FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	ETCS – EVC – Modul pro výpočet brzdných křivek
Student:	Jiří Doležal
Vedoucí:	Ing. Jan Matoušek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

ETCS (European Train Control System) je jednotný celoevropský zabezpečovací systém. Práce je součástí trenažéru vlakového zabezpečovacího zařízení vyvíjeného ve spolupráci s Fakultou dopravní ČVUT.
ETCS zajišťuje, že rychlost a pozice vlaku zůstanou v povolených mezích. EVC (European Vital Computer) je centrální počítač vozidlové části systému a jeho součástí je výpočet brzdných křivek, který vychází z matematického modelu brzdných schopností vlaku a vlastností tratě. Cílem této práce je vytvořit modul pro výpočet brzdných křivek podle specifikace.

Pokyny pro vypracování:

1. Analyzujte současný stav projektu, soustřeďte se na komponentu EVC a výpočet brzdných křivek.
2. Analyzujte metodiku výpočtu brzdné křivky dle SUBSET-026 v3.6.0.
3. Pomocí metod softwarového inženýrství navrhněte nový modul pro výpočet brzdných křivek.
4. Implementujte a zdokumentujte navržený modul.
5. Modul podrobte vhodným testům (jednotkové, integrační, akceptační).
6. Porovnejte kvalitu výpočtu se stávajícími řešeními.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 6. února 2023 v Praze.

Bakalářská práce

ETCS EVC – BRZDNÉ KŘIVKY

Jiří Doležal

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Matoušek
11. května 2023

Odkaz na tuto práci: Doležal Jiří. *ETCS EVC – Brzdné křivky.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

V první řadě bych chtěl poděkovat vedoucímu mé práce, Ing. Janu Matouškovi za časté konzultace, cenné poznatky a hlavně přátelský přístup. Dále bych chtěl poděkovat doc. Ing. Martinu Lesovi, Ph.D, který mi pomohl vyřešit nejasnosti týkající se brzdných křivek. V neposlední řadě bych rád zmínil svého spolubydlícího, Michala Kubánka, který se mnou strávil spoustu bezesných nocí při psaní našich bakalářských prací.

Prohlášení

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

V Praze dne 11. května 2023 .....

Abstrakt

Tato bakalářská práce se zabývá evropským vlakovým zabezpečovačem ETCS, konkrétně trenažerem zmíněného zabezpečovače, který vzniká ve spolupráci Fakulty dopravní a Fakulty informačních technologií ČVUT v Praze. Práce se zaměřuje na výpočet brzdných křivek a s nimi spojených limitů. Vytvořené řešení počítá pro danou rychlost vlaku brzdné vzdálenosti a je tak nedílnou součástí mechanismu, který zabraňuje srážce vlaku s okolím. Přínosem této práce je srozumitelnost a přehlednost kódu brzdných křivek, díky čemuž je modul lehce použitelný i pro ostatní členy projektu. Další výhodou je strukturalizace kódu, která umožňuje kvalitní testování.

Klíčová slova European Train Control System, trenažér ETCS, brzdné křivky, Emergency Brake Deceleration, Service Brake Deceleration, Guidance curves, European Vital Computer, gamma vlak, lambda vlak, C++

Abstract

This bachelor's thesis deals with the European train security system ETCS, specifically the simulator of the mentioned security system, which is created in cooperation between the Faculty of Transport and the Faculty of Information Technologies of the Czech Technical University in Prague. The thesis focuses on the calculation of braking curves and the limits associated with them. The created solution calculates the train distance for a given speed and is thus an integral part of the mechanism that prevents the train from colliding with its surroundings. The benefit of this work is the comprehensibility and clarity of the braking curve code, which makes the module easy to use for other members of the project. Another advantage is the structuring of the code, which enables quality testing.

Keywords European Train Control System, ETCS simulator, braking curves, Emergency Brake Deceleration, Service Brake Deceleration, Guidance curves, European Vital Computer, gamma train, lambda train, C++

Seznam zkratek

BTM Balise Transmission module 6

DMI Driver Machine Interface 6

EBD Emergency Brake Deceleration 3, 8, 10–13, 16, 17, 24, 25, 27, 37, 38, 41

EBI Emergency Brake Intervention 8, 14, 25, 37, 38, 41

EOA End of Authority 16, 45

ERTMS European Rail Traffic Management System iii, 5–7, 33

ETCS European Train Control System iii, 1, 3, 5, 6, 8, 11, 16, 17, 19, 31, 33, 39, 45, 46, 48, 49

EVC European Vital Computer 1, 6, 31, 37, 39, 46, 48

GSM-R Global System for Mobile Communications - Railway iii, 5–7

GUI Guidance curve 3, 10, 13, 16, 24, 25, 27, 35, 37, 38, 41

I Indication 8, 13, 25

JRU Juridical Recording Unit 6

LOA Limit of Authority 16

P Permitted speed 8, 13, 25

RBC Radio Block Center 5

SBD Service Brake Deceleration v, 3, 10, 13, 16, 24, 25, 27, 37, 38, 41

SBI Service Brake Intervention 8, 13, 14, 25, 37, 38, 41

TIU Train Interface Unit 6

W Warning 8, 13, 25

Kapitola 1
Úvod

ETCS je evropský vlakový zabezpečovací systém. Měl by postupně nahradit přes 20 různých národních systémů vlakových zabezpečovačů a tak dosáhnout interoperability v železniční dopravě, tj. vedení vlaků po celém území Evropy bez nutnosti výměn hnacích vozidel na hranicích, popřípadě bez nutnosti vybavení hnacích vozidel různými národními systémy. [1]

Fakulta dopravní ČVUT v Praze se rozhodla ve spolupráci s Fakultou informačních technologií ČVUT v Praze vyvinout trenažer, který by měl českým strojvedoucím pomoci se připravit na používání ETCS na území ČR.

Na zmiňovaném trenažeru jsem začal pracovat v rámci předmětu BI-SP1 a pokračoval na jeho vývoji i v předmětu BI-SP2. Jakožto člen týmu, který měl na starosti implementaci EVC, což je jedna z komponent ETCS, jsem měl na starosti přidání několika nových funkcí, ale hlavně jsem se věnoval přepisu původního kódu, který byl velmi špatně strukturován. Během tohoto přepisu jsem společně se svými kolegy narazil na problematiku brzdných křivek. K dispozici jsme měli modul pro výpočet těchto brzdných vzdáleností, avšak k modulu neexistovala žádná dokumentace a ani programátoři, kteří ho vyvíjeli, neprokázali dostatečnou znalost tohoto problému. Rozhodl jsem se tak otázce brzdných křivek věnovat v rámci své baklářské práce.

Sestrojení nového modulu brzdných křivek, který je srozumitelný a jednoduše použitelný, velkou měrou usnadní práci na projektu dalším programátorům, což je velmi důležité vzhledem k vysoké frekvenci, jakou se členové týmu mění.

Ve druhé kapitole bude vyjasněno, co je cílem této práce. Dále bude provedena podrobná analýza brzdných křivek vlaku na základě dokumentů zveřejněných Evropskou železniční agenturou a bude rozebrán i současný stav projektu trenažeru ETCS, konkrétně modulu brzdných křivek. Ve čtvrté kapitole bude představen návrh implementace kódu brzdných křivek za pomocí diagramů tříd a dalších nástrojů softwarového inženýrství. Kapitola následující se bude věnovat implementaci. Budou ukázány některé části kódu nově vzniklého modulu a také budou popsány problémy, jež vývoj modulu provázely. Závěrem bude zhodnoceno naplnění cílů práce.

Kapitola 2
Cíl práce

Cílem této práce je navrhnout a implementovat nový modul pro výpočet křivek EBD, SBD, GUI a s nimi spojených limitů rychlosti. Nově vzniklý modul bude vycházet z důkladné analýzy původního projektu pro výpočet brzdných křivek a dokumentace od Evropské železniční agentury, konrétně SUBSET-026-3 [2], který nově obsahuje vzorce, jež upřesňují výpočet výše zmíněných křivek.

Proces, při kterém jsou křivky kalkulovány, musí být dostatečně rychlý, aby mohl být použit v projektu trenažeru ETCS, a dobře testovatelný, čehož se dosáhne rozdělením kódu na menší logické celky. Tyto celky pak budou vybaveny jednotkovými testy, díky nimž lze předcházet chybám v produkci.

Kapitola 3
Analýza

V této kapitole bude představen systém ERTMS a budou uvedeny důvody, které vedly k jeho zavedení do vlakové dopravy. Stěžejní části systému ETCS budou podrobně popsány a dále bude vysvětleno, co jsou brzdné křivky vlaku, a zdůvodněno, proč je jejich korektní výpočet nepostradatelný pro řízení a bezpečnost železniční dopravy. Závěrem bude osvětlno, v jakém stavu se projekt trenažeru a nynější modul brzdných křivek nachází, a budou uvedeny funkční a nefunkční požadavky k této práci.

3.1 ERTMS

ERTMS je evropský systém řízení železniční dopravy. Na jeho implementaci se podílejí nejen státy Evropské unie, ale například i Norsko, Švýcarsko, Čína nebo Turecko a Saudská Arábie. Systém se skládá ze dvou hlavních částí: ETCS, evropský vlakový zabezpečovací systém, a GSM-R, globální systém pro mobilní komunikace pro železniční aplikace. [3]

3.1.1 ETCS

Na rozdíl od starších zabezpečovacích systémů je ETCS použitelný pro všechny druhy vlakových služeb. Klíčem k této univerzálnosti je stupeň funkční autonomie přidělený palubnímu zařízení. Zahrnuje všechny obecné funkce nezbytné pro sledování a správu rychlosti.

ETCS se skládá ze dvou částí — traťové a vozidlové. Informace mezi nimi probíhají v podobě datových přenosů. [4]

3.1.1.1 Traťová část

3.1.1.1.1 Eurobalíza Pasivní zařízení, které je nainstalováno na trati za účelem sběru a uchování informací jako například rychlostní limity, poloha či sklon. Eurobalíza nepotřebuje elektrické napájení, jelikož energie je dodána skrze anténu při přejezdu vlaku. [5]

3.1.1.1.2 Radiobloková centrála (RBC) Bezpečnostní zařízení užívané v ETCS L2 a L3, které skrze rádiovou komunikaci uděluje vlakům opravnění k jízdě (MA - Movement Authority). RBC získává informace z pevné části zabezpečovacího zařízení a z informací z jednotlivých vozidel. [5]

3.1.1.2 Vozidlová část

3.1.1.2.1 Centrální počítač (EVC) EVC je jádrem palubního zařízení ETCS. Přijímá údaje, vypočítává brzdné křivky, dohlíží na jízdu vlaku a je jednotkou, se kterou interagují všechny ostatní zařízení vlaku. [5]

3.1.1.2.2 Zobrazovací jednotka (DMI) Rozhraní, které propojuje ETCS a řidiče. Jedná se o LCD dotykový panel, který zobrazuje potřebná data (např. rychlost, režim atp.) a zároveň umožňuje řidiči zadat potřebné údaje. [5]

3.1.1.2.3 Jednotka vlakového rozhraní (TIU) Rozhraní, které umožňuje ETCS vyměňovat si informace a vydávat příkazy kolejovým vozidlům (např. ETCS odešle kolejovým vozidlům příkaz k použití brzd). [5]

3.1.1.2.4 Záznamová jednotka (JRU) Poskytuje funkce „černé skříňky“, tj. ukládá nejdůležitější data a proměnné z jízd vlaků, což umožňuje pozdější analýzu. [5]

3.1.1.2.5 Přenosový modul balízy (BTM) Modul uvnitř palubního zařízení ETCS, který vysílá signál k napájení balíz a zároveň přijímá a zpracovává signál z trati na vlak. [5]

3.1.1.2.6 Odometrie Slouží k měření rychlosti a ujeté vzdálenosti. [5]

3.1.1.3 Úrovně ETCS

Pro ještě lepší pochopení fungování Evropského vlakového zabezpečovače je potřeba zmínit, že ETCS má pět funkčních úrovní. Tyto úrovně určují použitelný rozsah funkcí.

- **ETCS L0** Zařízení hlídá pouze maximální rychlost. Vozidlo se pohybuje po tratích, kde není traťová část zabezpečovače nainstalována.
- **ETCS LSTM** Vozidlo s mobilní částí ETCS se pohybuje po tratích vybavených národním vlakovým zabezpečovačem. Zařízení ETCS z něj přijímá informace prostřednictvím STM (Specific Transmission Module).
- **ETCS L1** Zahrnuje nepřetržitý dohled nad pohybem vlaku (tj. palubní počítač nepřetržitě dohlíží na maximální povolenou rychlost a vypočítává brzdnou křivku až do konce oprávnění k jízdě), zatímco mezi vlakem a traťovou kolejí dochází k nepřetržité komunikaci, obvykle prostřednictvím eurobalíz.
- **ETCS L2** Zahrnuje nepřetržitý dohled nad pohybem vlaku s nepřetržitou komunikací přes GSM-R mezi vlakem a traťovou částí.
- **ETCS L3** Velmi podobná úrovni 2. Hlavní rozdíl spočívá v tom, že umístění a integrita vlaku jsou spravovány v rámci systému ERTMS, tj. není potřeba traťových návěstidel nebo systémů detekce vlaků na trati kromě eurobalíz. [4]

3.1.2 GSM-R

Mezinárodní standard bezdrátové komunikace určený pro železniční aplikace.

GSM-R podporuje bezpečnou a spolehlivou komunikaci mezi řidičem a signalizátorem za použití nejúčinnějších technologií a procesů. [6]

3.1.2.1 Výhody GSM-R

- **Zlepšení dostupnosti** GSM-R zajišťuje přímou komunikaci mezi řidičem a signalizátorem za všech okolností. To zahrnuje oblasti, jako jsou tunely a hluboké zářezy, kde dříve nebyla možná radiová komunikace.
- **Zvyšuje bezpečnost pro řidiče, týmy údržby a cestující** Zajišťuje rychlejší a efektivnější reakce na potenciální nebezpečí s aplikacemi, jako je železniční nouzové volání. Eliminuje potřebu strojvedoucích vystoupit z vlaku v případě problému.
- **Snížení provozních nákladů** GSM-R snižuje náklady na průběžnou údržbu, zvyšuje spolehlivost a poskytuje základ pro digitálně podporovanou železniční síť.
- **Odklon od analogu** Předchozí komunikace mezi řidičem a signalizátorem se spoléhala na analogové rádiové sítě. Ty měly omezenou funkčnost a jejich údržba byla stále složitější nákladnější. [6]

3.1.3 Důvody vedoucí k zavedení ERTMS do vlakové dopravy

V současné době existuje v Evropské unii více než 20 systémů řízení vlaků. Každý vlak používaný národní železniční společností musí být vybaven alespoň jedním systémem, ale někdy i více, aby mohl bezpečně jezdit v rámci dané země.

Každý systém je samostatný a není interoperabilní, a proto vyžaduje rozsáhlou integraci, což zvyšuje celkové náklady na dodání pro přeshraniční provoz. To omezuje hospodářskou soutěž a brání konkurenceschopnosti evropského železničního sektoru vůči silniční dopravě vytvářením technických překážek pro mezinárodní cesty. Například vlakové soupravy Thalys jezdící mezi Paříží – Bruselem – Kolínem nad Rýnem a Amsterdamem musí být vybaveny 7 různými typy systémů řízení vlaků, což přináší značné náklady.

Kromě toho je ERTMS pravděpodobně nejvýkonnějším systémem řízení vlaků na světě, což potvrzuje i stále větší popularita systému i ve státech ležící mimo Evropu (např. v Číně, v Jižní Koreji nebo v Saudské Arábii). Tato výkonnost přináší značné úspory nákladů, ale také zvyšuje bezpečnost, kapacitu a spolehlivost vlakových spojů.

Tím, že zvyšuje konkurenceschopnost železničního sektoru, pomáhá ERTMS vyrovnat podmínky pro silniční a železniční dopravu a v konečném důsledku poskytuje významné přínosy pro životní prostředí. [1]

3.2 Brzdné křivky

ETCS dohlíží, že se vlak vždy pohybuje v rámci povolené rychlosti a brzdných limitů a v případě potřeby dá příkaz k zásahu brzdového systému, aby se zabránilo jakémukoli riziku srážky s okolím vlaku. Palubní počítač ETCS k tomu musí předvídat pokles rychlosti vlaku v budoucnu. Tato předpověď poklesu rychlosti v závislosti na vzdálenosti je nazývána brzdná křivka a lze ji vypočítat z matematického modelu dynamiky brzdění vlaku a vlastností trati. Na základě brzdných křivek vykalkuluje pak palubní počítač ETCS brzdné vzdálenosti. Ty jsou poté používány k udržování vlaku v takových rychlostních mezích, aby bylo možno v případě potřeby včas zabrzdit. [2, 3.13.1]

3.2.1 Harmonizované brzdné křivky

Sjednocení výpočtu brzdných křivek bylo v ještě nedávné minulosti vcelku kontroverzní téma. Hlavním důvodem bylo rozdělení odpovědnosti mezi železniční podnik a správce infrastruktury vynucené směrnicemi EU.

Pozdější specifikace ETCS baseline 2 [7] pak položily základní principy pro brzdné křivky a související informace zobrazované řidiči, ale stále neexistovala žádná metoda či algoritmus pro jejich výpočet, což s sebou neslo následující problémy:

- Algoritmy různých dodavatelů vedou k různým brzdným dráhám pro daný typ kolejových vozidel. Konstrukce traťového systému ETCS tak není závislá pouze na výkonu brzdového systému kolejového vozidla, ale také na samotném palubním dodavateli ETCS.
- U přeshraničních vlaků rozdíly ve vnitrostátních pravidlech/postupech vyžadují zavedení několika brzdových systémů do palubního systému ETCS, což s sebou logicky nese zvýšení nákladů na provoz.

