学位請求論文

直流饋電系と列車群制御の
統合インテリジェントシステム化

指導教官　曽根　悟 教授

東京大学大学院工学系研究科

電気工学専攻 27105

高木　亮

1994年12月20日

目次

付録 I　RTSS マニュアル・序論

付録 II　RTSS マニュアル・プログラムの概要

付録 III　RTSS マニュアル・アルゴリズム詳説

付録 IV　RTSS マニュアル・使い方

図一覧

表一覧

I

序論

1

鉄道とインテリジェント化・統合システム化

近年，コンピュータ技術の急速な進展に伴い，**インテリジェンス**ないしは**インテリジェント化**という言葉がいろいろな場面で頻繁に使われるようになってきた。しかし，その言葉の意味は相変わらず使う人や使われる場面によってかなりまちまちである。

本論文は，その表題にもある通り,「直流饋電系と列車群制御の統合インテリジェントシステム化」がテーマである。そこで，本章では本論文における「インテリジェント化」ないし「統合化」とは何かを簡単に定義づけ，筆者の立場を明確にすることを試みる。

⟨1.1⟩　インテリジェント化

まずは「インテリジェント化」についてである。ここで重要と思われることは,「インテリジェンス」と「インテリジェント化」というのは必ずしも同一ではない，ということだ。ここでは，まずは簡単に，システムを「インテリジェンス」を持った状態に近づける操作が「インテリジェント化」であるという程度に考えておくことにしよう。

⟨1.1.1⟩　「インテリジェンス」，または人工知能とは

人間は機械や他の動物などに比べて非常に高いインテリジェンスを持っているとされる。機械にも人間と同じようなインテリジェンスを持たせたいという希望は，人類の一つの大きな夢であった。人間がロボットと，相手が機械であることを意識しないで話ができる，といった SF 的な世界はまだまだ遠い先の話としても，ここ2〜3年のパーソナルコンピュータの能力向上に見られるように，この夢の実現までの距離が確かに短くなりつつあることはまちがいない。しかし，これだけの能力向上があってもなお人間の「知能」は機械にとって越え難い，遠い目標であることもまちがいない。

このような現段階においてもなお「**人工知能**」と呼ばれているものは確かにある。そうなると，一般論としてどのレベルの能力を持っていれば「知能」ないし「インテリジェンス」と呼びうるのか，という問題が浮上する。最近は一般に「**推論**」とか「**学習**」を行う程度のレベルでないとインテリジェンスとは呼び得ないとされている。しかし,「学習」ないしは「推論」とはそもそもどういうものをいうのか，などと問題をつきつめてゆくと，その細部は人によってかなりまちまちであることがわかる。

ここでは「学習」は置いておくこととして，主として「推論」についてもう少し突っ込んだ検討を試みよう。「広辞苑」[8]によれば,「推論」とは「推理」のことであり,「推理」とは「あらかじめ知られていることから新しい知識・結論を導き出すこと」とある。この定義によれば，四則演算などの「単なる計算」は

「推論」とは一般にはいえない。これが「推論」ならば，電卓もインテリジェンスを持った機械ということになってしまう。これは一般的な認識からは大きくずれているといわざるを得ない。

一方，方程式の「展開」や「因数分解」は推論の簡単な例だ，といわれる。また，ゲーム（例えば「オセロゲーム」や「将棋」や「囲碁」）の「次の一手」を考えるのは高度な推論である，とされる。ところが，現在これらを行うソフトウェアが現実に存在する。方程式の展開や因数分解ができるツールとしては，すでに *Mathematica* なるもの[9]が市販されて広く用いられている。一方，ゲームについては残念ながら筆者は「将棋」「囲碁」が満足にできないので，これらのソフトウェアの強さを計ることはできない。しかし,「オセロゲーム」は相当強いソフトウェアが 1986 年ころの段階ですでにあり，筆者の力ではそのソフトに勝つこともできなかった。「囲碁」についてはまだ開発途中のようだが，これも近年だいぶレベルアップしてきたようだ。

これら「すでに実現してしまったソフトウェア」は，その実現法がすでにわかってしまった現在，単なるアルゴリズムに過ぎなくなった，といういい方もできる。単なるアルゴリズムということであれば，例えば非線形連立方程式を解く Newton-Raphson 法や，微分方程式を解く Runge-Kutta 法と同じレベルである。Newton-Raphson 法や Runge-Kutta 法はともかく，すでにソフトウェアとして実現してしまった手法を「インテリジェンス」と呼ぶことには抵抗を感じる。

このようなことから考えると，人間の知能に比べて機械のそれが相変わらず相当劣る現状においては，人工知能などと呼ばれているものは多分に「未来指向」的な要素を含んでいる，といういい方ができよう。すなわち，現在実現していない「人間の能力」を機械に実現させることができたなら，それが人工知能ないし機械のインテリジェンスである，といういい方が可能だ。こういう定義にしてしまうと厳密には人工知能は存在し得ないということになってしまうが,「現在」という言葉に適切な「幅」を持たせることによって（あるいは「新規性」といった言葉を用いて定義を適切に書き換えることにより），実現してしまった人工知能も当分は「人工知能」としての存在を許されることになる。

⟨1.1.2⟩　鉄道とインテリジェント化

機械の「インテリジェンス」ないし人工知能について⟨1.1.1⟩で行ったような定義しかできないとなると,「インテリジェンス」を目指したシステムの改良という「**インテリジェント化**」の定義はさらに曖昧な意味あいにならざるを得ない。しかし,「未来指向」的な人工知能の定義にあっては「現在」の把握が重要であると思われる。すなわち，鉄道のシステムの現状をより「インテリジェンスに近づける」技術がインテリジェント化ということになる。

