Міністерство освіти і науки України

Український державний університет науки і технологій

Кафедра «Автоматика та телекомунікації»

ДОВІДКА

про відсутність плагіату у випускній кваліфікаційній роботі

За результатами перевірки випускної кваліфікаційної роботи (ВКР) здобувача вищої освіти освітнього ступеня (ОС) «магістр»

Семенова Михайла Володимировича

(прізвище, ім'я, по батькові)

на тему: Моделювання раціональних тягових розрахунків за двома показниками

в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР В. Лагута (підпис) Лагута Василь Васильович (прізвище, ім'я, по батькові)

20.12.2021р

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський державний університет науки і технологій

Кафедра «Автоматика та телекомунікації»

«ДО ЗАХИСТУ»
Завідувач кафедри

_____ Гаврилюк В.І.

«_16_» _____12_____ 2021 р.

МАГІСТЕРСЬКА РОБОТА

на здобуття ОКР «магістр»

Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»

Тема: «МОДЕЛЮВАННЯ ТЯГОВИХ РОЗРАХУНКІВ ЗА ДВОМА ПОКАЗНИКАМИ »

Theme: «Modeling rational traction calculations based on two criteria»

Керівник дипломної роботи доцент _____ Лагута В.В.

Виконавець, студент групи 977-М1 _____ Семенов М.В.

Student Semenov Michailo

(family name)

Дніпро

2021

Дніпровський державний університет науки і технологій

Кафедра «Автоматика та телекомунікації»
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»

«ЗАТВЕРДЖУЮ»
Завідувач кафедри

_____ Гаврилюк В.І.
(підпис) (ПІБ)

« 22 » жовтня 2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеню «магістр»
Семенов Михайло Володимирович
(Прізвище та ім'я по батькові)

Тема роботи Моделювання раціональних тягових розрахунків за двома показниками

Затверджена наказом по університету № 630-ст від « 19 » жовтень 2020 р.

2. Термін подання студентом закінченої роботи «15» грудень 2021 р.

3. Вихідні дані до роботи План і профіль залізничної дільниці, тягові характеристики, параметри поїзда

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Обсяг %	Кількість креслень
ВСТУП	5	1
1. ОГЛЯД РОБІТ ПО ОПТИМІЗАЦІЇ ТЯГОВИХ РОЗРАХУНКІВ	10	1
2. МОДЕЛЮВАННЯ РУХУ ПОТЯГУ	15	2
3. ЗАДАЧА БАГАТОКРИТЕРІАЛЬНОЇ ОПТИМІЗАЦІЇ. ПРИНЦИП ПАРЕТО	10	2
4. ОПТИМАЛЬНІ ТЯГОВІ РОЗРАХУНКИ НА МНОЖЕННІ ПАРЕТО	15	2
5. ВИЗНАЧЕННЯ РЕЖИМІВ РУХУ ЕЛЕКТРОВОЗА ДС – 3	40	2
6. АЛГОРИТМ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ ЕФЕКТИВНИХ ТЯГОВИХ РОЗРАХУНКІВ	5	1

Студент Семенов М.В.

Науковий керівник .Лагута В.В.

РЕФЕРАТ

Об’єм пояснювальної записки: 95 сторінок, 3 таблиці, 13 рисунків, 38 джерел літератури.

Ключові слова: тягові розрахунки, багатокритеріальна оптимізація, принцип Парето, моделювання руху поїзду, динамічне програмування, ефективні траєкторії руху поїзда.

Завданням даної магістерської роботи було автоматизація розрахунків визначення ефективних тягових траєкторій руху поїзда.

В першому розділі розглядалися роботи по оптимізації тягових розрахунків.

Застосування методів векторної оптимізації до вирішення двокритеріальної задачі оптимізації тягових розрахунків

В другому розділі розглядалось моделювання руху потягу.

При імітаційному моделюванні рівняння руху визначається блоком вибору машиністом режиму руху і вирішується для кожного поїзда окремо з урахуванням координат і швидкостей попереду і ззаду наступних поїздів.

В третьому розділі розглядалися задача багатокритеріальної оптимізації та принципу Парето.

До цих пір ми розглядали завдання оптимізації, де ясний критерій (показник ефективності) по якому проводиться оцінка ефективності проектованого об'єкта, тобто потрібно звернути в min (max) один єдиний показник.

В четвертому розділі розглядались оптимальні тягові розрахунки на множенні Парето.

В п’ятому розділі визначалися ефективні траєкторії руху поїзда та відповідні режими руху електровоза ДС – 3.

В шостому розділі розроблявся алгоритм та математичне забезпечення задачі ефективних тягових розрахунків.

ЗМІСТ

ВСТУП

Тягові розрахунки є прикладної частиною теорії тяги поїздів і дозволяють вирішувати численні практичні задачі, що виникають при проектуванні та експлуатації залізниць. До числа найважливіших задач відносяться:

- визначення маси вантажних складів при заданому типі локомотива відповідно до профілю, швидкісті руху і часу ходу по ділянках і окремим перегонах;
- визначення необхідних параметрів локомотива для забезпечення заданої пропускної і провізної здатності ділянки;
- складання графіку руху поїздів – основного документа роботи залізничного транспорту;
- вибір найбільш раціонального розміщення станцій, зупиночних і роздільних пунктів при проектуванні залізниць;
- визначення параметрів системи енергопостачання при електрифікації залізниці: розміщення тягових підстанцій і визначення їх потужності, розрахунок тягової мережі та інше.

На залізничному транспорті методи розробки тягових розрахунків і необхідні для їх виконання нормативи регламентуються Правилами тягових розрахунків (ПТР) для поїзної роботи [1-6].

В даний час тягові розрахунки виконуються, переважно, з допомогою засобів обчислювальної техніки. Однак для математичного формулювання задачі необхідно враховувати фізичну суть явищ, що супроводжують процес руху поїзда, основні прийоми та способи тягових розрахунків. У більшості випадків тягові розрахунки вимагають оперативності їх проведення.

Розробка програмного забезпечення тягових розрахунків, що враховує інтереси всіх служб залізничного транспорту, навряд чи доцільно і можливо в зв'язку з обробкою значної кількості інформації і труднощів в розстановці пріоритетів. Проте розробка методів оптимальних тягових розрахунків з точки зору теорії оптимального управління дозволить оцінити наявні

можливості та розробити перспективні напрямки впровадження оптимальних тягових розрахунків. Сьогодні, мабуть, найактуальнішою проблемою є проблема економії енергоресурсів. У той же час необхідно вантаж доставляти під час, а в багатьох випадках – в найкоротші терміни. У цьому й полягає особливість у виборі методів розв'язання задачі.

Актуальність

Одним з радикальних способів, що забезпечує стійкість на ринку надання транспортних послуг, є економія енергоресурсів. Залізнична мережа України органічно зливається з залізницями Росії, Білорусії, Польщі, Чехословаччини, Румунії та ін. Вигідне розташування України має великий потенціал до збільшення транзитних перевезень. Незважаючи на істотне зниження обсягів перевезень, умови роботи залізничних підприємств України залишаються важкими і це в першу чергу пов'язано із щорічним зростанням цін на енергоносії. Значна частка використання енергії припадає на забезпечення тяги поїздів. Проблемі економії енергоресурсів приділяється постійна і пильна увага у всіх галузях промисловості і не тільки на транспорті.

В рішенні Комісії Організації співробітництва залізниць (ОСЖД) щодо інфраструктури і рухомого складу, який набрав чинності 30 жовтня 2003р. надано рекомендації, спрямовані на реалізацію енергетичної політики залізничного транспорту країн-членів організації ОСЖД і підвищення ефективності використання енергії в процесі перевезень на електрифікованих лініях [7].

Розрахунок оптимальних дільничних швидкостей ведення поїзда згідно з правилами тягових розрахунків для поїзної роботи, а також вибір відповідних режимів з управління тяговими електродвигунами та побудова режимних карт (РК), є відомими базовими задачами для реалізації енергозберігаючої технології процесу перевезень. Оскільки при оптимізації за критерієм мінімуму електроспоживання витрата електроенергії не зв'язується ні з періодами доби, ні з місцем електроспоживання, в даний час

використання цих методик має обмежене застосування. Зокрема, такі режими можуть бути рекомендовані, якщо в умовах ОРЕ (оптовий ринок електроенергії) зміна ціни електроенергії за час руху можна не враховувати, а при використанні ДТ (диференційований тариф оплати електричної енергії) рух поїзда відбувається в одній тарифної зони [7].

Сьогодні важливою проблемою є створення компромісно-оптимальних режимів тяги поїздів для показників споживання та часу доставки вантажу. Вартісні показники ефективності руху поїздів вимагають нових підходів до розробки методів оптимального розрахунку режимних карт ведення поїздів. Для аналізу доцільності переходу на режими руху, оптимальні за вартістю електроенергії, необхідно виконати дослідження компромісно-оптимальних рішень, ефективних для вектора показників:

- витрати електроенергії;
- вартості електроенергії при заданому обсязі перевезень;
- графік руху.

Компромісно-оптимальні режими представляють набір умовно-оптимальних режимів руху (або ж дільничних швидкостей), які застосовуються залежно від заданої переваги характеристик векторної цільової функції [7].

У відповідності із Концепцією науково-технічного розвитку рухомого складу залізниць України на 2020–2021 роки основним напрямком науково-технічного розвитку локомотивного господарства на перспективу є

- зменшення вартості тягового рухомого складу (ТРС), що закупляється;
- скорочення витрат палива і електроенергії на тягу;
- підвищення продуктивності локомотива за рахунок високої ефективності використання ТРС;
- передбачення суттєвого зменшення вартості життєвого циклу кожної одиниці ТРС.

Мета дослідження.

Удосконалення програмного забезпечення ефективних тягових розрахунків на основі проведення обчислювальних експериментів.

Методи дослідження.

Методи дослідження вибиралися виходячи з постановок вирішуваних задач з урахуванням особливостей досліджуваних об'єктів.

Теоретичною базою дослідження є праці вітчизняних і зарубіжних учених у галузі тяги поїздів, системного аналізу, чисельних методів рішення диференціальних рівнянь, теорії оптимального управління, математичної статистики, обчислювальної математики та програмування. Для дослідження оптимальних режимів і алгоритмів керування локомотивом використані методи імітаційного моделювання на ЕОМ та експериментальні дослідження в умовах експлуатації.

Для досягнення мети були поставлені і вирішені наступні задачі:

- аналіз математичних методів оптимізації режимів ведення поїзда по ділянці залізничної мережі;
- побудова математичної моделі руху поїзда, адаптованої для вирішення тягово-оптимізаційних задач;
- розробка алгоритмічного та програмного забезпечення з розрахунку і видачі індивідуальних режимних карт ведення поїзда з урахуванням мінімізації витрат енергії та часу.

РОЗДІЛ 1. ОГЛЯД РОБІТ ПО ОПТИМІЗАЦІЇ ТЯГОВИХ РОЗРАХУНКІВ

1.1 Аналіз пристосування схем і методів до обчислювальної техніки

Характерною особливістю робіт за оптимальними тяговим розрахунками є пристосування схем і методів до обчислювальної техніки [8-10]. У роботі [10] проводиться деяка намітка на оптимізацію з використанням принципу Беллмана [11-12]. Як критерій оптимальності взято експлуатаційні витрати. Запропонований алгоритм відшукання оптимальної траєкторії містить наступні елементи

- для заданої початкової швидкості виробляються тягові розрахунки при різних режимах на першому кроці;

- для отриманих швидкостей робляться тягові розрахунки при різних режимах на другому кроці. Якщо нові отримані швидкості виявляться «близькими» (різниця за абсолютною величиною $< \varepsilon$), то проводиться порівняння відповідних варіантів по накопичених за два кроки експлуатаційних витрат і залишаються тільки ті незрівнянні траєкторії, у яких експлуатаційні витрати менше;

- траєкторії що залишилися продовжують і знову робится порівняння по експлуатаційним витратам;

- за цією схемою ведеться розрахунок і далі, поки на останньому кроці не залишиться варіант з необхідною кінцевою швидкістю і найменшими експлуатаційними витратами.

Щоб скоротити час рахунку, автори пропонують використовувати послідовні наближення в просторі кривих швидкостей $\{v(s)\}$, тобто на першому етапі ε береться значним, а потім ε зменшується, одночасно зменшується і інтервал можливої варіації швидкості.

Крім наведених вище міркувань авторами роботи [10] не вказується ані конкретна реалізація даного алгоритму, ні справедливість даної процедури

стосовно вибраного критерію оптимальності, ні збіжність процесу послідовних наближень у просторі швидкостей.

Розвитком роботи [10] є роботи [13-15]. За критерій оптимальності взято наведені витрати, що визначаються за формулою

$$\acute{y} = \tilde{n}_t t + c_A A \qquad (1.1)$$

де

t - час ходу по ділянці в годинах;

А - витрати енергії на пересування потягу в кВт.год;

c_A - витрати в грошовому еквіваленті, пропорційні 1 кВт.год отриманої енергії по шинам тягової підстанції, що включають вартість електричної енергії та частину ремонтних витрат, які залежать від механічної роботи і її складових;

c_t - витрати в грошовому еквіваленті, пропорційні 1 поїздо-часу, що включають витрати з утримання рухомого складу в тій частині, яка залежить від часу.

Якщо дивитися на праву частину (1.1) як на лінійні члени розкладання наведених витрат в ряд Тейлора, то це співвідношення в якійсь мірі можна визнати доцільним.

Як і в роботі [10], автором [14-15] в основу отримання алгоритму покладено принцип Беллмана. До достоїнств робіт [14, 15] можна віднести той факт, що зазначений алгоритм був реалізований у вигляді конкретного програмного забезпечення і були проведені конкретні експериментальні розрахунки, які дозволили накопичити певний досвід за оптимальними тяговим розрахунками.

Стосовно до критерію оптимальності у формі (1.1) було встановлено, що прийнятною, з практичної точки зору, сітка швидкості повинна бути близько 1 км / год, а по колії - 500м.

Оскільки зазначені вище роботи були одними з перших за своєю шириною поставлених задач і глибині розробок, то є й ряд недоліків з боку

оптимальних тягових розрахунків (нечіткість постановки задачі, відсутність адекватної математичної моделі).

Недотримання основних положень тягне за собою деяку некоректність у постановці звичайних класичних варіаційних задач. Наприклад в [14, 15] вказується, що «за розробленим алгоритмом складена і налагоджена програма побудови кривих руху потягу ..., що забезпечує вибір оптимального режиму ведення поїзда, виходячи з критерію мінімуму наведених витрат ... »і далі - « програма дозволяє встановлювати оптимальні часи ходу, витрати електроенергії, вартість поїздки і т.д » - це повний абсурд, маючи на увазі оптимізаційну постановку задачі з однією функцією мети. З цього випливає, що розроблена програма здатна визначати (будувати) криві швидкості, на яких реалізується (задовольняється) кілька функцій цілі (критеріїв). Це непорозуміння можна віднести за рахунок некоректного поводження з термінологією і поняттями математичної теорії оптимальних управлінь.

Дуже важливим, в сенсі додатків, є клас задач, що звуться ізоперіметричними (коли час ходу поїзда від одного пункту до іншого має бути суворо одного певного значення).

Взагалі при вирішенні таких об'ємних задач зі складною логікою, як задачі тягових розрахунків, велике значення має метод програмування, що визначає ту чи іншу гнучкість програми при вирішенні конкретних практичних задач.

1.2 Аналіз основних методів по оптимізації тягових розрахунків

У роботах [13, 16] будуються оптимальні криві або просто криві швидкості руху поїзда. У роботі [13] зроблено облік падіння напруги в контактному проводі. Важливість подібних робіт очевидна.

Одним з напрямів впровадження методів кібернетики на транспорті, є розробка таких обчислювальних систем, які дозволяли б оптимально управляти поїздом, як в замкнутому циклі (автоматичне керування), так і в режимі рекомендацій (підказок). Практика показала, що для більшої

ефективності необхідно розробити досить дієві математичні методи розв'язання задач оптимальних тягових розрахунків, на підставі яких можна було б розробити ті чи інші алгоритми для конкретних інженерних задач, з наступним уточненням і доробкою на реальних процесах керування поїздами або в системах управління. Дана проблема висвітлена в роботах [17-20].

Робота [21] присвячена оптимального вибору режиму руху потягу на основі принципу максимуму Понтрягіна та динамічного програмування, розглянуті задачі визначення режимів ведення поїзда оптимальні за швидкодією і мінімальної витрати енергії. На підставі якісних досліджень поведінки інтегральних кривих розроблено метод побудови області допустимих по температурі перегріву локомотива. Розроблено метод змінної сітки для більш ефективного використання динамічного програмування. Дано узагальнення ізоперіметрическої задачі на випадок оптимального розподілу часу ходу по перегону.

Подальший розвиток проблема тягових розрахунків отримала в роботах [22-28]. У даних роботах використовується як класичне варіаційне числення, так і методи математичної теорії оптимального управління, що з'явилися у фундаментальних працях Понтрягіна і Белмана. Проблема оптимальних режимів управління локомотивом розглядається як інженерна завдання. В основу методів рішення покладений в більшості випадків принцип максимуму.

У [29] викладено теорія тяги поїздів та тягові розрахунки, розглянуто сили тяги, опору і гальмування потягів, розрахунок руху поїздів, енергетика тяги, експлуатаційні випробування, методи тягових розрахунків на ЕОМ великовагових складів. Всі задачі розглянуто з точки зору керованого руху.

Роботи [30-33] присвячені, в основному, розробці методів оптимізації режимів водіння поїздів, заснованих на використанні сучасної математичної теорії управління і обчислювальних засобів. Критерієм оптимальності в більшості виконаних робіт служить мінімум витрат енергії на тягу поїздів, хоча зустрічається також застосування інших показників ефективності

організації перевізного процесу, наприклад час руху поїзда по ділянці, що використовується в задачах на швидкодію або точність виконання заданого часу ходу і т.п. Проте незалежно від прийнятого критерію і параметра оптимізації задача вибору оптимальних режимів водіння поїзда розглядалися в однокрітеріальній постановці.

Застосування методів векторної оптимізації до вирішення двухкрітеріальної задачі оптимізації тягових розрахунків викладені в роботах [34-36]. У роботах досліджується рішення двухкрітеріальних задач методом векторної оптимізації. Для аналізу можливих шляхів вирішення використовується метод параметризації, проводиться аналіз завдання тягових розрахунків як завдання векторної оптимізації. Запропонований метод оптимізації грунтується на якісному дослідженні режимів руху на елементарному відрізку колії. До недоліків можна віднести відсутність чисельних методів векторної оптимізації орієнтованих до використання обчислювальної техніки що реалізують даний метод.

Ідея переходу на енергооптімальний графік руху не нова - вчені працюють у цьому напрямку більш десяти років. Тепер з'явилися технічні можливості для впровадження розробки - системи автоведення, якими обладнано близько 1,3 тис. пасажирських локомотивів, що дозволяє використовувати спеціальну програму «Енергооптімальний тяговий розрахунок», створену в ВНІІЖТі.

Програма дозволяє розрахувати оптимальний режим руху поїзда з урахуванням безлічі критеріїв: профілю колії, обмеження швидкості руху, місць випробування гальм, складовою, характеристик локомотива і так далі. При виникненні будь-яких змін в дорозі, приміром, непередбаченої зупинки, відбувається автоматичний перерахунок швидкості руху, а система автоведенія з максимальною точністю проведе поїзд по ділянці в відповідно до нових параметрів. Дослідна експлуатація показала, що економія електроенергії може досягати 5-12%. Причому нововведення жодним чином не відбилося на щільному приміському русі двох столиць, що було однією з

умов. Поступово під новий графік після відповідного навчання машиністів будуть підлаштовуватися поїзда, не обладнані системами автоведенія.

Надалі енергооптімальний графік буде перенесений і на вантажний рух. З технічної точки зору це набагато складніше, тому що тут немає чіткої прив'язки до розкладу. Але зусилля варті того - економія очікується набагато більше.

Що ж стосується впровадження енергооптімального графіка у вантажному повідомленні, то тут, на думку багатьох вчених, можуть виникнути серйозні складності. На мережі чимало ділянок з обмеженням швидкісного режиму, які можуть стати перешкодою для впровадження технології. Для зняття обмежень будуть потрібні серйозні фінансові та витрати часу.

Активно ведуться дослідження з проблеми економії електроенергії на тягу пасажирських поїздів і в Європі. Так у Нідерландах залізниці Нідерландів (NS) уклали з міністерством економіки довгострокову угоду про ефективне використання енергоресурсів. Відповідно до цього NS взяли на себе зобов'язання підвищити на 11% ефективність використання електроенергії. Угода торкнулося наступних п'яти підрозділів залізниць:

- NS Reizigers (пасажирські перевезення);
- NS RailInfraBeheer (інфраструктура);
- NS Stations (станції);
- NS Vastgoed (будинки та споруди);
- NedTrain, раніше NS Materieel (рухомий склад).

Вантажні перевезення в Нідерландах виконує компанія Railion, що не входить до складу NS. На тягу поїздів вона щорічно споживає близько 0,1 ТВТ · год електроенергії і 16 млн. л дизельного палива.

Кожне з підрозділів NS представило план економії енергії (ЕВР), який визначав:

- норми споживання і баланс енергії по всіх процесів;
- коефіцієнт підвищення ефективності енергоспоживання;

- коло відповідальності за економію енергії в підрозділі;
- резерви економії енергії.

Таблиця 1.1 - План споживання та економії енергії на NS

Підрозділ	Споживання первинної енергії в 1997 р.			Економія в період 1997 - 2010 рр..,%
	ТДж	ТВТ · год	%	
NS Reizigers	9 104	2,53	86	10
NS RailInfraBeheer	7 30	0,2	7	8
NS Stations	2 18	0,06	2	13
NS Vastgoed	3 75	0,11	3	25
NedTrain	1 91	0,05	2	8
І т о г о:	1 0 618	2,95	100	11

У табл. 1.1 подано контрольні цифри плану економії енергії. Середня для всіх підрозділів NS величина економії склала 11%. Найкращі результати прогнозуються для підрозділу NS Reizigers (NSR), у якого при показнику економії, близькому до середнього, споживання енергії найбільша (86%) [38].

Висновок

В даному розділі було розглянуто алгоритми визначення оптимальних тягових траєкторій і містить наступні елементи. Проаналізовано методи оптимізації режимів водіння поїздів, заснованих на використанні сучасної математичної теорії управління і обчислювальних засобів. Критерієм оптимальності в більшості виконаних робіт служить мінімум витрат енергії на тягу поїздів, хоча зустрічається також застосування інших показників

ефективності організації перевізного процесу. З аналізу джерел витікає, що сьогодні активно ведуться дослідження з проблеми економії електроенергії на тягу поїздів.