Výše zmíněné problémy vedly k další harmonizaci brzdných křivek, která se promítla do specifikace ETCS baseline 3 [8]. [9, 2.2.]

3.2.2 Význam brzdných křivek

Brzdné křivky jsou naprosto nezbytné k dosažení základní funkčnosti ETCS a to z následujících důvodů:

- **Umožňují ETCS fungovat jako „záchranný padák“** Brzdná křivka související s poklesem rychlosti v důsledku nouzové brzdy se nazývá křivka EBD. Z EBD a naměřené rychlosti vlaku počítač ETCS vykalkuluje (několikrát za sekundu) vzdálenost potřebnou k zastavení vlaku od doby, kdy by palubní ETCS přikázalo zásah nouzové brzdy. Tato vzdálenost určuje místo zvané EBI, tj. bod, za kterým ETCS zasáhne bez přičinění odpovědného člověka.
- **Asistují řidiči s brzděním** Kromě funkčnosti padáku poskytuje ETCS řidiči informace týkající se brzdění. Jeho účelem je pomoci řidiči udržováním rychlosti vlaku v příslušných mezích. Palubní ETCS proto vypočítává i další limity dohledu: Indikace (I), Povolená rychlost (P), Varování (W) a Zásah provozní brzdy (SBI). Skládají se z lokací, které když vlak projede, předá systém řidiči potřebné informace skrze odpovídající grafiku. [9, 3.1.]

3.2.3 Gamma a lambda metoda

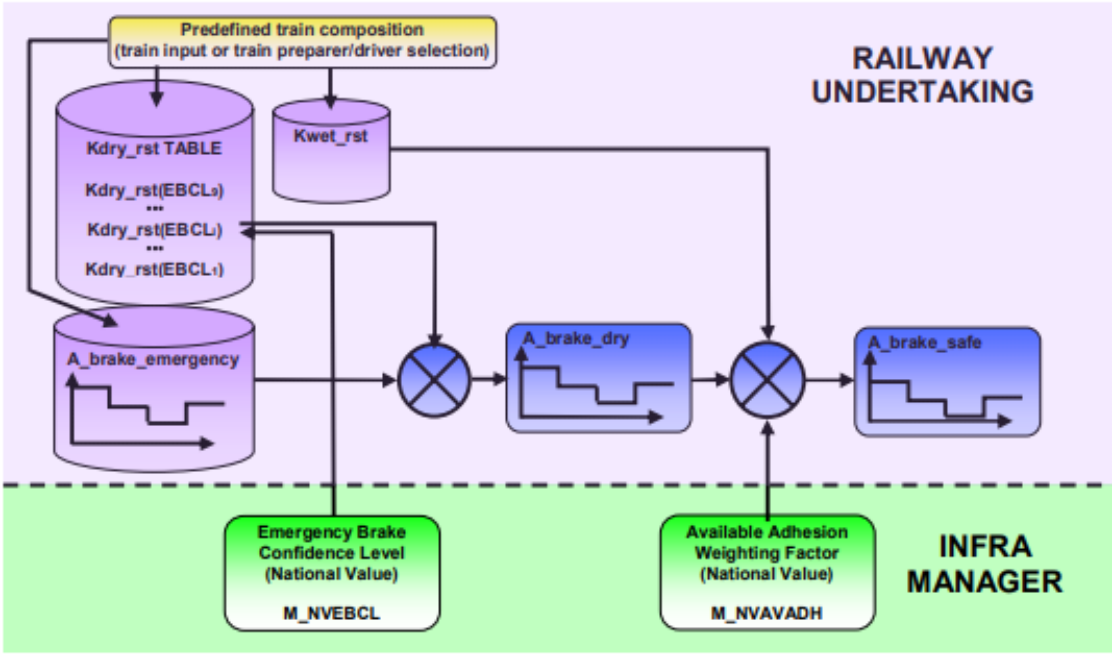
Brzdné křivky se určují pro ucelené vlakové soupravy, jež mají pevně dané řazení a pro soupravy sestavené z hnacího vozidla a různých vozů. Podle toho, o jaký typ soupravy se jedná, rozlišujeme tzv. gamma a lambda metodu výpočtu.

V případě ucelených souprav je možné určit, zda brzdná síla je závislá na obsazenosti (počet cestujících) vlaku. U takovýchto souprav mluvíme o tzv. gamma metodě.

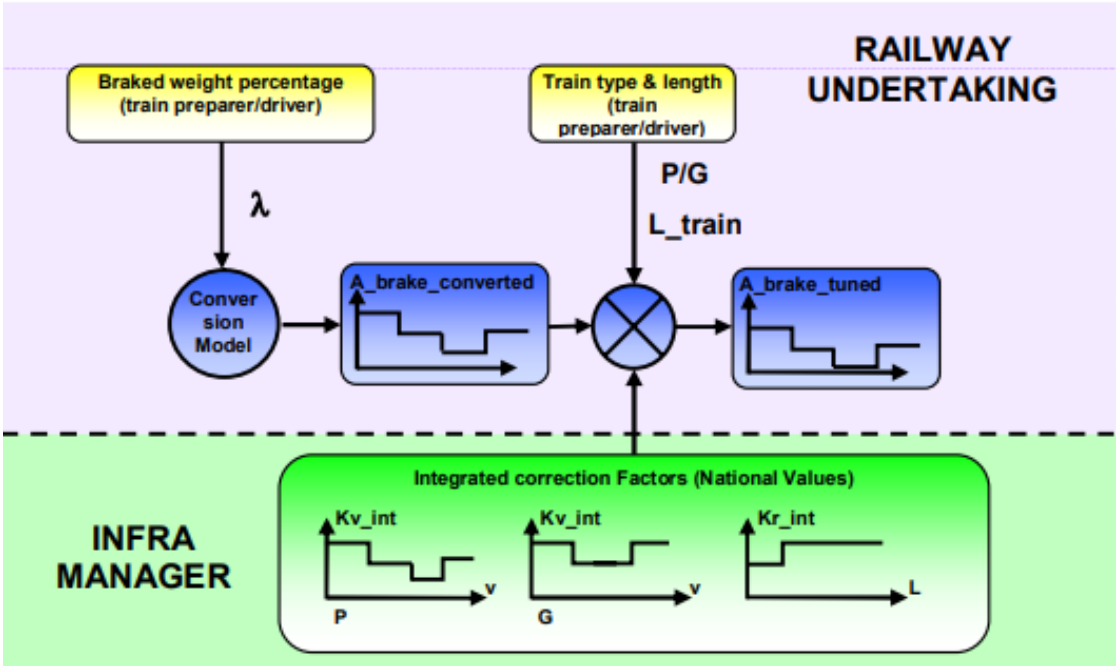
Přístup gamma nelze uplatnit u souprav typu lokomotiva plus vozy, tedy různé sestavy vozidel. U různě složených souprav mluvíme o tzv. lambda metodě. Pro jednotlivá vozidla totiž není známá závislost brzdné síly na rychlosti a závislost brzdné síly na čase. Tyto závislosti jsou důležité pro stanovení brzdné křivky. U náhodně sestavených souprav je známé pouze brzdné procento, které udává podíl brzdící váhy napsané na vozidle a hmotnosti vlaku. Díky brzdícímu procentu se snadno stanoví reálné brzdící schopnosti vlaku. [9, 3.3.3.] [10, 2.3]

Tabulka 3.1 Rozdíly mezi gamma a lambda metodou [10, Table 4]

Typ	Lambda	Gamma
Přesnost	Nízká/omezená	Vysoká
Počet parametrů	Několik	Mnoho
Primárně používané pro:	Nákladní vlaky, kde jsou parametry brzdění neznámé	Ucelené vlakové soupravy, kde je brzdná schopnost detailně zadefinována



Obrázek 3.1 Odhad brzdné křivky pomocí gamma modelu [9, Figure 6]



Obrázek 3.2 Odhad brzdné křivky pomocí lambda modelu [9, Figure 7]

3.2.4 Křivky zpomalení

Pro výpočet křivek EBD, SBD a GUI je nejprve potřeba si zadefinovat funkce A_brake_emergency(V), A_brake_service(V) a A_brake_normal_service(V). Tyto funkce popisují zpomalení způsobené brzděním příslušné brzdy.

- A_brake_emergency(V) = zpomalení způsobené použitím nouzové brzdy
- A_brake_service(V) = plné zpomalení provozní brzdy
- A_brake_normal_service(V) = normální zpomalení provozní brzdy [2, 3.13.2.2.3]

A_brake_emergency(V) a A_brake_service(V) se modulu brzdných křivek předá jakožto skoková funkce. [2, 3.13.2.2.3]

Speciálním případem je funkce A_brake_normal_service(V). Ta se nepředává pouze jako skoková funkce, ale jako celá sada brzdných parametrů a je pro ni možné definovat až dvě sady skokových funkcí po třech modelech:

- jedna sada je použitelná pro nákladní vlaky
- druhá sada je použitelná pro vlaky osobní [2, 3.13.2.2.3.1.9]

Součástí A_brake_normal_service(V) jsou také dvě pivotní hodnoty A_SB01 a A_SB12, které se užívají pro určení konkrétní skokové funkce, která bude použita při výpočtech. Určení výsledné funkce probíhá podle následujících pravidel:

- Pokud A_brake_service(V = 0) ≤ A_SB01
  - A_brake_normal_service(V) = A_brake_normal_service_0(V)
- Pokud A_brake_service(V = 0) ≤ A_SB12
  - A_brake_normal_service(V) = A_brake_normal_service_1(V)
- Pokud A_SB12 < A_brake_service(V = 0)
  - A_brake_normal_service(V) = A_brake_normal_service_2(V) [2, 3.13.2.2.3.1.10]

3.2.4.1 EBD

EBD je parabolická křivka, která počítá zpomalení vyplývající z použití záchranné brzdy a vlastností trati. Za tímto účelem je zpomalení nouzové brzdy modelováno pomocí skokové funkce bezpečného zpomalení vůči rychlosti (A_brake_safe), zatímco sklony trati jsou odesílány traťovým systémem ETCS jako skoková funkce sklonů vůči vzdálenosti.

Pří výpočtu se zpomalení způsobené brzděním nouzové brzdy A_brake_emergency(V) přenásobí odpovídajícími korekčními faktory, které jsou rozepsány v podkapitole 3.2.7, z čehož vznikne zpomalení, které je bezpečnostně relevantní a nazývá se A_brake_safe(V). [9, 3.3.1]

- $A\_brake\_safe(V) = Kdry\_rst(V, M\_NVEBCL^{1}) \cdot (Kwet\_rst(V) + M\_NVAVADH^{2} \cdot (1 - Kwet\_rst(V))) \cdot A\_brake\_emergency(V)$ pro gamma metodu výpočtu
- $A\_brake\_safe(V) = Kv\_int(V) \cdot Kr\_int(L\_TRAIN) \cdot A\_brake\_emergency(V)$ pro lambda metodu výpočtu [2, 3.13.6.2.1]

Taktéž sklon svahu je přenásoben korekčními hodnotami a to pro zohlednění rotující hmoty. Rotující hmota může být například sypký náklad vlaku, který svým pohybem v nákladním prostoru ovlivňuje brzdnou sílu soupravy. Výsledkem je proměnná A_gradient, která reprezentuje zrychlení/zpomalení vzhledem ke sklonu trati. [2, 3.13.4.3.2]

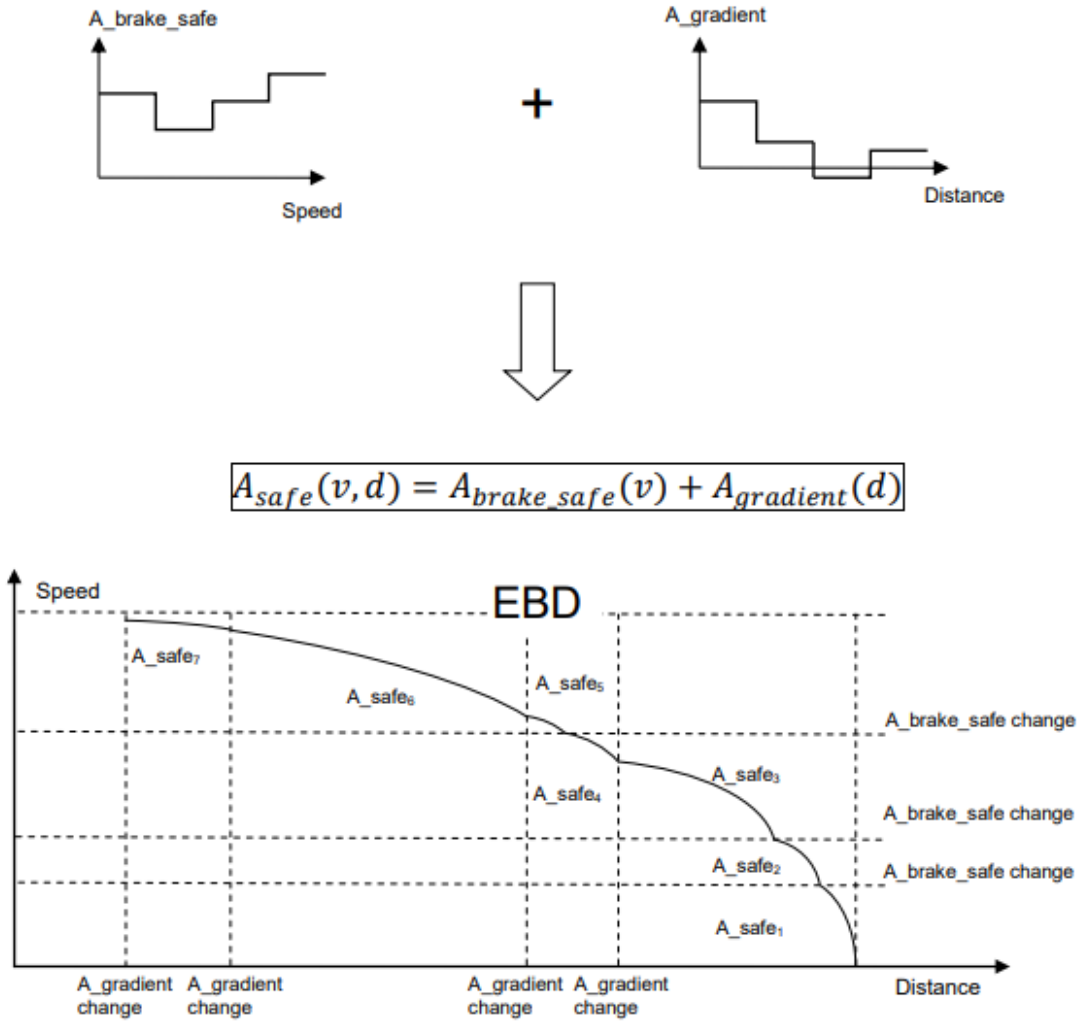
- Pokud není známa hodnota rotující hmoty:
  - Stoupání: A_gradient = g * grad / (1000+10*M_rotating_max)
  - Klesání: A_gradient = g * grad / (1000+10*M_rotating_min)
- Pokud je známa hodnota rotující hmoty:
  - Stoupání: A_gradient = g * grad / (1000+10*M_rotating_nom)
  - Klesání: A_gradient = g * grad / (1000+10*M_rotating_nom)

g = gravitační zrychlení
grad = hodnoty gradientu v ‰
M_rotating_nom = procentuální hodnota nominální rotující hmoty vzhledem k hmotnosti vlaku
M_rotating_max = maximální rotující hmota jako procento z totální hmotnosti vlaku
M_rotating_min = minimální rotující hmota jako procento z totální hmotnosti vlaku [2, 3.13.4.3.2]

[1] Úroveň spolehlivosti pro bezpečné zpomalení nouzové brzdy na suchých kolejích [11, s. 7.5.1.75.1]
[2] Váhový faktor pro dostupnou přilnavost kolejnice [11, s. 7.5.1.73.1]

3.2.4.1.1 Kompenzace délky vlaku Zrychlení/zpomalení vzhledem ke sklonu trati (tj. A_gradient) musí být vypočítáno z nejnižší (s přihlédnutím ke znaménku) hodnoty gradientu mezi polohou fiktivního předku vlaku a polohou fiktivního konce vlaku. [2, 3.13.4.2]

Na závěr se pro kalkulaci EBD využije zpomalení A_safe(V,d), které bere v potaz jak zpomalení v důsledku použití záchranné brzdy, tak i zpomalení vzhledem k vlastnostem trati. [2, s. 3.13.6.2.1.1]



Obrázek 3.3 Konstrukce EBD [9, Figure 4]

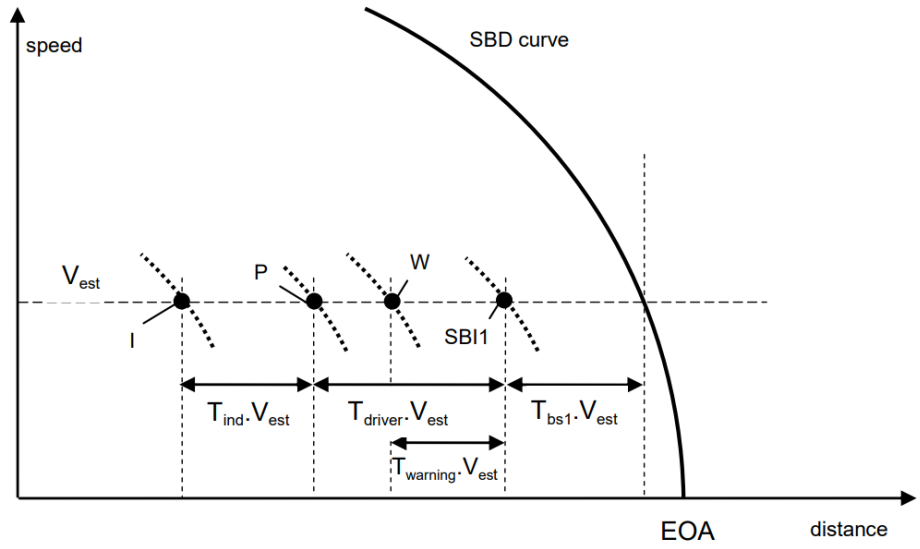
3.2.4.2 SBD

Další křivkou, kterou musí systém vlakového zabezpečovače vypočítat je SBD. Tato křivka je podobná EBD s tím rozdílem, že se bere v úvahu plné zpomalení při použití provozní brzdy namísto záchranné. Z SBD se poté počítá celá řada rychlostních limitů jako například indikace (I), povolená rychlost (P), varování (W) a samozřejmě i zásah provozní brzdy (SBI). [2, 3.11.11]

Pro kalkulaci SBD je nutné získat zpomalení A_expected(V,d), které bere v potaz jednak zpomalení vlivem plného pužití provozní brzdy, jednak zpomalení způsobené sklonem trati.

