements(void)`: Vytváří dle `nElem` pole elementů a překrývajících se částí.
- `CompScen(int ind, cube & T_old, vec Param, vec & W_old,vec & X_av_old, cube & Tf, double htcl)`: Počítá jednotlivé scénáře.
- `OptimiseScen(int ind, vec & pElemi, double v_z, double htc0, arma::vec & Wi, vec & X_av_oldi, cube & Tf,double htcl)`: Optimalizační algoritmus popsaný v příloze A.
- `ControlPoints`: Třída popisující kontrolní body. Jsou uchovávány jako řádky matice, kde je popsána jejich reálná poloha a hodnota teploty.

Po spuštění úlohy a načtení parametrů dochází k nadělení výpočetní oblasti na `nElem` elementů. Pak vstupujeme do hlavního iteračního cyklu `while` PHA. Na jeho začátku se vyskytuje paralelizovaný cyklus `for`, ve kterém se počítají jednotlivé scénáře. Dále následuje samotné průměrování řešení PHA. Z důvodu vyšší efektivity se každý přesah průměruje zvlášť (pole `X_av(nElem,1)` a až po provedení všech nezbytných operací se zasadí zpátky do celkového vektoru průměrných řešení `X_s_av`. Celý proces pokračuje do té doby, než dosáhneme pevně nastaveného počtu iterací.

Paralelizace se povoluje (zakazuje) odkomentováním (zakomentováním) příkazu `#pragma parallel for` uvnitř metody `ComputePHA`. Nutné je také správně nastavit hodnotu přepínače `parallel` na `true` (`false`).

V případě sériového výpočtu se průměrování obchází tím, že jako levostrannou OP s-tého elementu dosadíme první řez v daném přesahu, ale z elementu $s-1$. Podstatně se tím zrychlí konvergence, ale ztratíme možnost paralelizace.

Při používání programu musíme nejdříve nastavit počet `nElem`. Ideální je počet $4-5$. Poté musíme do souboru `ControlPoints.mat` napsat odpovídající počet kontrolních bodů, které **nesmí** být v oblastech překryvu. Pro zjištění správných teplot je vhodné provést simulaci numerického modelu pro rychlost v_{z0}. Dále je nutné nastavit počet iterací algoritmu.

K vizualizacím používáme program *Octave*. Pokud není na systému nainstalován, nelze vykreslovat obrázky. Nicméně program bude fungovat.

Veškeré programy vznikly v užitím prostředí Eclipse CDT [12] pro systémy založené na linuxu. Potřebné balíčky ke spuštění programu a kompilaci jsou popsány v souboru „readme.txt“ v elektronické příloze. Vzhledem k použití paralelizace je vhodnější program kompilovat ze zdrojového kódu, než použít program vygenerovaný na jiném počítači.

E Obsah přiloženého CD

Na CD je přiloženo následující

- Elektronická verze práce
- Program pro výpočet numerického modelu diferenční metodou
- Program pro výpočet úlohy optimálního řízení pomocí upraveného PHA
- Soubor `readme.txt` popisující postup pro spuštění a kompilaci programů