ただし，インテリジェンスに近づいた結果としてシステムが「使いにくく」なったら，それは意味がないというべきだろう。そういう意味からすれば，現在より「**気の効いた**[10]」（**「clever な」**）働きをするシステムへの改良，という定義は妥当である。

いうまでもなく，従来の鉄道でもインテリジェント化が行われてこなかったわけではない。部分的には本格的な「人工知能」の適用も行われている（ダイヤ作成支援ツール[11]，列車群制御支援ツール[12]など)。また，最近になって多くのインテリジェント化手法の提案がまとめられた文献[13]もある。当然本論文においてもいくつかのインテリジェント化手法の提案が行われる。そのなかには，インテリジェンスという見方からするとあまりにプリミティブに過ぎるものもある。空気バネのレベリングバルブと応荷重装置はインテリジェント化の一例とされている[10]が，これなどはプリミティブな手法によるインテリジェント化の最たる例だろう。インテリジェント化がそのまま「人工知能」の導入などの高度な手段を伴うわけでは必ずしもない。

また，何が「気の効いた」システムか，というのも問題である。例えば，本論文では再三「地上電力設備の利用率向上」という言葉が出てくる。サービス向上のためにはまだまだ輸送力の増強が必要である，というのが日本の多くの鉄道における現状である。その輸送力増強のために電力設備の容量不足が生じる

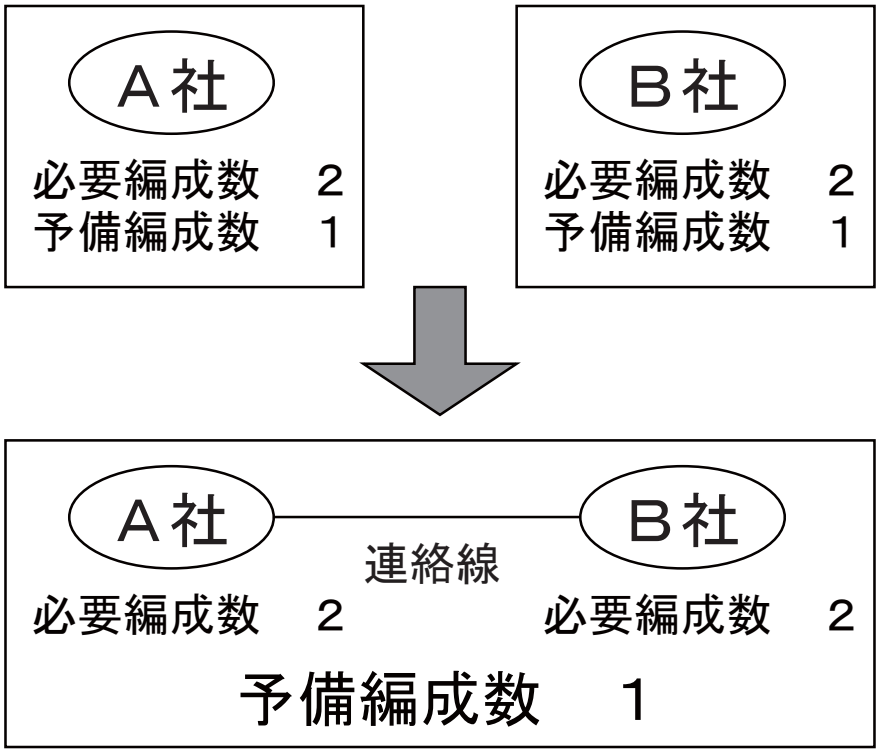


図 1.1: 統合化の例

が，従来の方策はそれを電力設備の増強で補おうというものだった。これでは巨大な地上設備を現状以上に抱えることになり,「気の効いた」システムとはいえない。本論文では，これ以外の手段によって変電設備の利用率向上をはかることにより，輸送力増強のための電力設備容量をいわば「生み出す」手法を提案する。

このように，本論文においては「大きなハードウェアの新たな導入なしに，システムの抱える問題を clever に解決する技術」を「インテリジェント化技術」と定義することにしよう。

⟨1.2⟩　統合インテリジェント化

統合化とは，複数の独立したサブシステムをそれぞれ単独に運用するのではなく，何らかの手段により 1 つのより大きなシステムにまとめることで，経済性などの改善を図ろうとするものである。

⟨1.2.1⟩　統合化と予備冗長性

例として，比較的車両数が少なく，線路が独立している鉄道会社 2 社が存在している場合（図1.1）を考える。両線とも普段必要な電車は 2 編成だけだとしよう。この場合でも，検査などのために予備車は必要であるが，線路がつながっておらず 2 社が別々に予備車を持つ場合，予備車は各社 1 編成ずつ必要である。ところが，両社の線路をどこかでつないで（新たな連絡線の建設），車両を共通運用できるようにすれば，予備車は両社合わせて 1 編成ですむだろう。

この例では，統合化されるサブシステムが 2 社の鉄道会社，統合化の手段は新たな連絡線の建設である。複数の独立したシステムをこのように統合化する狙いのひとつは，この例のような予備冗長性の融通による減少にある。

この例では統合化のために連絡線という比較的「大きなハードウェア」を「新たに導入」しているが，統合インテリジェント化ではこのようなことは行わず，大きなハードウェアについては既存のものの範囲内で改善を図る。上の車両運用の共通化の例でいえば，わざわざ新規に連絡線を建設しなくても線路はどこかのルートを通じてつながっていることが多いから，それを使えばハードウェアの追加は必要ない。