РОЗДІЛ 2. МОДЕЛЮВАННЯ РУХУ ПОТЯГУ

2.1 Моделювання процесу руху

При імітаційному моделюванні рівняння руху визначається блоком вибору машиністом режиму руху і вирішується для кожного поїзда окремо з урахуванням координат і швидкостей попереду і ззаду наступних поїздів. У загальному вигляді вирішуване рівняння виглядає наступним чином:

$$\frac{dV}{dt} = (f_k - f_t - \omega_0 - \omega_B \pm i_k)\varepsilon, \qquad (2.1)$$

де ($f_к$ - f_t - w_0 - $w_в$ + $i_к$) – питомі сили, що діють на поїзд при його русі (при цьому деякі параметри можуть мати нульове значення в залежності від умов руху); $f_к$ - питома сила тяги поїзда; $f_т$ - питома сила гальмування ; w0 - питомий основний опір руху; ± $i_к$ - додатковий опір руху від ухилу і кривизни шляху; $w_в$ - питома сила опору від повітряного середовища; ε - коефіцієнт прискорення поїзда.

Уявімо рівняння (1.1) у вигляді, зручному для інтегрування:

$$dt = \frac{dV}{\varepsilon(f_k - \omega_k)} = \frac{dV}{f(V)},$$

проінтегрувавши його, отримаємо:

$$\int_{t_n}^{t_{n+1}} dt = \frac{1}{\varepsilon}\int_{V_n}^{V_{n+1}} \frac{dV}{f(V)}, \qquad \text{або} \qquad t_{n+1} - t_n = \frac{1}{\varepsilon}\int_{V_n}^{V_{n+1}} \frac{dV}{(f_k - \omega_k)}, \qquad (2.2)$$

де $w_к = f_т + w_0 + w_в + i_к$.

Помноживши ліву і праву частину рівняння (2.2) на швидкість, визначену в будь-якій точці шляху (V = f (S)), отримаємо:

$$\mathrm{Vdt} = \mathrm{VS} = \frac{\mathrm{VdV}}{\varepsilon(\mathrm{f_k} - \omega_\mathrm{k})} = \frac{\mathrm{VdV}}{\mathrm{f(V)}}.$$

Проінтегрувавши цей вираз, отримаємо:

$$\int_{S_n}^{S_{n+1}} dS = \frac{1}{\varepsilon}\int_{V_n}^{V_{n+1}} \left(\frac{VdV}{f(V)}\right), \qquad \text{або} \qquad S_{n+1} - S_n = \frac{1}{\varepsilon}\int_{V_n}^{V_{n+1}} \left(\frac{VdV}{f_k - \omega_k}\right).$$

Позначивши $V = \frac{dV}{f(t)} = f(t)$, маємо $dS = f(t)dt$.

Так як $dt = \frac{dV}{f(V)}$, можна записати:

$$\int_{S_n}^{S_{n+1}} dS = S_{n+1} - S_n = \int_{V_n}^{V_{n+1}} V dt = \int_{V_n}^{V_{n+1}} f(t) dt, \qquad (2.3)$$

Якщо прийняти, що на інтервалі швидкості (V1 - V2) сила, що діє на поїзд ($f_к$ - $w_к$), постійна, то з рівняння (2.1) отримаємо:

$$t_2 - t_1 = \frac{1}{\varepsilon(f_k)} \int_{V_1}^{V_2} dV,$$

звідки

$$\Delta t = (t_1 - t_2) = \frac{V_2 - V_1}{120(f_k - \omega_k)} \ (год), \quad \text{або} \ \Delta t = \frac{V_2 - V_1}{2(f_k - \omega_k)1{,}2} \ (хв), \qquad (2.4)$$

або

$\frac{30(V_2 - V_1)}{(f_k - \omega_k)1{,}2}$ (с).

З (2.3) при постійній силі ($f_к$ - $w_к$) отримаємо:

$$\Delta S = S_2 - S_1 = \frac{1}{120(f_k - \omega_k)1{,}2} \int_{V_1}^{V_2} V dV = \frac{(V_2^2 - V_1^2)}{2 \times 12(f_k - \omega_k)1{,}2} \qquad (км)$$

або

$$\frac{4{,}17(V_2^2 - V_1^2)}{(f_k - \omega_k)1{,}2} \qquad (м).$$

Залежність t = f (S) виходить шляхом виключення V із знайдених залежностей V = f (t) і V = f (S). З (2.4) маємо:

$$V_2 - V_1 = 120(t_2 - t_1)(f_k - \omega_k)1{,}2, \qquad (2.5)$$

Підставляючи цей вираз в (2.5), знаходимо:

$$(S_2 - S_1)[км] = 500(V_1 + V_2)(t_2 + t_1)(год,)$$

або $(S_2 - S_1)[м] = \frac{(V_1 + V_2)(t_1 + t_1)}{7{,}2}$,

де t вимірюється в [с], V - [км / год].

З вищенаведених співвідношень отримуємо систему рівнянь виду:

$$\begin{cases} V_{n+1} = V_n + \frac{1}{30} \Delta t (f_k - \omega_k); \\ S_{n+1} = S_n + \Delta t \frac{V_n}{3{,}6}; \\ t_{n+1} = t_n + \Delta t. \end{cases}$$

При моделюванні руху на ділянці враховується довжина поїзда при переході з одного ухилу на інший. Якщо передня частина поїзда вагою (P + Q_1) і довжиною L_1 вступила на ухил i_2, а друга його частина вагою (Q - Q_1) знаходиться ще на ухилі i_1, то питомий опір руху поїзда від ухилу визначається з виразу

$$\frac{(P+Q_1)i_2+(Q+Q_1)}{P+Q}=i_1+\frac{P+Q_1}{P+Q}(i_2-i_1),$$

де P – вага локомотива; Q – вага складу з вантажем або з пасажирами.

Крім цього, проводиться перевірка складу на рушання з місця. Для розрахунку енергетично оптимальної траєкторії руху поїздів, в моделі проводиться розрахунок витрат електроенергії з урахуванням рекуперації, який здійснюється за формулою

$$A=A_d-A_{рек,}$$

де A_d – витрата енергії на рух поїзда, Вт× год; $A_{рек}$ - повернення енергії в контактну мережу на ділянках рекуперативного гальмування;

$$A_d=\frac{U}{60}\sum\left(I_{ср}\times\Delta t\right);\quad A_{рек}=\frac{U}{60}\sum\left(I_{ср}\times\Delta t\right),$$

де S ($I_{ср}$ Δt) - ампер-хвилини, відповідно, споживані з контактної мережі або повертаються в неї; U - напруга контактної мережі.

2.2 Рівняння руху потягу

Так як основною цільовою функцією системи є управління рухом поїздів, то розробка імітаційних моделей окремих елементів технологічного комплексу починається з розгляду питань моделювання процесів руху (моделі класу М).

Для управління процесом руху поїздів вже розроблені різні варіанти побудови математичних моделей руху на ділянках залізниць. Застосувавши математичну схему, основні результати досліджень у розглянутій області можна узагальнити наступними основними положеннями.

1. При моделюванні процесів руху на ділянці довжиною L в якості обслуговуючого приладу розглядається ця ділянка. У загальному випадку він

характеризується набором параметрів, істотних для розрахунків часу його заняття і звільнення. До таких параметрів відносяться характеристики плану і профілю ділянки, величина постійних і тимчасових обмежень швидкості та ін

2. При моделюванні процесів руху в якості заявок розглядаються поїзда, що характеризуються множинами статистичних $СЗ_i$ і динамічних $dЗ_i$ параметрів. Безліч $СЗ_i$ в загальному випадку включає такі параметри, як сила тяги F_i і гальмування $f_{ті}$ складу, вага складу P_i та інші характеристики рухомого складу.

В якості основних динамічних параметрів $dЗ_i$ приймаються швидкість V_i і час руху поїзда Δt_i по дискретним відрізкам шляху (дискрет-дільницям) довжиною ΔI. Обслуговуючий прилад ΔI розглядається у вигляді багатоканального пристрою з динамічним числом каналів обслуговування $У_i$, яке визначається виразом

$$Y_1 = \left|\frac{\Delta I}{l_{ni}}\right|,$$

де l_{ni} – довжина і-го потягу.

У процесі руху поїзда по дискрет-ділянкам Δl (рис.2.1) складається наступна технологічна ситуація по часу: (t_1, t_3, t_2 і t_4), де

t_1 позначає час заняття поїздом даної ділянки Δi_i; t_2 - звільнення попереднього Δi_{i-1}; t_3 - заняття наступного Δi_{i+1}; t_4 - звільнення зайнятої ділянки Δi_i.

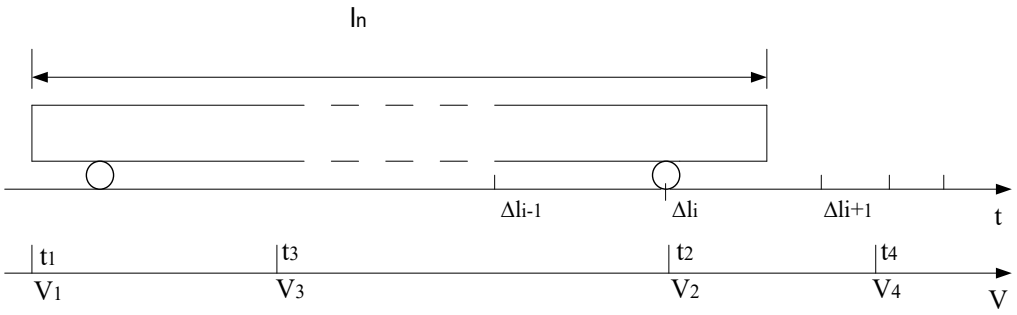


Рисунок 2.1 - Схема просування складу моделі

Значення швидкостей руху поїзда в моменти часу t_1, t_3, t_2, t_4 співвідносяться залежно від режиму руху таким чином:

($V_1 = V_3 = V_2 = V_4$) - при рівномірному русі поїзда;

($V_1 < V_3 < V_2 < V_4$) - в режимі прискорення руху поїзда;

($V_1 > V_3 > V_2 > V_4$) - в режимі гальмування поїзда.

Запропонований підхід дозволяє звести моделювання безперервного процесу руху до розрахунку значень швидкостей в заданих точках шляху, де ці значення істотні для моделювання роботи всього технологічного комплексу.

3. Функція обслуговування заявки $З_i$ по динамічному параметру f_i (визначення швидкостей V_1 - V_4) розраховується по заданому рівнянню руху V = f (S) в двох можливих режимах - без переривання (рівняння в межах ділянки не змінюється) і з перериванням (рівняння руху змінюється).

4. Функція обслуговування заявки $З_i$ за часом f_i визначається за обчисленими значеннями швидкостей V_1 - V_4 з урахуванням прийнятої

апроксимації рівняння руху в межах ділянки залізниці L.

При оцінці точності апроксимації встановлено, що у всіх випадках

достатня точність моделювання досягається при апроксимації по середній швидкості руху з розрахунком часу заняття обслуговуючого приладу π_i довжиною Δ_l по наступним рівнянням:

$$V_1 = \frac{V_1+V_3}{2}; \qquad \tau_1 = \frac{\Delta l}{V_1} = \frac{2\Delta l}{V_1+V_3};$$

$$V_2 = \frac{V_2+V_4}{2}; \qquad \tau_2 = \frac{l_{ni}}{V_2} = \frac{2l_{ni}}{V_2+V_4};$$

$$\xi = \xi_1 + \xi_2,$$

де V_1 - середня швидкість руху першої осі; t_1 – час руху першої осі; V_2 - середня швидкість руху останньої осі; t_2 – час руху останньої осі; t – час заняття обслуговуючого приладу рі довжиною Δl.

У разі переривання процесу обслуговування заявки під час руху першої осі (момент τ', $t_1 \le \tau' \le t_3$) час заняття

$$\tau' = \frac{2\Delta l}{V_1 + V'_3} + \frac{2l_{ni}}{V'_2 + V'_4}.$$

При перериванні процесу обслуговування заявки під час виходу останньої осі (момент τ", $t_2 \leq t" \leq t_4$) час заняття Δl буде дорівнювати

$$\tau" = \tau_1 + \tau"_2 = \frac{2\Delta l}{V_1+V_3} + \frac{2l_{ni}}{V_2+V'_4}.$$

Наведена математична модель забезпечує досить малий час рахунку, але точність обчислень визначається прийнятою довжиною обслуговуючого приладу Δl і адекватністю залежностей V = f (S) реальному процесу руху поїздів, які апроксимуються. При цьому ці залежності не враховують взаємодію сусідніх поїздів при їх русі.

Висновок

При імітаційному моделюванні рівняння руху визначається блоком вибору машиністом режиму руху і вирішується для кожного поїзда окремо з урахуванням координат і швидкостей попереду і ззаду наступних поїздів. При моделюванні руху на ділянці враховується довжина поїзда при переході з одного ухилу на інший. Так як основною цільовою функцією системи є затарати енергії, то розробка імітаційних моделей окремих елементів технологічного комплексу починається з розгляду питань моделювання процесів руху (моделі класу М).

РОЗДІЛ 3. ЗАДАЧА БАГАТОКРИТЕРІАЛЬНОЇ ОПТИМІЗАЦІЇ. ПРИНЦИП ПАРЕТО

До цих пір ми розглядали завдання оптимізації, де ясний критерій (показник ефективності) по якому проводиться оцінка ефективності проектованого об'єкта, тобто потрібно звернути в min (max) один єдиний показник. На жаль, такі завдання на практиці зустрічаються рідко.

Доводиться розглядати додаткові критерії (показники ефективності). Чим більше критеріїв якості вводиться в розгляд, тим більш повну характеристику достоїнств і недоліків проектованого об'єкта можна отримати. Таким чином, завдання проектування складних систем завжди багатокритеріальні, тому що при виборі найкращого варіанта доводиться враховувати багато різних вимог, пред'явлених до системи (об'єкту).

Вперше проблема багатокритеріальної оптимізації виникла у Італійського економіста В. Парето при математичному дослідженні товарного обсягу. Надалі інтерес до проблеми векторної оптимізації посилився у зв'язку з розробкою і широким використанням обчислювальної техніки в роботах все тих же економістів-математиків. І вже пізніше стало ясно, що багатокритеріальні задачі виникають не тільки в економіці, але і в техніці.

При вирішенні завдань слід основну увагу звернути на попередньо тільних етап – складання математичної моделі (ММ) і на заключному етапі – всебічний аналіз отриманого оптимального рішення.

Складання математичної моделі починається з вибору змінних, сукупність числових значень, яких однозначно визначає один з варіантів процесу. Після вибору змінних необхідно по тексту задачі скласти обмеження, яким ці змінні повинні задовольняти. При цьому потрібно стежити, щоб в модель були включені всі обмеження, а в той же час не було жодного зайвого або записаного в більш жорсткою, ніж вимагається умовами задачі, формі.

Нарешті, складається цільова функція (функції), яка в мате математичній формі відображає критерій (критерії) вибору кращого варіанта. Після складання математичної моделі необхідно розглянути можливі шляхи її спрощення та вибрати відповідний обчислювальний метод для вирішення задачі.

Математичні моделі можуть бути функціональними, якщо вони ото-зображують фізичні або інформаційні процеси, що протікають в модельований об'єкті, і структурними, якщо вони відображають тільки структурні (наприклад, геометричні властивості об'єктів. Функціональні моделі найчастіше являють собою системи рівнянь, а структурні моделі – це графи, матриці.

У математичній моделі об'єктів проектування зазвичай виділяють властивості систем, елементів систем і зовнішнього середовища, в якій повинен діяти об'єкт. Кількісні подання цих властивостей називають параметрами, тобто фігурують у математичній моделі об'єктів проектування величини називають параметрами. Параметр – це величина, що характеризує властивості або режим його функціонування

Розрізняють вихідні параметри як величини, що характеризують властивості системи, зовнішні параметри як величини, що характеризують властивості зовнішнього середовища, внутрішні параметри як величини, що характеризують властивості елементів системи.

Параметри елементів об'єкта називають внутрішніми параметрами, величини. Отже, внутрішні параметри характеризують властивості елементів проектованого об'єкту (проектні параметри).

Ті внутрішні параметри, які є незалежними один від одного і можуть змінюватися в деяких межах, називаються керованими параметрами (незалежними).

Параметри, що характеризують властивості об'єкта, називають вихідними параметрами.

Параметри, що характеризують властивості зовнішньої по відношенню до даного об'єкту середовища, називають зовнішніми параметрами.

Постановка задачі багатокритеріальної оптимізації

Передбачається, що $m \geq 2$, при $m = 1$ задача оптимізації є одно-критеріальною (скалярною).

Задачі оптимізації, в яких є не одна, а кілька цільових функцій (критеріїв), отримали назву багатокритеріальних задач оптимізації.

Критерії F_i (X), i = 1,2,. . . , M, утворюють векторний критерій F (X) = (F_1, F_2, ..., F_m). Тому в літературі також використовують термін "векторна оптимізація".

Нехай $X_1 \epsilon D$, тоді

F_1 (X1) - локальна оцінка рішення X_1 по 1 - му критерію або критерієм F_1;

F_2 (X1) - локальна оцінка рішення X_1 по 2 - му критерію або критерієм F_2;

.

.

.

Fm (X_1) - локальна оцінка рішення X_1 по m - му критерію або критерієм Fm;

F (X_1) = (F_1 (X_1), F_2 (X_1), Fm (X_1)) - векторна оцінка для вирішення X_1.

Для пояснення суті завдань використовують геометричну інтерпретацію, пов'язану з введенням n - вимірного простору E_n простору параметрів проектування (керованих параметрів) і m - мірного простору E_m вихідних параметрів. Кожній точці простору E_n і E_m відповідають вектори X і Y значень змінних проектування та вихідних параметрів відповідного об'єкта, що проектується.

Отже, допустимої області D (образ) можна поставити у відповідності деякий безліч оцінок. Це безліч будемо позначати Y_D і його будемо називати критеріальним простором або областю критеріїв (областю оцінок), тобто Y_D = F (D) - прообраз безлічі D.

Сформулюємо задачу багатокритеріальної оптимізації. Вона має вигляд:

min F (X) min F (X)

$X \epsilon D$ або

$h_k (X) = 0, k = 1,2,. . . , K;$ (3.1)

$g_j (X) \ ^3 \ 0, j = 1,2,. . . , J.$

Завдання багатокритеріальної оптимізації може бути сформульована таким чином, наприклад:

в квадраті $D = \{-1 \le x_1 \le 1, -1 \le x_2 \le 1\}$ задані два критерії які бажано мінімізувати.

Питається, чи можна знайти рішення, одночасно задовольняючи всім цим вимогам? З усією відвертістю відповімо: ні. Рішення, що звертає в мінімум один якийсь показник, як правило, не звертає ні в мінімум, ні в максимум інші. Тому часто вживана формулювання: "досягти максимального ефекту при мінімальних витратах" являє собою не більш ніж фразу і при науковому аналізі повинна бути відкинута . Теоретично можна уявити собі випадок, коли на множині D виявиться одна альтернатива (рішення), в якій всі m критеріїв приймають найменші значення; вона і є найкращою. Однак на практиці такі випадки майже не зустрічаються, і виникає питання, як же тоді здійснювати вибір. Як правило, критерії суперечливі, тобто зменшення одного критерію веде до збільшення інших критеріїв.

При розробці методів вирішення доводиться вирішувати специфічні проблеми. Розглянемо ці проблеми докладніше.

1.Непорівнянність рішень.

Основна складність логічного аналізу багатокритеріальних задач полягає в тому, що в них, на відміну від «звичайних» (одно-крітеріальним) завдань з'являється ефект непорівнянність варіантів (рішень).

2.Нормалізація критеріїв.

Так як приватні критерії мають різний фізичний зміст, тобто вимірюються в різних одиницях; масштаби їх не співмірні, тому неможливо порівняння якості отриманих результатів за кожним критерієм.

Операція приведення масштабів локальних критеріїв до єдиного, зазвичай безрозмірного, носить назву нормалізації критеріїв.

Після нормалізації приватних критеріїв векторні критерії набувають деякі корисні властивості. Головне з них – будь-яка перестановка приватних критеріїв призводить до векторної оцінкою, яка входить в безліч векторних оцінок (значень вихідної векторної оцінки). За допомогою нормалізації приватних критеріїв будуються покрокові математичні алгоритми звуження вихідної безлічі D до єдиного рішення. Нормалізація приватних критеріїв використовується, наприклад, при побудові адитивного критерію оптимальності.

3.Вибір принципу оптимальності.

Тобто потрібно визначити правило, яке дозволило б сказати яке рішення краще. Вибір принципу оптимальності - основна проблема векторної оптимізації. Формально описати принцип оптимальності (критерії «правильності рішення») - виявляється затрудненим.

4.Облік пріоритету критеріїв.

Зазвичай з фізичного змісту задачі випливає, що локальні критерії мають різну важливість при вирішенні задачі, тобто один локальний критерій має якийсь пріоритет над іншим локальним критерієм. Це слід враховувати при виборі принципу оптимальності і визначенні області можливих рішень, віддаючи перевагу більш важливим критеріям.

5.Обчислення оптимуму ЗВО.

Зараз досягнуті певні успіхи в області вирішення задач математичного програмування (МП). Так за одними даними, методів одно-крітеріальної оптимізації та їх модифікацій більш 500 (п'ятисот), за іншими – їх кількість перевалила за кілька тисяч!

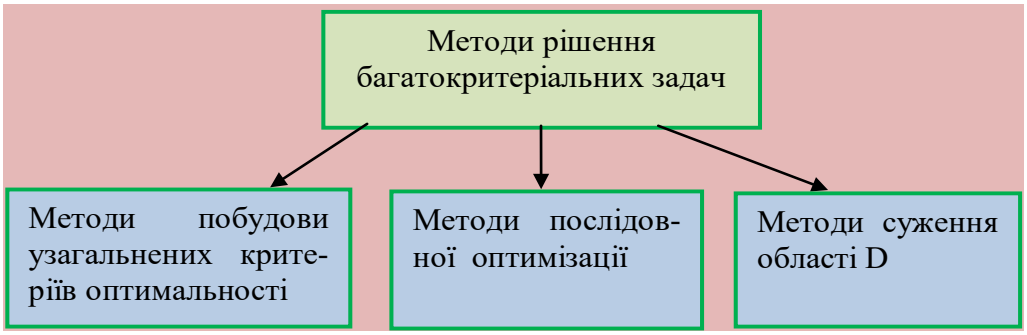


Рис. 3.1 - Методи розв'язання багатокритеріальних задач

В науці і техніці досить актуальні задачі багатокритеріальної оптимізації, що вимагають одночасної оптимізації відразу за декількома функціями (критеріям). Поняттям в багатокритеріальної оптимізації є - Парето-оптимальна (не домінуюча) альтернатива, тому пошук прийнятної («оптимальної») альтернативи, що є рішенням багатокритеріальної задачі, слід виконувати на безлічі не домінуючих альтернатив. Саме тому так актуальні методи, що дозволяють виділяти підмножини Парето-оптимальних альтернатив з безлічі можливих альтернатив.