A_expected(V,d) = A_brake_service(V,d) + A_gradient(d), kde výpočet A_gradient(d) je stejný jako v podkapitole 3.2.4.1 [2, 3.13.6.3.1.3]

.



Obrázek 3.4 Příklad křivky SBD a s ní spojených rychlostních limitů [2, Figure 46]

3.2.4.3 GUI

Účelem naváděcí křivky GUI je poskytnout pohodlný způsob brzdění řidiče, aby se zabránilo nadměrnému opotřebení brzd a ušetřila se trakční energie. Pokud není GUI zadefinována skrze národní hodnoty, palubní zařízení vypočítá pomocnou naváděcí křivku na základě normálního zpomalení provozní brzdy a sklonu trati. [2, 3.13.8.5]

- Pro kladné hodnoty sklonu platí:
  - $A\_normal\_service(V,d) = A\_brake\_normal\_service(V,d) + A\_gradient(d) - Kn + (V) \cdot grad/1000$
- Pro záporné hodnoty sklonu platí:
  - $A\_normal\_service(V,d) = A\_brake\_normal\_service(V,d) + A\_gradient(d) - Kn - (V) \cdot grad/1000$

A_gradient(d) je stejný jako v podkapitole 3.2.4.1, grad jsou hodnoty gradientu v ‰ a Kn jsou korekční faktory, které jsou podrobněji rozerpsány v podkapitole 3.2.7. [2, 3.13.6.4.3]

3.2.5 Limity odvozené z brzdných křivek

- **Zásah nouzové brzdy** (Emergency brake intervention) — EBI
- **Zásah provozní brzdy** (Service brake intervention) — SBI
- **varování** (Warning) — W
- **Povolená rychlost** (Permitted speed) — P
- **Indikace** (Indication) — I

Účelem limitu zásahu nouzové brzdy je zajistit, aby vlak zůstal v rámci různých limitů (vzdálenost/rychlost) stanovených tratí.

Účelem všech ostatních limitů dohledu je pomoci řidiči předcházet zásahu nouzové brzdy. [2, 3.13.9]

3.2.5.1 Vzorce pro výpočet limitů dohledu

Prvně se vypočítají hodnoty pomocných proměnných V_{delta0}, V_{delta1}, V_{delta2}, V_{bec}, D_{bec}, které se používají při výpočtu limitů.

- $V_{delta0} = V_{ura}$, kde V_{ura} symbolizuje kompenzaci nepřesnosti rychlosti podle SUBSET-041 [12, 5.3.1.2], nebo $V_{delta0} = 0$ (Pokud je kompenzace nepřesnosti rychlosti předána skrze národní hodnoty)
- $V_{delta1} = A_{est1} \cdot T_{traction}$, kde A_{est1} je momentální zrychlení vlaku limitované na kladné hodnoty
- $V_{delta2} = A_{est2} \cdot T_{berem}$, kde A_{est2} je momentální zrychlení vlaku limitované na hodnoty 0—$0.4 m/s^2$
- $V_{bec} = max(V_{est} + V_{delta0} + V_{delta1}, V_{target}) + V_{delta2}$
- $D_{bec} = max(V_{est} + V_{delta0} + V_{delta1}/2, V_{target}) \cdot T_{traction}$ $+ (max(V_{est} + V_{delta0} + V_{delta1}, V_{target} + V_{delta2}/2) \cdot T_{berem}$ [2, 3.13.9.3.2.10]

3.2.5.1.1 Kalkulace EBI $d_{EBI}(V_est) = d_{EBD}(V_{bec}) - D_{bec}$ [2, 3.13.9.3.2.12]

3.2.5.1.2 Kalkulace SBI1 $d_{SBI1}(V_est) = d_{SBD}(V_{est}) - V_{est} \cdot T_{bs1}$, kde T_{bs1} je čas zprovoznění brzdy [2, 3.13.9.3.3.1]

3.2.5.1.3 Kalkulace SBI2 $d_{SBI2}(V_est) = d_{EBI}(V_{est}) - V_{est} \cdot T_{bs2}$, kde T_{bs2} je čas zprovoznění brzdy [2, 3.13.9.3.3.2]

3.2.5.1.4 Kalkulace W $d_W(V_{est}) = d_{SBI}(Vest) - V_{est} \cdot T_{warning}$, kde $T_{warning}$ je fixní hodnota [2, 3.13.9.3.4.1]

3.2.5.1.5 Kalkulace P $d_P(V_{est}) = d_{SBI}(V_{est}) - V_{est} \cdot T_{driver}$, kde T_{driver} je fixní hodnota [2, 3.13.9.3.5.1]

3.2.5.1.6 Kalkulace I $d_I(V_{est}) = d_P(V_{est}) - V_{est} \cdot T_{indication}$, kde $T_{indication} = max(0, 8 \cdot T_{bs}, 5s) + T_{driver}$ [2, 3.13.9.3.6.1]

3.2.5.2 Nepřesnost v měření rychlosti

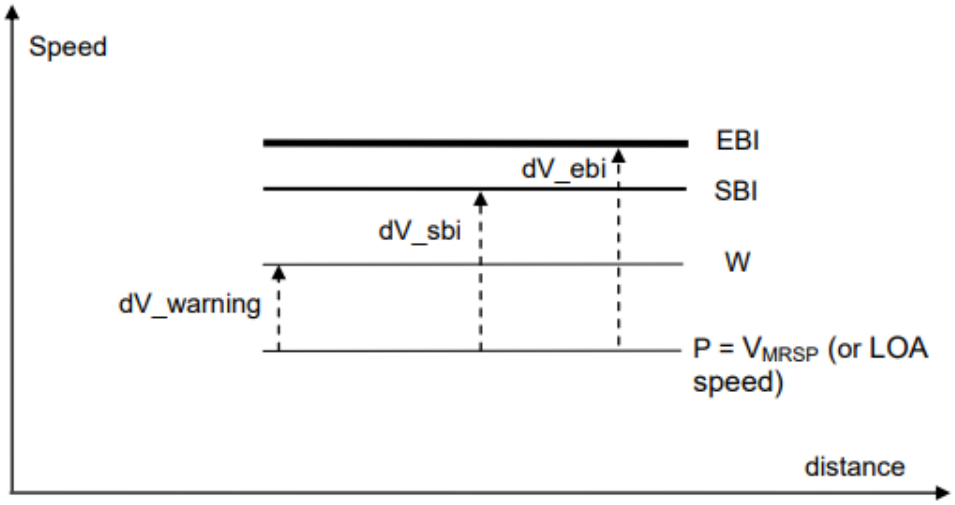
Vzorec pro kompenzaci této nepřesnosti je definován v SUBSET-041 [12, s. 5.3.1.2] a vypadá následovně:

- Pro rychlosti menší než 30 km/h platí:
  - nepřesnost měření = ±2 km/h
- Pro rychlosti z intervalu 30 km/h až 500 km/h (včetně) platí:
  - nepřesnost měření se lineárně se zvyšuje až do hodnoty ±12 km/h
- Pro rychlosti vyšší než 500 km/h platí:
  - nepřesnost měření = ±12 km/h

3.2.5.3 Rozdílové hodnoty

Rozdílové hodnoty (z anglického difference values) stanovují rychlostní rozdíl mezi danými limity rychlosti. Lze rozlišit tři rozdílové hodnoty dV_ebi, dV_sbi a dV_warning.

- dV_ebi je rychlostní rozdíl mezi zásahem nouzové brzdy a povolenou rychlostí
- dV_sbi je rychlostní rozdíl mezi zásahem provozní brzdy a povolenou rychlostí
- dV_warning je rychlostní rozdíl mezi varováním a povolenou rychlostí [2, Figure 43]



Obrázek 3.5 Znázornění rozdílových hodnot dV_ebi, dV_sbi a dV_warning [2, Figure 43]

Vzorec pro jejich výpočet vypadá následovně:

- Pokud V > V_min:
  - $C = (dV\_max - dV\_min)/(V\_max - V\_min)$
  - $dV = min((dV\_min + C \cdot (V - V\_min)), dV\_max)$
- Jinak:
  - $dV = dV\_min$

kde V je daná rychlost a hodnoty dV_min, dV_max, V_min, V_max jsou konstanty daných limitů z podkapitoly 4.3.

3.2.6 Vstupní parametry

Pro správný výpočet brzdných křivkek je zapotřebí řada vstupních parametrů. Ty lze rozdělit do čtyř kategorií: parametry fyzikální, konstanty, traťová data a palubní parametry. [9, 3.2.]

3.2.6.1 Fyzikální parametry

Vyplývají z měření v reálném čase palubním zařízením ETCS: okamžitá poloha, rychlost a zrychlení. [9, 3.2.]

3.2.6.2 Konstanty

Většinou se týkají ergonomie samotného modelu brzdné křivky (například reakční rychlost řidiče). [9, 3.2.]

3.2.6.3 Traťová data

- **Omezení rychlosti související s tratí**
  - Mezi parametry omezení rychlosti a vzdálenosti patří i *typ cíle*. Když je cílová rychlost na konci úseku, kde je vlaku garantována autorita pohybu, nula, cílový bod se nazývá EOA. Když cílová rychlost není nula, nazývá se LOA.
- **Sklon trati**
- **Národní hodnoty**
  - Mezi nejvýznamnější národní hodnoty patří sada korekčních faktorů Kv_int(V), Kr_int(l) a Kt_int. Ty platí pro palubní zařízení pouze při použití lambda konverze a jsou součástí výpočtu křivky EBD. Podrobněji se jim věnuje kapitola 3.2.7.
- **Konkrétní omezení rychlosti a vzdálenosti** [2, 3.13.2.3]

3.2.6.4 Palubní parametry

- **Trakční model** — Funkce, která popisuje časovou prodlevu při spuštění traction cut-off příkazu.
- **Brzdné modely, nebo brzdné procento** — Pro ucelené vlakové soupravy jsou známy brzdné modely, ze kterých se poté vypočítávají křivky EBD, SBD a GUI. Pro ostatní soupravy se používá konverze z brzdného procenta.
- **Pozice brzdy** — Poloha brzdy definuje chování brzdy pro konkrétní typy vlaků. Rozlišujeme tři různé polohy, ve kterých se brzda může nacházet — osbní vlak v P, nákladní vlak v P, nákladní vlak v G.
- **Rozhraní traction cut-off** — Definuje, zda je příkaz k odpojení trakce implementován.
- **Délka vlaku**
- **Nominální rotující hmota** — Používá se pro kompenzaci brzdných křivek při jízdě do/z kopce (rotující hmota jako například náklad štěrku se uvnitř vlaku pohybuje a ovlivňuje brzdící schopnosti)
- **Korekční faktory** — Kdry_rst(V, EBCL), Kwet_rst(V), Kn+(V) a Kn-(V), podrobnější jsou popsány v kapitole 3.2.7. [2, 3.13.2]

Mezi posledními dvěma kategoriemi vstupních parametrů je třeba věnovat zvláštní péči těm, které přispívají k výpočtu křivky EBD. Celková bezpečnost železničního systému totiž velmi závisí na skutečnosti, že vlaky budou účinně brzděny podle předpokládané EBD. Ta proto musí splňovat příslušnou bezpečnost, která je požadována pro provoz vlaků ETCS na dané infrastruktuře. [9]

3.2.7 Korekční faktory

Korekční faktory se používají napříč všemi výpočty a i díky nim splňují výsledné křivky potřebnou bezpečnost. [2, 3.13.2.1.3]

Korekční faktory Kv_int(V), Kr_int(l) a Kt_int se používají při výpočtu EBD, avšak pouze používá-li se lambda metoda a s ní spojená konverze. [2, 3.13.2.3.7]

Korekční faktory Kdry_rst(V, EBCL) a Kwet_rst(V) se taktéž používají při kalkulaci EBD, ale naopak pouze, pokud se jedná o ucelenou vlakovou soupravu, a tudíž se používá gamma metoda výpočtu. [2, 3.13.2.2.9.1]

3.2.7.0.1 Kv_int(V) Korekční faktor závislý na rychlosti, který je předán jako součást národních hodnot a to ve formě skokové funkce. [2, 3.13.2.3.7]

Lze definovat až 2 sady Kv_int. Sady Kv_int se týkají následujících typů vlaků:

- Nákladní vlaky
- Konvenční osobní vlaky
  - Pro osobní vlaky se dokonce předávájí dvě různé skokové funkce Kv_int_x_a a Kv_int_x_b a hodnoty A_P12 a A_P12, ze kterých se podle následujících pravidel vypočítá výsledný korekční faktor Kv_int_x:
    - A_ebmax = maximální zpomalení nouzové brzdy
    - $Kv\_int\_x = Kv\_int\_x\_a$, pokud $A\_ebmax \leq A\_P12$.
    - $Kv\_int\_x = Kv\_int\_x\_b$, pokud $A\_ebmax \geq A\_P23$.
    - $Kv\_int\_x = Kv\_int\_x\_a + (A\_ebmax - A\_P12)/(A\_P23 - A\_P12) \cdot (Kv\_int\_x\_b - Kv\_int\_x\_a)$, pokud $A\_P12 < A\_ebmax < A\_P23$. [2, 3.13.2.3.7]

3.2.7.0.2 Kr_int(l) Korekční faktor závislý na délce vlaku, který je předán jako součást národních hodnot a to ve formě skokové funkce. [2, 3.13.2.3.7]

3.2.7.0.3 Kt_int Korekční faktor pro dobu potřebnou k plému zprovznění brzdy, který je taktéž předán jako součást národních hodnot jakožto desetinné číslo. [2, 3.13.2.3.7]

3.2.7.0.4 Kdry_rst(V, EBCL) Pro danou úroveň spolehlivosti bezpečného zpomalení nouzové brzdy (EBCL) se korekční faktor Kdry_rst(V) předá jako skoková funkce rychlosti, se stejnými kroky jako A_brake_emergency(V) (viz 3.2.4). Je nedílnou součástí brzdných modelů gamma vlaku a představuje ztrátu zpomalení na suchých kolejích. [2, 3.13.2.2.9.1]

3.2.7.0.5 Kwet_rst(V) Korekční faktor Kwet_rst(V) musí být taktéž předán jako skoková funkce, se stejnými kroky jako A_brake_emergency(V) (viz 3.2.4). A stejně jako Kdry_rst(V, EBCL) je součástí brzdných modelů gamma vlaku a představuje ztrátu zpomalení na mokrých kolejích. [2, 3.13.2.2.9.1]

3.2.7.0.6 Kn+(V) a Kn-(V) Korekční faktory závislé na rychlosti pro kladný, respektive záporný sklon trati. [2, 3.13.2.2.9.2]

3.2.8 Konverzní modely

U vlaků s proměnným složením se charakteristika brzd liší spolu se složením vlaku. V tomto případě není vhodné předprogramovat parametry brzdy nutné pro výpočet brzdných křivek. Jediný praktickým způsobem, jak získat správné hodnoty pro aktuální složení vlaku, je zahrnout do procesu zadávání dat řidičem. Od řidiče však nelze však očekávat znalost hodnoty zpomalení vlaku a doby náběhu brzd. Konverzní modely jsou proto definovány tak, aby převáděly jednoduché parametry zadané řidičem (brzdné procento a poloha brzdy) na parametry odpovídajícího modelu brzdy. [2, 3.13.3]

Obecně však platí, že metoda lambda vede k pozvolnějším brzdným křivkám než metoda gamma. [13]

3.2.8.1 Konverze brzdného procenta

Brzdné procento se využívá ke zjištění skokových funkcí $A\_brake\_emergency(V)$ a $A\_brake\_service(V)$, které vyjadřují zpomalení vlaku při použití nouzové, respektive provozní brzdy. [2, 3.13.3]

Výpočet těchto dvou proměnných se liší v jediném a to hned prvním kroku. Tím je výpočet parametru λ_O. Pro nouzovou brzdu platí $\lambda_O = \lambda$, zatímco pro provozní brzdu platí $\lambda_O = min(\lambda, 135)$. [2, A.3.7.1]

Rychlostní limit pro první skok je počítán v km/h podle vzorce, který je dál uveden: $V\_lim = x \cdot \lambda_O^y$, kde $x = 16.85$ a $y = 0.428$. [2, A.3.7.3]

První skok se pak značí AD_0 (hovoříme o prvnín skoku, avšak číslujeme od nuly) a vyjadřuje zpomalení v m/s^2 pro rychlosti 0—V_lim. Lze jej spočítat následovně: $AD\_0 = A \cdot \lambda_O + B$, kde $A = 0.0075$ a $B = 0.076$. [2, A.3.7.4]

Zbylé skoky lze vypočítat jakožto výsledek polynomu třetího řádu, který vypadá takto: $AD\_n = a3\_n \cdot \lambda_O^3 + a2\_n \cdot \lambda_O 2 + a1\_n \cdot \lambda_O + a0\_n$ [2, A.3.7.5]
Hodnoty, kterých n nabývá, jsou 1—5 a platí pro rychlosti uvedené dále (rychlosti jsou v km/h)

- **n = 1**
  - **V_lim < 100** platí pro rychlosti v rozmezí V_lim—100
  - **V_lim ≥ 100** nepočítá se
- **n = 2**
  - **V_lim < 100** platí pro rychlosti v rozmezí 100—120
  - **100 < V_lim < 120** platí pro rychlosti v rozmezí V_lim—120
  - **V_lim ≥ 120** nepočítá se
- **n = 3**
  - **V_lim < 120** platí pro rychlosti v rozmezí 120—150
  - **120 < V_lim < 150** platí pro rychlosti v rozmezí V_lim—150
  - **V_lim ≥ 150** nepočítá se
- **n = 4**
  - **V_lim < 150** platí pro rychlosti v rozmezí 150—180
  - **150 < V_lim < 180** platí pro rychlosti v rozmezí V_lim—180
  - **V_lim ≥ 180** nepočítá se

- **n = 5**
  - **V_lim > 180** platí pro rychlosti větší než V_lim
  - **V_lim ≤ 180** platí pro rychlosti větší než 180 [2, A.3.7.5]

  Koeficienty polynomů jsou zadefinovány v tabulce níže.