⟨1.2.2⟩　統合化とインテリジェント化

統合化のもう一つの狙いは，⟨1.1⟩で定義した「インテリジェント化」，すなわち「大きなハードウェアの追加なしに，システムの抱える問題を clever に解決する技術」を適用できる範囲を拡大することにある。

上の2つの鉄道会社の車両運用の共通化の例でも，予備車両数の低減によるコストダウン効果以外にいろいろな可能性が出てくる。例えば，統合された2社のどちらでも通常の運用をこなすためには2編成必要であるものの，2編成必要なのはピーク時の場合で，オフピーク時には1編成あれば十分であるとしよう。ここで，一方の会社の沿線で大きなイベントがあり，一時的に大きな輸送需要があり3編成が必要となった場合，もう一方の会社が通常使っている2編成のうち1編成を借りてきて使うような柔軟な車両運用が可能になろう。このように，統合化前にはできなかったことが可能になることは，インテリジェント化技術の適用可能性が統合化によって拡大していることを意味する。

⟨1.2.3⟩　鉄道システムと統合インテリジェント化

鉄道システムは，他の交通機関に比べ格段に高い安全性，省エネルギー特性，優れた対環境特性を有する。この特性を社会にとってより生かすためには，鉄道が他の交通機関に比べてより高い競争力を持つことが望まれる。昨今盛んな列車の高速化，高密度化はこの流れにそったものである。

鉄道システムは大規模システムであり，いくつかの比較的独立したサブシステムから構成されている。しかし，サブシステムは互いに完全に独立なわけではなく，あるサブシステムに余裕を多く与えるとほかのサブシステムにも結果的に余裕が生まれるようなケースは数多い。現実には，各種のサブシステムの設計においてこのことをほとんど無視して余裕を設定しているため，鉄道システム全体としてみると過大な余裕を持っていることが多い。

⟨1.2⟩で述べた統合インテリジェント化技術によって，これらの余裕をサービス改善に転用したり，同一のサービスを提供するために要する資源を減少させることを狙うことができるようになる。システムを組み合わせることにより初めて可能となる新しいサービスの可能性も数多い。ところが，従来実施されてきた鉄道のインテリジェント化は，比較的小さなサブシステムの中に留まることが多く，その効果も鉄道システムの限られた範囲にしか行き渡らなかった。このような状況を打破するものとして，統合インテリジェント化技術には大きな期待を寄せることができる。

この統合インテリジェント化技術の狙い目には多様なものがあり，制御の戦略もじつにさまざまである。これらの戦略，得られる可能性のある効果を体系化し，どの程度の効果が期待できるのかを，シミュレーションによって明らかにすることが求められる。

その一方で，統合インテリジェント化では大きなハードウェアの「新たな導入」なしに改善を達成しようとする。新たな導入がなければシステムの能力は潜在的なものも含めて見れば上がってはいないのだから，達成できる改善の度合にはいうまでもなく限度があることも理解すべきだ。従って，改善の可能性だけでなく，統合インテリジェント化による改善の極限を見きわめることも重要であろう。

2

本研究の目的および本論文の構成

⟨2.1⟩ 本研究の目的

本研究では，⟨1.2⟩で述べた統合インテリジェント化技術を，鉄道を構成するサブシステムのうち電鉄饋電システム及び列車群制御システムに応用し，鉄道システムが現在持っている余裕をうまく活用することにより，サービス提供に要する資源を減少させる，ないしは同一の資源のもとでサービスを改善させることを狙う。

(1) まず，本研究におけるさまざまな検討に利用するために，直流饋電システムの正確なシミュレーションモデルを構築し，その妥当性を検証する。なお，このシミュレーションプログラムは，本研究用のみならず直流饋電システムのシミュレーション一般に利用可能であるように，可能な限りの汎用化を図ったプログラムとし，これ自体を成果物として公開できるように配慮する。
(2) 統合化以前の饋電システムについての最適化を行う。(1) にて構築したシミュレーションモデルを用い，さまざまなケースを想定して饋電システムの最適化を行う。
(3) 饋電システム・列車群制御システムの統合インテリジェント化技術にはいろいろな可能性と戦略とがあるが，これらをまず体系化する。次いで，(1)にて構築したシミュレーションモデルを用い，さまざまな統合インテリジェント化技術の可能性の評価を行う。

⟨2.2⟩ 本論文の構成

第 II 部では，饋電システムと列車群制御システムについて説明したのち，このシステムのかかえる問題と統合インテリジェント化技術によって考えられる解決の方策をまとめる。

第 III 部では，本研究の死命を制する重要なツールである直流饋電システムシミュレーションプログラム RTSS について，その特徴を述べたのち，実際のシステムに関する測定結果と RTSS の出力データとの比較，さらに RTSS と従来の考え方で設計されたプログラムとの結果を比較して，RTSS のシミュレーションプログラムとしての有効性の高さを検証する。

こののち，**第 IV 部・第 V 部**で，**第 II 部**でまとめた統合インテリジェント化技術の定量的可能性を RTSS を利用して議論する。

第IV部では，饋電システムの省エネルギー化手法について，変電所の V–I 特性の最適化，変電所電圧のリアルタイム制御，列車主回路電力制御による回生失効防止の3つを検討している。V–I 特性の最適化は統合インテリジェント化以前の段階の話だが，このレベルの改善可能性もまだかなり残っていることを示す。また，変電所電圧リアルタイム制御によって V–I 特性最適化よりさらに1段進んだ改善可能性があることをシミュレーションによって明らかにする。これら，変電所電圧制御に関する検討ののち，列車主回路電力制御の概念が提出され，これによる回生失効防止制御の効果がアピールされる。