Рішення $X \epsilon D$ набір його оцінок за всіма критеріями, тобто набір (F_1 (X), F_2 (X), ..., F_m (X)), є векторна оцінка рішення X. Векторна оцінка X містить повну інформацію про цінності (корисності) цього рішення для ОПР і порівняння будь-яких двох рішень замінюється порівняння їх векторних оцінок. Нехай в МЗО потрібно отримати менші значення кожного приватного критерію (мінімізувати приватні критерії) F_i (X).

Нехай є два рішення X_1 і X_2. Кажуть, що рішення X_1 краще (переважніше, ефективніше, домінує) рішення X_2, якщо F_i (X1) <= F_i (X_2) для всіх i = 1, m, і хоча б для одного j - го критерію виконується суворе нерівність F_i (X_1) <F_i (X_2) або рішення X_2 називається домінуючим, якщо існує рішення X_1, не гірше ніж X_2, тобто для будь оптимізується функції F_i, I = 1, 2, ..., m,

F_i (X2) ≤ F_i (X_1) при максимізації функції F_i,

F_i (X2) ≤ F_i (X_1) при мінімізації F_i.

У разі домінування при переході від X_2 до X_1 нічого не буде програно ні по одному з приватних критеріїв, але у відношенні j - го приватного

критерію точно буде отриманий виграш. Кажуть, що рішення X_1 краще (переважно) рішення X_2. Стратегія $X_1 \in D$ називається ефективною (оптимальної по Парето), якщо не існує стратегії $X_2 \in D$ такий, що $F_i (X_2) \leq Fi (X_1)$, i = 1,. . ., m, $F (X_2) \neq F (X_1)$, або якщо рішення не домінує ніяким іншим рішенням, то воно називається не домінуючим або оптимальним в сенсі Парето.

Очевидно, тоді в складі безлічі D немає сенсу зберігати рішення X_2, воно витісняється (або, як кажуть, «домінує») рішенням X_1. Гаразд, викинемо, рішення X_2 як неконкурентоспроможне і перейдемо до порівняння інших рішень по всім критеріям. В результаті такої процедури відкидання свідомо непридатних, невигідних рішень безліч D зазвичай сильно зменшується: у ньому зберігаються тільки так звані ефективні (інакше «паретовське») рішення, характерні тим, що ні для одного з них не існує домінуючого рішення. Безліч таких точок і називається множиною точок оптимальних за Парето. Безліч точок оптимальних за Парето лежать між точками оптимумів, отриманих при рішенні задачі математичного програмування для кожного приватного критерію. У літературі безліч точок оптимальних за Парето, як правило, позначають буквою P ($P \in D$).

Безліч векторних оцінок, які відповідають безлічі ефективних точок, називають областю компромісів (переговорним безліччю) або безліччю Парето в області критеріїв.

Оптимальність за Парето означає, що не можна далі поліпшувати значення одного критерію, не погіршуючи при цьому хоча б одного з решти.

Проілюструємо прийом виділення паретовських рішень на прикладі задачі з двома критеріями F_1 і F_2 (обидва потрібно максимізувати). Безліч D складається з 11 можливих рішень. Кожному рішенню відповідають певні значення показників F_1 і F_2. Нехай є наступні векторні оцінки: $F (X_1) = (2, 4)$, $F (X_2) = (3, 5)$, $F (X_3) = (3, 3)$, $F (X_4) = (5; 2)$, $F (X_5) = (4, 3)$, $F (X_6) = (1, 3)$, $F (X_7) = (2, 3)$, $F (X_8) = (3, 2)$, $F (X_9) = (2; 2)$, $F (X_{10}) = (3; 1)$, $F (X_{11}) = (2; 1)$. Векторні оцінки результатів представимо точками координатної площини (по

осі абсцис відкладаємо значення критерію F_1, а по осі ординат - значення критерію F_2). Використовуємо принцип оптимальності за Парето для виділення ефективних рішень. Рішення X_1 витісняється рішенням X_2, рішення X_2 краще рішень X_3, X_7, X_8, X_9, X_{10} і X_{11}. Рішення X_4 за першим критерієм краще рішення X_5, а по другому навпаки, тобто маємо не поліпшуюче рішення, і т.д. Після проведеного аналізу у нас залишилися три рішення X_2, X_4, X_5 оптимальних за Парето. Побудуємо критеріальний простір для нашої задачі. Як відомо парі чисел відповідає точка на площині. Занумеруємо точки відповідно номеру рішення (рис. 3.2). З малюнка видно, що ефективні точки лежать на правій верхній границі області можливих рішень

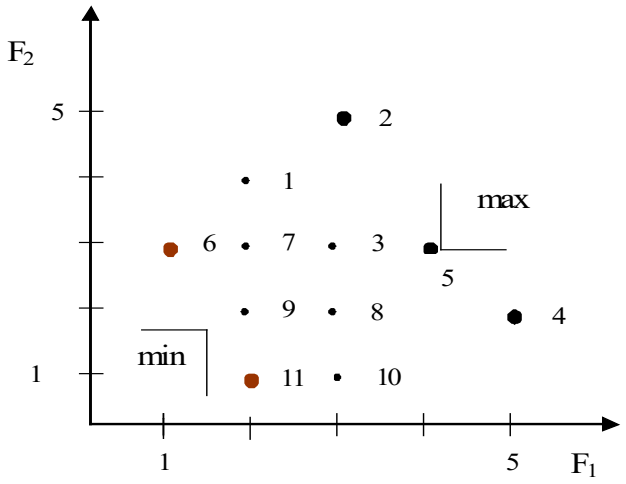


Рисунок 3.2 - Критеріальний простір для задачі з двома критеріями X_1 і X_2.

Коли з безлічі можливих рішень виділені ефективні, «переговори» можуть вестися вже в межах цього «ефективного» безлічі. На рис 3. утворюють три рішення X_2, X_4, X_5; із них X_4 краще за критерієм F_1, а рішення X_2 за критерієм F_2. Справа ОПР, вибрати той варіант, який для нього кращий і «прийнятний» за обома критеріями.

У випадку, коли безліч допустимих результатів є безперервним, їх векторні оцінки «заповнюють» деяку область Y_D на площині і получається «картинка» на зразок зображеної на рис. 3.3 У цьому випадку безліч Парето-

оптимальних оцінок (червона лінія) являє собою частину кордону Y_D, образно кажучи, її «південно-західний» кордон . Якщо критерії максимізують то –« північно- східний » кордон області Y_D.

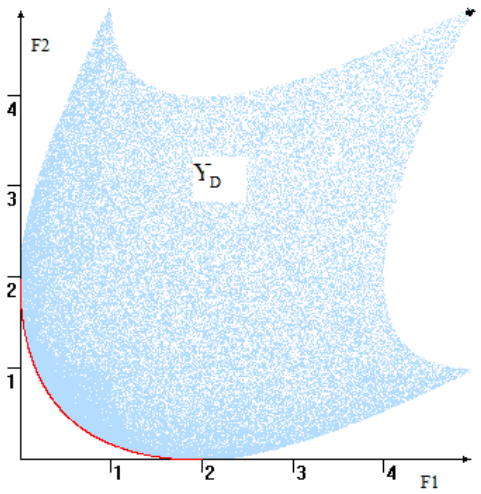


Рисунок 3.3 - Область Y_D, яка являє собою безліч результатів векторної оцінки.

Висновок

Складання математичної моделі починається з вибору змінних, сукупність числових значень, яких однозначно визначає один з варіантів процесу. Після вибору змінних необхідно по тексту задачі скласти обмеження, яким ці змінні повинні задовольняти. При цьому потрібно стежити, щоб в модель були включені всі обмеження, а в той же час не було жодного зайвого або записаного в більш жорсткою, ніж вимагається умовами задачі, формі. Досить актуальні задачі багатокритеріальної оптимізації, що вимагають одночасної оптимізації відразу за декількома функціями (критеріям). Поняттям в багатокритеріальної оптимізації є – Парето-оптимальна (не домінуюча) альтернатива, тому пошук прийнятної («оптимальної») альтернативи, слід виконувати на безлічі не домінуючих альтернатив. Безліч векторних оцінок, які відповідають безлічі ефективних точок, називають областю компромісів (переговорним безліччю) або безліччю Парето в області критеріїв.

РОЗДІЛ 4. ОПТИМАЛЬНІ ТЯГОВІ РОЗРАХУНКИ НА МНОЖЕННІ ПАРЕТО

4.1 Метод динамічного програмування Беллмана

Динамічне програмування (ДП) використовується, коли рішення задачі на кожному етапі є рішенням деякої задачі. Ідея ґрунтується на запам'ятовуванні рішення підзадач і побудови через них рішення вихідної задачі. За рахунок запам'ятовування рішень підзадач перебір значно скорочується.

Метод динамічного програмування.

Зазвичай завдання, до якого застосовується метод динамічного програмування, представляється багатостадійним процесом. Досліджуючи процес S розбивається на стадії, які характеризуються N координатами S_j, $j=\overline{0,N}$. Процес на кожній стадії може перебувати в декількох станах s_{ij}, $i=\overline{0,M}$. На кожній стадії здійснюється вибір управління для переходу в необхідний стан наступній стадії. Призначення переходу зі стану S_j в будь допустиме стан S_{j+1} назвемо локальним управлінням u. Кожному з станів процесу S приписується множена $U(S)$ можливих локальних управлінь, що здійснюють переходи в інші стани. Реалізація локального управління $u \in U(S)$ має деякий якість $c(u)$. Потрібно визначити послідовність локального управління, що переводять систему з початкового стану S_0 в деякий кінцевий стан S_N, з найкращим показником якості $C(S_0)$, що дорівнює сумі складових якості її локальних управлінь. Така послідовність локальних управлінь називається оптимальним управлінням для здійснення переходу зі стану S_0 з показником якості $C(S_0)$. Аналогічних сенс вкладається і в поняття оптимального управління з будь-якого іншого стану S_j процесу S.

В основу методу динамічного програмування покладено принцип оптимальності Беллмана: в розглянутій стадії S_j слід вибирати таке локальне

управління $u \in U(S)$, яке в сукупності з оптимальним управлінням із стану S_{j+1} складе оптимальне управління зі стану S_j. Цей принцип записується так

$$C(S_j) = \min_{u \in U(S_j)} \{c(u) + C(U(S_{j+1}))\}. \qquad (4.1)$$

Критерій якості локального управління може бути і протилежним критерієм оптимального управління, ніж у виразі (4.1).

Пошук мінімального шляху в графі.

Якщо безліч локальних управлінь конечне, то можлива інтерпретація принципу оптимальності за допомогою графів. Визначимо зважений орієнтований граф $G = (V, E)$ (далі граф), вершини якого відповідають станам системи, а дуги – всіляким переходам зі стану в стан; ваги дуг покладаються рівними оцінками якості відповідних локальних управлінь.

Оптимальним орієнтованим шляхом з деякої вершини $v \in V$ назвемо шлях мінімальної довжини $C(v)$ з вершини v в одну з вершин, відповідних кінцевому стану.

Розглянемо елементи орграфа G: вершину v стадії j і безліч всіх дуг (v,v_1), (v,v_2) ,…, (v,v_m), що виходять з v. Дуга (v,v_i) вагою $c(v,v_i) = c(u^i)$ відповідає локальному управлінню $u^i \in U(S)$, $i = \overline{1,m}$, переводить систему S в деякий стан стадії S_{j+1}. Принцип оптимальності в термінах теорії графів формулюється так: будучи в вершині v, слід вибирати ту дугу (v,v_i), яка разом з оптимальним шляхом з вершини v_i, - складе оптимальний шлях з вершини v:

$$C(v) = \min_{i=\overline{1,m}} \{c(v,v_i) + C(v_i)\}.$$

При визначенні $C(v)$ необхідно знати величини $C(v_i)$, $i = \overline{1,m}$. Це можливо, якщо граф допускає топологічну сортування (є безконтурним).

Алгоритм пошуку оптимальних шляхів у зв'язному безконтурному орграфі, що використовує принцип оптимальності Беллмана.

Крок 0. Провести топологічну сортування вершин орграфа $G = (V, E)$,, пронумерувавши їх числами $1, 2, \ldots, n$. Покласти $step_1 = n$ і перейти до наступного кроку.

Крок k , $(k \geq 1)$. 1. Якщо $step_k = 0$, то алгоритм закінчено. Якщо $step_k \neq 0$, то вибрати вершину v з номером $step_k$. Якщо немає дуг з початком у вершині v, то покласти $C(v) = 0$, $top(v) = \varnothing$, $step_{k+1} = step_k - 1$ і перейти до наступного кроку.

2. Якщо маються дуги з початком у вершині v, то вибрати серед них дугу (v, w) з мінімально можливим значенням

$c(v, w) + C(w)$.

Покласти

$C(v) = c(v, w) + C(w)$, $top(v) = w$, $step_{k+1} = step_k - 1$

і перейти до наступного кроку.

Часова складність алгоритму дорівнює

$$O\left(|E| \max_{e \in E} \log_2 c(e) \right), \qquad (4.2)$$

де $c(e)$ - вага дуги e.

4.2 Схема динамічного програмування

Розглядається задача, яка змістовно відома як задача оптимальних тягових розрахунків. Величини

$f(s)$, $t(s)$ – показники, що відображають перевізний процес і являють собою витрати енергоресурсів (електроенергія, паливо) і часу на доставку вантажу;

s – координата колії. Поїзд розглядається як тверде тіло з масою зосередженої в його центрі. Рівняння руху потягу враховуються як в [24]. Вважаються заданими

– поздовжній профіль колії;

– обмеження швидкості по колії проходження;

– маса складу;

– тип вагонів, навантаження на вісь;

– маса електровоза;

– тягові характеристики електровоза;

– обмеження часу проходження;

– початкова і кінцева швидкість;

– довжина ділянки колії.

З точки зору витрат енергоресурсів на рух виникає задача про побудову закону керування потягом, де критерієм оптимальності є витрата енергоресурсів. Критичним залишається вимога витрат часу на проходження поїзда для даної ділянки.

Нехай

s – координата колії, $0 \le s \le l$;

l – довжина ділянки колії (значення кінцевої координати ділянки);

$v(s)$ – швидкість руху поїзда;

$f(v(s))$ – витрати енергоресурсів;

$t(v(s))$ – функція витрат часу в залежності від обраної швидкості руху;

$\overline{T}$ – час руху по ділянці.

Задача на оптимальне управління рухом поїзда з мінімальною витратою енергії коротко можна сформулювати так: знайти таке допустиме управління $v(s)$, при якому відповідний витрат енергоресурсів був би мінімальним і виконувався графік руху на даній ділянці.

Зазвичай задача оптимального управління руху поїзда з мінімальним витратами енергоресурсів має вигляд

$$\min_{v(s)} f(s) \qquad (4.3)$$

за умови

$$t(v(s)) = \overline{T}, \quad 0 \le s \le l. \qquad (4.4)$$

Управлінням є швидкість руху.

Модель (4.2)–(4.4) враховує не всі обмеження. Необхідно при розрахунках ще врахувати й інші чинники: початкову та кінцеву швидкість, характеристики локомотива (обмеження на питому дотичну силу, обмеження на питому гальмівну силу, ККД і ін.), обмеження швидкісного режиму, перегрів тягового двигуна.

Пом'якшимо жорстку умову щодо часу проходження (4.4) і замість рівності (4.2) будемо розглядати обмеження

$$t(v(s)) \le \overline{T}, \quad 0 \le s \le l. \qquad (4.4)$$

Модель (4.3)–(4.4') є неперервною. Для побудови схеми розв’язку задачі перейдемо до відповідної дискретної моделі.

Розіб'ємо ділянку колії $0 \le s \le l$ на N елементарних ділянок $\{[s_{j-1}, s_j]\}$, $j = 1,\dots,N$. У точках розбиття s_j швидкість $v = v(s_j)$ може приймати кінцеву безліч значень V_j, $j = 1,\dots,N$

$$V_j = \{v_i(s_j)\}, \quad i = 1 \dots m_j,$$

де m_j – кількість елементів у множині V_j.

Величина m_j визначається обмеженнями на швидкість руху в точці s_j та у, спосіб дискретизації $v(s_j)$ (регулярний крок розбиття, нерегулярний крок розбиття, величина кроку розбиття). Залежно від вибраної швидкості руху $v \in V_j$ в точці розбиття колії s_j елементарна ділянка $[s_{j-1}, s_j]$ може бути прослідувана за час $t_j = t(V_j)$ – невід’ємна величина, при цьому витрати енергоресурсів складуть $f_j = f(V_j)$ – також невід'ємна величина. Витрати енергоресурсів на ділянці $0 \le s \le l = s_N$ являють собою суму всіх витрат на елементарних ділянках. Витрати часу для $0 \le s \le l = s_N$ представляють суму часу відповідних встановленому енергоресурсу на елементарних ділянках. Інакше, функція витрат енергоресурсів і функція витрат часу є адитивні функції, визначені на кінцевих множинах V_j.

Потрібно вибрати такий режим руху потягу v_k , $k = 0 \ldots N$ (v_0 и v_N надані), при якому сумарні витрати енергоресурсів були б мінімальними, і при цьому загальні витрати часу не виводили з встановленого графіка руху (сумарні витрати часу не перевершували заданої величини $\overline{T}$).

Розглянута задача про оптимальний рух поїзда з мінімальними витратами енергоресурсів (4.3)–(4.4') є канонічною задачою про розподіл ресурсу [25, 26]. Для її рішення пропонується схема методу динамічного програмування. Замість рекурентних рівнянь використовується покрокове обчислення безлічі точок, оптимальних за Парето, на площині значень цільової функції й ресурсу.

У прийнятих позначеннях формальна постановка задачі запишеться так. Знайти мінімум суми

$$\sum_{j=1}^{N} f_j(v_i), \quad v_i \in V_j, \quad j = 1 \ldots N \qquad (4.5)$$

при обмеженнях

$$\sum_{j=1}^{N} t_j(v_i) \le \overline{T}, \quad v_i \in V_j, \quad j = 1 \ldots N. \qquad (4.6)$$

Передбачається, що безліч (4.6) допустимих рішень не порожньо.

Схема динамічного програмування

Задача (4.5)–(4.6) представляє собою відому задачу оптимального розподілу ресурсу, для вирішення якої використовується, зазвичай, метод динамічного програмування [11, 12]. Наведемо основні співвідношення цього методу. Позначимо через $B_j(u)$ оптимум наступної задачі: знайти мінімум суми

$$\sum_{k=1}^{j} f_k(v_k)$$

при обмеженнях $\sum_{i=1}^{j} t_i(v_i) \le u \quad v_i \in V_i, \ j = 1 \ldots N$, де j приймає значення $1, \ldots, N$, $0 < u \le \overline{T}$.

Очевидно, величина $B_N(\overline{T})$ дорівнює оптимуму вихідної задачі (4.5), (4.6). Її розрахунок проводиться за рекурентним рівнянням

$$\begin{cases} B_j(u) = \min\limits_{v_j \in V_j \setminus t_j(v_j) \le u} \left\{ B_{j-1}[u - t_j(v_j)] + f_j(v_j) \right\} \\ 0 < u \le \overline{T} \quad j = 1,\ldots,N. \end{cases} \tag{4.7}$$

При такій організації обчислень необхідно покласти $B_0(u) = +\infty$, $0 < u \le \overline{T}$, і $B_j(u) = 0$, якщо мінімум в (4.7) береться по порожній безлічі.

При великих значеннях $\overline{T}$ і N розрахунок з використанням рівнянь (4.7) вимагає значного обсягу пам'яті і часу рахунку. Нижче пропонується підхід, який дозволяє істотно заощаджувати обчислювальні ресурси.

На площині двох змінних введемо відношення часткового порядку

$$(x, y) \prec (z, w) \Leftrightarrow x \le y, \quad z \le w.$$

Нехай A – деяка безліч точок на площині. Точки з A, мінімальні щодо часткового порядку, називають оптимальними за Парето або просто паретовськими. Розглянемо безліч точок вигляду

$$F = \sum_{i=1}^{j} f_i(v_i), \quad T = \sum_{i=1}^{j} t_i(v_i),$$

де вектор $(v_1, v_2, \ldots, v_j)$ пробігає всі значення, що задовольняють умовам

$$\sum_{i=1}^{j} t_i(v_i) \le \overline{T}, \quad v_i \in V_i, \quad i = 1 \ldots j.$$

Сукупність паретовських точок цієї множини позначимо через S_j. З кількох рівних паретовських точок у безліч S_j включається тільки одна. Позначимо через (F_{jk}, T_{jk}), $k = 1,\ldots,K_j$, точки безлічі S_j, нумеруючи їх за зростанням координат, тобто

$$F_{j1} < F_{j2} < \ldots F_{jK_j}, \quad T_{j1} < T_{j2} < \ldots T_{jK_j}.$$

Неважко бачити, що $B_j(T_{jk}) = F_{jk}$, $k = 1,\ldots,K_j$. Функція $B_j(u)$ є неубуваючою по аргументу u при даному j. Її графік складається з ділянок

постійності і точок зростання, які і складають безліч S_j. Таким чином, безліч S_j містить всю необхідну інформацію про функції в мінімальному обсязі.

Безлічі S_j, $j=1,\ldots,N$ п ерераховуються по кроках, аналогічно рівнянь (4.7). На початковому кроці вважаємо $S_0 = \{(0,0)\}$. Опишемо спільний крок. Нехай вже побудовано безліч

$$S_{j-1} = \left\{F_{j-1,k}, T_{j-1,k}, \quad k=1,\ldots,K_{j-1}\right\}.$$

Розглянемо безліч точок (F,T) вигляду

$$F = F_{j-1,k} + f_j(v_j), \quad T = T_{j-1,k} + t_j(v_j),$$

де $k=1,\ldots,K_{j-1}$, а змінна v_j пробігає всі значення, що задовольняють умовам

$$T_{j-1,k} + t_j(v_j) \le \overline{T}, \quad v_j \in V_j.$$

Виділяючи з цієї множини паретовські точки і залишаючи з рівних точок тільки одну, отримуємо безліч

$$S_j = \left\{\left(F_{j,k}, T_{j,k}\right), \quad k=1,\ldots,K_j\right\}.$$

Цей процес завершується побудовою безлічі

$$S_N = \left\{\left(F_{N,k}, T_{N,k}\right), \; k=1,\ldots,K_N\right\}.$$

Величина F_{NK_N} дорівнює оптимуму початкової задачі. Відповідне вказаному оптимуму значення T_{NK_N} є витратами часу. Тут використано таку

властивість рішення: перспективні пари (F,T) утворюють безліч Парето, а всі інші можна видалити (але можна і залишити). У реальній реалізації представленого алгоритму попередньо виділяються для кожного значення індексу $j=1,\ldots,N$ паретовскі точки безлічі

$$\left\{f_j(v_j), t_j(v_j), v_j \in V_j\right\}$$

і використовуються в розрахунках тільки вони.

Якщо перспективні пари (F,T) не вилучати, то серед безлічі пар (F,T) можна знайти такі, які оптимізують час. У самому сприятливому випадку

серед безлічі паретовськіх пар можна вибрати найбільш підходящі до умов графіку руху за витратами часу та енергоресурсів.