Tabulka 3.2 Koeficienty polynomu sloužícího ke konverzi brzdného procenta v závislosti na m a n [2, A.3.7.6]

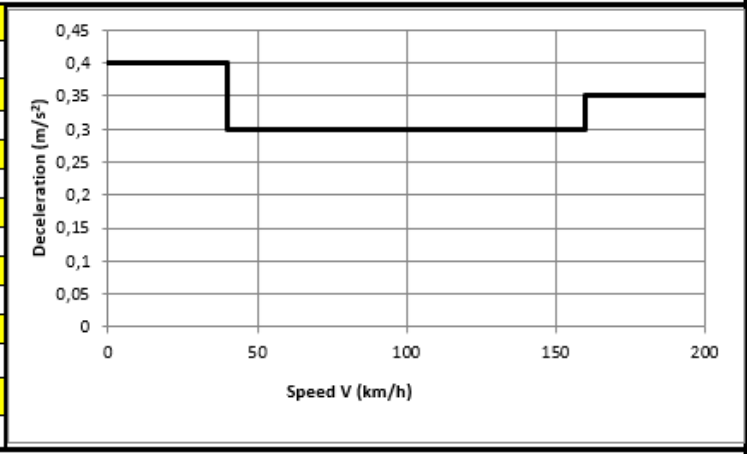
am_n	m = 3	m = 2	m = 1	m = 0
n = 1	$-6,30 \cdot 10^{-7}$	$6,10 \cdot 10^{-5}$	$4,72 \cdot 10^{-3}$	$0,0663$
n = 2	$2,73 \cdot 10^{-7}$	$-4,54 \cdot 10^{-6}$	$5,14 \cdot 10^{-3}$	$0,1300$
n = 3	$5,58 \cdot 10^{-8}$	$-6,76 \cdot 10^{-6}$	$5,81 \cdot 10^{-3}$	$0,0479$
n = 4	$3 \cdot 10^{-8}$	$-3,85 \cdot 10^{-6}$	$5,52 \cdot 10^{-3}$	$0,0480$
n = 5	$3,23 \cdot 10^{-9}$	$1,66 \cdot 10^{-6}$	$5,06 \cdot 10^{-3}$	$0,0559$

Konverze vstupních hodnot na křivku A_brake_normal_service(V) není v SUBSET-026-3 [2] popsána. Z tabulek Braking curves simulation tool v4.2 [14], oficiálního nástroje, který umožňuje předem určit všechny brzdné dráhy tak, jak by byly vypočítány palubním zařízením ETCS, však vyplývá, že pro tuto konverzi se využívá předem daných konstant A_SB01, A_SB12 a skokových funkcí, z nichž je jedna vybrána jako A_brake_normal_service(V) a to podle vzroce uvedeného v kapitole 3.2.4.

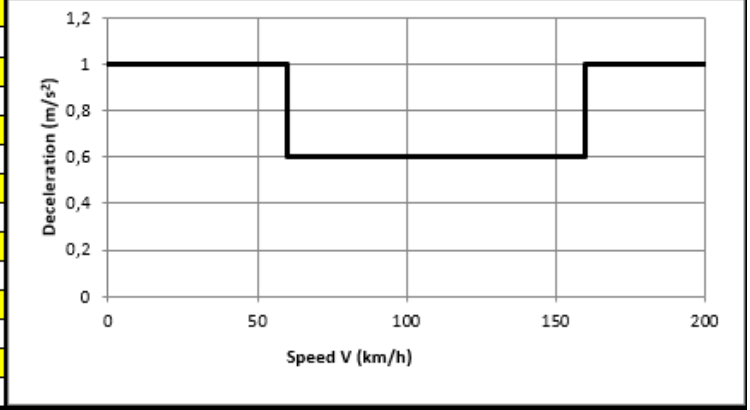
A_SB01 (m/s²)	1,2
A_SB12 (m/s²)	1,3

Obrázek 3.6 Hodnota parametrů A_SB01 a A_SB12 při lambda konverzi [14]

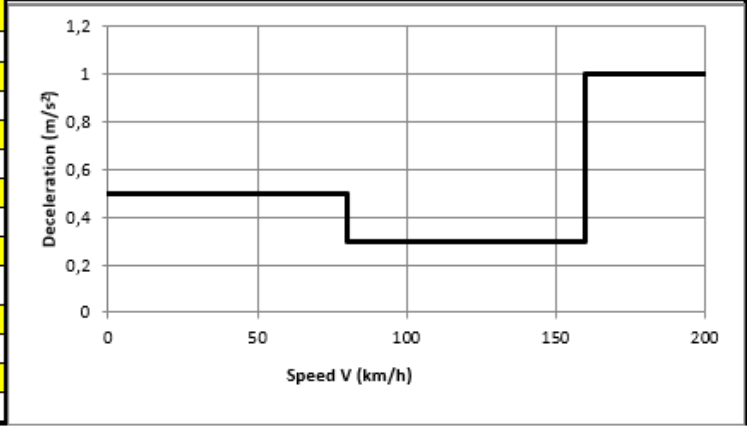
Normal service brake deceleration (A_brake_normal_service_0) - Passenger/freight P			
V0	0	AD0	0,4
V1	40	AD0	0,4
	40	AD1	0,3
V2	160	AD1	0,3
	160	AD2	0,3
V3	160	AD2	0,3
	160	AD3	0,35
V4	160	AD3	0,35
	160	AD4	0,35
V5	160	AD4	0,35
	160	AD5	0,35
V6	180	AD5	0,35
	180	AD6	0,35
v7	200	AD6	0,35



Normal service brake deceleration (A_brake_normal_service_1) - Passenger/freight P			
V0	0	AD0	1
V1	60	AD0	1
	60	AD1	0,6
V2	160	AD1	0,6
	160	AD2	1
V3	160	AD2	1
	160	AD3	1
V4	160	AD3	1
	160	AD4	1
V5	160	AD4	1
	160	AD5	1
V6	160	AD5	1
	160	AD6	1
v7	200	AD6	1



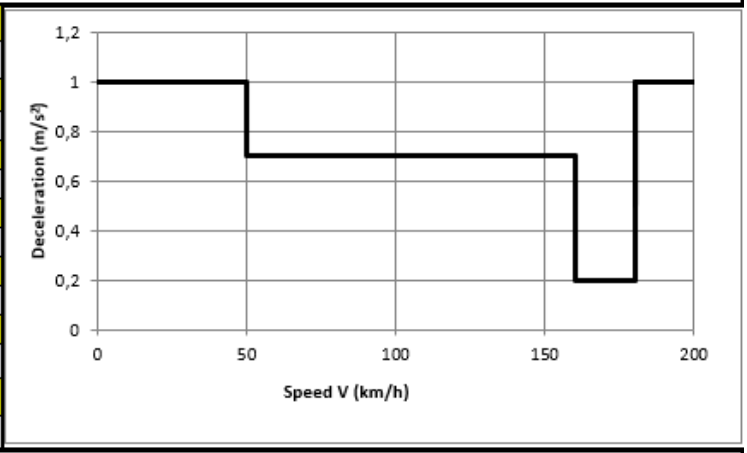
Normal service brake deceleration (A_brake_normal_service_2) - Passenger/freight P			
V0	0	AD0	0,5
V1	80	AD0	0,5
	80	AD1	0,3
V2	160	AD1	0,3
	160	AD2	1
V3	160	AD2	1
	160	AD3	1
V4	160	AD3	1
	160	AD4	1
V5	160	AD4	1
	160	AD5	1
V6	160	AD5	1
	160	AD6	1
v7	200	AD6	1



Obrázek 3.7 Skokové funkce sloužící k získání A_brake_normal_service(V) pro vlakové soupravy s brzdou v pozici P [14]

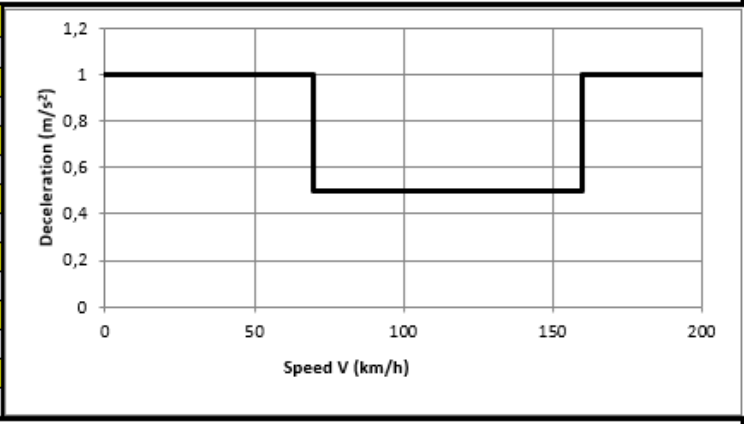
Normal service brake deceleration (A_brake_normal_service_0) - Freight G

V0	0	AD0	1
V1	50	AD0	1
	50	AD1	0,7
V2	160	AD1	0,7
	160	AD2	0,2
V3	180	AD2	0,2
	180	AD3	1
V4	180	AD3	1
	180	AD4	0,5
V5	180	AD4	0,5
	180	AD5	1
V6	180	AD5	1
	180	AD6	1
v7	200	AD6	1



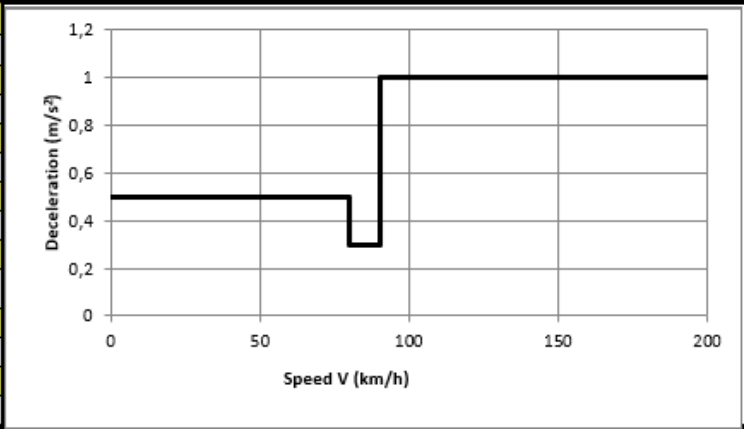
Normal service brake deceleration (A_brake_normal_service_1) - Freight G

V0	0	AD0	1
V1	70	AD0	1
	70	AD1	0,5
V2	160	AD1	0,5
	160	AD2	1
V3	160	AD2	1
	160	AD3	1
V4	160	AD3	1
	160	AD4	1
V5	160	AD4	1
	160	AD5	1
V6	160	AD5	1
	160	AD6	1
v7	200	AD6	1



Normal service brake deceleration (A_brake_normal_service_2) - Freight G

V0	0	AD0	0,5
V1	80	AD0	0,5
	80	AD1	0,3
V2	90	AD1	0,3
	90	AD2	1
V3	120	AD2	1
	120	AD3	1
V4	160	AD3	1
	160	AD4	1
V5	160	AD4	1
	160	AD5	1
V6	160	AD5	1
	160	AD6	1
v7	200	AD6	1



Obrázek 3.8 Skokové funkce sloužící k získání A_brake_normal_service(V) pro vlakové soupravy s brzdou v pozici G [14]

3.2.8.2 Konverze polohy brzdy

Na základě polohy brzdy, délky vlaku a popřípadě i cílové rychlosti lze vypočítat čas potřebný k dosažení plné brzdné síly jak pro nouzovou, tak i provozní brzdu. [2, 3.13.3.4]

3.2.8.2.1 Výpočet času k dosažení plné brzdné síly nouzové brzdy Výpočet času k dosažení plné brzdné síly nouzové brzdy se řídí následujícím vzrocem: $T\_brake\_basic\_eb = a + b \cdot (L/100) + c \cdot (L/100)^2$. Proměnné L, a, b, c se mění v závislosti na poloze brzdy, jak je možno vidět níže. [2, A.3.8]

- **Osobní vlak s brzdou v pozici P**
  - $L = max$(400m, délka vlaku v metrech)
  - $a = 2,30$
  - $b = 0$
  - $c = 0,17$ [2, A.3.8.1]
- **Nákladní vlak s brzdou v pozici P**
  - **délka vlaku** $< 900m$
    - $L = max$(400m, délka vlaku v metrech)
    - $a = -0,40$
    - $b = 0$
    - $c = 0,17$
  - $900m <$ **délka vlaku** $\leq 1500m$
    - $L = max$(400m, délka vlaku v metrech)
    - $a = -0,40$
    - $b = 1,60$
    - $c = 0,03$ [2, A.3.8.2]
- **Nákladní vlak s brzdou v pozici G**
  - **délka vlaku** $< 900m$
    - $L =$ délka vlaku
    - $a = 12$
    - $b = 0$
    - $c = 0,05$
  - $900m <$ **délka vlaku** $\leq 1500m$
    - $L =$ délka vlaku
    - $a = -0,40$
    - $b = 1,60$
    - $c = 0,03$ [2, A.3.8.3]

V závislosti na cílové rychlosti je poté možné rozlišit dva různé časy potřebné k dosažení plné brzdné síly nouzové brzdy — T_brake_emergency_cm0 a T_brake_emergency_cmt.

- **cílová rychlost = 0** $T\_brake\_emergency\_cm0 = T\_brake\_basic\_eb$
- **cílová rychlost** $\neq$ **0** $T\_brake\_emergency\_cmt = kto \cdot T\_brake\_basic\_eb$ [2, A.3.8.4]
  - $kto = 1 + Ct$
    - $Ct = 0,16$ pro nákladní vlaky s brzdou v pozici G
    - $Ct = 0,20$ pro nákladní vlaky s brzdou v pozici P
    - $Ct = 0,20$ pro osobní vlaky [2, A.3.8.5]

3.2.8.2.2 Výpočet času k dosažení plné brzdné síly provozní brzdy Výpočet je prakticky totožný jako u brzdy nouzové. Využívá se téměř stejný vzorec:
$T\_brake\_basic\_sb = a + b \cdot (L/100) + c \cdot (L/100)^2$.

Liší se však hodnotách proměnných L, a, b, c, jak lze vidět níže. [2, A.3.9]

- **Osobní vlak s brzdou v pozici P**
  - L = délka vlaku
  - $a = 3$
  - $b = 1,50$
  - $c = 0,10$ [2, A.3.9.1]
- **Nákladní vlak s brzdou v pozici P**
  - **délka vlaku** $< 900m$
    - L = délka vlaku
    - $a = 3$
    - $b = 2,77$
    - $c = 0$
  - $900m <$ **délka vlaku** $\leq 1500m$
    - L = délka vlaku
    - $a = 10,50$
    - $b = 0,32$
    - $c = 0$ [2, A.3.9.2]
- **Nákladní vlak s brzdou v pozici G**
  - **délka vlaku** $< 900m$
    - $L = max$(400m, délka vlaku v metrech)
    - $a = 3$
    - $b = 2,77$
    - $c = 0$
  - $900m <$ **délka vlaku** $\leq 1500m$
    -
    - $L = max$(400m, délka vlaku v metrech)
    - $a = 10,50$
    - $b = 0,32$
    - $c = 0,18$ [2, A.3.9.3]

I u provozní brzdy rozlišujeme na základě cílové rychlosti dva různé časy potřebné k dosažení plné brzdné síly provozní brzdy — T_brake_service_cm0 a T_brake_service_cmt.

- **cílová rychlost = 0** $T\_brake\_service\_cm0 = T\_brake\_basic\_sb$
- **cílová rychlost ≠ 0** $T\_brake\_service\_cmt = kto \cdot T\_brake\_basic\_sb$ [2, A.3.9.4]
  - hodnota kto se zjistí stejným způsobem jako u brzdy nouzové - 3.2.8.2.1

3.3 Stav původního modulu brzdných křivek

Kalkulace v původním modulu brzdných křivek vychází z Excel dokumentu Evropské železniční agentury Braking curves simulation tool v3.0 [15]. Kód, který byl původně napsán v jazyce Visual Basic byl převeden do jazyka C++ a umístěn do odděleného repozitáře.