設備利用率向上の可能性についても**第IV部**で検討される。変電所の電圧制御による電流のピークカットに簡単に触れたのち，列車主回路電力制御を変電所電流ピークカット制御に応用する方法を詳しく述べる。列車主回路電力制御によって非常に大きなピークカット効果を得ることができることが，シミュレーションによって示される。

ここまでは饋電システム中心の議論だったが，**第V部**では列車群制御に一歩踏み込み，列車ダイヤの動的な変更を議論する。まず，変電所容量が部分的に低くなった場合，その領域では列車がゆっくり走ることが可能なようにダイヤの余裕時分を再配分する形の余裕融通が議論され，1変電所が完全に脱落してもほぼ通常の運転を継続できる見通しが示される。さらに，列車運行乱れ時について，一部の列車の走行パターンを変更することで省エネルギー化が図られることもシミュレーションで示される。これら統合インテリジェント化技術による改善の極限をみきわめるための最適制御問題への定式化も同時に提案される。一連の問題の多くが列車群の最適走行パターン問題に定式化できることが示されたのち，数値解を求めるアルゴリズムと，それによって実際にいくつかの問題を数値的に解いた例が提示される。

第VI部では成果をまとめ，今後の課題について述べる。

II

饋電システムと列車群制御システム

3

本論文で取り扱うシステムの定義

⟨3.1⟩　鉄道システムを構成するサブシステム

鉄道システムは大規模な工学的システムであり，いくつかのサブシステムから成り立っている。電気鉄道のインテリジェント化の現状と可能性・展望・課題をまとめた電気学会技術報告[13]によれば，

(1) 車両サブシステム
(2) 電力サブシステム（饋電システム）
(3) 運転保安サブシステム（信号システム）
(4) 運行管理サブシステム（列車群制御システム）
(5) 営業・旅客サービスサブシステム
(6) 保守・防災サブシステム
(7) 情報伝送サブシステム

の7つがサブシステムとしてあげられている。

いうまでもなく鉄道は大規模システムであるから，これら7つのサブシステムもそれ自体が相当な大規模システムであり，それぞれがまた多くのサブシステムから成り立っている。さらに，それぞれのシステムの境界は必ずしも明確でないことがある。例えば，保守・防災サブシステムや情報伝送サブシステムは車両や駅のコンポーネンツとしてこれらに分散して存在する部分もあるだろう。あるいは，電力サブシステムは負荷としての車両も含むから，車両サブシステムは電力サブシステムに含まれるといういい方もできる。このように，サブシステムは互いに独立に存在しているわけではないことに注意しておく必要がある。

これらのサブシステムのうち，電力サブシステム（饋電システム）と運行管理サブシステム（列車群制御システム）を組み合わせ，統合化・インテリジェント化の検討を行うのが本論文の主たる目的である。そこで，以下でこれら2つのシステムについて説明する。

⟨3.2⟩　饋電システム

饋電システムとは，電鉄用変電所ならびに変電所から供給された電力を電気車に伝える電線路のことをいう。饋電システムには直流および交流の2方式があるが，本研究が対象とするのは専ら直流システムである。