Ефективним рішенням багатокритеріальної задачі називають оптимальне по Парето рішення [27]. Пошук ефективного рішення називають ще програмуванням на множині Парето. Чисельна реалізація методу динамічного програмування на множинах Парето дозволяє застосовувати обмеження на використання ресурсу.

Задача на оптимальне управління рухом поїзда з мінімальною витратою енергії та обмеженням часу можна звести до задачі оптимального розподілу ресурсу, для вирішення якої використано метод динамічного програмування на сукупності паретовскіх точок безлічі пар $(F,T) = (енергія, час)$. Рішення засноване на паретовскіх точках неєдине. На безлічі рішень по Парето вибирається одне найбільш підходяще по компромісу щодо організації перевізного процесу для даної ділянки.

Висновок

Динамічне програмування (ДП) використовується, коли рішення задачі на кожному етапі є рішенням деякої задачі. Ідея ґрунтується на запам'ятовуванні рішення підзадач і побудови через них рішення вихідної задачі. За рахунок запам'ятовування рішень підзадач перебір значно скорочується. Зазвичай завдання, до якого застосовується метод динамічного програмування, представляється багатостадійним процесом. В основу методу динамічного програмування покладено принцип оптимальності Беллмана: в розглянутій стадії S_j слід вибирати таке локальне управління $u \in U(S)$, яке в сукупності з оптимальним управлінням із стану S_{j+1} складе оптимальне управління зі стану S_j.

РОЗДІЛ 5. ВИЗНАЧЕННЯ РЕЖИМІВ РУХУ ЕЛЕКТРОВОЗА ДС – 3

5.1 Розрахунок режимів

Рівняння руху поїзда записується так:

$$\frac{dv}{dt} = \frac{\xi}{1+\gamma} f_y(v) \quad \text{або} \frac{dv}{dt} = \zeta f_y(v)\,, \qquad (5.1)$$

$$\zeta = \frac{\xi}{1} + y;$$

ξ – залежить від розмірності величини;

$1+y$ – коефіцієнт інерції;

$f_y(v)$ – рівнодіюча питома сила.

Приклад розмірності:

$[v]$ = км/г;

$[t]$ = г;

$[s]$ = км;

$[f_y] = \frac{H}{кН}$.

Тоді ξ = 127;

В звичайних розрахунках приймати $1+y$=1,06.

Рівнодіюча сила, яка діє на електровоз визначається:

$f_y(v) = f_k(v) - w_o(v) - i(s)$ – режим тяги;

$f_y(v) = -\{w_{ox}(v) + i(s)\}$ – режим вибігу;

$f_y(v) = -\{w_{ox}(v) + i(s) + b_k(v)\}$ – режим гальмування.

Позитивний напрямок сил у відповідності з рис. 5.1

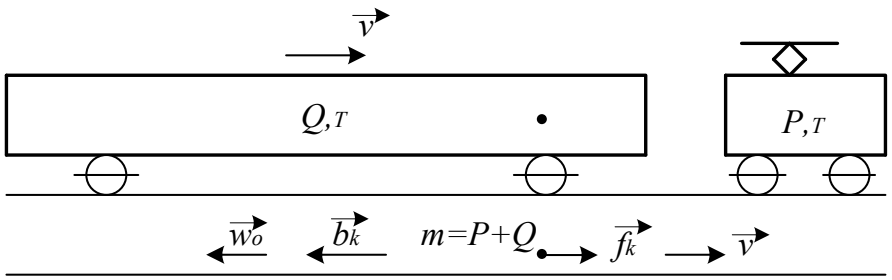


Рисунок 5.1 - Напрямок руху сил.

Основний питомий опір руху потягу:

- тяга

$$w_o = \frac{w_o^{'}P + w_0^{"}Q}{P + Q};$$

- вибіг (гальмування)

$$w_{ox} = \frac{w_x P + w_o^{"}Q}{P + Q},$$

де $w_o^{'}\ i\ \ w_x$ – основний питомий опір руху електровоза відповідно в режимі тяги та вибігу (табл.1.2);

$w_o^{"}$ – основний питомий опір руху состава.

Основний питомий опір руху состава вантажного потягу

$w_o^{"} = \alpha w_{o4}^{"} + \beta w_{o8}^{"}$,

де $\alpha\ та\ \beta$ - доля ваги 4-х та 8-и-вісних вагонів у вазі потягу ($\alpha + \beta = 1$).

Основний питомий опір руху состава пасажирського потягу

$w_o^{"} = w_o^{"}(\text{пасажирських}) + w_{\text{пг}}$,

де - $w_{пг}$ - опір руху під вагонних генераторів

$$w_{пг} = \begin{cases} 0\ при\ 0 \le v \le 120\ км/г \\ = \dfrac{136P}{q_o v}, \dfrac{H}{kH}, \end{cases}$$

де Р – середня потужність під вагонних генераторів на один вагон потягу, кВт (Р ≅ 9 кВт).

Таблиця 1.2 - Основний питомий опір руху електровоза

Тип рухомого складу	Тип колії	
	ланковий	безстикової
Електровоз під струмом	$w_o^{'} = 1{,}9 + 0{,}01v + 0{,}0003v^2$	$w_o^{'} = 1{,}9 + 0{,}008v + 0{,}00025v^2$
Електровоз на вибігу	$w_x = 2{,}4 + 0{,}011v + 0{,}00035v^2$	$w_x = 2{,}4 + 0{,}009v + 0{,}00035v^2$
Пасажирські вагони	$w_o^{"} = 0{,}7 + \frac{8 + 0{,}18v + 0{,}003v^2}{q_o}$	$w_o^{"} = 0{,}7 + \frac{8 + 0{,}16v + 0{,}0023v^2}{q_o}$
4-х вісні грузові	$w_o^{"} = 0{,}7 + \frac{3 + 0{,}1v + 0{,}0025v^2}{q_o}$	$w_o^{"} = 0{,}7 + \frac{3 + 0{,}9v + 0{,}002v^2}{q_o}$
8-и вісні цистерни	$w_o^{"} = 0{,}7 + \frac{6 + 0{,}038v + 0{,}0021v^2}{q_o}$	$w_o^{"} = 0{,}7 + \frac{6 + 0{,}026v + 0{,}0017v^2}{q_o}$

Питома гальмівна сила може приймати значення

$$0 \le b_k \le mb_k, \frac{H}{kH} = (\frac{кгс}{т})$$

де b_k – гальмівна сила при екстреному гальмуванні;

m – коефіцієнт, який визначає ступінь використання гальмівних засобів.

Приймають $m = \begin{cases} 0{,}5 - \text{вантажні потяги;} \\ 0{,}6 - \text{пасажирські потяги.} \end{cases}$

Питома гальмівна сила при екстреному гальмуванні

$$b_k = 1000\vartheta_p\varphi_{кр},$$

де ϑ_p - розрахунковий гальмівний коефіцієнт потягу;

$\varphi_{кр}$ - розрахунковий коефіцієнт тертя колодки об бандаж (диск).

Гальмівний коефіцієнт

$$\vartheta_p = \begin{cases} 0{,}33 - \text{вантажні потяги;} \\ 0{,}60 - \text{пасажирські потяги.} \end{cases}$$

Розрахунковий коефіцієнт тертя

$\varphi_{кр} = 0{,}27\dfrac{v+100}{5v+100}-$ стандартні чугунні колодки з підвищеним вмістом фосфору;

$\varphi_{кр} = 0{,}36\dfrac{v+150}{2v+150}-$ композиційні колодки.

Питома сила тяги.

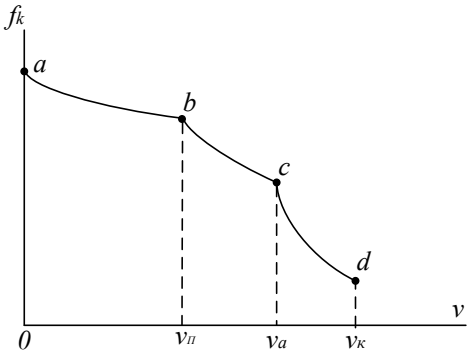
Область допустимих значень

$$0 \le f_k \le f_k(v),$$

де $f_k(v)$ - гранична тягова характеристика в граничних одиницях.

Гранична тягова характеристика

$$f_k(v) = min\{f_{ab}(v), f_{bc}(v), f_{cd}(v)\}.$$



Ділянка ab – обмеження по зчепленню

$$f_{ab} = \frac{F_{зп}(v)}{(p+Q)\times 9{,}81}, \quad \frac{H}{kH} = \left(\frac{кгс}{T}\right),$$

де сила зчеплення

$$F_{зч} = 1000P\psi_k \times 9{,}81, \qquad H,$$

розрахунковий коефіцієнт зчеплення

$$\psi_к = a_o + \frac{a_1}{a_2 + a_3 v} - a_4 v,$$

де $a_o - a_4$ – числові коефіцієнти.

Ділянка bc ($v_П < v \le v_a$) – характеристика постійної потужності.

$$f_{bc} = \frac{f_{ab}(v_П)\times v_П}{v}.$$

Ділянка cd ($v_a < v \le v_k$) – серієсна характеристика.

$$f_{cd} = \frac{f_{ab}(v_П) \times v_П \times v_a}{v^2}.$$

Якщо ділянки cd не має, як у електровоза ДС3, в початкових даних приймати $v_a = v_k$.

Змінні константи в розрахункових формулах та варіанти розрахунку.

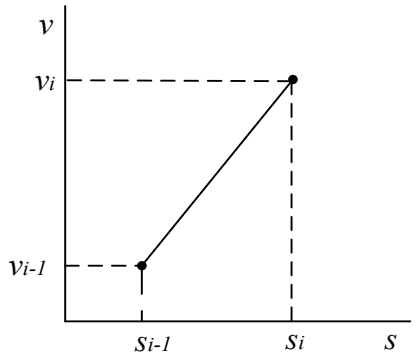
P, Q, $v_П$, v_a, v_k, $a_o - a_4$, v_p, q_o, $P_{пг}$.

Коливатися можуть:

- тип колії ;
- тип гальмівних колодок;
- тип потягу.

Рівнодіюча питома сила, яка забезпечує перехід від v_{i-1} до v_i , дорівнює

$$f_{yi} = \frac{v_1^2 - v_{i-1}^2}{2\zeta(s_i - s_{i-1})}$$



(при [v] = км/г; [f_y] = $\frac{H}{kH}$ та 1+ $\gamma = 1{,}06$ має ζ=120).

або при [s] = м

$$f_{yi} = \frac{500[v_i^2 - v_{i-1}^2]}{\zeta(s_i - s_{i-1})}.$$

Для вибору режиму розраховуємо:

- середнє значення швидкості

$$v_{ci} = 0{,}5(v_{i-1} + v_i);$$

- опір руху для режимів
  - вибігу $-w_{oxi}\,(v_{ci})$,
  - тяги $-w_{oi}\,(v_{ci})$

- значення сили тяги $f_k(v)$ та гальмівної сили b_k з умови переходу в стан *i* – 1 в стан *i*:

$$f_{ki} = f_{yi} + w_o(v_{ci}) + i(s);$$

$$b_{ki} = -\{f_{yi} + w_{ox}(v_{ci}) + i(s)\},$$

де *i* – алгебраїчне значення сили тяги та гальмівної сили:

$$f_k(v) = min\{f_{ab}(v), f_{bc}(v), f_{cd}(v)\};$$

$$b_k(v_c) = m1000\vartheta_p\varphi_{кр}(v_c),$$

Приймаємо режим відповідно до табл.5.1

Таблиця 5.1 – Режими руху електровоза ДС-3

Признак	Режим	Умови реалізації переходу	Опір руху потягу
$f_k \le 0$ *та* $b_k \le 0$	Вибіг	Завжди	$w = \begin{cases} w_{ox} \text{ при } b_k = 0; \\ 0{,}5(w_{ox} + w_0) \text{ при } b_k < 0 \end{cases}$
$f_k > 0\ ;\ b_k < 0$	Тяга	$f_k \le f_k(v_c)$	w=w$_o$
$f_k < 0\ ;\ b_k > 0$	Гальмування	$b_k \le mb_k(v_c)$	w=w$_{ox}$

Правило визначення витрат часу

$$\Delta t_i = \frac{2(s_i - s_{i-1})}{v_i + v_{i-1}}.$$

Правило визначення витрат електроенергії ([*s*] = км; [f_k] = Н/кН)

На елементі $s_i - s_{i-1}$

$$\Delta A_i = 2{,}725\frac{f_{ki}(s_i - s_{i-1})}{\eta(F_k^*, v^*)}, \qquad \frac{Вт \times год}{т}$$

де $F_k^* = \frac{f_{ki} \times 9{,}81 \times (P+Q)}{F_{kH}}$; $v^* = \frac{v}{v_H}$,

($F_{kH}, v_H -$ сила тяги та швидкість руху номінального режиму електровоза).

Алгоритм визначення $\eta(F_k^*, v^*)$ – окремо.

Сумарний питома витрата на тягу

$$a = \frac{\sum_{i=1}^{n} \Delta A_i}{s_n - s_o}, \qquad \frac{\text{Вт} \times \text{год}}{\text{т} \times \text{км}}.$$

Робота сил опору руху

$$\Delta A_{wi} = 2{,}725 w_i (s_i - s_{i-1}), \quad \frac{\text{Вт} \times \text{год}}{\text{т}}.$$

$$a_w = \frac{\sum_{i=1}^{n} \Delta A_{wi}}{s_n - s_o}, \qquad \frac{\text{Вт} \times \text{год}}{\text{т} \times \text{км}}.$$

Робота сил гальмування

$$\Delta A_{bi} = 2{,}725 b_{ki} (s_i - s_{i-1}), \quad \frac{\text{Вт} \times \text{год}}{\text{т}}.$$

$$a_b = \frac{\sum_{i=1}^{n} \Delta A_{bi}}{s_n - s_o}, \qquad \frac{\text{Вт} \times \text{час}}{\text{т} \times \text{км}}.$$

Можливе повернення електроенергії за рахунок рекуперації

$$a_p = a_B \left[\text{при } v \geq v_{\text{рек}} (\approx 40 \text{ км/г})\right] \times \eta_p,$$

де $\eta_p -$ к.к.д. системи рекуперативного гальмування (приближена);

$v_{\text{рек}} -$ швидкість заміщення рекуперації пневматичними гальмами.