Nejdůležitější třídou původního modulu brzdných křivek je BrakingCurveCalculator, kde dochází k výpočtu křivek EBD, SBD, GUI a s nimi spojených limitů rychlolsti.
Třída BrakingCurveInputs pak definuje podobu dat, které modul přijímá na vstupu.
Třídy GammaTrainDeceleration a LambdaTrainDeceleration se pak zabývají výpočtem zpomalení vlaku. Dále je do jazyka C++ přeformulována Excel funkce VLOOKUP, která slouží k vertikálnímu vyhledávání dat v tabulce [16]. Tento postup jen činí kód tohoto modulu méně čitelným. Je nelogické přepisovat funkci z programu Excel, namísto toho by se měly využít nástroje, které nabízí jazyk C++. Ostatní třídy jsou pak jakýmsi mixem zbylých výpočtů.

Struktura kódu původního modulu brzdných křivek vypadá následovně:

```
BrakingCurves
├── vlookup
│   ├── vlookup.cpp
│   └── vlookup.hpp
├── BrakingCurveCalculator.cpp
├── BrakingCurveCalculator.hpp
├── BrakingCurveCalculatorGetters.cpp
├── BrakingCurveInputs.hpp
├── BrakingCurveInterface.hpp
├── GammaTrainDeceleration.cpp
├── LambdaTrainDeceleration
├── Module1.cpp
├── Module2.cpp
└── Module3.cpp
```

Jak lze z kostry výše vypozorovat, kód je v původním modulu brzdných křivek dělen jen minimálně. Navíc se v jednom celku často vyskytují kusy kódu, které spolu logicky vůbec nesouvisí, či je celý celek pojmenován nesourodně s jeho obsahem (například „Module3“).

Veškeré třídy, funkce a metody pak v naprosté většině nejsou doplněny doprovodnými komentáři, a pokud ano, jedná se jen o odkaz do referenčího Excel dokumetnu. Jednotlivé metody a funkce jsou navíc velmi dlouhé, ty nejdelší se pohybují v řádech vyšších stovek řádků, což ještě více zhoršuje čitelnost a srozumitelnost kódu.

Projekt není vybaven žádným testovacím rozhraním, obsahuje pouze pár složek s testovacími daty, které ovšem ani zdaleka nepokrývají celý rozsah výpočtu brzdných křivek. Vzhledem ke skladbě kódu tohoto modulu to je však pochopitelné, protože takto strukturalizovaný kód ani příliš testovat nelze.

Jako další velký problém tohoto modulu se ukázalo nedostatečné porozumění kódu ze strany programátorů, což zapříčinilo, že z modulu brzdných křivek vznikla jakási černá skříňka — nikdo neví, co dané kusy kódu dělají, ale výsledné křivky zdají se být v pořádku.

Je velmi pravděpodobné, že v blízké budoucnosti Evropská železniční agentura vydá další specifikace pro výpočet brzdných křivek. V tu chvíli se stane tento modul zastaralým a kvůli své nerozšiřitelnosti poustpně i nepoužitelným.

3.4 Funkční a nefunkční požadavky

3.4.1 Funkční požadavky

- **F1** — **Kalkulace brzdných křivek EBD, SBD, GUI** Vzniklý modul bude schopen pro validní vstupní data vypočítat brzdné křivky EBD, SBD a GUI dle SUBSET-026-3 v360 [2].
- **F2** — **Dopočítání brzdných limitů** Z křivek EBD a SBD odvodí modul brzdných křivek následující limity: zásah nouzové brzdy (EBI), zásah provozní brzdy (SBI), varování (W), povolenou rychlost (P) a indikaci (I).

3.4.2 Nefunkční požadavky

- **N1** — **Programovací jazyk** Pro kalkulaci brzdných křivek bude použit jazyk C++.
- **N2** — **Verze** Výchozí verze subsetů bude 3.6.0.
- **N3** — **Testovatelnost** Struktura procesu výpočtu brzdných křivek bude vytvořena tak, aby bylo možné jednoduše přidávat nové testy. Každá třída pak bude vybavena jednotkovými testy.
- **N4** — **Srozumitelnost** Kód bude přehledný a pochopitelný i pro nově příchozí členy týmu. Kód bude strukturalizován do menších logických celků a metody a třídy budou doplněny vysvětlujícími komentáři.
- **N5** — **Rychlost** Kalkulace brzdných limitů proběhne až několikrát za vteřinu.

Kapitola 4
Návrh

V této kapitole bude předveden návrh nového modulu pro výpočet brzdných křivek a bude naznačeno, jak bude kód rozdělen. Dále bude pomocí diagramu tříd ukázáno, jak budou některé třídy vypadat, a v neposlední řadě bude popsán proces výpočtu brzdných křivek v tomto modulu.

4.1 Dělení kódu

Proces výpočtu brzdných křivek bude sestrojen jakožto samostatný modul, který může být eventuálně znovupoužit i na jiném projektu. Veškerý kód bude situován do složky **BrakingCurves**, která bude obsahovat další podsložku — **BrakingCurvesCalculation**

Výpočetní část bude dále rozdělena následovně:

- **BrakingCurvesCalculator** — hlavní třída, která bude sdružovat všechny části výpočtu
- **LambdaConversions** — repozitář, který bude obsahovat kód pro konverzi vstupních parametrů při použití lambda metody
- **Decelerations** — repozitář, kde bude uložen kód pro kalkulaci modelů zpomalení
- **DifferenceValues** — zde bude probíhat výpočet rozdílových hodnot podle vzorce z podkapitoly 3.2.5.3
- **SpeedInaccuracy** — adresář pro třídu vypočítavající nepřesnost rychlosti
- **BrakeDecelerationCurves** — repozitář, kde budou vznikat brzdné křivky EBD, SBD a GUI
- **SupervisionLimits** — v tomto místě budou dopočítávány limity dohledu
- **Gradient** — složka, jež bude určena pro operace spojené se sklonem trati

Další dělení kódu bude do hpp a cpp souborů tak, aby se zachovala co největší soudržnost a co nejmenší závislost mezi soubory.

Kostra kódu bude vypadat takto:

```
BrakingCurves
└── src
    ├── Calculation
    │   ├── LambdaConversions
    │   │   ├── LambdaConversions
    │   │   └── PolynomialCalculation
    │   ├── Decelerations
    │   │   ├── ASafe
    │   │   ├── AExpected
    │   │   └── ANormalService
    │   ├── DifferenceValues
    │   ├── SpeedInaccuracy
    │   ├── Gradient
    │   ├── BrakeDecelerationCurves
    │   │   ├── EBD
    │   │   ├── SBD
    │   │   └── GUI
    │   └── SupervisionLimits
    │       ├── EBI
    │       ├── SBI
    │       ├── SBI1
    │       ├── SBI2
    │       └── WarningPermittedSpeedIndication
    └── BrakingCurvesCalculator
```

4.2 Výčtové typy

Výčtový typ (v angličtině označovaný jako „enumerated type“ nebo zkráceně „enum“) je datový typ tvořený konečnou omezenou množinou pojmenovaných hodnot. Každému členu tohoto typu odpovídá zpravidla celočíselná konstanta. Výhodou jejich použití je výrazně lepší čitelnost, přehlednost a hlavně srozumitelnost kódu oproti použití nic neříkajících číselných konstant. [17]

V tomto projektu budou použity následující výčtové typy:

- **TrainType** — Typ vlaku, který dále určuje způsob výpočtu křivek.
- **BrakePosition** — Kde se nachází brzdící mechanismus.
- **TargetType** — Typ cíle, ke kterému se křivka počítá.

```
enum TrainType { GAMMA, LAMBDA };
enum BrakePosition { PASSENGER_P, FREIGHT_P, FREIGHT_G };
enum TargetType { LOA_MRSP, EOA_SVL};
```

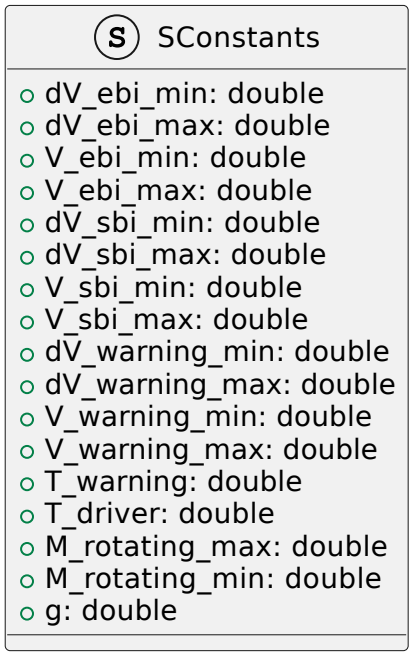
Obrázek 4.1 Demonstrace výčtových typů, které budou při výpočtu použity

4.3 Konstanty

Napříč všemi výpočty se používá hned několik konstant, které jsou zadefinované v SUBSET-026-3 [2]. Pro zpřehlednění kódu budou tyto neproměnné veličiny umístěny do jedné struktury. Jednotlivé hodnoty budou statické (statické proměnné můžeme použít aniž bychom vytvořili příslušný objekt) a označeny klíčovým slovem **constexpr**, díky čemuž jednotlivá přiřazení proběhnou již za kompilace, což povede k zlepšení výkonu programu.

Tabulka 4.1 Seznam brzdných konstant a jejich popis [2, A.3.1]

Název	Hodnota	Popis
dV_ebi_min	7.5 km/h	Minimální rozdíl mezi povolenou rychlostí a limitem pro včasný zásah nouzové brzdy.
dV_ebi_max	15 km/h	Maximální rozdíl mezi povolenou rychlostí a limitem pro včasný zásah nouzové brzdy.
V_ebi_min	110 km/h	Hodnota MRSP, kde dV_ebi začíná narůstat do dV_ebi_max.
V_ebi_max	210 km/h	Hodnota MRSP, kde dV_ebi přestává narůstat do dV_ebi_max.
dV_sbi_min	5.5 km/h	Minimální rozdíl mezi povolenou rychlostí a limitem pro včasný zásah provozní brzdy.
dV_sbi_max	10 km/h	Maximální rozdíl mezi povolenou rychlostí a limitem pro včasný zásah provozní brzdy.
V_sbi_min	110 km/h	Hodnota MRSP, kde dV_sbi začíná narůstat do dV_sbi_max.
V_sbi_max	210 km/h	Hodnota MRSP, kde dV_sbi přestává narůstat do dV_sbi_max.
dV_warning_min	4 km/h	Minimální rozdíl mezi povolenou rychlostí a limitem pro varování řidiče.
dV_warning_max	5 km/h	Maximální rozdíl mezi povolenou rychlostí a limitem pro varování řidiče.
V_warning_min	110 km/h	Hodnota MRSP, kde dV_warning začíná narůstat do dV_warning_max.
V_warning_max	140 km/h	Hodnota MRSP, kde dV_warning přestává narůstat do dV_warning_max.
T_warning	2 s	Čas mezi varováním a zásahem provozní brzdy.
T_driver	4 s	Reakční čas řidiče mezi povolenou rychlostí a zásahem provozní brzdy.
M_rotating_max	15 %	Maximální hodnota rotující hmoty vyjádřena v procentech celkové hmotnosti vlaku.
M_rotating_min	2 s %	Minimální hodnota rotující hmoty vyjádřena v procentech celkové hmotnosti vlaku.
g	$9.81 m/s^2$	gravitační zrychlení



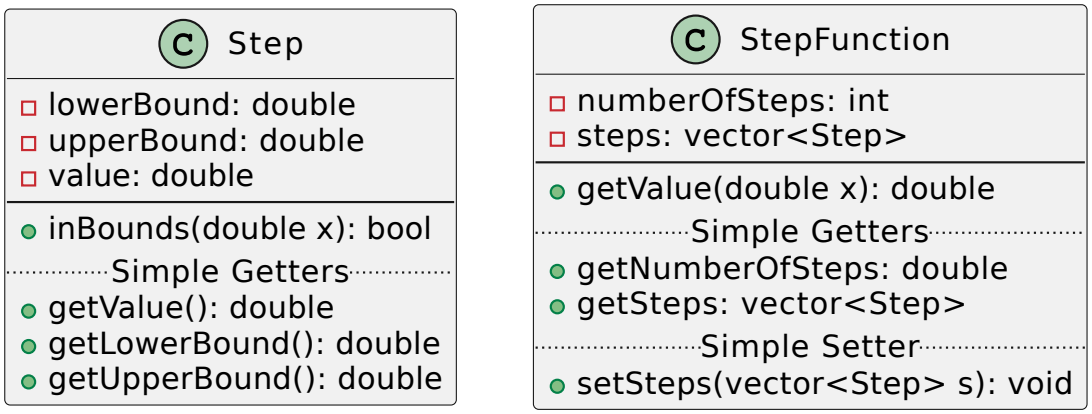
Obrázek 4.2 Diagram struktury SConstants

4.4 Modely

Pro lepší čitelnost a zjednodušení práce je třeba vytvořit datové struktury pro skokové funkce, vstupní či výstupní data a brzdné parametry.

4.4.1 Skoková funkce

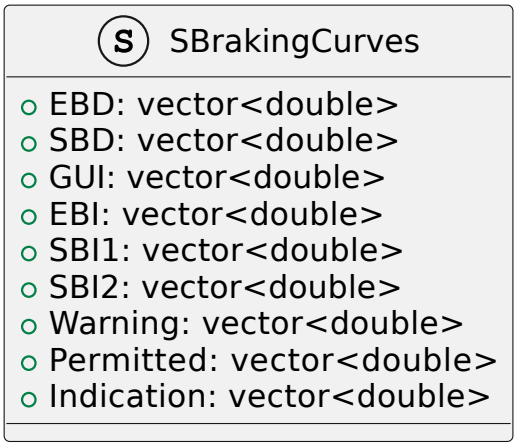
Skokové funkce se budou skládat z jednoho či více skoků, které budou obsahovat údaje o spodní a horní hranici a samotnou hodnotu skoku. Za pomoci metody inBounds(double x) bude poté možno zjistit, zda hodnota x patří do daného skoku, tzn. nachází se mezi spodní a horní hranicí (včetně). Samotná StepFunction pak bude kontrolovat, zda skoky začínají od nuly a zda na sebe navazují. Pokud tomu tak nebude, bude vyhozena výjimka. Diagramy třídy Step a StepFunction jsou podrobně rozveden na obrázku 4.3.



Obrázek 4.3 Diagramy tříd Step a StepFunction

4.4.2 Výstupní parametry

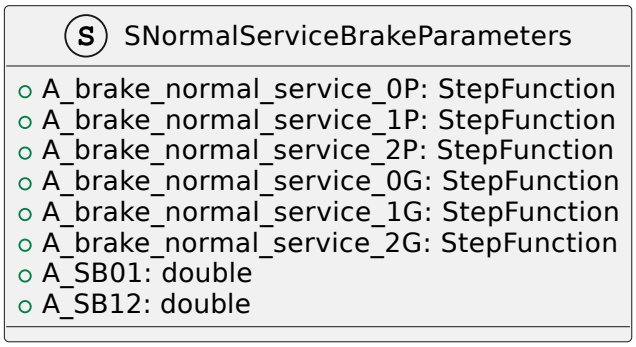
Výsledné hodnoty brzdných křivek budou reprezentovány sekvencí po sobě jdoucích hodnot, které budou uloženy v datovém kontajneru std::vector [18], tak jak tomu bylo v původním modulu brzdných křivek. Stejné výstupní parametry ulehčí integraci do projektu trenažeru ETCS. Výsledné křivky pak budou zabaleny do struktury SBrakingCruves, která bude předávána do EVC. Návrh této struktury si lze prohlédnout na obrázku 4.4.



Obrázek 4.4 Diagram struktury SBrakingCurves

4.4.3 Brzdné parametry

Parametry normálního zpomalení provozní brzdy nejsou vyjádřeny pouze jednou skokovou funkcí, narozdíl od brzdy nouzové či plného zpomalení brzdy provozní, ale obsahuje skokových funkcí hned šest — dvě sady (jednu pro osobní a jednu pro nákladní vlak) po třech skokových funkcích. SNormaServiceBrakeParameters, jak bude struktura pro uchovávání dat týkajících se normálního zpomalení provozní brzdy pojmenována, bude pak obsahovat ještě hodnoty A_SB01 a A_SB02, které se používají k finálnímu určení hodnoty A_brake_normal_service (viz 3.2.4).