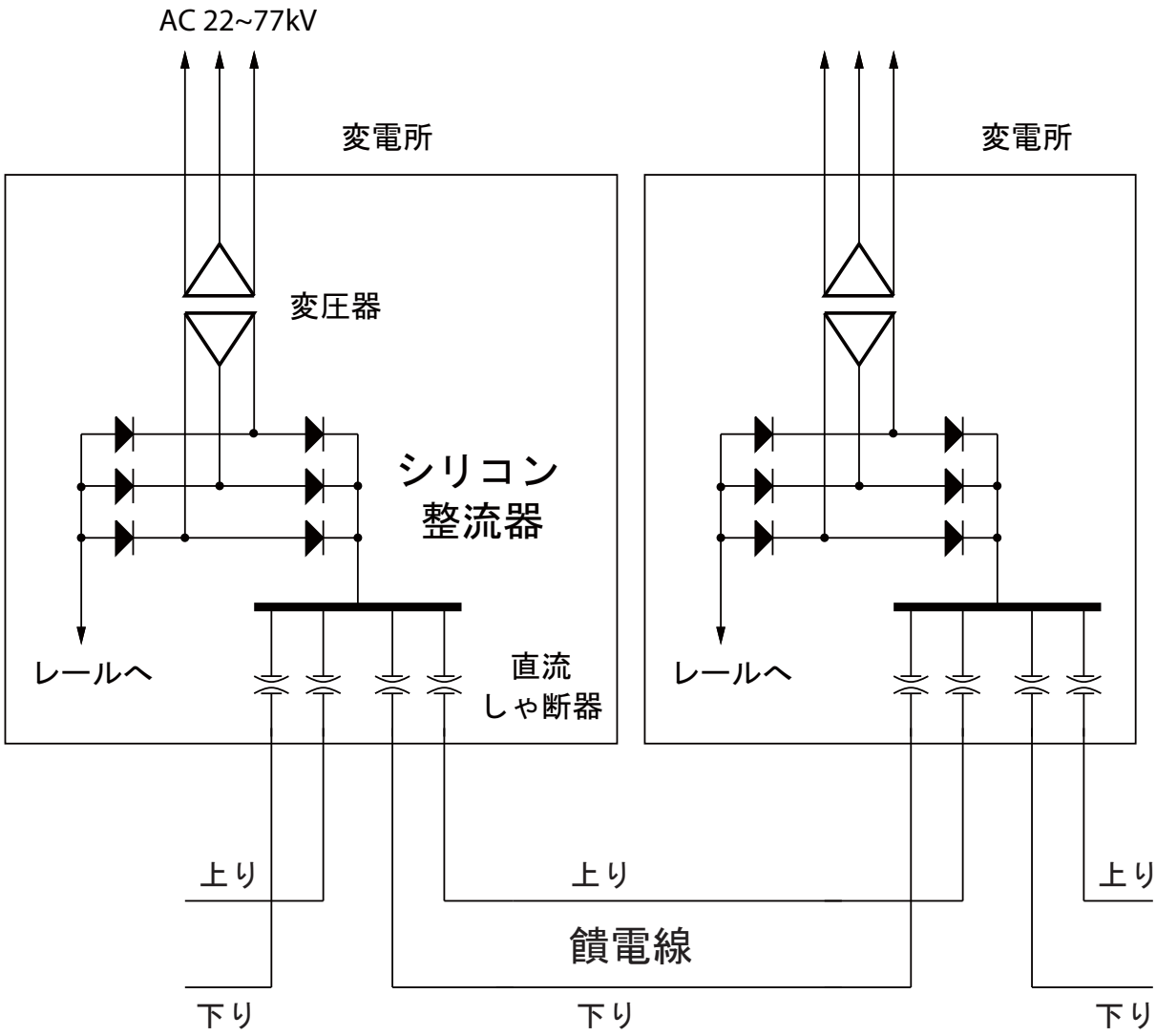


図 3.1: 直流饋電設備概要

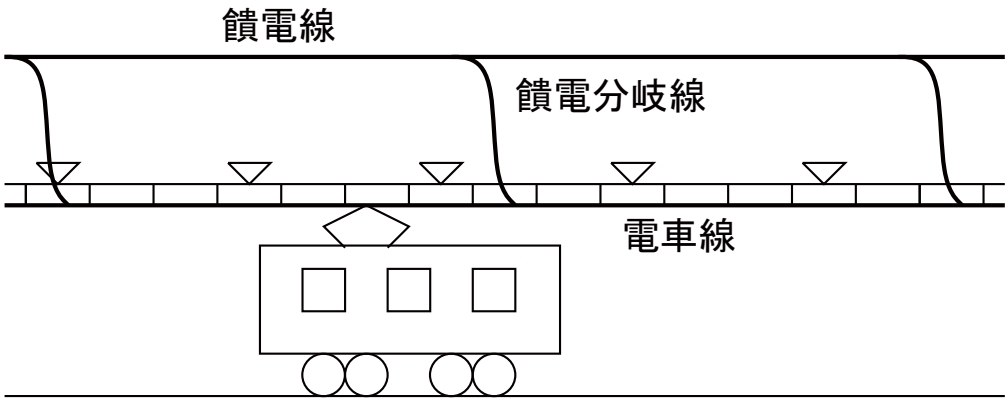


図 3.2: 架空電車線路概要

直流饋電システムは，地上側が図3.1および図3.2で示される**電鉄用変電所**および**饋電系統**（**饋電線**，**電車線**，帰線としてのレール），および負荷としての列車によって構成される大規模システムである。最近は，**電力指令所**などと称し，変電所にて得られるさまざまな情報を集中管理し，制御（変電所変成器の接続・解放，夜間工事のための停電制御など）も行う場所が整備されている。

現在，直流電鉄用変電所は，シリコン・ダイオードをコンバータに用いている。この変電所は鉄道関係者の間では通常**シリコン変電所**と呼ばれる（ちなみに **SR** と略されることが多い）が，この呼び名は水銀整流器や回転変流機が主流だった当時，シリコンダイオードによるコンバータがこれらに対してシリコン整流器と呼ばれたことからきている古い言葉である。

近年は水銀整流器や回転変流機はほとんど姿を消したといってよい。そのうえ，サイリスタをコンバータに用いた**サイリスタ変電所**と呼ばれるものも現れ始めた。サイリスタもシリコン半導体でできている，ということからすれば，サイリスタ変電所に対するシリコン変電所という名前はよくないと考えられる。そこで，本論文ではダイオードコンバータの変電所のことを**ダイオード変電所**という名称で呼んで区別することにする。

直流電気車では，ブレーキ時にモータを発電機として運転することにより，運動エネルギーを電力として車両から饋電システムに返還する**電力回生ブレーキ**を搭載した車両が，最近は主要路線ではほとんどを占めるようになった。なかでも，回生能力が高いほか，高い粘着特性，省保守性，軽量化など数多くのメリットを持つ**インバータ制御電気車**の普及がめざましく，現在日本で新造される車両のほとんどはインバータ制御電気車という状況である。そこで，本論文でも特に断らない限りインバータ制御で回生ブレーキを持った電気車を対象に議論を進める。

交流饋電システムでは変圧器を通して饋電システムから電力系統への電力の逆流があり得るが，ダイオード変電所では直流側から交流側への電力の逆流はおきないために，饋電システム内に回生電力を消費する負荷が不足している場合，回生車が回生能力を生かせない**回生失効**の問題が生じる。コンバータをサイリスタ化（サイリスタ変電所化）しても，それだけでは直流側の極性が定まっているために逆流はできない。