5.2 Результуючі характеристики

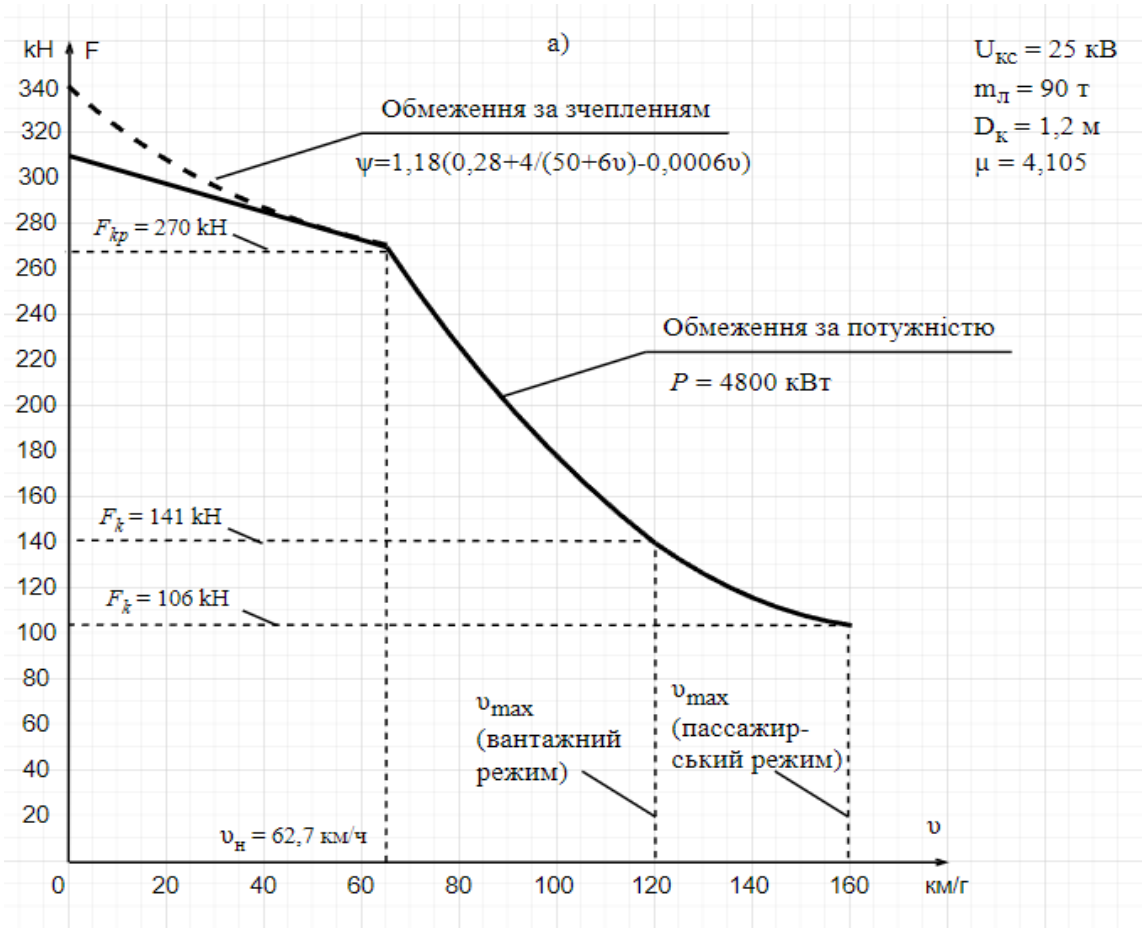


Рисунок 5.2(а) - Тягова характеристика електровозу ДС-3

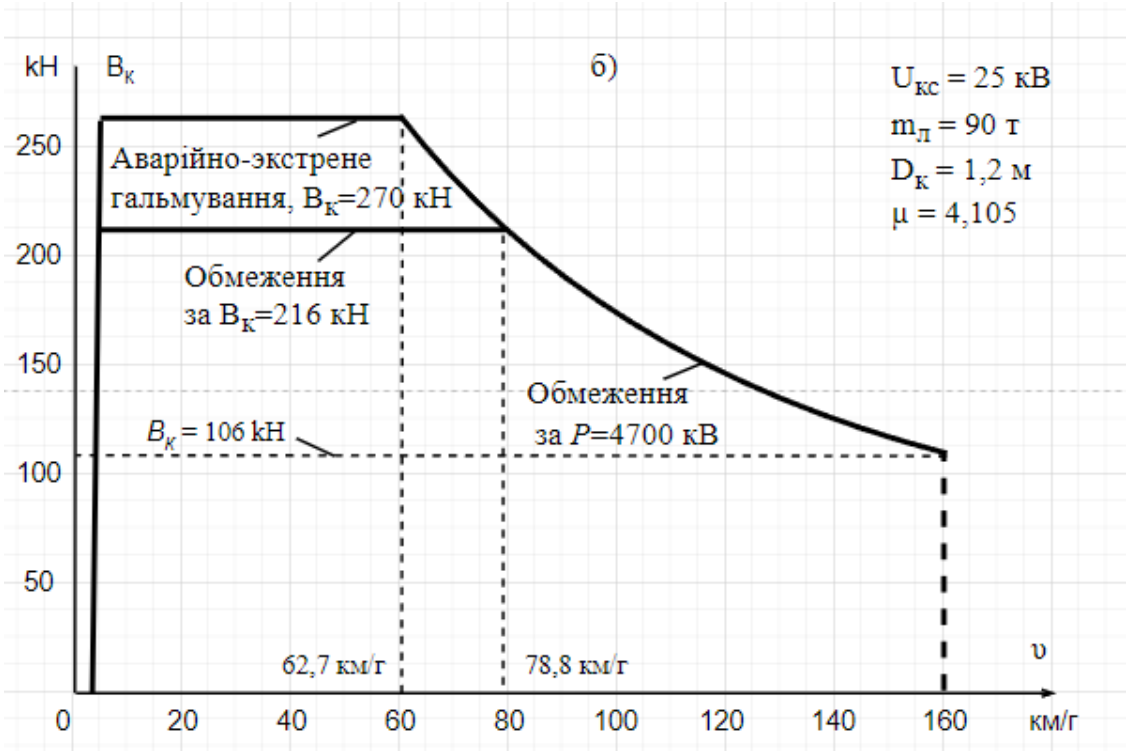


Рисунок 5.2(б)- Гальмівна характеристика електровозу ДС3

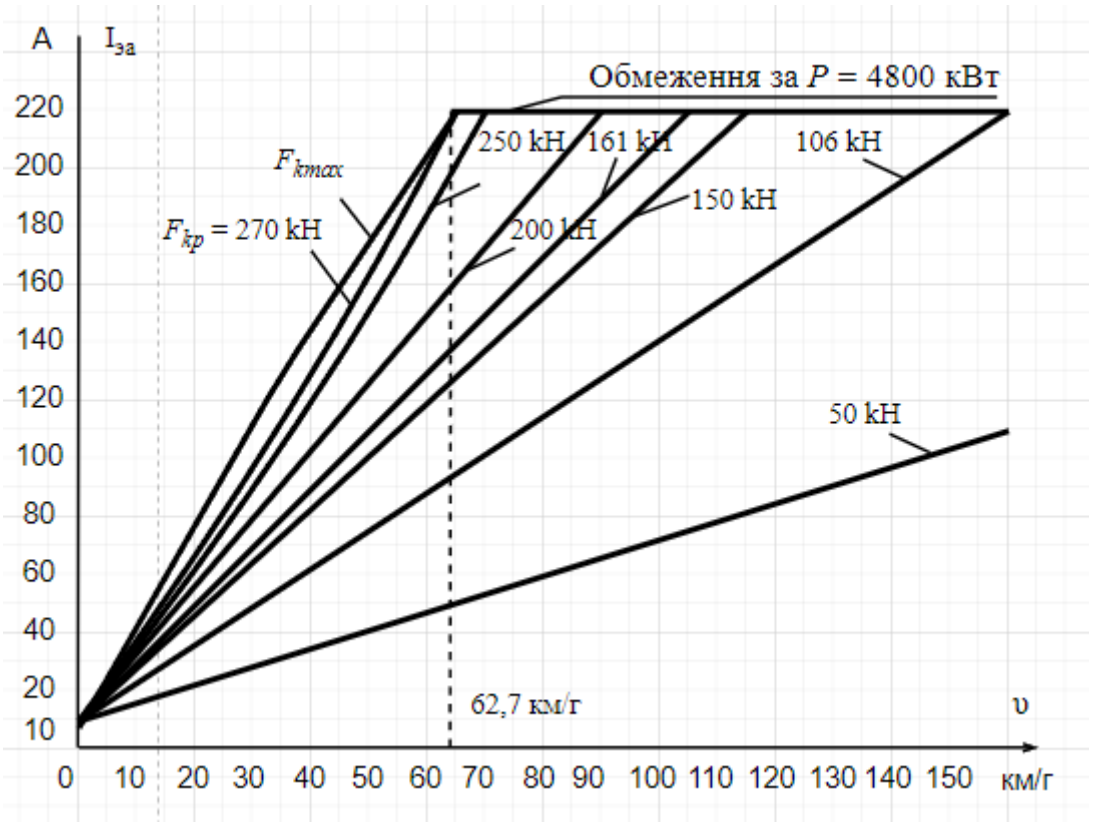


Рисунок 5.3 - Токова характеристика електровозу ДС3 в режимі тяги

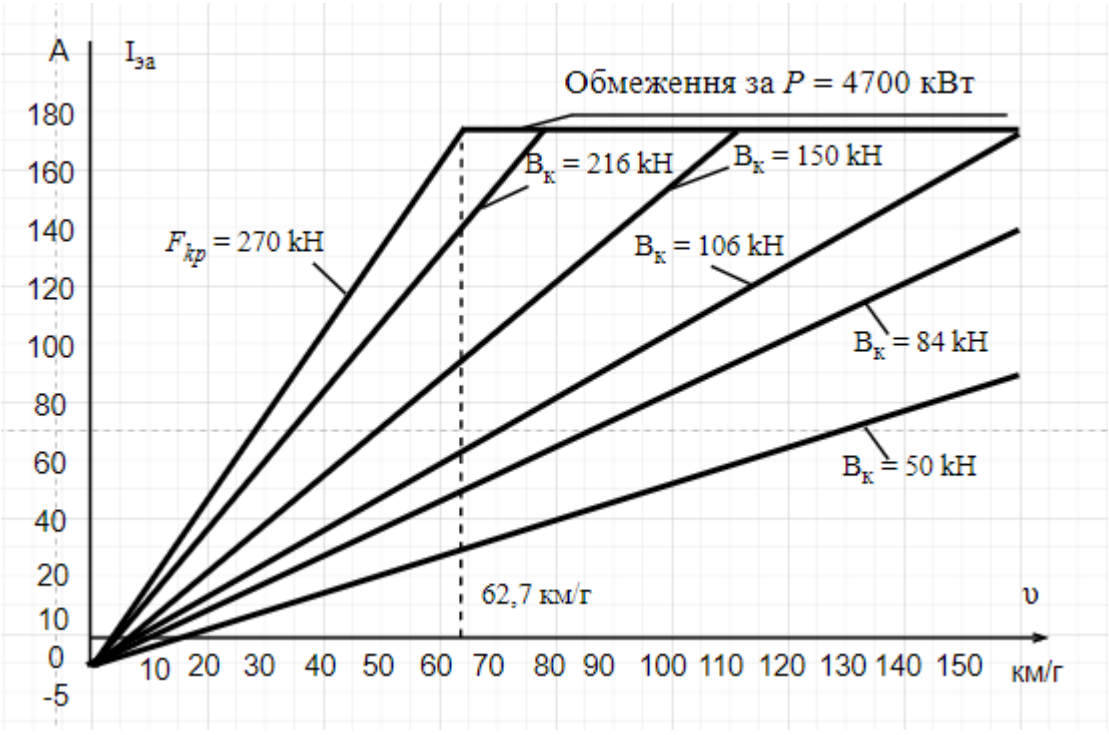


Рисунок 5.4 - Токова характеристика електровозу ДС3 в режимі рекуперації

РОЗДІЛ 6. АЛГОРИТМ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ ЕФЕКТИВНИХ ТЯГОВИХ РОЗРАХУНКІВ.

6.1 Обмежуючі стани поведінки потягу в моделі тяги

Модулювання процесу тяги та руху потягу проводиться в послідовності:

- побудова словесно-описової моделі;
- побудова на її основі математичної моделі, що визначає закон руху;
- дослідження моделі, яка орієнтована на досягнення мети в межах допустимих станів потягу;
- натуральний експеримент, який визначає адекватність моделі оригіналу;
- коректування результатів дослідження та приймання рішення.

Припущення, яке виходить в результаті побудови описової моделі, яка являється основною, згідно якої проводять математичний опис поведінки системи. В теорії тяги потягів прийняті наступні припущення:

1. Механічний рух потяга можна описати математично, як рух математичної точки з одним ступенем свободи (утримуючі зв’язки).

Цілеспрямований рух потягу можна описати одним диференційним рівнянням, як рух матеріальної точки з одним степенем свободи. Це спрощує тягові розрахунки, але вносить деяку помилку.

2. Для пророкування руху потягу достатньо урахувати тільки зовнішні сили, які визначають цілеспрямований рух та співпадають з його напрямом.

В тягових розрахунках ураховуються тільки ті зовнішні сили, які співпадають з напрямком руху потягу. Якщо колінеарні сили приложені до однієї точки, тоді їх можна алгебраїчно складати та розглядати рух системи під дією результуючої сили.

3. Пророкування руху можливо по входам та виходам системи без урахування процесів всередині системи, що дозволяє використовувати в тягових розрахунках статичні характеристики сил тяги, гальмування та опору руху.

Створити типові динамічні характеристики мережі доріг не представляють можливості із-за складної фізичної природи перехідних процесів та залежностей їх від самих різноманітних місцевих умов та організації руху потягів. Тому в тягових розрахунках доводиться використовувати статичні характеристики в якості загальносітьовій апріорній інформації. В тягових розрахунках при зміні позицій контролера та ручки гальмівного крана машиніста або переході центра ваги потяга кордонів елемента профілю різноманітної крутизни умовно приймають стрибкоподібну зміну рівнодіючої сили.

4. Для спрощення тягових розрахунків допустима кусково-лінійна апроксимація неперервної нелінійної функції сил потягу, що дозволяє використовувати принцип суперпозицій та вирішити диференційні рівняння в формі задачі Коши.

Функції сил, визначаючих рух системи, нелінійні, що потребує рішення складних нелінійних диференційних рівнянь. При цьому властивості системи залежать від її поведінки та отже принцип суперпозицій використовувати неможна.

Лінеаризація нелінійної функції проводять в кордонах допустимого, використовують принцип суперпозицій та вирішують диференційні рівняння в формі задачі Коши. Лінеаризація проводиться методами квантування по принципу малих відхилень. При цьому неперервну нелінійну функцію рівнодіючих сил заміняють дискретною кусково-лінійною по малим інтервалам швидкості руху. Тоді в межах кожного інтервалу можна прийняти силу постійною і відповідній середній швидкості в інтервалі. По нормативам ПТР допустима апроксимація по інтервалам швидкості не більше 10 км/год.

Для побудови моделі потягу його можна вважати незмінною системою з одним ступенем свободи, на яку діють тільки зовнішні сили, прикладенні до центру ваги потягу в середині його довжини та співпадаючими з напрямом руху потягу. Якщо сили залежать від швидкості, тоді рух можна пророкувати тільки шляхом рішення диференційних рівнянь. Для потяга з одним ступенем свободи, достатньо одного диференційного рівняння.

В кожному інтервалі швидкості допустимо приймати рівнодіючу силу постійною по значенню, відповідною середовищу арифметичної швидкості в інтервалі та заданому режимі руху в залежності від поєднання сил розрізняють наступні режими руху:

- тяга – взаємодіють сила тяги локомотива і сила опору;
- холостого ходу – діє тільки сила опору руху;
- гальмування – взаємодіють гальмівна сила та сила опору руху.

При зміні режимів або зовнішньому навантаженні передбачається стрибкоподібна зміна рівнодіючої сили. В залежності від відношення сил визначається характер руху потягу. Наприклад, прискорений рух – якщо сила тяги більше сили опору руху, уповільнення – якщо сила тяги менша сили опору , рівномірне – якщо вони рівні між собою.

Потяг володіє властивостями стійкості, тому вони завжди прагнуть до рівномірної швидкості при будь-якому режимі руху.

6.2 Вихідна інформація

В результаті розрахунку необхідно визначити швидкість та час руху, витрату електроенергії, перевищення температури обмоток електричних машин, механічну роботу сил тяги, експлуатаційний к.к.д. локомотива та деякі інші. Проводиться тягово-енергетичні розрахунки на комп’ютері використовується для розробки режимних карт водіння потягів, розрахунок норм витрат електроенергії на тягу.

Сучасні розрахункові засоби дозволяють проводити багатоваріантні розрахунки показників роботі локомотива, моделювання технологічних

процесів водіння потягів з метою оптимізації режимів та вишукування резервів збільшення ефективності тяги.

Для тягово-енергетичних розрахунків необхідно підготовити вихідну інформацію у вигляді дискретного запису в пам'яті комп'ютера. Вихідну інформацію можна розділити на чотири групи: данні про залізничну ділянку, про локомотив, про рухомий склад, довідкові дані.

Інформація про розрахункову ділянку вміщає: данні про спрямлений профіль колії з вказівкою довжини та крутизни елементів профілю; обмеження по швидкості руху; встановленні швидкості руху по перегону; про довготривалі та постійні обмеження швидкості руху з вказівкою координат та довжини ділянки, величини допустимої швидкості; пункти проб гальм (в програмі не передбачено) на ефективність дії з вказівкою довжини до місця перевірки гальм.

Інформація про локомотив вміщає відомості про тягові, токові та теплові характеристики, про допустиму температуру нагріву електричних машин, про к.к.д., про характеристики витрат електроенергії в різноманітних режимах руху, про напругу на контактній мережі.

Інформація про состав включає відомості про основні питомі опори руху вагонів різноманітних типів. Необхідні відомості: про зчіпну вагу, вагу потяга, тип вагонів, про відсоткові відношення типів вагонів в складі, про вагу вагона, віднесеній до однієї колісної пари.

Вся інформація про ділянку, рухомий склад та локомотив в перспективі повинна організовуватися в відповідні бази даних (бібліотеки).

6.3 Розробка алгоритму та програми

В основу тягово-енергетичних розрахунків положено вибір ефективного

(по Парето) режиму управління локомотивом, при якому витрати електроенергії E(S) та часу t(S) визначаються у векторним критеріям

$$min\begin{bmatrix}E(S)\\t(S)\end{bmatrix}, \qquad S \in [0, S_k],$$

S – координата шляху, S_k – довжина перегону для якого проводиться ефективні тягові розрахунки.

Режим управління локомотива (в програмі в якості управління взаємодії взята швидкість руху V(S)), як згадувалось раніше) вибирається для кожного шага інтегрування виходячи із заданих умов руху з урахуванням обмежень, накладається на фазові координати шляху S, швидкості руху V та часу руху t. Допустима область управління (відносна швидкість V) являє собою кусково-лінійну функцію розривами першого роду.

Під раціональним режимом прийнято розуміти такий режим руху, який забезпечує максимальну провізну здатність перегону при мінімум затрат електроенергії на механічну роботу по переміщенню потягу по ділянці за встановленим графіком часу руху.

Під ефективним режимом будемо розуміти такий режим руху, який відповідає непорівнянної по Парето траєкторією руху потягу відносно витрат енергії та часу у розглядає мій координаті траєкторій S.

Основні модулі програми

Математичне забезпечення визначається ефективних траєкторій в задачі тягових розрахунках складається з наступних модулів.

1. Модуль вводу вхідних даних.

Визначається наступні величини, які вводяться:

- ухили даної ділянки колії, задаються послідовністю пар $(\Delta S_j\ ,\ i_j)$,

ΔS_j – довжина j – ї ділянки колії, який має ухил i_j;

- план ділянки шляху, задається послідовністю пар $(\Delta S_j\ ,\ R_j)$, ΔS_j –довжина j – ї ділянки плану шляху, який має радіус кривої R_j;

- допустима швидкість руху по перегону, задаються послідовність пар $(\Delta S_j\ ,V_j)$, ΔS_j – довжина j – ї ділянки відрізку шляху, на якому максимальна швидкість руху потягу визначається величиною V_j;

- довжина рухомого складу, [м];
- вага рухомого складу, [т];
- кількість вагонів в складі;
- середнє навантаження на вісь,[т/вісь].

Програма в представленій редакції виконує ефективні тягові розрахунки пасажирського потягу з електровозом ДС – 3.

2. Модуль констант.

В цьому модулі визначаються величини, які не міняються: вага локомотива, режим руху, різноманітні коефіцієнти та ін.

3. Модуль дискретизації колії.

В цьому модулі виконується розділення ділянки колії на відрізки з урахуванням профілю та плану колії, урахуванням початку та закінчення руху (дискретизація нерегулярна).

4. Модуль дискретизації швидкості руху потягу.

В цьому модулі виконується розкладання допустимих діапазонів швидкості вдаль шляху на допустимі відрізки (дискретизація нерегулярна).

Модуль дискретизації шляху та модуль дискретизації швидкості забезпечує побудову мережі, яка служить для проведення процедури динамічного програмування, формування незрівнянних траєкторій по Парето.

5. Модуль розрахунку середньої швидкості на ділянці шляху, яка розглядається.

6. Модуль визначення допустимої максимальної швидкості в даній координаті шляху S.

7. Модуль визначення часу слідування на ділянці шляху, яка розглядається.

8. Модуль визначення усередненого узагальненого ухилу «профіль-план» (усереднений ухил).

Приведений ухил визначається з урахуванням профілю та ухилу для кожного вагону потягу.

9. Модуль визначення граничної величини тягової характеристики електровоза ДС – 3.

В даному модулі визначається граничне значення тягової характеристики.

10. Модуль визначення граничної величини характеристики рекуперативного гальмування ДС – 3 .

11. Модуль визначення сили струму, яка відповідає поточній швидкості руху потягу та сили тяги.

12. Модуль визначення сили струму в режимі рекуперації в залежності від швидкості руху та сили гальмування.

13. Модуль визначення режиму руху («тяга», «холостий хід», «гальмування»). Вхідними параметрами являється: швидкість руху на початку відрізку, швидкість руху в кінці відрізка, приведений ухил на початку відрізку, приведений ухил в кінці відрізку.

14. Модуль виводу результатів розрахунку.

Функціональні модулі наведено в Додатках 1.

Опис головного модуля

Розглянемо порядок роботи алгоритму ефективних тягових розрахунків. Приведемо етапи алгоритму.

I. Ініціалізація модуля константи.

II. Ввід вхідних даних.

III. Дискретизація шляху та швидкості.

На цьому етапі проходить побудова сітки для виконання процедури динамічного програмування визначення ефективних траєкторій тягових розрахунків.

IV. Ініціалізація початкової точки шляху.

Визначається стан начальної точки, яка відповідає потягу на початку руху. Стан потягу в дискретній координаті задається

наступні атрибути: сила тяги, сила гальмування, тяговий струм (з пантографу), час необхідний для преходу з одного до іншого стану наступної стадії, режим руху.

V. Вважаємо поточну розглядаємо точку шляху з індексом 2(наступна дискретна точка від початкової).

VI. Присвоєння стану (допустимої, не допустимої, присвоєння необхідних атрибутів) поточна точка шляху в залежності від швидкості, яка розглядається.

Точка шляху відповідає поняттю стадії в динамічному програмуванні. В задачі кожної стадії відповідає множені станів.

Для поточної стадії будуються можливі переходи з стану попередньої стадії в допустимий стан поточної. Кожний можливий перехід характеризується вектором («витрачається енергія», необхідний час переходу).

VII. Для кожного стану розглядається стадії, формується множина траєкторії по Парето.

На цьому етапі формується всі можливі зворотні траєкторії у вигляді списку траєкторій для кожного стану (в дискретному динамічному програмуванні це буде вершина графа). Проглядаються всі стани стадії і в кожному стані із сформованого списку зворотних траєкторій, характеризується вектором («енергія», «час»), будується множина Парето. Отримана множина Парето траєкторій закріплюється станом і в подальшому служить в якості вхідних даних в наступній стадії.

Приклад визначення ефективних траєкторій.

Вхідні дані

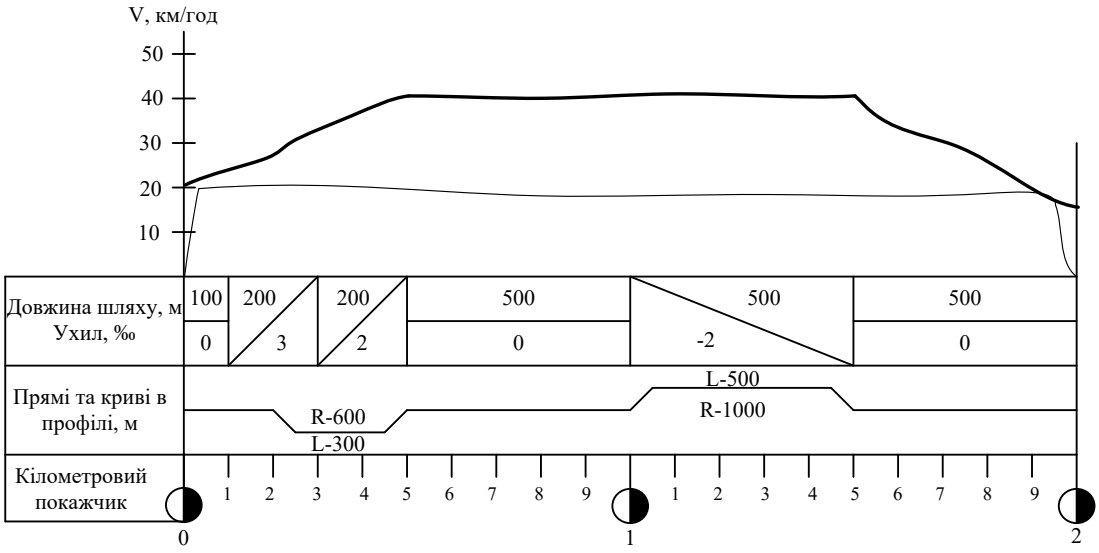


Рисунок 6.1 - Профіль колії

Профіль колії:

> Muklon:=[

#[длина участка (м), уклон на участке (промилле)],

[100, 0],

[200, 3],

[200, 2],

[500, 0],

[500,-2],

[500, 0]

]:

План колії:

Mplan:=[

#[длина участка (м), радиус кривой на участке (м)],

[200, 0],

[300, 600],

[500, 0],

[500, 1000],

[500, 0]

]:

Vlim :=[

```
#[длина участка (м), максимальная скорость на участке (км/ч)],
[ 500, 20],
[1000, 40],
[ 500, 20]
]:
# Количество вагонов в составе
Npas :=  16:
# Средняя нагрузка на ось (т/ось)
q0 := 13.75:
# Нач  и конечная скорость (км/час)
Vstart:=0:   Vstop:=0:
```

Числовий приклад

Временно-энергетические характеристики эффективных траекторий движений

Количество вагонов 16;
Длина поезда, [м] 417.0;
Масса состава, [т] 960.0;
Масса поезда, [т] 1050.0;

Длина пути 2000; Начальная скорость   0;   Конечная скорость   0; Стадий 8

Сетка скорости

0 5 10 15 20 25 30 35 40

Сетка пути

0 100 300 500 1000 1500 1800 2000

Траекторий 13

Траект.	Энергия	Время	Режим	Скоростной режим
1	31.43	9.61	Выбег	0 15 10 20 20 15 10 0
2	31.96	8.85	Выбег	0 20 20 20 20 15 10 0
3	37.41	8.57	Тормоз	0 15 10 20 20 15 15 0
4	37.93	7.81	Тормоз	0 20 20 20 20 15 15 0
5	38.68	7.24	Тормоз	0 20 20 20 20 15 20 0
6	43.08	6.90	Тормоз	0 20 20 20 20 20 20 0
7	43.17	6.80	Тормоз	0 20 20 10 30 20 20 0
8	46.68	6.30	Тормоз	0 20 20 20 30 20 20 0

9	49.85	5.89	Тормоз	0	20	20	20	35	25	20	0
10	51.83	5.73	Тормоз	0	20	20	20	35	30	20	0
11	52.47	5.58	Тормоз	0	20	20	20	40	30	20	0
12	55.15	5.45	Тормоз	0	20	20	20	40	35	20	0
13	67.89	5.35	Тормоз	0	20	20	20	40	40	20	0

Графіки траєкторій

Траектория движения 1

Путь м	Скорость км.час	Время мин	Энергия квт/час	Режим	F тяги кН	I тяги А	B торм кН	I рек. А
0	0	0.00	0.00	Тяга	129.80	210.99	0.00	0.00
100	15	0.80	7.33	Выбег	0.00	0.00	0.00	0.00
300	10	1.76	220	Тяга	109.11	300.08	0.00	0.00
500	20	2.56	181	Тяга	33.62	170.72	0.00	0.00
1000	20	4.06	28.43	Выбег	0.00	0.00	0.00	0.00
1500	15	5.77	28.43	Тормоз	0.00	0.00	6.61	0.00
1800	10	7.21	28.43	Выбег	0.00	0.00	0.00	0.00
2000	0	9.61	28.43	Stop				

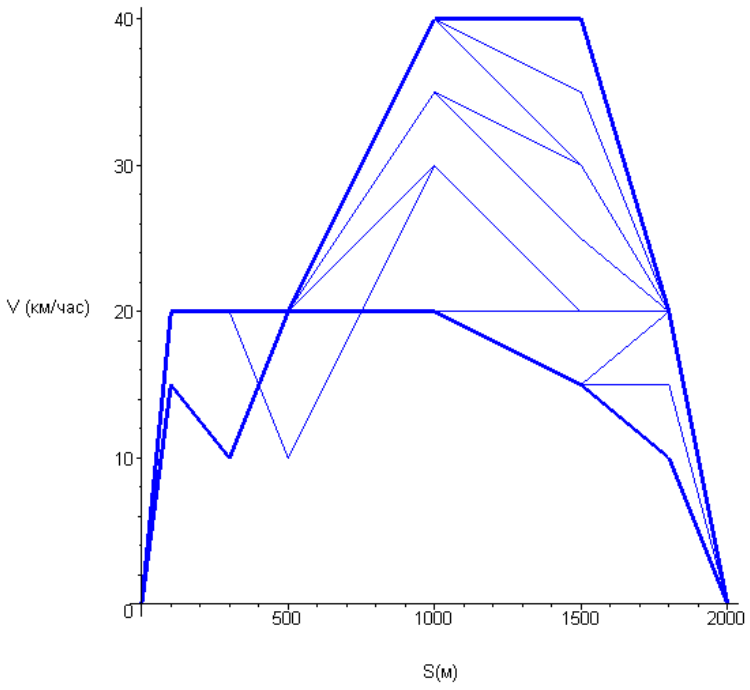


Рисунок 6.2 - Ефективні траєкторії

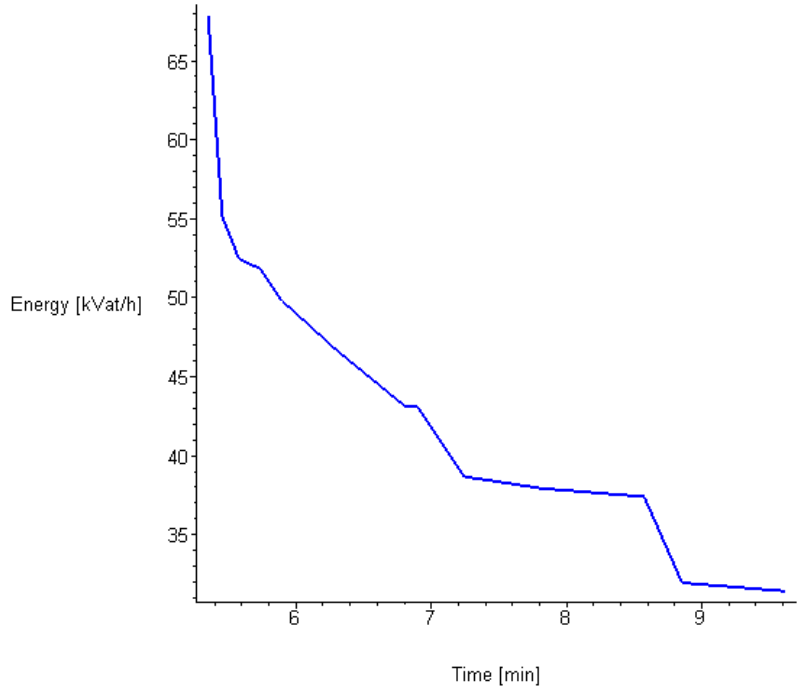


Рисунок 6.3 - Залежність енергії від часу для ефективних траєкторій

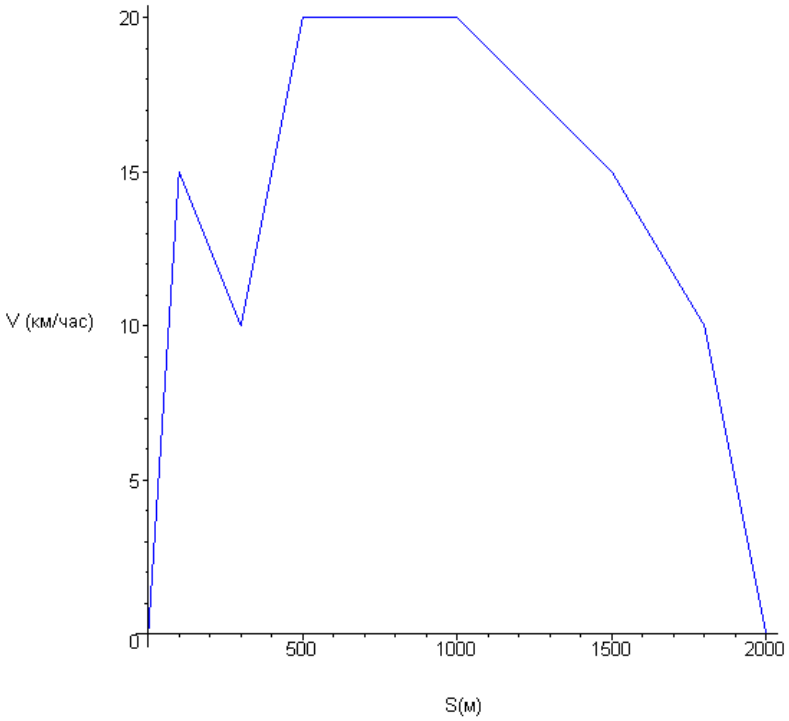


Рисунок 6.4 - Залежність швидкості руху від довжини шляху.