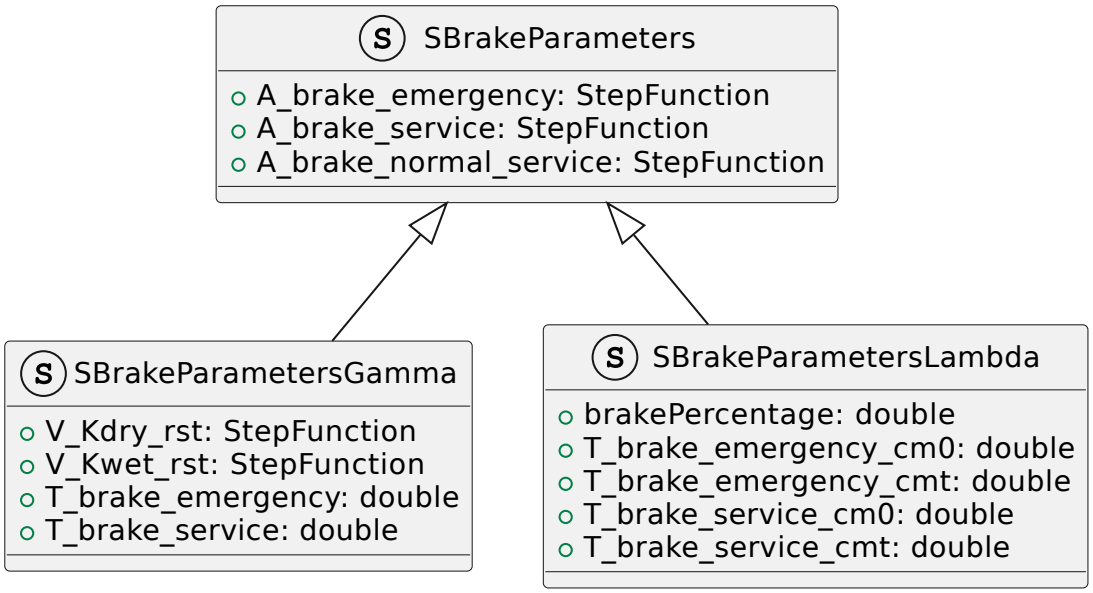
Obrázek 4.5 Diagram struktury SNormalServiceBrakeParameters

Brzdné parametry se liší podle toho, jestli se jedná o gamma vlak, nebo lambda vlak. Některé proměnné však mají společné - konrétně se jedná o skokové funkce A_brake_emergency, A_brake_service a sadu A_brake_normal_service. Z tohoto důvodu budou struktury SBrakeParametersGamma a SBrakeParametersLambda, reprezentující brzdné parametry gamma a lambda vlaku, odvozeny prostřednictvím dědičnosti od struktury SBrakeParameters, která v sobě bude uchovávat hodnoty skokových funkcí zmíněných výše.

Struktura SBrakeParametersGamma pak bude dále obsahovat korekční faktory Kdry_rst(V) a Kwet_rst(V),podrobně popsány v podkapitole 3.2.7, a hodnoty T_brake_emergency a T_brake_service, které se dále používají pro výpočet času zprovoznění brzd.

Struktura SBrakeParametersLambda bude sloužit k uchování výsledků lambda konverze, a tudíž bude obsahovat i následující proměnné:

- *brakePercentage* - brzdné procento, ze kterého jsou ostatní hodnoty odvozeny
- *T_brake_emergency_cm0* - hodnota využívající se k výpočtu doby zprovznění nouzové brzdy, když je cílová rychlost rovna 0
- *T_brake_emergency_cmt* - hodnota využívající se k výpočtu doby zprovznění nouzové brzdy, když cílová rychlost není rovna 0
- *T_brake_service_cm0* - hodnota využívající se k výpočtu doby zprovznění provozní brzdy, když je cílová rychlost rovna 0
- *T_brake_service_cmt* - hodnota využívající se k výpočtu doby zprovznění provozní brzdy, když cílová rychlost není rovna 0



Obrázek 4.6 Diagramy struktur SBrakeParameters, SBrakeParametersGamma a SBrakeParametersLambda

4.4.4 Vstupní parametry

Vstupní parametry budou rozděleny do tří různých struktur — SArgumentsInitial, SArgumentsNational a SArgumentsVariable.

SArgumentsInitial budou uchovávat počáteční parametry nezbytné k výpočtu brzdných křivek. Patří mezi ně například typ vlaku, pozice brzdy či délka soupravy. Mimo jiné bude obsahovat i brzdné parametry pro vlak typu gamma zmíněné v sekci 4.4.3, nebo brzdné procento. Jelikož není dopředu známo, jakého typu vlak je, nelze přesně určit typ brzdných parametrů.
Ve svém kódu proto využiji šablonu třídy std::variant. Jakákoli instance std::variant totiž v daném okamžiku obsahuje buď hodnotu jednoho ze svých alternativních typů, nebo nemá žádnou hodnotu [19].

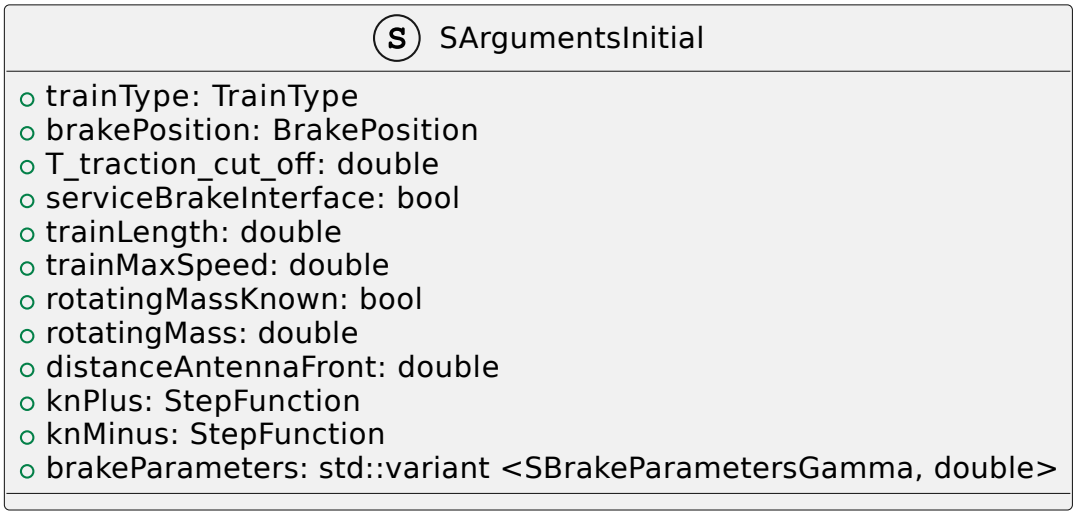
K uchování argumentů, které se často mění, slouží struktura SArgumentsVariable. Do této skupiny patří například cílová rychlost, vzdálenost cíle či sklon trati.

SArgumentsNational je pak struktura zabalující národní hodnoty. Národní hodnoty ERTMS/ETCS jsou souborem parametrů definovaných manažerem infrastruktury pro přizpůsobení chování palubního systému tak, aby splňovalo výkonnostní a bezpečnostní požadavky dané země [11].

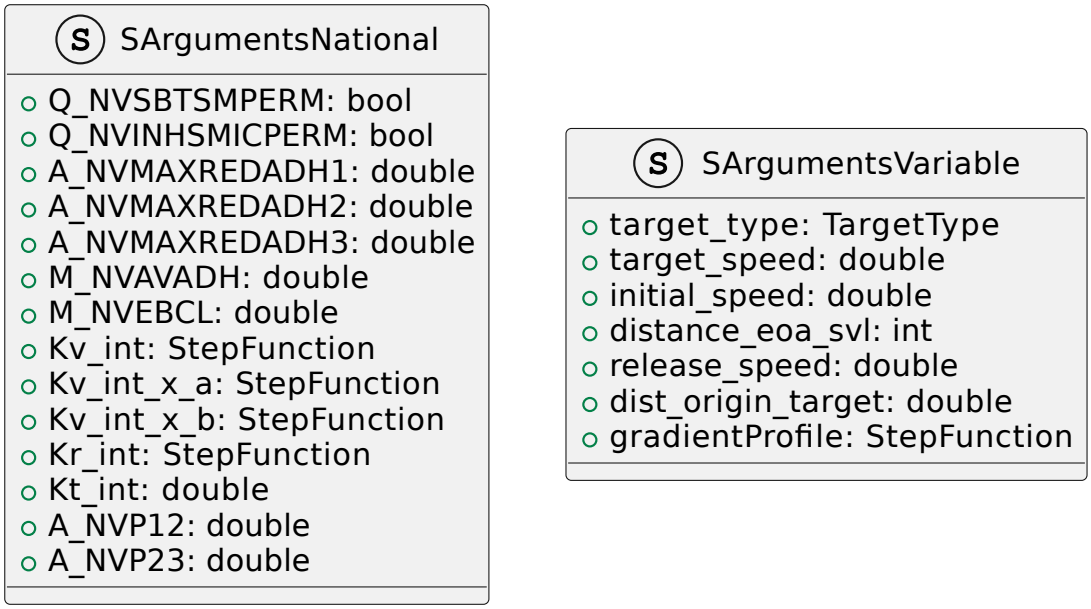
V celém systému ETCS se používá velké množství těchto hodnot, avšak ke komputaci brzdných křivek postačí proměnné popsané v tabulce 4.2.

Tabulka 4.2 Národní hodnoty používané při výpočtu brzdných křivek [14]

Národní hodnota	Popis
Q_NVSBTSMPERM	Povolení použití provozní brzdy při sledování cílové rychlosti
Q_NVINHSMICPERM	Povolení zakázat kompenzaci nepřesnosti měření rychlosti
A_NVMAXREDADH1	Maximální hodnota zpomalení za podmínek snížené adheze(1)
A_NVMAXREDADH2	Maximální hodnota zpomalení za podmínek snížené adheze(2)
A_NVMAXREDADH3	Maximální hodnota zpomalení za podmínek snížené adheze(3)
M_NVAVADH	Váhový faktor pro dostupnou adhezi kola/kolejnice
M_NVEBCL	Úroveň spolehlivosti pro bezpečné zpomalení nouzové brzdy na suchých kolejích
Kv_int	Korekční faktor závislý na rychlosti
Kv_int_x_a	Podskupina Kv_int pro konvenční osobní vlaky
Kv_int_x_b	Podskupina Kv_int pro konvenční osobní vlaky
Kr_int	Korekční faktor závislý na délce vlaku
Kt_int	Korekční faktor pro dobu zprovoznění brzdy
A_NVP12	Dolní mez zpomalení k určení sady Kv_int, která se má použít
A_NVP23	Horní mez zpomalení k určení sady Kv_int, která se má použít



Obrázek 4.7 Diagram struktury SArgumentsInitial



Obrázek 4.8 Diagramy struktur SArgumentsNational a SArgumentsVariable

4.5 Lambda metoda - konverzní modely

Pokud se jedná o vlak typu lambda, ke konverzi brzdného procenta a dalších parametrů se použije třída **LambdaConversions**. Tato třída v konstruktoru přijme následující parametry: brzdné procento, délku vlaku a pozici brzdy. Všechny tyto parametry pak budou privátně uloženy a později použity při konverzi.

Třída bude dále obsahovat metody s názvy *getDecelerationSpeedModelEB*, *getDecelerationSpeedModelSB*, *getDecelerationSpeedModelNSB*, *getBuildUpTimeEBcm0*, *getBuildUpTimeEBcmt*, *getBuildUpTimeSBcm0*, *getBuildUpTimeSBcmt*, *getBrakeParametersLambda* a *calculatePolynomials*.

Metody getDecelerationSpeedModelEB a getDecelerationSpeedModelSB budou sloužit k transformaci brzdného procenta na skokovou funkci reprezentující deceleraci nouzové, respektive provozní brzdy. Tato konverze bude probíhat podle zadefinovaných vzorců zmíněných v podkapitole 3.2.8. Při obou kalkulacích bude volána metoda calculatePolynomials, která bude představovat společnou část výpočtu pro obě brzdy. Nedojde tak ke zbytečné duplikaci kódu.

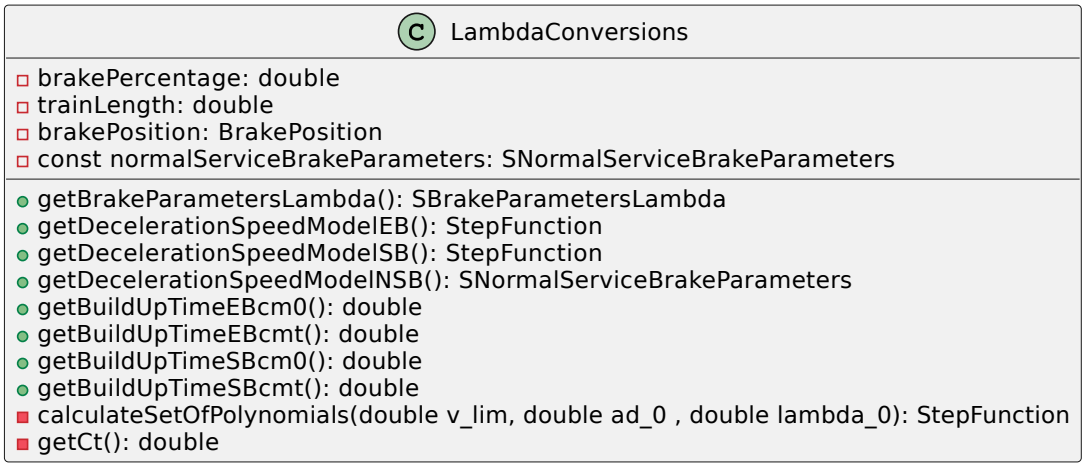
Metoda getDecelerationSpeedModelNSB bude vracet parametry pro výpočet křivek GUI, které budou ve třídě LambdaConversions uloženy jako konstantní hodnoty.

Metody getBuildUpTimeEBcm0 a getBuildUpTimeSBcm0 pak přetransformují zbylé parametry (tj. délka vlaku, a pozice brzdy) na čas potřebný k dosažení plné brzdné síly nouzové a provozní brzdy pro cílovou rychlost rovnou nule.

Metody getBuildUpTimeEBcmt a getBuildUpTimeSBcmt poskytnou naopak čas potřebný k dosažení plné brzdné síly nouzové a provozní brzdy pro cílovou rychlost různou od nuly.

Ze třídy LambdaConversions pak bude možné získat všechny brzdné parametry pomocí metody getBrakeParametersLambda.

Pro lepší přehlednost kódu bude vytvořena ještě jedna třída s názvem **PolynomialCalculation**, která bude mít na starost kalkulaci polynomu z 3.2.8.1 po dosazení konkrétních hodnot za proměnné.

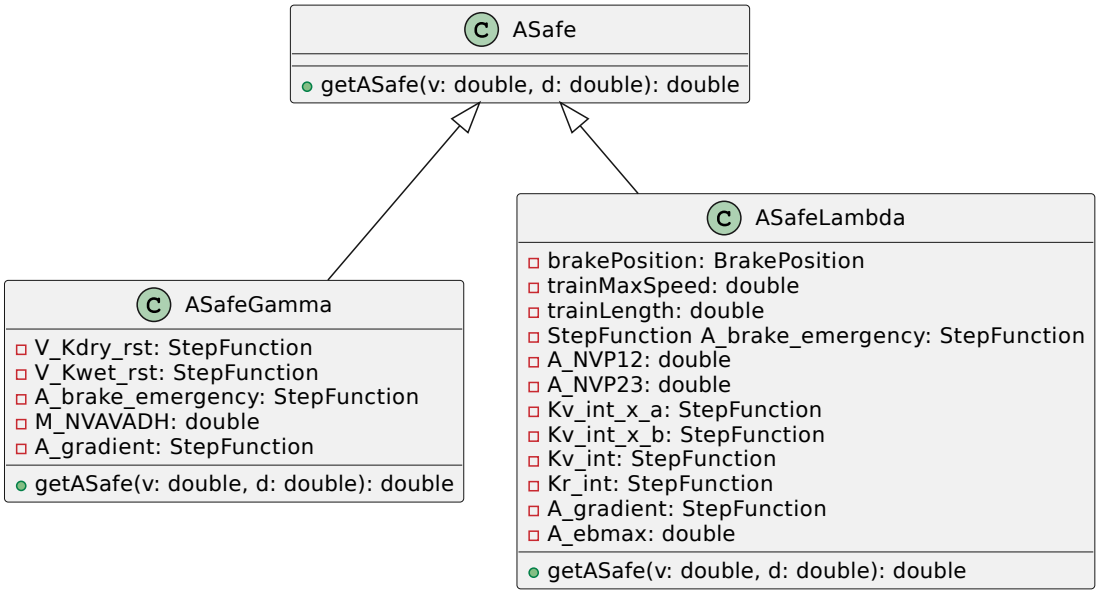


Obrázek 4.9 Diagram třídy LambdaConversions

4.6 Třídy zpomalení

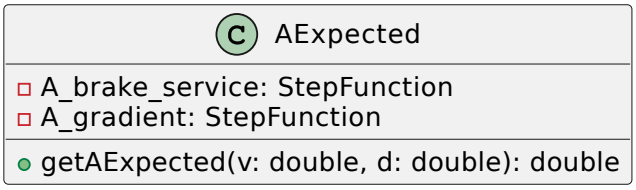
Lze rozlišit tři druhy zpomalení — A_safe, A_expected a A_normal_service. Pro výpočet daného zpomalení se využije hodnot zpomalení příslušných brzd a korekčních faktorů.

Výpočet A_Safe se jako jediný bude lišit podle typu vlaku, tzn. pro vlak typu gamma se využijí jiné korekční faktory než pro vlak typu lambda. Z tohoto důvodu bude využita dědičnost, kterou jazyk C++ poskytuje, a budou sestrojeny dvě různé třídy ASafeGamma a ASafeLambda, které budou mít společného předka — ASafe, jak je možno vidět na diagramu 4.10.

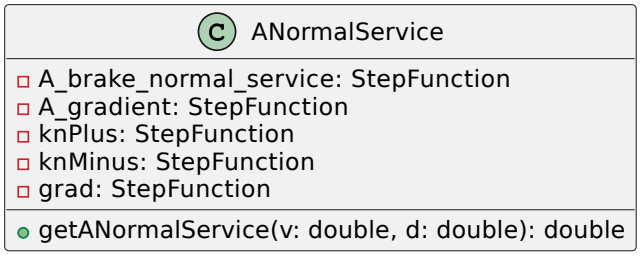


Obrázek 4.10 Diagramy tříd ASafe, ASafeGamma a ASafeLambda

Pro kalkulaci zpomalení A_expected a A_normal_service pak budou sestrojeny třídy AExpected a ANormalService, které budou přijímat a uchovávat parametry potřebné k výpočtu daných zpomalení.