回生失効は，列車の回生エネルギーの有効活用ができなくなるデメリットがある他，失効時の回生ブレーキと空気ブレーキの切り替わり時に空気ブレーキの立ち上がりが遅いため，あるいは勾配路線では抑速回生ブレーキが失効するため，などの理由で運転上支障をきたす場合もある。最近では，この問題に対処するために，各種の**回生電力吸収装置**が実用化されている。**回生インバータ**，**フライホイール**，**抵抗チョッパ**などが代表例であり，地上の変電所や駅構内などに設ける。コンバータをサイリスタとし，逆流時は直流側の極性をサイリスタスイッチ等で切替えることにより，コンバータ・インバータ両用システムとしている例もある。地上設置のこれら装置に対し，車両にブレーキ抵抗を持たせる場合もある。

フライホイールやバッテリポストなどの蓄エネルギー設備を，電源（変電所）が不足している場所に電圧補償のために設置することもある。電力系統からの送電線の設置が難しかったりコストが高かったりする場合に採用されるが，現在のところ設置事例はきわめて少ない。

⟨3.3⟩　列車群制御システム

列車群制御システムとは，列車が遅延した場合などに列車群に対し列車の運行を正常に戻すための制御（駅での出発抑止，駅間走行時分の延長や短縮などの指示）をかけるためのシステムをいう。現実の鉄道システムではこれを完全に自動で行うことはなく，センタ（路線あたりひとつとはかぎらない）にいる指令員とのマン・マシン・インタフェースによって運行乱れからの回復のための戦略を定めるようになっている。

鉄道という大規模システムのサブシステムとしての列車群制御システムは，饋電システムに比べるとま

だ確立された構成がある段階ではない。その一方では，列車群制御の機能の他に鉄道システム全体の監視などの機能を盛り込んだ**鉄道トータルシステム**という考え方や，それを採り入れたシステムも現れつつある。これらのシステムは当然電力システムも取り込んだものになっているが，電力システムとの統合化のレベルは本研究で取り扱うような内容からは程遠い消極的なレベルに留まっている。特に残念なことは，駅務省力化機器や駅の案内装置など，一般の乗客との接点となる設備もこのシステムに取り込まれていることだ。肝心の使い勝手が改善されていれば残念なことはないのだが，乗客の目からみたとき従来の「トータルシステム化」されていない鉄道と比べて目立った改善がみられないのである。

本研究では「列車群制御システム」として次のようなものを考える。このシステムの基本となるのは制御の主体となる場所（インテリジェンスの所在場所でもある）である。これを**列車群制御センタ**（またはセンタ）と呼ぶことにする。これに，センタに列車の位置を信号系（軌道回路）からの情報として獲得・表示するシステム，列車や線路に制御のための指示を行う装置が含まれる。

例えば，従来CTC とかTTC とか呼ばれていたものはこの列車群制御システムのサブシステムとなる。ATO 装置も場合によっては含まれる。なお，本研究にあっては自動運転を前提とは必ずしもしない。

⟨3.4⟩　饋電システム・列車群制御システムにおけるインテリジェント化の範囲

⟨1.1⟩において,「インテリジェント化」とは「大きなハードウェアの新たな導入なしに clever な問題解決を図る技術である」と定義したが，饋電システム・列車群制御システムにおいて,「大きなハードウェアの新たな導入」とは例えば次のようなものを指す。

- 変電所の新設・容量増
- 饋電線の太さ増加，本数増
- 回生電力吸収装置の新設

一方，例えば容量増を伴わない変電所のサイリスタコンバータ化や列車のVVVF インバータ制御化は，老朽取り替えのさいに行えばよく，ここではインテリジェント化の定義における「大きなハードウェアの新たな導入」にはあたらないものとする。

4

統合化鉄道電力システムとその可能性

本論文では，饋電システムと列車群制御システムとの統合インテリジェント化を目指す。そこで，最終的にこれらが統合化されたシステムを**統合化鉄道電力システム**と呼ぶことにする。本章では，統合化前の饋電システム・列車群制御システムにあってどんな問題があり，統合化鉄道電力システムにおいてそれがどのように解決できる可能性があるかを，体系的に論じる。

⟨4.1⟩　省エネルギー化

同一のサービスを提供するために要するエネルギーは，当然のことながら小さい方がよい。そのための技術的可能性をいくつか列挙してみよう。

⟨4.1.1⟩　列車運転における省エネルギー化

列車の走行パターンを最適化することによって，省エネルギー化を図ることが可能だ。一般に，定められた駅間走行時分で走る駅間走行パターンは無数に存在するが，これらのうちもっとも列車消費エネルギーの小さいものを選ぶことによって省エネルギー化を図ることができる。

ただし，一般に1列車のみの最適パターンと列車群としてみた場合の最適パターンが必ずしも一致しないことに注意すべきだ。回生ブレーキを持たない電気車では，これらの違いは主として動力特性が電車線電圧に依存することから生じる。一方，回生ブレーキを利用する電車では，回生ブレーキの有効性が列車群の動きに大きく依存するため，この違いは非回生車の場合に比べて非常に大きくなる可能性がある。

変電所が饋電システムから電力系統側への電力の逆流を許さない条件下にあっては，回生車と力行車が共存する場合のほうが，そうでない場合より回生電力を有効活用できる。この性質を利用するため，着発（回生車・力行車）の競合をうまく作るようにダイヤ作成の段階から配慮することも考えられる。

⟨4.1.2⟩　饋電システム（変電所）の制御による省エネルギー化

饋電システムの諸定数・諸特性を変更したり，列車群制御システムから得られる情報を饋電システム側で活用することにより，サービスレベルの変更なしに省エネルギー化を図る余地が残っている。