Програмне забезпечення задачі, Maple

```
> restart;
> with(linalg):   with(queue):
> Muklon:=[
#[длина участка (м), уклон на участке (промилле)],
[100, 0],
[200, 3],
[200, 2],
[500, 0],
[500,-2],
[500, 0]
]:

Mplan:=[
#[длина участка (м), радиус кривой на участке (м)],
[200,    0],
[300,  600],
[500,    0],
[500, 1000],
[500,    0]
]:

Vlim :=[
#[длина участка (м), максимальная скорость на участке (км/ч)],
[ 500, 20],
[1000, 40],
[ 500, 20]
]:
```

```
# Длина состава (м), (только состав)
# :=400.0:

# Масса состава (т), (только состав)
# Msost := 880.0:

# Количество вагонов в составе
Npas :=  16:

# Средняя нагрузка на ось (т/ось)
q0 := 13.75:

# Нач  и конечная скорость (км/час)
Vstart:=0:   Vstop:=0:

>
> _Const:=proc()
# Постоянные данные

global DlinaSost, Msost, Npas, q0,
       DlinaLok, Mlok, lVagon, mLok05, mVagon05,
       Mpoezda, DlinaPoezda, P, Q, PQ,
       RegXX, RegTiaga, RegTormoz,
       Zveno, NeStyk, Putx,
       ChugTorm, CompTorm, mPas,
       TetaPas, ProcChug,   ProcComp,
       Auskor, Atorm :

assign('DlinaLok' = 17.):  # [м]
assign('Mlok' = 90.):      # масса локом, (т)
```

```
    assign('lVagon' = 25.):     # длина пасс вагона, [м]
    assign('mLok05' = 45.):     # половина масса локом,
(т)
    assign('mVagon05' = 30.):   # половина масса вагона,
(т)
    assign('DlinaSost' = 0.):   # длина состава(только
вагонов), [м]
    DlinaSost:=Npas*lVagon :
    assign('Msost' = 0.):       # масса состава(только
вагонов), (т)
    Msost:=Npas*2.*mVagon05:
    assign('Mpoezda'=0):   assign('DlinaPoezda'=0):

    assign('XX'    =1):     assign('Tiaga' =2):
    assign('Tormoz'=3):     assign('RegNo' =0):

    assign('Zveno'=1):      assign('NeStyk'=0):
    assign('Putx'=0):       Putx:= NeStyk:

    assign('ChugTorm'=1  ):   assign('CompTorm'=2 ):
    assign('mPas'    =0.6):   assign('TetaPas' =0.6):
    assign('ProcChug'=50 ):   assign('ProcComp'=50):

    Mpoezda := Mlok + Msost:
    DlinaPoezda := DlinaLok + DlinaSost:
    assign('Auskor'=0.35 ):  # assign('Auskor'=0.3 ):
    assign('Atorm'=-1.0 ):   # assign('Atorm'=-0.6 ):

    assign('P'=0. ):   assign('Q'=0. ): assign('PQ'=0.
):
    P := Mlok:   Q := Msost:   PQ := Mpoezda:
```

```
    assign( 'Fnominal'=1. ):  assign('Vnominal'=1. ):
    assign( 'Err'=0 ):

    end proc:
    >
    > _Const():

    > Setap_setV := proc(dv)
    # Построение:
    #     - сетки по скорости, Vset
    #     - разбиение на участки с одинаковыми уклонами,
Suklon, Uklon
    #         - разбиение на участки с одинаковыми
кривизнами, Sradius, Wr

    global Vset,   Nv,
           Vlim,   SVlim,   limV,
           Muklon, Suklon,  Uklon,  Err,
           Mplan,  Sradius, Radius, Wr,  DlinaPoezda:

    local
nvlim,j,Vmax,i,Nuklon,Nradius,indLradius,indRadius:

    Vlim  := convert(Vlim, matrix):
    nvlim                :=                rowdim(Vlim):
assign(SVlim=array(0..nvlim)):
    assign(limV=array(1..nvlim)):        #   ограничение
скорости по участкам

    SVlim[0] := 0:
```

```
for j from 1 to nvlim do
    SVlim[j]:= SVlim[j-1] + Vlim[j,1]:
    limV[j] := Vlim[j,2]:
end do:

for j from 1 to nvlim do
    if irem(limV[j],dv) <> 0 then
       return false:
    end if:
end do:
Vmax:=limV[1]:
for j from 1 to nvlim do
    if Vmax < limV[j] then
       Vmax:=limV[j]
    end if
end do:
assign('Nv'=0):  Nv:=Vmax/dv:
assign(Vset=array(0..Nv)):
for i from 0 to Nv do  Vset[i] := dv*i end do:

Muklon:=convert(Muklon, matrix):
Nuklon := rowdim(Muklon):
assign(Suklon=array(0..Nuklon )):
assign( Uklon=array(1..Nuklon)):    # [ promile ]

Suklon[0] := 0:
for j from 1 to Nuklon do
    Suklon[j]:= Suklon[j-1] + Muklon[j,1]:
    Uklon[j] := Muklon[j,2]:
end do:
```

```
    Mplan:=convert(Mplan, matrix):
    Nradius:= rowdim(Mplan):
    assign(Sradius=array(0..Nradius)):
    assign(Radius=array(1..Nradius)):   # кривизна

    indLradius:=1:  indRadius:=2:
    Sradius[0] := 0:
    for j from 1 to Nradius do
        Radius[j] := Mplan[j,indRadius]:
        Sradius[j]       :=       Sradius[j-1]       +
Mplan[j,indLradius]:
    end do:

    return true:
    end proc:

    > Setap_setS := proc()
    # Построение:
    #    - сетки по пути, Sset
    #      assign(Sset=array(1..Ns)):

    global Vmax,   Vset,    Nv,  Err,
           Vlim,   SVlim,   limV,
           Muklon, Suklon,  Uklon, Sset,
           Mplan,  Sradius, Radius,    DlinaPoezda:

    local  Sn, ds, s,
           stick,
startS0,testS0,finSn0,testSn0,start,fin,
           NS0,NSn,  nvlim,  Nradius,Nuklon,  i,  finS,
d,dd,Ns:
```

```
    Sn:=0: ds:=500:  stick:=200:  Err:=0:

    startS0 :=[0, 20, 50, 100, 300, 500]:
    testS0  :=[0, 100, 300, 500]:
    finSn0  :=[300, 100, 50, 20, 20, 10]:
    testSn0 :=[300, 200]:

    start:=testS0:  fin:=testSn0:  NS0:=vectdim(start):
NSn:=vectdim(fin):

    nvlim  := rowdim(Vlim):
    Nradius:= rowdim(Mplan):
    Nuklon := rowdim(Muklon):

    if ( SVlim[nvlim]=Suklon[Nuklon]
         and Suklon[Nuklon]=Sradius[Nradius] )
    then
        Sn:=Suklon[Nuklon]:
        s:=new(): for i from 1 to  NS0 do  enqueue( s,
start[i] ): end do:
        finS:=Sn - sum('fin[k]','k'=1..NSn):
        d := start[NS0]:  dd := finS - d:
        if dd < 0 then
           Sn:=0:  return Sn:
        end if:
        while  dd  >  ds   do  d:=d+ds:  enqueue(s,  d):
dd:=finS - d: end do:
        if dd >= stick then
           d := d +  dd:  enqueue(s, d):
```

```
            for  i  from  1  to  NSn  do    d:=d+fin[i]:
enqueue(s, d): end do:
         end if:
         if dd > 0 and dd < stick then
            d := d +  dd + fin[1]:  enqueue(s, d):
            if NSn > 1 then for i from 2 to NSn do
                                   d:=d+fin[i]:  enqueue(s,
d): end do:
            end if:
         end if:
         if dd = 0 then
            for  i  from  1  to  NSn  do    d:=d+fin[i]:
enqueue(s, d): end do:
         end if:

     else
         Err:=1:  return -1:
     end if:

     Ns:=length(s):
     assign(Sset=array(1..Ns)):
     # UklonSet := array(0..Ns):
     for i from 1 to Ns do Sset[i]:=dequeue(s): end do:
     return Sn:
     end proc:

     >
     # nvlim  := rowdim(Vlim);  Nradius:= rowdim(Mplan);
Nuklon := rowdim(Muklon);
     > dv:=5:    Setap_setV(dv);
```