Obrázek 4.11 Diagram třídy AExpected



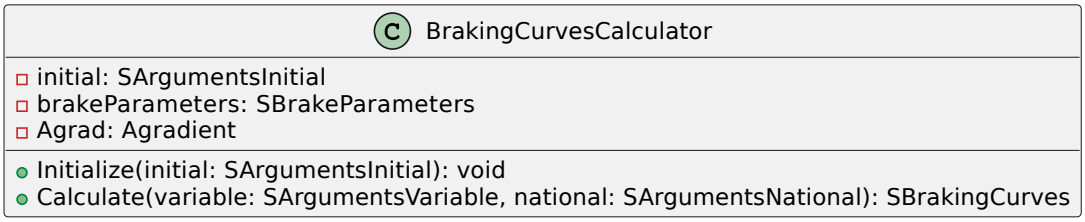
Obrázek 4.12 Diagram třídy ANormalService

4.7 Třídy pro kalkulaci brzdných křivek

Pro jednotlivé křivky budou vytvořeny stejnojmenné třídy, kde proběhne výpočet křivek EBD, SBD, GUI a limitů EBI, SBI1, SBI2, varování, povolené rychlosti a indikace. Pomocí veřejně přístupných statických metod bude možné si tyto křivky ze třídy vytáhnout a to ve formě vector<double> (viz 4.4.2).

4.8 Třída BrakingCurvesCalculator

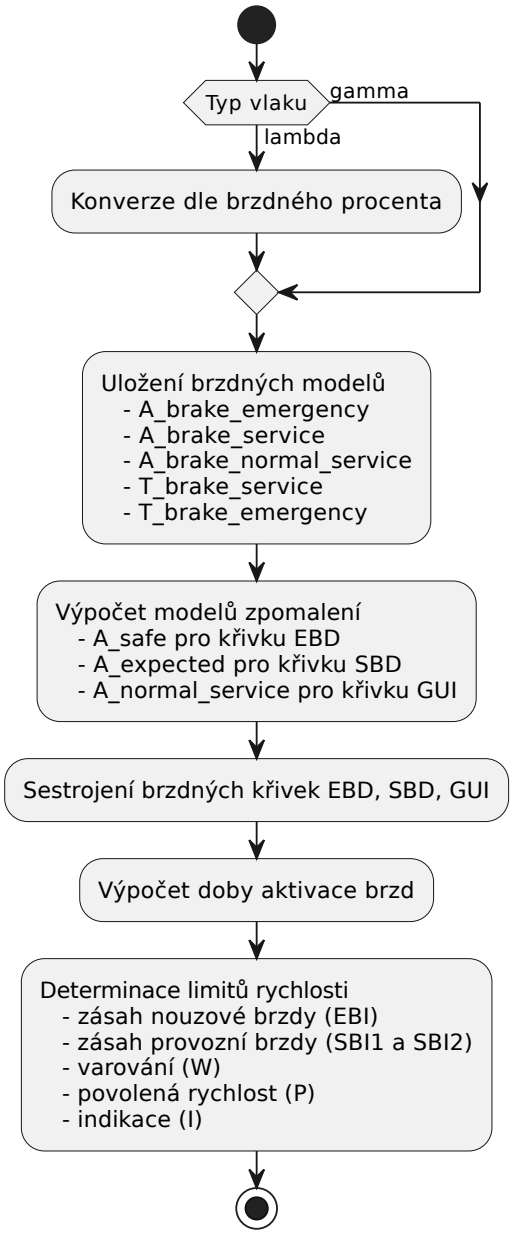
Tato třída sjednocuje celý výpočet a bude volána z EVC. Po spuštění vlaku bude incializována vstupními parametry SArgumentsInitial (podrobně popsány v sekci 4.4.4) za pomocí metody *Initialize*. Metoda *Calculate* poté bude volána k výpočtu brzdných křivek a jejím výstupem bude struktura SBrakingCurves (viz 4.4.2).



Obrázek 4.13 Diagram třídy BrakingCurvesCalculator

4.9 Proces výpočtu brzdných křivek

Jak bylo zmíněno v kapitole 3.2.3, výpočet brzdných křivek se liší podle typu vlaku. Jedná-li se o lambda metodu výpočtu, je potřeba provést konverzi brzdného procenta na brzdné parametry. V případě gamma metody jsou brzdné parametry jasně definovány a předány modulu brzdných křivek ve formě vstupních dat. Po uložení těchto brzdných parametrů je výpočet totožný. Dojde k výpočtu modelů zpomalení A_safe, A_expected a A_normal_service, které se dále využijí ke kalkulaci křivek EBD, SBD a GUI. Poté se vykalkuluje čas potřebný k plnému zporovznění brzd a z již vypočítaných křivek se dopočítají brzdné limity EBI, SBI1, SBI2, varování, povolené rychlosti a indikace. Celý tento proces je znázorněn diagramem 4.14. [2, Figure 28]



Obrázek 4.14 Diagram procesu výpočtu brzdných křivek [2, Figure 28]

Kapitola 5
Implementace

V rámci této bakalářské práci byl naprogramován nový modul pro kalkulaci brzdných křivek vlaku jedoucího pod dohledem vlakového zabezpečovače ETCS. Kód je strukturován do menších logických celků, tak jak bylo navrhnuto v podkapitole 4.1, a je doplněn vysvětlujícími komentáři u všech metod či částí implementace, které nejsou na první pohled jasně pochopitelné.

Jakožto rozhraní mezi novým modulem brzdných křivek a EVC působí třída BrakingCurvesCalculator. Ta, jak je uvedeno v podkapitole 4.8, sdružuje všechny části výpočtu.

Tuto třídu je nutné vždy po spuštění vlaku inicializovat počátečními parametry jako je například délka vlaku. Tyto parametry se uloží do privátní proměnné a za pomocí některých z nich se zkonstruuje třída Agradient, která zaštiťuje operace spojené s korekcí sklonu trati. Dále se uloží brzdné parametry v závislosti na typu vlaku.

```
void BrakingCurvesCalculator::Initialize(shared_ptr<SArgumentsInitial> initial){
    this->initial = initial;
    /**
     * Initialize Agradient
     */
    Agrad = Agradient(initial->rotatingMassKnown,
                      initial->rotatingMass,
                      initial->trainLength);
    /**
     * Obtain brake parameters gamma or brake percentage
     * Brake percentage is then converted to brake parameters lambda
     */
    try{
        if (initial->trainType == TrainType::GAMMA) {
            brakeParameters = make_shared<SBrakeParametersGamma>
                    (std::get<SBrakeParametersGamma>(initial->brakeParameters));
        }else{
            double brakePercentage = get<double>(initial->brakeParameters);
            LambdaConversions lambdaConversions(brakePercentage, initial->
    trainLength, initial->brakePosition);
            brakeParameters = make_shared<SBrakeParametersLambda>
                    (lambdaConversions.getBrakeParametersLambda());
        }
    }catch (const bad_variant_access& ex) {
        throw invalid_argument("The type of train and brakeParameters in initial
     data do not correspond.");
    }
}
```

Výpis kódu 5.1 Inicializace třídy BrakingCurvesCalculator

Po inicializaci třídy BrakingCurvesCalculator lze zavolat metodu *Calculate*. Ta v první části, jak lze vidět ve výpisu kódu 6.3, zkontroluje, zda byla třída inicializována, a poté získá hodnotu A_gradient reprezentující sklon trati po korekci. Dále jsou za pomocí brzdných parametrů sestrojeny ukazatele na třídy A_expected a A_normal_service představující zpomalení vlaku.

Během procesu konstrukce třídy zpomalení A_safe je potom nutné provést tzv. „downcast“ ukazatele na brzdné parametry. Jinými slovy je potřeba převést ukazatel základní třídy SBrakeParameters na ukazatel třídy odvozené, tj. SBrakeParametersGamma, nebo SBrakeParametersLambda [20]. Poté se využije polymorfismu (polymorfismus v C++ znamená, že stejná entita se v různých scénářích může chovat odlišně [21]), který jazyk C++ poskytuje, a do proměnné A_safe se uloží ukazatel na instanci třídy ASafeGamma, nebo ASafeLambda.

```
if (initial == nullptr || brakeParameters == nullptr) {
    //! Calculator has to be initialized first!
    throw logic_error("Braking curves were not initialized!");
}
StepFunction A_gradient = Agrad.getAGradient(variable->gradientProfile);
shared_ptr<AExpected> A_expected = make_shared<AExpected>(
                                        brakeParameters->A_brake_service,
                                        A_gradient);
StepFunction normalServiceBrakeParams =
    brakeParameters->A_brake_normal_service.getNormalServiceBrake(
       brakeParameters->A_brake_service.getValue(0),initial->brakePosition);
shared_ptr<ANormalService> A_normal_service = make_shared<ANormalService>(
                                                    normalServiceBrakeParams,
                                                    A_gradient, initial->knPlus,
                                                    initial->knMinus,
                                                    variable->gradientProfile);
shared_ptr<ASafe> A_safe;
/**
 * ASafe - different initialization for gamma and lambda
 */
if (initial->trainType == TrainType::GAMMA) {
    /**
     * Dont have to check the cast, because from initialize we know,
     * that SBrakeParametersGamma is there
     */
    auto brakeParametersGamma = dynamic_pointer_cast<SBrakeParametersGamma>(
                                    brakeParameters);
    A_safe = make_shared<ASafeGamma>(brakeParametersGamma->V_Kdry_rst,
                                     brakeParametersGamma->V_Kwet_rst,
                                     brakeParametersGamma->A_brake_emergency,
                                     national->M_NVAVADH, A_gradient);
}else{
    auto brakeParametersLambda = dynamic_pointer_cast<SBrakeParametersLambda>(
                                   brakeParameters);
    A_safe  = make_shared<ASafeLambda>(initial->brakePosition,
                                       initial->trainMaxSpeed,
                                       initial->trainLength,
                                       brakeParametersLambda->A_brake_emergency,
                                       national->A_NVP12, national->A_NVP23,
                                       national->Kv_int_x_a,
                                       national->Kv_int_x_b,
                                       national->Kv_int,
                                       national->Kr_int, A_gradient);
}
```

Výpis kódu 5.2 Část procesu kalkulace třídy BrakingCurvesCalculator

Na závěr metody *Calculate* se za pomocí příslušných tříd a metod vypočítají všechny potřebné brzdné křivky a uloží se do výsledné struktury SBrakingCurves.

5.1 Třídy limitů rychlosti a brzdných křivek

Mezi tyto třídy patří: EBD, SBD, GUI, EBI, SBI1, SBI2 a WarningPermittedIndication, která zaštiťuje výpočet hned tří brzdných limitů.

Jejich implementace je velmi podobná, liší se hlavně ve vstupních parametrech a korekčních faktorech použitých při výpočtu.

Jako příklad bude použita třída SBD. Pokud je typ cíle EOA_SvL, parametry jsou nainicializovány na nulovou hodnotu a poté se v cyklu postupně po desetinách vypočítává brzdná vzdálenost pro rychlosti v km/h. Využívá se při tom zpomalení A_expected, které je jedním ze vstupních parametrů.

```
vector<double> SBD::getSBD(double initial_speed, TargetType targetType,
                           double dist_origin_target,
                           shared_ptr<AExpected> AExpected) {

vector<double> sbdCurve ((initial_speed + 30)*10, NAN);
if (targetType == TargetType::EOA_SVL) {
    double v_sbd = 0;
    double fullServiceDeceleration =
        AExpected->getAExpected(v_sbd, dist_origin_target);

    int iterator = 0;
    double lastSBDValue = 0;
    sbdCurve[iterator++] = lastSBDValue;

    for (double speed = 0.1;
         speed <= (initial_speed + 30 + 0.001);
         speed += 0.1) {
        //! Speed is in km/h by default, we need to convert it to m/s
        double currentSpeedInMetresPerSecond = speed/3.6;
        double previousSpeedInMetresPerSecond = (speed-0.1)/3.6;
        /**
         * Formula taken from Braking curves simulation tool v3.0
         */
        double nextValue = lastSBDValue
            + ((pow(currentSpeedInMetresPerSecond, 2)
            - pow(previousSpeedInMetresPerSecond, 2))
            / (2*fullServiceDeceleration));

        /**
         * If the value is greater than target distance
         * we reached the end of computation
         */
        if (nextValue >= dist_origin_target)
            break;
        //! Adding next value to the result and increasing iterator
        sbdCurve[iterator++] = nextValue;
        lastSBDValue = nextValue;
        fullServiceDeceleration =
            AExpected->getAExpected(speed, dist_origin_target - lastSBDValue);
    }
}
return sbdCurve;

}
```

Výpis kódu 5.3 Inicializace třídy BrakingCurvesCalculator

5.2 Další třídy

Mezi další třídy, které se v novém modulu brzdných křivek používají, patří například třída SpeedInaccuracy či třída DifferenceValues.

Třída SpeedInaccuracy počítá nepřesnost měření rychlosti zmíněnou v podkapitole 3.2.5.2.

```
double SpeedInaccuracy::getSpeedInaccuracy(double speed) {
    double S41_speed_inacc;
    if (speed <= 30.0) //! +- 2 km/h for speed lower than 30 km/h
            S41_speed_inacc = 2.0;
    else if (speed >= 500.0)  //! +- 12 km/h at 500 km/h
            S41_speed_inacc = 12.0;
    else //! increasing linearly up to +- 12 km/h
            S41_speed_inacc = 2.0 + ((10.0 * (speed - 30.0)) / 470.0);

    return S41_speed_inacc;
}
```

Výpis kódu 5.4 Implementace třídy SpeedInaccuracy

Třída DifferrenceValues obaluje výpočty hodnot dV_ebi, dV_sbi a dV_warning podle podkapitoly 3.2.5.3.

```
double DifferenceValues::getDifferenceValue(double ceiling_speed,
                                            double V_min,
                                            double V_max,
                                            double dV_min,
                                            double dV_max) {
    /**
     * SUBSET-026-3 - 3.13.9.2.3
     */
    if (ceiling_speed > V_min) {
        double C = (dV_max - dV_min) / (V_max - V_min);
        return std::min((dV_min + C * (ceiling_speed - V_min)), dV_max);
    }
    return dV_min;
}

double DifferenceValues::dV_ebi(double speed) {
    return getDifferenceValue(speed, SConstants::V_ebi_min,
                              SConstants::V_ebi_max, SConstants::dV_ebi_min,
                              SConstants::dV_ebi_max);
}

double DifferenceValues::dV_sbi(double speed) {
    return getDifferenceValue(speed, SConstants::V_sbi_min,
                              SConstants::V_sbi_max, SConstants::dV_sbi_min,
                              SConstants::dV_sbi_max);
}

double DifferenceValues::dV_warning(double speed) {
    return getDifferenceValue(speed, SConstants::V_warning_min,
                              SConstants::V_warning_max,
                              SConstants::dV_warning_min,
                              SConstants::dV_warning_max);
}
```

Výpis kódu 5.5 Implementace třídy DifferrenceValues

5.3 Problémy s implementací

5.3.1 Ztráta přesnosti při počítání s plovoucí desetinou čárkou

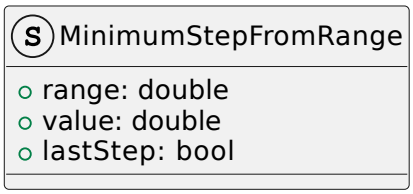
Desítkové hodnoty s plovoucí desetinnou čárkou obecně nemají přesnou binární reprezentaci. Toto je vedlejší efekt toho, jak CPU reprezentuje data s pohyblivou řádovou čárkou. Z tohoto důvodu může dojít ke ztrátě přesnosti a některé operace s plovoucí desetinnou čárkou mohou způsobit neočekávané výsledky. [22]

Vzhledem k tomu, že brzdné vzdálenosti se v nově vzniklém modulu brzdných křivek počítají v cyklu, kde se rychlost iteruje po desetinách, některá porovnání desetinných čísel nevycházela tak, jak by se očekávalo. Tento bug byl vyřešen přičtením fiktivní hodnoty („dummy value“). Toto řešení však zhoršuje přehlednost a srozumitelnost kódu.

5.3.2 Implementace kompenzace délky vlaku při kalkulaci A_gradient

A_gradient představuje zrychlení/zpomalení vzhledem ke sklonu trati a při jeho kalkulaci musí být zohledněna délka vlaku tak, jak je uvedeno v paragrafu 3.2.4.1.1. Hodnota A_gradient je reprezentována skokovou funkcí, která však na tuto operaci (hledání minima v daném intervalu a jeho přepsání se do intervalu dalšího) nebyla úplně připravena. Ve třídě Agradient, která výpočet A_gradient zaštiťuje, tak vznikla privátní struktura MinimumStepFromRange a metoda getMinimumStepFromRange, které pomáhají tento problém řešit.

Metoda getMinimumFromRange totiž najde skok z daného intervalu skokové funkce, který má nejnižší hodnotu, a vrátí společně s touto hodnotou i jeho velikost (pokud je skok v daném intervalu jen částečně, vrací se pouze velikost části skoku ležící v intervalu) a informaci o tom, zda se jedná o skok poslední, či nikoliv. Výstupem této metody pak je již zmiňovaná struktura MinimumStepFromRange.