回生失効は，饋電システム内に回生電力を消費する負荷が不足している場合に起きることは，すでに⟨3.2⟩に述べた。ところで，長い直流電化区間であれば，回生車からなるべく遠くの負荷にまで電力がとどくようにすれば，回生失効の確率も減少することが期待される。そのためには回生車が変電所の**無負荷時**

送出電圧よりなるべく高い電圧で電力を回生すればいいが，回生車の出し得る電圧には当然限度がある。そこで，変電所の無負荷時送出電圧を高めから低めに変更すれば，同じ電圧でより遠方まで回生電力を送ることができるようになる。このように，変電所の無負荷時送出電圧を下げると回生車が回生絞り込みを行う確率が減少し，変電所入力エネルギーの低減につながる。

ただし，電圧を下げると一般には変電所電流が増大し，列車の力行性能も悪化する。これらは**饋電線損失**の増大や**力行時間**の増大につながるので，あまり極端に下げると逆に変電所入力エネルギーは増えてしまう。従って，変電所送出電圧には最適値が存在することが知られている。

送出電圧の変更は，ダイオード変電所にあっては変電所の**変圧器1次側タップ**切替えによって行える。その場合は選べる無負荷時送出電圧の値が変圧器タップに対応した離散的な値だけとなる。また，1日の時間帯別に送出電圧を変更したい場合，自動タップチェンジャなどの設備が新たに必要となる。サイリスタ変電所であれば，この種の設備は必要なく，しかも連続的に送出電圧を設定可能となる。

一方，サイリスタ変電所を前提にすれば，変電所の **V–I（電圧–電流）特性**を変更することで，従来のダイオード変電所よりも無負荷時送出電圧をさらに下げ，同時に重負荷時の送出電圧を高い値に保つことができるようになる。このほか，V–I特性上で定電圧領域を広くとればとるほど，変電所間の横流が低減され，饋電線損失が下がることによる省エネルギー化の可能性も出てくる（ただし，あまりこれを極端にし過ぎると変電所のピーク電流値が上昇してしまう）。

さきにのべたように，変電所の電圧低下は回生電力の円滑な流通の面での有利性と力行性能低下・饋電線損失増大の面での不利とを併せ持つ。ここで有利性について考えてみると，回生車と力行負荷とが同時に饋電システム内に存在しなければ，変電所電圧を低くする意味はそもそもないことに気づくだろう。そこで，各瞬時の列車の位置・速度・状態などに応じて変電所電圧の**リアルタイム制御**を行い，回生車・力行車の共存時にのみ変電所電圧を下げるようにすることで，電圧低下のデメリットを抑え，さらに省エネルギー化を図ることも可能だ。

⟨4.1.3⟩　列車主回路電力制御による回生失効防止

饋電システムの問題とされる回生失効に対しては，通常変電所の制御で対処されることが多かった。しかし，列車の主回路電力を制御することにより，問題を解決することもできる。饋電システムの評価改善を列車の制御により達成しようとする一連の制御戦略を**列車主回路電力制御**，または単に**主回路電力制御**と呼ぶ。回生失効防止に主回路電力制御を使う場合には，回生車の多いときに惰行車を力行させることで回生失効防止を図ることができる。すなわち，列車の運動エネルギーをフライホイールのように使おうというアイディアである。

この場合，力行により惰行車が饋電システムから受けとったエネルギーは，当然のことながらあとで何らかの形で饋電システム側に返却しなければ省エネルギーにはならない。次駅が近いところで回生失効防止のための力行を行うと，これを饋電システム側に返却する余地のないまま駅の停車位置に停止するためのブレーキに入ってしまい，その駅に早着する。駅停車時分がその早着分だけ短縮できればよいが，出発時分は通常早めることはできないから，その分だけ駅停車時分が増大するだけのことになる可能性が強い。

それでも，回生失効が減ることによりブレーキシュー摩耗の低減ははかれるものと思われる。また，回生ブレーキが完全に無効になると，電車は主回路を開いてしまうので，再び饋電システムが回生電力を受けとれる状況に戻っても電力回生ができなくなる。従って，この方法で主回路が開かれるほどの回生失効を防止するだけで，早着した列車のエネルギーが回収できなくてもそれ以上の大きな効果が得られる可能性がある。

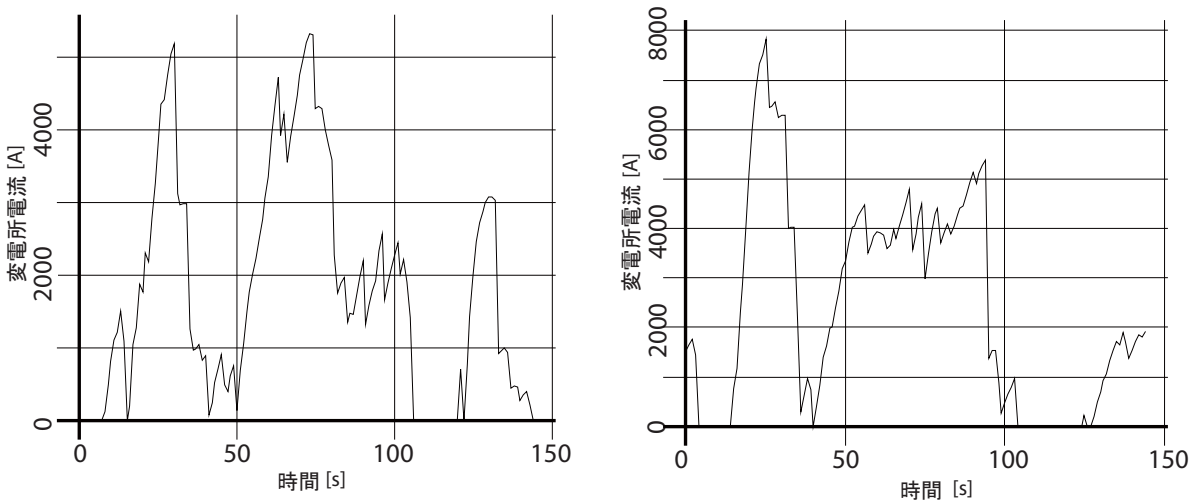


図 4.1: 変電所の出力電流の変動の例 (144秒周期)