true

```
> sn:=Setap_setS();
```

$$sn := 2000$$

```
> V_sredn := proc( v1, v2 )
local v:
v:=(v1+v2)/2.:
return v:
end proc:

>
> Vmax_dopust := proc(s)
# Определение наибольшей допустимой скорости v
# в рассматриваемой точке пути

local  v, m, j:
global SVlim, limV:

# Vlim   := convert(Vlim, matrix): - матрица длин участков с один. скоростью
#   assign(SVlim=array(0..nvlim)):      - границы участков с одинакрвыми скоростями
#  assign(limV=array(1..nvlim)):      - ограничение скорости по участкам

m:=vectdim( limV ):
for j from 1 to m do
    if SVlim[j-1] < s and s <= SVlim[j] then
       v := limV[j]: return v:
    end if:
end do:
end proc:
```

```
>
>t_perehod := proc( ds, v )
local t: t:=ds/(v*1000./3600.): return t:
end proc:

>
>V_dopust := proc( v, s )
# Определение допусти скорости v
# в рассматриваемой точке пути s.
# V_dopust - логическая функция

local  vmax, v_, m, j:
global SVlim, limV:

# Vlim   := convert(Vlim, matrix): - матрица длин участков с один. скоростью
#   assign(SVlim=array(0..nvlim)):      - границы участков с одинакрвыми скоростями
#  assign(limV=array(1..nvlim)):      - ограничение скорости по участкам

# v_ := 1:
m:=vectdim( limV ):
for j from 1 to m do
    if SVlim[j-1] <= s and s <= SVlim[j] then
       vmax := limV[j]: break:
    end if:
end do:
if   v <= vmax then
    return true:
else
```

```
        return false:
    end if:
    end proc:

    >
    > _ik := proc(xx)
    # Усредненный обобщенный "профиль+план"
    #  Строим "уср. профиль" по сетке оптимизации
    # - подсчитываем, задавая x
    # Sset -   сетка по пути,
    #          assign(Sset=array(1..Ns)):

    global Npas,
           Muklon, Suklon,  Uklon,  Sset,
           Mplan,  Sradius, Radius,
           DlinaLok, lVagon, mLok05, mVagon05, PQ:

    local  y, Nradius, Nuklon, iProfil, rPlan, wr,
           Ns, SumUklon, i, j, k, u1, u2, s, u, x :

    Ns:=vectdim( Sset ):  x:=xx;
    Nradius:=    rowdim(Mplan):            Nuklon    :=
rowdim(Muklon):

    s:=new():  enqueue(s, x):           # начало поезда
с локомотива
    x:=x-DlinaLok:   enqueue(s, x):  # конец локомотива
и начало состава

    for j from 1 to Npas do
        x:=x-lVagon:
```

```
        enqueue(s, x):   # концы вагонов в составе
    end do:

    u:=new():   SumUklon:= 0.:
    while not empty(s) do
       y:=dequeue(s):
       if y>0 then
          for i from 1 to Nuklon do
              if   (Suklon[i-1]   <=   y)   and   (y   <=
Suklon[i]) then
                 iProfil := Uklon[i]: break:
              end if:
          end do:
       else
          iProfil:=Uklon[1]:
       end if:

       if y>0 then
          for i from 1 to Nradius do
              if   (Sradius[i-1]   <=   y)   and   (y   <=
Sradius[i]) then
                 rPlan := Radius[i]: break:
              end if:
          end do:
       else
          rPlan:=Radius[1]:
       end if:

    # Vmax_dopust(y):
       wr:=0: if rPlan<>0 then wr:=700./rPlan: end if:
       enqueue(u, iProfil+wr):
```

```
end do:
# s:='s':

u1:=dequeue(u):  u2:=dequeue(u):
SumUklon := SumUklon +(u1+u2)*mLok05:
while not empty(u) do
    u1:=u2:   u2:=dequeue(u):
    SumUklon := SumUklon +(u1+u2)*mVagon05:
end do:

SumUklon := SumUklon/PQ:
return SumUklon:

end proc:

> Ns:=vectdim(Sset):

> _Fk := proc(v)
# Тяговая характеристика ДС3
# Определение граничного значения силы тяги
local f:
if v <= 62.7 then
    f := -0.638*v+310.0:
else
    f := 16929.0/v:
end if:
return f:
end proc:

>
```

```
> _Bk := proc(v)
# Хар-ка рекуперативного торможения ДС3
# Опред B максимального для  заданного v
local B:
if (v < 3.5) then B := 0.0: return B: end if:
if (3.5 <= v) and (v <=  5.5) then
   B := 108.*v - 378.:  return B: end if:
if 5.5  <  v and v <= 78.8 then
   B := 216.:  return B:
else
   B := 17020.08/v:  return B:
end if:
end proc:

>
> _k:=proc(x2,y2)
# Тангенсы наклонов токовой хар-ки
local x1, y1, k;
x1:=0.: y1:=10.:
k:=(y2-y1)/(x2-x1):
return k:
> end proc:

> # _k(62.7,220);
> yk:=[0.58333, 1.30769, 1.83333, 2.01149, 2.4982,
3.1250, 3.34928]:
> xfk:=[50, 106, 150, 161, 200, 250, 270]:
>  plot([[50,  0.58333],  [106,  1.30769],  [150,
1.83333], [161, 2.01149],
```

```
          [200,   2.4982],   [250,   3.1250],   [270,
3.34928]], style=point):

    > _Etta:=proc(f,v)
    local Fkzv, vzv:
    global Fnominal, Vnominal, Mpoezda:
    return 1.:  # zaglushka

    vzv:=v/Vnominal:
    Fkzv:=f*9.81*Mpoezda/Fnominal:
    # тут определяется этта (Fkzv,vzv)
    end proc:

    > A_tiaga:=proc(v,Fk)
    # Токовая хар-ка
    local k, A:
    # Коэфф хар-ки
    k:=-0.03841151  : (см оригинал)
    # Строим (находим) саму хар-ку (прямую)
    A := k*v+10.: if A>220. then A:=220. end if: return
A:
    end proc:
    >
    > # A_tiaga(100,100);

    > A_recup:=proc(v,Bk)
    # Токовая хар-ка
    local k, A:
    # Коэфф хар-ки
    k:=0.01457228438 : (см оригинал)
```

```
    # Строим (находим) саму хар-ку (прямую)
    A := k*v-5.:  if A>172. then A:=172. end if: return
A:
    end proc:
    >
    _Regim:=proc( V1, V2, ds, Uklon1, Uklon2 )

    local Vsr, Vsr2, W01, Wx, W02, W0x, W0,
          fy, fk, bk, Regim, SrednUklon,
          dXX, fyXX, dW025, eps_fy, dW0, cntXX :

    global DlinaSost, Msost, Npas, q0,
           Mpoezda, DlinaPoezda,
           P, Q, PQ,
           XX,  Tiaga, Tormoz, RegNo,
           Zveno, NeStyk, Putx,
           ChugTorm, CompTorm, mPas,
           TetaPas, ProcChug,   ProcComp,
           Auskor, Atorm,
           DlinaLok, Mlok:

    Vsr:= (V1+V2)/2.:  Vsr2:=Vsr*Vsr:  eps_fy:=0.15:

    if Putx = Zveno then
              W01:=     1.9+0.010*Vsr+0.00030*Vsr2:
              Wx :=     2.4+0.011*Vsr+0.00035*Vsr2:
              W02:=0.7+(8.0+0.180*Vsr+0.00030*Vsr2)/q0:
           else    #  Putx = NeStyk
              W01:=     1.9+0.008*Vsr+0.00025*Vsr2:
              Wx :=     2.4+0.009*Vsr+0.00035*Vsr2:
              W02:=0.7+(8.0+0.160*Vsr+0.00230*Vsr2)/q0:
```

```
            end if:

    #            W02:=1.2+0.012*Vsr+0.002*Vsr2:  #  from
Kokurin

    # ... плюс подвагон генераторы
    if Vsr > 20 then
       W02:= W02+136.*9./(q0*Vsr)
    end if:

    # correction...
    if Vsr < 20 then
       W02:= W02*35./(15.+Vsr):
    end if:
    # при трогании...
    if V1 = 0 then
       W02:= 28./(7.+0.1*q0):
    end if:

    W0:=       (W01*P+W02*Q)/Mpoezda:                W0x:=
(Wx*P+W02*Q)/Mpoezda:

    SrednUklon:= (Uklon1+Uklon2)/2.:
    fy:= 500.*(V2*V2-V1*V1)/( 120.*ds ):
    fk:= fy+W0+SrednUklon:  bk:= -(fy+W0x+SrednUklon):

    if fk<=0 and bk<=0 then
       Regim:= XX:
       return Regim, fk, bk:
    end if:
```

```
    dW025:=  abs(  W0x-(Wx*P+W02*(Q+25.))/(Mpoezda+25.)
):
    dXX:= abs(W0-W0x):
    fyXX:=-(W0x+SrednUklon):
    cntXX:=0:

    if abs( (fy-fyXX)/fyXX ) <= eps_fy then
       Regim:= XX:
       return Regim, fk, bk:
    end if:

    if    (    abs(    bk    )<=0.5+dW025    )    and
(abs(fk)<=0.5+dW025+dXX )  then
       Regim:= XX:
       return Regim, fk, bk:
    end if:

    if fk> 0 and bk< 0 then
       Regim:= Tiaga:
       return Regim, fk, bk:
    end if:

    if fk< 0 and bk> 0 then
       Regim:= Tormoz:
       return Regim, fk, bk:
    end if:
     Regim:= XX:
       return Regim, fk, bk:

    end proc:
```

```
>
>GoV1V2 := proc( V1, V2, ds, Uklon1, Uklon2 )
#
# ds [m]
# V1, V2 [км\ч]
#
global P, Q, PQ, Mpoezda,
       XX, Tiaga, Tormoz, RegNo,
       Zveno, NeStyk, Putx,
       mPas, ProcChug, ProcComp, TetaPas,
       Auskor, Atorm,
       DlinaLok, Mlok:

local a, t, v, go,
      Bmax, Bt, Fk, Fmax,
      regim, fk, bt, FiKr :

v:=(V1+V2)/2.:  t:=ds/(v*1000./3600.):
a:=((V2-V1)*1000./3600.)/t:
go:=false:

if a >= 0 and a <= Auskor  then go:=true: end if:
if a <  0 and a >= Atorm   then go:=true: end if:

if go then
   regim,  fk,  bt  :=_Regim(  V1,  V2,  ds,  Uklon1,
Uklon2):

   if  regim = Tiaga  then
       Fk:= fk * Mpoezda*9.81/1000.:
       Fmax := _Fk(v):
```

```
            if Fk > Fmax then
               return RegNo, fk, 0., 0., 0:
            else
               return Tiaga, fk, 0., A_tiaga(v,Fk), t:
            end if:
         end if:

         if  regim = Tormoz  then
            Bt:= bt * Mpoezda*9.81/1000.:
            if  v>20  then                          #   торм
рекуперацией
               Bmax := _Bk(v):
               if Bt > Bmax then
                  return RegNo, 0., Bt, 0., t:
               else
                  return  Tormoz,  0.,  Bt,  A_recup(v,Bt),
t:
               end if:
            else          # v<20- электропневмотормоз
               # Коэфф трения
               FiKr:=(
ProcChug*0.27*(v+100.)/(5.*v+100.)+
          ProcComp*0.36*(v+150.)/(2.*v+150.) )/100.:
               Bmax:= 1000*mPas*TetaPas*FiKr :
               if bt > Bmax then
                  return RegNo, 0., Bt, 0., t:
               else
                  return Tormoz, 0., Bt, 5., t:
               end if:
            end if:
         end if:
```

```
    if  regim = XX then
        return XX, 0., 0., 0., t:
    end if:

    return RegNo, 0., 0., 0., 0:
end if:

return RegNo, 0., 0., 0., 0:
end proc:

> _RegTxt := proc(cod)
local ans:
if cod=1 then ans:=`Выбег `: return ans: end if:
if cod=2 then ans:=`Тяга  `: return ans: end if:
if cod=3 then ans:=`Тормоз`: return ans: end if:
end proc:

> # reg, Fk, Bt, A, t := GoV1V2(90, 86.8, 500, 0.,
0.);

>
> DlinaLok, Npas;
                          17., 16
>  sn;
                          2000
> # Err:=0:
sErr:=0:  ErrStage:=0:  cnt:=0:
S1:0:  S2:=0:  uklonS1:=0:  uklonS2:=0:
dA:=0:
>
```

```
    # Основная задача
    #================================================
    #
    # Vset(i), i=0..Nv
    # Sset(i), i=1..Ns
      sost   := table([(regim_)=1, (Fk_)=2., (Bt_)=3.,
(I_)=4.,
                       (dt_)=5., (pnt_)=0, (dA_)=0. ]):
    # sost0  := table([(regim_)=0, (Fk_)=0., (Bt_)=0.,
(I_)=0.,
    #                 (dt_)=0., (pnt_)=0, (dA)=0. ]):
    # z       := table([(regim_)=1, (Fk_)=2., (Bt_)=3.,
(I_)=4.,
    #                 (dt_)=5., (pnt_)=0 ]):

    Stage1 := array(0..Nv):  Stage2 := array(0..Nv):
    Sled1  := array(0..Nv):  Sled2  := array(0..Nv):

    for j from 0 to Nv do
        Stage1[j] := new():
        Stage2[j] := new():
        Sled1[j]  := new():
        Sled2[j]  := new():
    end do:

    V1dostup:=array[0..Nv]:   V2dostup:=array[0..Nv]:

    for j from 0 to Nv do
        V1dostup[j]:=false:  V2dostup[j]:=false:
    end do:
```

```
Ns:=vectdim(Sset):

#   Установить нач вершину (скорость)
#   для нулевой стадии
iVstart:=Vstart/dv:  V1dostup[iVstart]:=true:

S1:=Sset[1]:  uklonS1:=_ik(S1):

enqueue( Stage1[iVstart], Tiaga ):   # regim
enqueue( Stage1[iVstart], 0.0 ):     # Fk
enqueue( Stage1[iVstart], 0.0 ):     # Bt
enqueue( Stage1[iVstart], 0.0 ):     # amper
enqueue( Stage1[iVstart], 0.0 ):     # dt
enqueue( Stage1[iVstart], iVstart):  # pnt
enqueue( Stage1[iVstart], 0   ):     # En
enqueue( Stage1[iVstart], 0   ):     # Tim

  # sost[regim_] := regim:
  # sost[Fk_ ] := Fk:
  # sost[Bt_ ] := Bt:
  # sost[I_  ] := amper:
  # sost[dt_ ] := dt:
  # sost[pnt_] := iv1:
  # sost[SumE_]:= En:
  # sost[SumT_]:= Tim:

for i from 0 to Nv do
      enqueue( Sled1[i], i ):
end do:

>
```

```
for iSset from 2 to Ns do
# формируем доступ к вершинам текущей стадии
if Err>0 then break: end if:
ds := Sset[iSset] - Sset[iSset-1]:
S1:=Sset[iSset-1]:  S2:=Sset[iSset]:
uklonS1:=_ik(S1):  uklonS2:=_ik(S2):

if iSset < Ns then
V2dostup[0]:=false: # Двиг без остановки
   for i from 1 to Nv do
       if V_dopust(Vset[i], Sset[iSset]) then
          V2dostup[i]:=true:
       else
          V2dostup[i]:=false:
       end if:
   end do:
else # iSset = Ns ...-конечная точка
     for i from 0 to Nv do
         V2dostup[i]:=false:
     end do:
     iVstop:=Vstop/dv:   V2dostup[iVstop]:=true:
end if:

cnt:=0;
# строим граф для текущей стадии
for iv2 from 0 to Nv do
    if V2dostup[iv2] then
       for iv1 from 0 to Nv do  # смотрим вершины
предыдущей стадии
            if V1dostup[iv1] then
```

```
                    #   print(   iv1,   iv2,   Vset[iv1],
Vset[iv2]):
                    regim, fk, Bt, amper, dt :=
                    GoV1V2(  Vset[iv1],  Vset[iv2],  ds,
uklonS1, uklonS2 ):
                    # print( regim, fk, Bt, amper, dt,
"\n" );
        if regim <> RegNo then   # пишем режим движения
                       cnt:=cnt+1:
                       dA:=0.0069444*amper*dt:
                       if regim = Tiaga then
                          v_:=V_sredn(          Vset[iv1],
Vset[iv2] ):
                          dA:=dA/_Etta(fk,v_):
                       end if:
                       # sost[regim_] := regim:
                       # sost[Fk_ ] := fk:
                       # sost[Bt_ ] := Bt:
                       # sost[I_  ] := amper:
                       # sost[dt_ ] := dt:
                       # sost[pnt_] := iv1:
                       # sost[dA_ ] := dA:
                       enqueue( Stage2[iv2], regim ):
                       enqueue( Stage2[iv2], dA    ):
                       enqueue( Stage2[iv2], Bt    ):
                       enqueue( Stage2[iv2], amper ):
                       enqueue( Stage2[iv2], dt    ):
                       enqueue( Stage2[iv2], iv1   ):
                       # print( sost );
                    end if:
```

```
                end if:
            end do: # смотрим вершины предыдущей стадии
        end if:
    end do: # строим граф для текущей стадии
    # printf("====================== \n"):

    if    cnt=0    then    Err:=2:    ErrStage:=iSset:
sErr:=sSet[iSset]: break:  end if:

    # формируем маршруты по Парето для iv2 текущей
стадии
    for iv2 from 0 to Nv do
          # printf("%30s\n",`-------------------------
-----------------------------`):
          # print('Вершина ', iv2):

           if length(Stage2[iv2]) <> 0 then # Есть
ребра в iv2 ?
              m2:=length(Stage2[iv2])/6:

              EnTimBack:= new():  BackPath:= new():

              # смотрим все входящие ребра
              # из вершин первой стадии
              # в вершину второй стадии iv2
              while not empty(Stage2[iv2]) do
                    # читаем одно ребро
                    m2Reg:= dequeue( Stage2[iv2] ):
                    m2Fk := dequeue( Stage2[iv2] ):
                    m2Bt := dequeue( Stage2[iv2] ):
                    m2Amp:= dequeue( Stage2[iv2] ):
```

```
                    m2Dt := dequeue( Stage2[iv2] ):
                    m2Pnt:= dequeue( Stage2[iv2] ):

                    # sost[regim_] := m2Reg:
                    # sost[Fk_ ] := m2Fk:
                    # sost[Bt_ ] := m2Bt:
                    # sost[I_  ] := m2Amp:
                    # sost[dt_ ] := m2Dt:
                    # sost[pnt_] := m2Pnt:
                    # print(sost);
                    if length(Stage1[m2Pnt]) <> 0 then

                        Back:=  new():      Rebra1  :=
new():

                       # читаем ссылки в вершине  m2Pnt
по Парето
                       # предыдущей стадии
                       m1:=length(Stage1[m2Pnt])/8:    #
к-во ссылок назад

                       #  смотрим  все  обратн  пути  по
Парето в вершине m2Pnt
                       # из первой стадии
                       while  not  empty(Stage1[m2Pnt])
do
                             # ...  читаем  одну  ссылку
(маршрут)
                             m1Reg:=           dequeue(
Stage1[m2Pnt] ):
```

```
                                m1Fk      :=      dequeue(
Stage1[m2Pnt] ):
                                m1Bt      :=      dequeue(
Stage1[m2Pnt] ):
                                m1Amp:=           dequeue(
Stage1[m2Pnt] ):
                                m1Dt      :=      dequeue(
Stage1[m2Pnt] ):
                                m1Pnt:=           dequeue(
Stage1[m2Pnt] ):
                                m1En      :=      dequeue(
Stage1[m2Pnt] ):
                                m1Tim:=           dequeue(
Stage1[m2Pnt] ):

                                 enqueue( Rebra1, m1Reg ):
                                 enqueue( Rebra1, m1Fk  ):
                                 enqueue( Rebra1, m1Bt  ):
                                 enqueue( Rebra1, m1Amp ):
                                 enqueue( Rebra1, m1Dt  ):
                                 enqueue( Rebra1, m1Pnt ):
                                 enqueue( Rebra1, m1En  ):
                                 enqueue( Rebra1, m1Tim ):

                                 En:=  m1En+m2Fk:    Tim:=
m1Tim + m2Dt:

                                 enqueue( EnTimBack, m2Reg
):
                                 enqueue( EnTimBack, m2Fk
):
```

```
                                enqueue( EnTimBack, m2Bt
):
                                enqueue( EnTimBack, m2Amp
):
                                enqueue( EnTimBack, m2Dt
):
                                enqueue( EnTimBack, m2Pnt
):
                                enqueue( EnTimBack, En
):
                                enqueue( EnTimBack, Tim
):

                                # выбираем маршрут (след)
для данного пути
                                for i from 1 to iSset-1
do
                                    m:=          dequeue(
Sled1[m2Pnt] ):
                                    enqueue( BackPath, m
):
                                    enqueue( Back, m ):
                                end do:
                                enqueue( BackPath, m2Pnt
):

                                # Надо брать для рассм-й
верш пред стадии
                                # следуюж обратн путь по
Парето
```

```
                        end   do:       #   while   not
empty(stage1[m2Pnt])
                                  # просмотрели все пути
...
                                  # надо взять следующее
ребро (вершину)
                                  # из первой стадии

               #  Восстановим  след  для  вершины  m2Pnt
стадии 1
                      # Восстановим маршрут
                      while not empty(Back) do
                            m:= dequeue( Back ):
                            enqueue( Sled1[m2Pnt], m ):
                      end do:
                      # Восстановим ребра
                      while not empty(Rebra1) do
                            m:= dequeue( Rebra1 ):
                            enqueue( Stage1[m2Pnt], m ):
                      end do:

                      end if: # length(stage1[m2Pnt]) <>
0 then

              end do:   # смотрим все входящие ребра в
iv2  из стад 1
                        # while not empty(stage2[iv2])
                        #... просм-ли вершину iv2

    #-- тут аналих на парето списка EnTinBack и запись
    # маршруты хранятся в BackPath
```

```
#>>>>>>>>>>>>>>>>>>>>  <<<<<<<<<<<<<<<<<<
```

```
    # к-во маршрутов в вершине
    m:= length(EnTimBack)/8:

    s:=array[1..m, 1..iSset]:
    reg:=array[1..m]:
    fk :=array[1..m]:
    bt :=array[1..m]:
    amp:=array[1..m]:
    dtm:=array[1..m]:
    pnt:=array[1..m]:
    en :=array[1..m]:
    tim:=array[1..m]:

ii:=1:
while not empty(EnTimBack) do
# читаем текущее ребро
      # m2Reg:=dequeue( EnTimBack ):
      # m2Fk :=dequeue( EnTimBack ):
      # m2Bt :=dequeue( EnTimBack ):
      # m2Amp:=dequeue( EnTimBack ):
      # m2Dt :=dequeue( EnTimBack ):
      # m2Pnt:=dequeue( EnTimBack ):
      # En   :=dequeue( EnTimBack ):
      # Tim  :=dequeue( EnTimBack ):

      reg[ii]:=dequeue( EnTimBack ):
      fk[ii] :=dequeue( EnTimBack ):
      bt[ii] :=dequeue( EnTimBack ):
      amp[ii]:=dequeue( EnTimBack ):
```

```
            dtm[ii]:=dequeue( EnTimBack ):
            pnt[ii]:=dequeue( EnTimBack ):
            en[ii] :=dequeue( EnTimBack ):
            tim[ii]:=dequeue( EnTimBack ):

        # читаем текущий маршрут
            for i from 1 to iSset do
                s[ii,i]:= dequeue( BackPath ):
            end do:
            ii:=ii+1:
        end do:
        ii:=ii-1:  # print(ii); print(m);

    # Тут можно исключить все маршруты t> T

        # выбор несравнимых по Парето вариантов
        if m > 1 then   # в вершине более одного
маршрута

        #               Прямой путь
        for iv from 1 to m-1 do
            for j from iv+1  to ii do

                Фильтр Парето

            end do:
        end do:

        #             Обратный путь
        for iv from m by -1  to 2 do
```

```
                Фильтр Парето

            end do:

            end if:    # m > 1

            # сформировали маршруты по парето
            # или содержится в вершине один маршрут.

            # теперь список маршрутов надо сохранить
            # Stage2[iv2]:= new():  Sled[iv2]:= new:
            clear(Stage2[iv2]):  clear(Sled2[iv2]):

            # Формируем маршруты вершины
            for iv from 1 to m do
                Формируем (см оригинал)
            end do:

            # удалить рабочие массивы
            reg:='reg':     fk   :='fk'   :     bt:='bt':
amp:='amp':
            dtm:='dtm':      pnt:='pnt':      en:='en':
tim:='tim':  s:='s':

          end if:  # length(Stage2[iv2]) <> 0

     end do:   # for iv2 --формируем маршруты Парето
текущей стадии (2)

    # текущая стадия становится предыдущей.
    # Меняем след1 на след2 и Stage1 на Stage2
```

```
      for j from 0 to Nv do
         clear(Stage1[j]):  Stage1[j]:= copy(Stage2[j]):
         clear(Stage2[j]):
         clear(Sled1[j]) :  Sled1[j] := copy(Sled2[j]) :
         clear(Sled2[j]) :
         V1dostup[j]:=V2dostup[j]:
       end do:
     # конец стадии.
     # можно смотреть результат по текущей стадии

     # >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     # Просмотр переходов в вершинах (отладка)
         FlgPrint:=false:
         if FlgPrint then
         printf("Стадия %2d \n", iSset):
         for i from 0 to Nv do
             printf("Вершина   %3d            Маршрутов%3d
Следов%3d\n",
                    i,              length(Stage1[i])/8,
length(Sled1[i])/iSset ):
             if length(Stage1[i]) > 0 then
                 Back:= new():   Rebra1:= new():
                 Rebra1:=                copy(Stage1[i]):
Back:=copy(Sled1[i]):
                 while not empty( Rebra1 ) do

                   # m2Reg:=dequeue( EnTimBack ):
                   # m2Fk :=dequeue( EnTimBack ):
                   # m2Bt :=dequeue( EnTimBack ):
                   # m2Amp:=dequeue( EnTimBack ):
```

```
                    # m2Dt :=dequeue( EnTimBack ):
                    # m2Pnt:=dequeue( EnTimBack ):
                    # En   :=dequeue( EnTimBack ):
                    # Tim  :=dequeue( EnTimBack ):
                    printf("  %7.2f  %7.2f  %7.2f  %7.2f%
7.2f %6d %7.2f %7.2f ",
                            dequeue(Rebra1),
dequeue(Rebra1),
                            dequeue(Rebra1),
dequeue(Rebra1),
                            dequeue(Rebra1),
                            dequeue(Rebra1),
                            dequeue(Rebra1),
dequeue(Rebra1) ):
                    printf("      Путь "):
                     for j from 1 to iSset do
                        printf("%3d", dequeue(Back) ):
                    end do:
                    printf("\n"):
                  end do:
               end if:
          end do:
          printf("%30s\n",`-----------------------------
---------------------------`):
          end if:  # FlgPrint

     end do:  # for iSset from 2 to 2 do ...to Ns do

     > if Err>0 then
        Err; ErrStage;  sErr;
     end if;
```

```
>
> # Nv, Ns; iVstop, dv, V2dostup[iVstop];

> # >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# Результаты в вершине iVstop последней стадии
    printf("\n"):
    printf("Временно-энергетические  характеристики
режимов движений\n"):
    printf("\n"):

    printf("Количество вагонов %2d;  \n", Npas):
    printf("Длина  поезда,    [м]  %6.1f;    \n",
DlinaPoezda):
    printf("Масса состава, [т] %6.1f;  \n", Msost):
    printf("Масса  поезда,    [т]  %6.1f;    \n",
Mpoezda):
    printf("\n"):
    printf("Длина  пути  %7d;   Начальная  скорость
%3d;   Конечная скорость %3d;   Стадий%3d  \n",
            Sset[Ns], dv*iVstart, dv*iVstop, Ns):

    printf("Сетка скорости  \n"):
    for j from 0 to Nv do
        printf("%4d ", Vset[j] ):
    end do:
    printf("\n"):

    printf("Сетка пути  \n"):
    for j from 1 to Ns do
        printf("%7d ", Sset[j] ):
```

```
        end do:
        printf("\n"):

        i:=iVstop:
        Nputx:=   length(Stage1[i])/8:    # Маршрутов
        Nsledov:= length(Sled1[i])/Ns:    # Следов
        printf("Режимов движений%4d    \n", Nputx ):

        printf("\n"):  printf("\n"):
        if length(Stage1[i]) > 0 then
           Back:= new():  Rebra1:= new():
           Rebra1:=                    copy(Stage1[i]):
Back:=copy(Sled1[i]):

           m:=Nputx:                                  # к-во
маршрутов в конечной вершине
           # sledPutxV:=array[1..m, 1..Ns]:
           assign( sledPutxV=array(1..m, 1..Ns)):
           regEndPutx:=array[1..m]:
               # fkPutx :=array[1..m]:
               # btPutx :=array[1..m]:
               # ampPutx:=array[1..m]:
               # dtmPutx:=array[1..m]:
               # pntPutx:=array[1..m]:
           EnPutx  :=array[1..m]:
           TimPutx :=array[1..m]:

           printf("Движение   Энергия    Время    Режим
Скоростной режим\n" ):
           iPutx:=1:  # номер маршрута
           while not empty( Rebra1 ) do
```

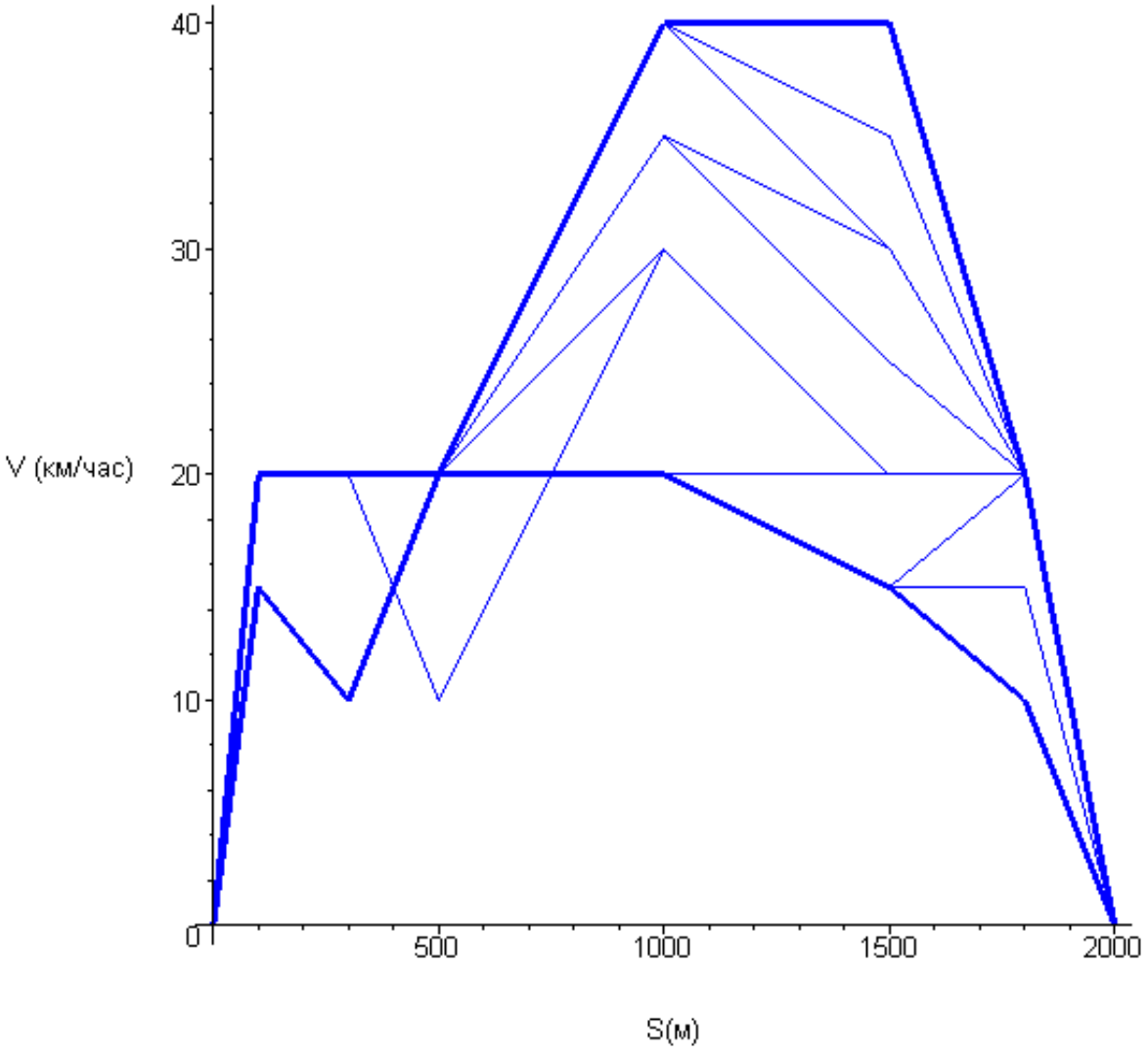
```
                    # m2Reg:=dequeue( EnTimBack ):
                    # m2Fk :=dequeue( EnTimBack ):
                    # m2Bt :=dequeue( EnTimBack ):
                    # m2Amp:=dequeue( EnTimBack ):
                    # m2Dt :=dequeue( EnTimBack ):
                    # m2Pnt:=dequeue( EnTimBack ):
                    # En   :=dequeue( EnTimBack ):
                    # Tim  :=dequeue( EnTimBack ):

                    regEndPutx[iPutx]:= dequeue(Rebra1):
                    dequeue(Rebra1):      dequeue(Rebra1):
dequeue(Rebra1):
                    dequeue(Rebra1): dequeue(Rebra1):
                    EnPutx[iPutx]:=       dequeue(Rebra1):
TimPutx[iPutx]:= dequeue(Rebra1):
                    printf(" %3d      %7.2f   %7.2f    %6s
",
                            iPutx,          EnPutx[iPutx],
TimPutx[iPutx]/60., _RegTxt(regEndPutx[iPutx]) ):
                    # Путь
                    dequeue(Back):             # Пропустим
повтор начальной вершины
                    for j from 1 to Ns-1 do  # ...читаем
далее
                        sledPutxV[iPutx,j]:=
dv*dequeue(Back):
                        printf("%4d",  sledPutxV[iPutx,j]
):
                    end do:
```

```
                    sledPutxV[iPutx,      Ns]:=dv*iVstop:
printf("%4d", sledPutxV[iPutx,j] ):
                    iPutx:=iPutx+1: printf("\n"):
            end do:
        end if:
        printf("\n"):
        printf("%30s\n",`-----------------------------
---------------------------`):
```