Obrázek 5.1 Diagram struktury MinimumStepFromRange

```
MinimumStepFromRange getMinimumStepFromRange(StepFunction &sf,
                                             double min, double max) {
    /**
     * This eases the work with method, cause we can insert values lower than 0
     * and they will be adjusted to 0
     */
    if (min < 0)
        min = 0;
    /**
     * This for loop finds all steps that are at least partly in range min-max
     */
    std::vector<Step> stepsInRange;
    for (Step step : sf.getSteps()) {
        if (step.getUpperBound() < min)
            continue;
        if (step.getLowerBound() > max)
            break;
        //! To get rid of edge cases
        if (step.getUpperBound() == min || step.getLowerBound() == max)
            continue;
        stepsInRange.push_back(step);
    }
    /**
     * Find the lowest step in range
     */
    Step res = stepsInRange[0];
    for (Step step : stepsInRange) {
        if (step.getValue() < res.getValue())
            res = step;
    }
    /**
     * Adjusting the lower bound value:
     *   if lower bound is lower than min, that means that step lays only
     *   partly in given interval, so we do not want to obtain the range
     *   of the whole step, but only range of the part that lays in
     *   said interval
     */
    double lb = (min > res.getLowerBound()) ? min : res.getLowerBound();
    /**
     * Find out whether the step is last in the step function or not
     */
    bool lastStep = (res.getUpperBound() == std::numeric_limits<double>::max())
                    ? true
                    : false;
    MinimumStepFromRange toReturn =
       MinimumStepFromRange(res.getUpperBound() - lb, res.getValue(), lastStep);
    return toReturn;
}
```

Výpis kódu 5.6 Implementace metody getMinimumFromRange

Kapitola 6
Testování

Všechny třídy a jejich veřejné metody jsou v novém modulu brzdných křivek vybaveny jednotkovými testy. V modulu se dále nachází dva velké akceptační testy, jejichž obsah by měl ověřit dva nejpravděpodobnější scénáře, které po integraci do projektu trenažeru ETCS mohou nastat. Jedním z těchto scénářů je inicializace třídy BrakingCurvesCalculator a následný výpočet brzdných křivek pro vlak typu gamma, typ cíle EOA a cílovou rychlost 0. Druhý scénář je velmi podobný, akorát se jedná o vlak typu lambda, tudíž je modulu brzdných křivek na vstupu předáno pouze brzdné procento, namísto brzdných parametrů vlaku, a musí tak dojít ještě ke konverzi.

Do projektu bylo také přidáno generování spustitelného souboru BrakingCurves_test, který hromadně spouští všechny testy a vypisuje jejich výsledek. Testování není jinak automatizováno.

Struktura adresáře /test, kde jsou tesy situovány, kopíruje strukturu adresáře /src z podkapitoly 4.1 a navíc je tam přidána složka TestData sloužící k uchovávání objemnějších referenčních dat.

Kostra adresáře /test vypadá následovně:

```
test
├── AcceptanceTests
├── Calculation
│   ├── LambdaConversions
│   ├── Decelerations
│   │   ├── ASafe
│   │   ├── AExpected
│   │   └── ANormalService
│   ├── DifferenceValues
│   ├── SpeedInaccuracy
│   ├── Gradient
│   ├── BrakeDecelerationCurves
│   │   ├── EBD
│   │   ├── SBD
│   │   └── GUI
│   └── SupervisionLimits
│       ├── EBI
│       ├── SBI
│       ├── SBI1
│       ├── SBI2
│       └── WarningPermittedSpeedIndication
└── TestData
```

K testování se využívá rozhraní GoogleTest, které je součástí i modulu EVC v projektu trenažeru ETCS.

6.1 GoogleTest

Rozhraní GoogleTest je založeno na tzv. asercích. Jedná se o tvrzení, u kterých se kontroluje, zda jsou pravdivá, či nikoliv. Výsledkem pak může být úspěch, nefatální selhání, nebo fatální selhání. Pokud dojde k fatálnímu selhání, přeruší se všechny aktuální funkce, jinak program normálně pokračuje.

Testy pak používají aserce k ověření chování testovaného kódu. Pokud test nedoběhne nebo obsahuje aserci, která byla vyhodnocena jakožto nepravdiá, pak celý test selže. V opačném případě se hovoří o úspěšném testu.

Jednotlivé testy se dále mohou sdružovat do testovacích sad. Do těchto sad by měly být umístěny testy, které ověřují funkčnost stejné třídy. Pokud existují testy, které spolu sdílí testovací data, mohou být umístěny do tzv. test fixture třídy, kde mají k testovacím datům společný přístup. [23]

Integrace testovacího rozhraní GoogleTest do nového modulu brzdných křivek probíhá skrze nástroj CMake.

```
include(FetchContent)
FetchContent_Declare(
        googletest
        URL https://github.com/google/googletest/archive/refs/tags/release
    -1.11.0.zip
)
// For Windows: Prevent overriding the parent project's compiler/linker settings
set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
FetchContent_MakeAvailable(googletest)
```

Výpis kódu 6.1 Integrace rozhraní GoogleTest do projektu CMake

6.1.1 Makra

Rozhraní GoogleTest poskytuje široké spektrum maker pro ověření chování kódu. Většina maker se dodává jako pár s variantou EXPECT_ a variantou ASSERT_.

Při selhání generují makra EXPECT_ nefatální selhání a umožňují, aby aktuální funkce pokračovala v chodu, zatímco makra ASSERT_ generují fatální selhání a přeruší aktuální funkci.

V novém modulu brzdných křivek byla použita následující makra:

- EXPECT_EQ
- EXPECT_NEAR
- EXPECT_DOUBLE_EQ
- EXPECT_TRUE
- EXPECT_FALSE
- ASSERT_TRUE
- ASSERT_NEAR

Makra ASSERT_TRUE a ASSERT_NEAR testují stejné vlastnosti jako makra EXPECT_TRUE a EXPECT_NEAR. [24]

6.1.1.0.1 EXPECT_EQ(hodnota1, hodnota2) Kontroluje rovnost parametrů hodnota1 a hodnota2. [24]

6.1.1.0.2 EXPECT_NEAR(hodnota1, hodnota2, abs_chyba) Ověřuje, že rozdíl mezi hodnota1 a hodnota2 nepřekračuje hranici absolutní chyby abs_chyba. [24]

6.1.1.0.3 EXPECT_DOUBLE_EQ(hodnota1, hodnota2) Kontroluje, že dvě desetinná čísla hodnota1 a hodnota2 jsou přibližně stejná (v rámci 4 ULP[1] od sebe). [24]

6.1.1.0.4 EXPECT_TRUE(podmínkap) Ověřuje, že podmínka je pravdivá. [24]

6.1.1.0.5 EXPECT_FALSE(podmínka) Ověřuje, že podmínka je nepravdivá. [24]

6.2 Testy nového modulu brzdných křivek

V nově vzniklém modulu brzdných křivek se nachází testovací sady pro testování jednoudchých tříd a struktur, třídy typu test fixture pro složitější třídy, kde je potřeba většího množství vstupních parametrů sdílených pro více testů, a testy akceptační.

Referenční data jsou uložena buďto přímo v testovacím souboru, nebo, jedná-li se o větší celky dat, ve formátu txt v adresáři testData.

```
TEST(StepTests, GettersTest) {
    Step step(0,100,50);
    EXPECT_EQ(step.getLowerBound(), 0);
    EXPECT_EQ(step.getUpperBound(), 100);
    EXPECT_EQ(step.getValue(), 50);
}

TEST(StepTests, InBoundsTest) {
    Step step(0,100,50);
    EXPECT_TRUE(step.inBounds(0));
    EXPECT_TRUE(step.inBounds(100));
    EXPECT_TRUE(step.inBounds(50));
    EXPECT_FALSE(step.inBounds(-5));
    EXPECT_FALSE(step.inBounds(101));
}

TEST(StepTests, InBoundsTest2) {
    Step step(40,60,0);

    for (int i = 20; i <= 40; i++) {
        EXPECT_FALSE(step.inBounds(i));
    }

    for (int j = 41; j <= 60; j++) {
        EXPECT_TRUE(step.inBounds(j));
    }

    for (int k = 61; k <= 80; k++) {
        EXPECT_FALSE(step.inBounds(k));
    }
}
```

Výpis kódu 6.2 Ukázka testů třídy Step

[1] unit of least precision

```
class SBDTest1 : public ::testing::Test {
protected:
    void SetUp() override {
        expected = make_shared<AExpected>(A_brake_service, A_gradient);
    }
    double initial_speed = 160.0;
    TargetType targetType = TargetType::EOA_SVL;
    double dist_origin_target = 7000;
    StepFunction A_brake_service =
        StepFunction({Step(0, 100, 0.8 ), Step(100, 160, 0.7),
                      Step(160, numeric_limits<double>::max(), 0.9)});
    StepFunction A_gradient =
        StepFunction({Step(0, 1299, 0.0285728155339806),
                      Step(1299, 4999, -0.0285728155339806),
                      Step(4999, numeric_limits<double>::max(), -0.0476921)});
    shared_ptr<AExpected> expected;
    };

TEST_F(SBDTest1, SBDTest1) {
    SBD curve = SBD();
    vector<double> sbd = curve.getSBD(initial_speed,
                                      targetType,
                                      dist_origin_target,
                                      expected);

    ifstream testData("../test/TestData/SBD/sbdValues1");
    int iterator = 0;
    while (!testData.eof()) {
        double testVal;
        testData >> testVal;
        EXPECT_NEAR(sbd[iterator++], testVal, 0.007);
    }
    testData.close();
}
```

Výpis kódu 6.3 Ukázka testů třídy Step

6.2.1 Výsledky testů

Všech 92 testů, kterými je nový modul brzdných křivek vybaven, prochází úspěšně. Průměrný čas provedení testů je 780 milisekund.

Naprostá většina referenčních dat pochází z dokumentu Evropské železniční agentury Braking curves tool v3.0 [15].

6.3 Integrace

Integraci (a s ní spojené integrační testy) do projektu trenažeru ETCS, konkrétně do komponenty EVC, má na starosti tým BI-SP1, jemuž byl zdrojový kód nového modulu brzdných křivek předán 5. 5. 2023, týden před odevzdáním této baklářské práce. Spolu s odevzdáním kódu proběhla informační schůzka s členy týmu SP1, kde jim bylo vysvětleno, jak by mělo začlenění nového modulu brzdných křivek do projektu trenažeru ETCS vypadat. V tuto chvíli integrace stále probíhá.

Kapitola 7
Závěr

Teoretická část této bakálářské práce seznamuje čtenáře se systémem ETCS a vysvětluje výhody jeho integrace do železniční dopravy. Dále objasňuje problematiku výpočtu brzdných křivek vlaku a popisuje dvě metody výpočtu těchto křivek — gamma a lambda. Popsán je i nynější stav modulu, který má kalkulaci brzdných křivek na starosti, a jsou popsány funkční a nefunkční požadavky této práce.

Praktická část pomocí metod softwarového inženýrství představuje návrh nového modulu pro výpočet brzdných křivek. Navržena je struktura kódu, stejně jako nejdůležitější modely a třídy. Tato část se dále věnuje implementaci kódu nového modulu brzdných křivek podle výše zmíněného návrhu. Popsány jsou nejen nejdůležiější části implementace, ale i problémy, které ji provázely. Na závěr této části je objasněno, jakým způsobem je nový modul brzdných křivek testován a zda testy proběhly úspěšně, či nikloiv.

Cíle práce byly splněny. Nový modul brzdných křivek produkuje křivky, které jsou v souladu s referenčními hodnotami. Výpočet je dostatečně rychlý (výpočet proběhne v řádech milisekund), modul je vybaven jednotkovými a akceptačními testy a díky kvalitní strukturalizaci kódu a doplnění o vysvětlující komentáře lze považovat nově vzniklý modul za srozumitelný pro nově příchozí programátory.

V budoucnu je možné nově vzniklý modul brzdných křivek obohatit o problematiku speciálních brzd či kalkulaci brzdných vzdáleností v oblastech se snížneou přilnavostí. Modul je taktéž připraven na rozšíření v případě publikace nových specifikací od Evropské železniční agentury.

Literatura

1. *ERTMS in brief* [online]. 2023. [cit. 2023-02-04]. Dostupné z: https://www.ertms.net/about-ertms/ertms-in-brief/.
2. EUROPEAN UNION AGENCY FOR RAILWAYS. *SUBSET-026-3 System Requirements Specification Chapter 3* [online]. 2016. Ver. 3.6.0 [cit. 2023-01-29]. Dostupné z: https://www.era.europa.eu/era-folder/set-specifications-3-etcs-b3-r2-gsm-r-b1.
3. LUICA, Pamela. *A race towards a EU-wide ERTMS network* [online]. 2021. [cit. 2023-01-25]. Dostupné z: https://www.railwaypro.com/wp/a-race-towards-a-eu-wide-ertms-network/.
4. STANLEY, Peter. *ETCS for engineers*. 1. vydání. Hamburg: Eurail Press, 2011. ISBN 9783777104164.
5. *Subsystems and constituents of the ERTMS* [online]. 2020. [cit. 2023-01-25]. Dostupné z: https://transport.ec.europa.eu/transport-modes/rail/ertms/how-does-it-work/subsystems-and-constituents-ertms_en.
6. NETWORK RAIL. *GSM-R: The railway's mobile communication system* [online]. 2016. [cit. 2023-02-01]. Dostupné z: https://www.networkrail.co.uk/running-the-railway/gsm-r-communicating-on-the-railway/.
7. EUROPEAN UNION AGENCY FOR RAILWAYS. *System Requirements Specification Chapter 2* [online]. 2022. [cit. 2023-04-25]. Dostupné z: https://www.era.europa.eu/era-folder/set-specifications-2-etcs-b3-mr1-gsm-r-b1.
8. EUROPEAN UNION AGENCY FOR RAILWAYS. *System Requirements Specification Chapter 3* [online]. 2022. [cit. 2023-04-25]. Dostupné z: https://www.era.europa.eu/era-folder/set-specifications-3-etcs-b3-r2-gsm-r-b1.
9. EUROPEAN UNION AGENCY FOR RAILWAYS. *INTRODUCTION TO ETCS BRAKING CURVES* [online]. 2020. [cit. 2023-03-19]. Dostupné z: https://www.era.europa.eu/system/files/2022-11/Introduction%20to%20ETCS%20braking%20curves.pdf.
10. LANDEX, Alex; JENSEN, Lars. *Infrastructure Capacity in the ERTMS Signaling System* [online]. 2019. [cit. 2023-03-20]. Dostupné z: https://ep.liu.se/ecp/069/040/ecp19069040.pdf.
11. EUROPEAN UNION AGENCY FOR RAILWAYS. *SUBSET-026-7 System Requirements Specification Chapter 7* [online]. 2016. Ver. 3.6.0 [cit. 2023-04-26]. Dostupné z: https://www.era.europa.eu/era-folder/set-specifications-3-etcs-b3-r2-gsm-r-b1.
12. EUROPEAN UNION AGENCY FOR RAILWAYS. *SUBSET-041 Performance Requirements for Interoperability* [online]. 2015. [cit. 2023-01-29]. Dostupné z: https://www.era.europa.eu/era-folder/set-specifications-3-etcs-b3-r2-gsm-r-b1.

13. HRUBAN, Ivo; NACHTIGALL, Petr; ŠTĚPÁN, Ondřej. *Přínosy zavedení ETCS z pohledu brzdných křivek* [online]. 2015. [cit. 2023-03-15]. Dostupné z: https://docplayer.cz/106609225-Vedeckotechnicky-sbornik-cd-c-40-2015.html.

14. EUROPEAN UNION AGENCY FOR RAILWAYS. *Braking curves simulation tool* [online]. 2018. Ver. 4.2 [cit. 2023-01-29]. Dostupné z: https://www.era.europa.eu/domains/infrastructure/european-rail-traffic-management-system-ertms_en.

15. EUROPEAN UNION AGENCY FOR RAILWAYS. *Braking curves simulation tool* [online]. 2009. Ver. 3.0 [cit. 2023-05-10]. Dostupné z: https://uloz.to/file/mGffpRohzIlJ/era-braking-curves-tool-v3-0-unlocked-xlsm.

16. MICROSOFT CORPORATION. *Microsoft VLOOKUP function* [online]. 2014. [cit. 2023-05-05]. Dostupné z: https://support.microsoft.com/en-us/office/vlookup-function-0bbc8083-26fe-4963-8ab8-93a18ad188a1.

17. LEHOCKÝ, Zdeněk. *C - 6. Lekce* [online]. 2005. [cit. 2023-05-10]. Dostupné z: http://programujte.com/clanek/1970010146-c-6-lekce/.

18. *STD::Vector* [online]. 2023. [cit. 2023-05-10]. Dostupné z: https://en.cppreference.com/w/cpp/container/vector.

19. *STD::Variant* [online]. 2023. [cit. 2023-05-10]. Dostupné z: https://en.cppreference.com/w/cpp/utility/variant.

20. JAVATPOINT. *Upcasting and downcasting in C++ - javatpoint* [online]. 2011. [cit. 2023-05-05]. Dostupné z: https://www.javatpoint.com/upcasting-and-downcasting-in-cpp.

21. GREAT LEARNING BLOG. *Polymorphism in C++ and types of polymorphism* [online]. 2022. [cit. 2023-05-07]. Dostupné z: https://www.mygreatlearning.com/blog/polymorphism-in-cpp/.

22. WHITNEY, Tyler. *Why floating-point numbers may lose precision* [online]. 2021. [cit. 2023-05-07]. Dostupné z: https://learn.microsoft.com/en-us/cpp/build/why-floating-point-numbers-may-lose-precision?view=msvc-170.

23. GOOGLETEST. *GoogleTest Primer* [online]. 2021. [cit. 2023-05-08]. Dostupné z: http://google.github.io/googletest/primer.html.

24. GOOGLETEST. *GoogleTest Assertions Reference* [online]. 2021. [cit. 2023-05-08]. Dostupné z: http://google.github.io/googletest/reference/assertions.html#success-failure.

Obsah přiloženého archivu

```
readme.txt ..... stručný popis obsahu média
src
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    thesis.pdf ..... text práce ve formátu PDF
```