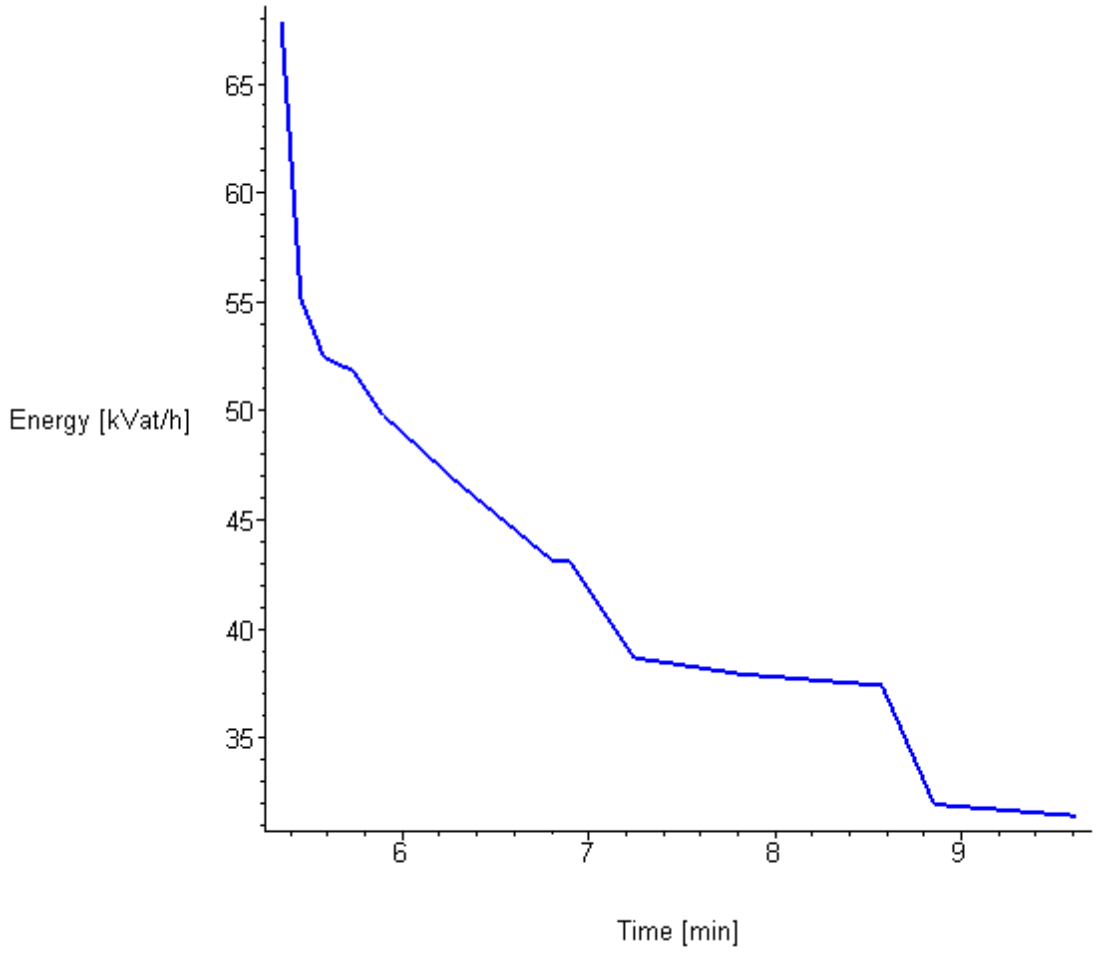
```
Временно-энергетические характеристики режимов движений

Количество вагонов 16;
Длина поезда,  [м]  417.0;
Масса состава, [т]  960.0;
Масса поезда,  [т] 1050.0;

Длина пути    2000;  Начальная скорость   0;   Конечная скорость   0;
Стадий  8
Сетка скорости
   0    5   10   15   20   25   30   35   40

Сетка пути
      0     100     300     500    1000    1500    1800    2000

Режимов движений  13


Движение  Энергия   Время   Режим      Скоростной режим
   1       31.43     9.61   Выбег      0  15  10  20  20  15  10   0
   2       31.96     8.85   Выбег      0  20  20  20  20  15  10   0
   3       37.41     8.57   Тормоз     0  15  10  20  20  15  15   0
   4       37.93     7.81   Тормоз     0  20  20  20  20  15  15   0
   5       38.68     7.24   Тормоз     0  20  20  20  20  15  20   0
   6       43.08     6.90   Тормоз     0  20  20  20  20  20  20   0
   7       43.17     6.80   Тормоз     0  20  20  10  30  20  20   0
   8       46.68     6.30   Тормоз     0  20  20  20  30  20  20   0
   9       49.85     5.89   Тормоз     0  20  20  20  35  25  20   0
  10       51.83     5.73   Тормоз     0  20  20  20  35  30  20   0
  11       52.47     5.58   Тормоз     0  20  20  20  40  30  20   0
  12       55.15     5.45   Тормоз     0  20  20  20  40  35  20   0
  13       67.89     5.35   Тормоз     0  20  20  20  40  40  20   0

------------------------------------------------------------
```

```
> # P  := [[TimPutx[n]/60.,EnPutx[n]] $n=1..Nputx]:
```

```
> # plot(P, style=line, symbol=asterisk, color=blue,
labels=["Time [min]", "Energy [kVat/h]"]);
> # ListPutx:=[1,2,3,4,5,6,7,8,9,10,11,12,13]:
> _Vplots:=proc(ListPutx)
local mPlotPutx, i, n, m, L, pPlot, PutxV,
      minPlot, maxPlot, a, colPutx, rowPutx;
PutxV:=  copy(sledPutxV):
mPlotPutx:=vectdim(ListPutx):
PutxV:=            convert(PutxV,            array):
colPutx:=coldim(PutxV):  rowPutx:=rowdim(PutxV):
pPlot:=array[1..mPlotPutx]:
i:=1:
for m in ListPutx do
   L:=[ [Sset[n],sledPutxV[m,n]]$n=1..colPutx ]:
   pPlot[i]:=plot(L,    style=line,    thickness=1,
color=blue):
   i:=i+1:
end do:
L:=[ [Sset[n],PutxV[1,n]]$n=1..colPutx ]:
minPlot:=plot(L,      style=line,      thickness=3,
color=blue, labels=["S(м)", "V (км/час)"]):
L:=[ [Sset[n],PutxV[rowPutx,n]]$n=1..colPutx ]:
maxPlot:=plot(L,      style=line,      thickness=3,
color=blue):
a:=plots[display]([minPlot,              maxPlot],
pPlot[n]$n=1..mPlotPutx):
return a:
end proc:
> a:=_Vplots([1,2,3,4,5,6,7,8,9,10,11,12,13]):  a;
```




```
> _plotEnergyTime:=proc(TimePutx, EnergyPutx)
  local n, Nputx, P:
  TimePutx:= convert(TimePutx, array): EnergyPutx:=
convert(EnergyPutx, array):
  Nputx:=vectdim(TimePutx):

  P:= [[TimePutx[n]/60.,EnergyPutx[n]] $n=1..Nputx]:
  plot(P, style=line, symbol=asterisk, color=blue,
thickness=2, labels=["Time [min]", "Energy [kVat/h]"]);
  end proc:

> _plotEnergyTime(TimPutx, EnPutx);
```



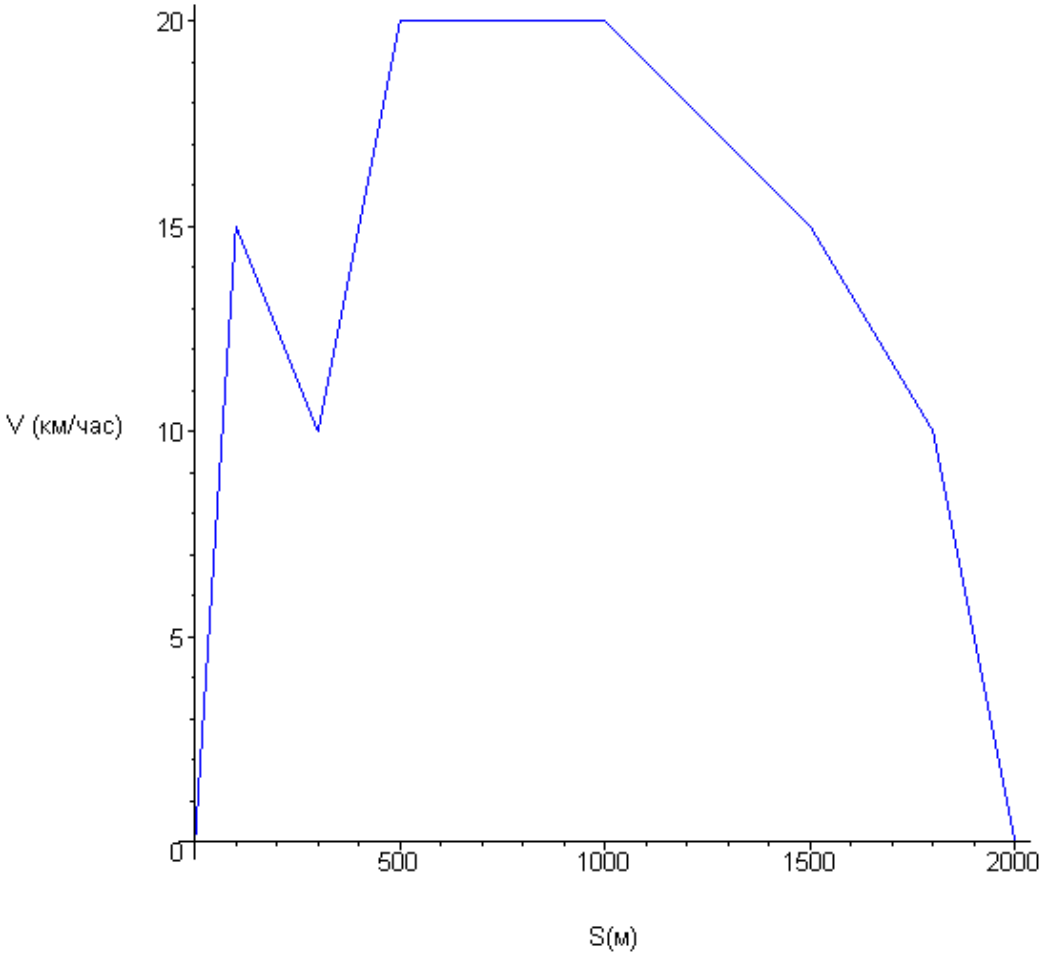

```
>
> _Move:=proc(iPutx)   #   = copy(sledPutxV):
# iPutx:=1:
local m, Time1, Time2, En1, En2, ds,j, iv1, iv2,
      uklonS1, uklonS2, S1, S2,
      regim, fk, Bt, amper, dt, Fk,
      Atiaga, Arecup, PutxV, L, n:

PutxV:=  copy(sledPutxV):
PutxV:= convert(PutxV, array): m:=coldim(PutxV):
Time1:=0.: Time2:=0.: En1:=0.:  En2:= 0.:
printf("\n"); printf("\n"); printf("\n");
printf("Режим движения%4d   \n", iPutx ):
```

```
    printf("   Путь   Скорость   Время   Энергия   Режим
F тяги  I тяги   B торм.  I рекуп. \n" ):
    printf("       m         км.час       мин      квт/час
кН       A       кН          A \n" ):
    for j from 1 to m-1 do
       ds  :=  Sset[j+1]  -  Sset[j]:       S1:=Sset[j]:
S2:=Sset[j+1]:
       uklonS1:=_ik(S1):  uklonS2:=_ik(S2):
       iv1:= PutxV[iPutx,j]:  iv2:= PutxV[iPutx,j+1]:
       regim, fk, Bt, amper, dt:= GoV1V2( iv1, iv2, ds,
uklonS1, uklonS2 ):
       Time2:=Time1+dt:
       Atiaga:= 0.:  Arecup:= 0.:  Fk:= 0.:
       if regim = Tiaga then
          Fk:= fk*Mpoezda*9.81/1000.:
          Atiaga:= amper:
          En2:= 0.0069444*amper*dt:
       end if:
        if regim = Tormoz then
          if  V_sredn(iv1,  iv2)  >=  20.0  then     #
...рекуперация
             Arecup:= amper:   En2:= 0.:
          else  #  ...электропневмо
             Arecup:= 0.:   En2:= 0.:
          end if:
       end if:
        if regim = XX then
           Arecup:= 0.:   Atiaga:=0.:  En2:= 0.:
        end if:
```

```
        printf("%6d    %5d     %6.2f    %6.2f    %6s   %7.2f
%6.2f  %7.2f   %6.2f",
                  Sset[j],   PutxV[iPutx,j],   Time1/60.,
En1, _RegTxt(regim),
                  Fk, Atiaga, Bt, Arecup );
        printf("\n"):
        Time1:= Time2:  En1:= En1+En2:
     end do:
     printf("%6d   %5d    %6.2f   %6.2f   %6s",
                  Sset[m],   PutxV[iPutx,m],   Time2/60.,
En1, `Stop` );
     printf("\n"):  printf("\n"):  printf("\n"):
     L:=[ [Sset[n],PutxV[iPutx,n]]$n=1..m ]:
     plot(L,   style=line,   thickness=1,   color=blue,
labels=["S(м)", "V (км/час)"]):
      end proc:
     > _Move(1);
```

Режим движения 1

Путь	Скорость	Время	Энергия	Режим	F тяги	I тяги	B торм.	I рекуп.
m	км.час	мин	квт/час		кН	A	кН	A
0	0	0.00	0.00	Тяга	129.80	21.99	0.00	0.00
100	15	0.80	7.33	Выбег	0.00	0.00	0.00	0.00
300	10	1.76	7.33	Тяга	109.11	30.08	0.00	0.00
500	20	2.56	17.36	Тяга	33.62	17.72	0.00	0.00
1000	20	4.06	28.43	Выбег	0.00	0.00	0.00	0.00
1500	15	5.77	28.43	Тормоз	0.00	0.00	6.61	0.00
1800	10	7.21	28.43	Выбег	0.00	0.00	0.00	0.00
2000	0	9.61	28.43	Stop				



Висновок

Розроблене програмне забезпечення передбачено тільки для визначення ефективних тягових розрахунків по Парето пасажирського потягу з локомотивом ДС – 3.

ВИСНОВОК

В роботі проаналізовано методи оптимізації стосовно проведення тягових розрахунків . Обрано найбільш відповідний метод за яким розроблено алгоритм та программу тягових розрахунків . обраним методом є метод що побудовано на основі принципа Паретто. З представлених досліджень можно зробити висновок що єфективні оптимальні траєкторії руху потягу може бути застосовано для організації руху поїздів . Перевагою ефективних траєкторій є надання вибору режиму руху потяга з декількох отриманих в результаті розрахунку на ЕОМ . Для проектування залізничної системи автоматики можє бути використана єфективна траєкторія за Паретто яяка мінімізує час Т. Тягові розрахунки є прикладної частиною теорії тяги поїздів і дозволяють вирішувати численні практичні задачі, що виникають при проектуванні та експлуатації залізниць. Найактуальнішою проблемою є проблема економії енергоресурсів. У той же час необхідно вантаж доставляти під час, а в багатьох випадках – в найкоротші терміни.

Під час розглядання даної проблеми були використанні принципи векторної оптимізації за принципом Парето.

Задачу на оптимальне управління рухом поїзда з ефективними витратами енергії та часу можна сформулювати так: знайти таке допустиме управління $v(s)$ - швидкість руху поїзда, при якому відповідні витрати енергоресурсів і часу були б оптимальними по Парето і при цьому виконувався графік руху на даній ділянці.

Для рішення поставленної задачі запропоновано схему методу динамічного програмування його дискретний варіант. Замість рекурентних рівнянь використовується покрокове обчислення безлічі точок, оптимальних за Парето, на площині значень показників витрат електроенергії $f(v(s))$ і часу $t(v(s))$. За отриманими розрахунками єфективну траєкторію руху поїзду що відповідає мінімальному витрату часу може бути використано ,при проєктуванні сучасних систем автоматики та телемеханіки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Подвижной состав и тяга поездов. Под ред. Докт. Техн. наук, проф. Н.А. Фуфрянского и канд. Техн. наук, доц. В.В. Деева. М., «Транспорт», 1979.
2. Правила тяговых расчётов для поездной работы. М., «Транспорт», 1985.
3. Гребенюк П.Т. и др. Справочник по тяговым расчётам, М., «Транспорт», 1987.
4. Подвижной состав и тяговое хозяйство железных дорог / Под ред. А. П. Третьякова. М.,1971.
5. Кокурин И.М., Кондратенко Л.Ф. Эксплуатационные основы устройств железнодорожной автоматики и телемеханики. —М.: Транспорт, 1989.
6. Тяговые расчёты. Методические указания к курсовому проектированию под редакцией Ю. Н. Ликратова. Новосибирск, 1989.
7. Рекомендации по обеспечению энергооптимального процесса перевозок на основе информационных технологий управления системами электрической тяги/ Решение Комиссии ОСЖД от 30 октября 2003 г.
8. Сидельников В.М. Производство тягових расчетов на ЭЦВМ. –Вестник ВНИИЖТ. –№7. -1961.
9. Сидельников В.М. Выбор оптимального режима управления локомотивом на ЭЦВМ. –Вестник ВНИИЖТ. –№2. -1965.
10. Шор Н.З., Берестовенко К.М., Росина Н.И. Автоматизация производства тягових расчетов на ЭЦВМ. –Киев. -1962.
11. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. -М.: Наука, 1965. _ 458 с.
12. Вентцель Е. С. Исследование операций: задачи, принципы, методология. 3-е изд. -М.: Дрофа, -2004. - 208 с.
13. Ерофеев Е.В. Выбор оптимального режима ведения поезда на АЦВМ с применением метода динамического программирования. –Транспорт. –Труды МИИТ. Вып. 228. -1967.

14. Сидельников В.М. Автоматизация выбора оптимального режима ведения поезда при электрической тяге на ЭЦВМ. Автореферат дисс. На соиск. уч. степ. к.т.н. –Москва. -1966.

15. Сидельников В.М. Автоматизация выбора оптимального режима ведения поезда при электрической тяге на ЭЦВМ. Дисс. на соиск. уч. степ. к.т.н. –Москва. -1966.

16. Пузанов Н.Я. Расчет на ЭЦВМ кривых движения. –Вестник ВНИИЖТ. –№4. -1966.

17. Ковальский А. Н. Система автоматического управления поездом метрополитена (САУ-М) и ее модернизация. – Труды МИИТ. -1968. – Вып. 276. –С. 3-13.

18. Система автоведения пассажирского поезда./Е. В. Ерофеев, Я. М. Головнчер, Н. И. Куренков и др.— Труды МИИТ. - Вып. 492. –С. 3-10.

19. Г а к к е л ь Е Я. Автомашинист для грузового тепловоза.— Труды ЛИИЖТ. -1964. -Вып. 232. -С. 3—8.

20. 3имарьков Б.Д. Локомотивом управляет автомат.— Электрическая и тепловозная тяга. -1973. - № *7. -С.* 21—22.

21. Босов А.А. Методы решения некоторых задач оптимальных тяговых расчетов на ЭЦВМ. / Диссертация на соискание ученой степени кандидата технических наук. –Днепропетровск. -1968.

22. К о с т р о м и н А. М. Оптимальное программное управление энергетическим оборудованием тепловоза. — Труды БелИИЖТ. -Вып. 89. -1970. С. 15—17.

23. К о с т р о м и н А. М. Методы определения оптимальных режимов вождения поездов. -БелИИЖТ. -Гомель -1974. —43 с.

24. К о с т р о м и н А. М. Об интегрировании уравнений движения поезда и расчете оптимальной траектории. — Труды БелИИЖТ. -Вып. 132. -1974. -С. 3—11.

25. К о с т р о м и и А. М., К е й з е р А . П . Расчет на ЭЦВМ оптимального процесса движения поезда методом локальных вариации. — Труды

БелИИЖТ. -Вып. 145. -1976. -С. 53—57.

26. К о с т р о м и н А. М. Реализация на ЭЦВМ статической модели тепловоза. -Труды БелИИЖТ. -Вып. 149. -1976. -С. 3—10.

27. К о с т р о м и н А. М., Р а ф а л о в с к и й В. В., Ф р е н к е л ь С. Я. Исследование на электронной модели системы оптимального управления тепловозом. - Труды БелИИЖТ. Вып. 149. - 1976. -С. 10—16.

28. К о с т р о м и н А. М. Об оптимальном управлении локомотивом при электрической тяге. -Труды БелИИЖТ. -Вып. 156. -1977, -С. 3—23.

29. Деев В.В. и др. Тяга поездов. М., «Транспорт», 1987.

30. Погосов В.Ю. Прогнозирование расхода электроэнергии на тягу поездов с учетом выброса параметров грузовых поездов и условий эксплуатации: Автореф дисс канд техн наукЖ 05.09.93 / МИИТ. –М. -1990. - 23 с.

31. Гетьман Г.К. Математические модели электроподвижного состава в задачах тягового обеспечения. / Сб. науч. Тр. НАН Украины. Ин-т математикм. –Киев. -1998. –С. 48-51.

32. Гетьман Г.К. Определение оптимальной по минимуму расхода энергии на движение поезда мощности локомотива / Зб. Наук. Праць. Випуск 39. ХарДАЗТ. –Харків. -2000. –С. 41-48.

33. Беляев А.В., Вольвич А.Г., Федорова Н.Ю. Алгоритм оптимального по расходу электроэнергии управления движения поезда / Сб. научн. Тр. Всерос. н.-и и проектно констр ин-т электровозостр. №39. -1998. –С. 160-169.

34. Босов А.А., Гетьман Г.К. Векторная оптимизация в задачах тяговых расчетов / Вісник Харківського державного політехнічного університету. Зб. Наук. Пр. Випуск 73. –Харків: ХДПУ. -1999. –С. 23-27.

35. Гетьман Г.К. Применение векторной оптимизации для решения задача тяговых расчетов / Вісник Харківського державного політехнічного університету. Зб. Наук. Пр. Випуск 62. –Харків: ХДПУ. -1999. –С. 12-19.

36. Босов А.А., Гетьман Г.К. Параметризация в задачах векторной оптимизации / Транспорт Зб. Наук. Пр. Випуск 5. –Дніпропетровськ: Наука і освіта. -2000. –С. 62-65.
37. РЖД-Партнер.RU, 7 мая 2009 г.
38. L. van Dongen, W. Fiechter. Elektrische Bahnen, 2000, № 11/12, S. 448 – 452.