⟨4.2⟩ 地上電力設備の機器利用率向上

電鉄変電所の電流は，図4.1に示すように変動が激しい。そのために，ピーク電流と平均電流との比が非常に大きくなっている。この状態では，変電所の機器利用率が低いと考えることができる。変電所電流の**ピークカット**によって，ピーク電流と平均電流との比が小さくなり，変電所の機器利用率が向上する可能性がある。なお，以下では変電所電流のピークカットのことを単にピークカットと称する。

また，直流変電所の容量設計において，かなり過大な冗長設計が行われている事実もある。これは，ある変電所が故障した場合その負荷を隣接変電所が背負うことを考慮するためである。統合化鉄道電力システムにおいては，ある程度の路線長がある場合に故障時の負荷を多数の変電所に均一にもたせるように制御をすることも可能である。このことを利用すれば冗長設計の考え方自体を変えることができ，変電所機器利用率のさらなる向上も図ることができる。

⟨4.2.1⟩ 変電所の制御によるピークカット

饋電システムの諸定数・諸特性を変更したり，列車群制御システムから得られる情報を饋電システム側で活用して制御を行ったりすることによりピークカットが可能だ。ただし，変電所が**ダイオード変電所**では難しいものと思われる。

サイリスタ変電所を前提にしたとき，変電所の V–I特性を変更することで，ダイオード変電所ではできなかったピークカットが可能になる。電流が大きくなり，ある電流値に到達したら定電流制御モードに移行し，これ以上電流が流れないようコンバータを制御することで可能になる。

列車の位置・速度・状態などの情報によって変電所電圧の**リアルタイム制御**を行うことで，ピークカットを図ることも可能だろう[1]。

⟨4.2.2⟩ 列車主回路電力制御によるピークカット

回生失効と同様に，変電所の制御でなく列車主回路電力制御によってもピークカットが可能だ。

この目的に主回路電力制御を応用する場合，変電所電流が過大な場合には力行車が電力を一時的に絞る制御，および惰行車が一時的に電力回生ブレーキで自車両の運動エネルギーを饋電システムに返却する制

御を行うことで，ピークカットを図ることができる。簡単にいえば，列車の運動エネルギーをフライホイールのように利用して饋電システムの救済を図る技術である。

このような制御は当然ダイヤの乱れの直接の原因になりうるが，通常の運行で無視できる範囲内（1秒程度）の運行乱れで相当な割合のピークカットが可能になることが示される。

⟨4.2.3⟩ デマンド管理

実際にエネルギーを減らす省エネルギー化のほかに，事業者にとっては電力会社に支払う料金を最小化したいという思惑もある。この両者が異なる課題としてここで取り上げられるのは，消費する電力エネルギーと支払う電力料金との関係にはさまざまな非線形性があるためである。このような観点からの検討も興味深い課題である。

例えば，契約電力を上回る電力をとろうとすると料金がはねあがる仕組みを回避するデマンド管理が考えられる。列車主回路電力制御の技術を利用した制御により容易に実現可能だろう。

なお，これらは評価量としてエネルギーのかわりに電力料金をとることで，省エネルギー化と同様の手法で検討できるものと考え，本研究では特にこの問題を取り扱うことはしない。

⟨4.3⟩ 列車群制御における饋電システムの制約の考慮

現状の列車群制御システムは，饋電システムの条件を考慮する制御はしていない。**鉄道トータルシステム**と呼ばれるものでも，饋電システムの条件を考慮に入れた列車群制御を積極的に行う機構は入っていないのが普通だ。しかし，列車の運行上調整可能な範囲があるなら，それを饋電システムの最適化ために利用するのはよいアイディアであろう。運行上の余裕を用いた饋電システム最適化のアイディアは次のように数多くあり，その効果も饋電システムだけの制御による最適化のそれより大きいと見られる。

⟨4.3.1⟩ 列車ダイヤ上の余裕の融通による饋電系の救済

列車ダイヤには必ず**余裕**があるが，この余裕をうまく利用して電力システムの救済を図ることができる。例えば，ある路線の変電所容量が変電所事故などにより局所的に不足している場合には，列車のダイヤ上の余裕時分を再配分して変電所容量が足りない区間では列車速度を抑制する制御により，サービスレベルを落さずに大幅なピーク抑制を図る可能性がある。

これ以外に，定常運転時には余裕時分を正確に「使い切る」制御や，着発（回生車・力行車）の競合をうまく作るように意図的にダイヤを小変更する可能性もある。

⟨4.3.2⟩ 列車運行乱れ時の制御の工夫

列車運行乱れ時の列車制御を工夫することによっても，省エネルギー化などを図る余地が残されている。

例えば，運行乱れが発生した結果として前方で大幅な減速を余儀なくされる場合，そのことを予め考慮して減速運転をすることによる省エネルギー化が考えられる。また，運行乱れ時にはすべての列車が回復につとめるよりは，特に遅れている1列車が回復につとめ，他の列車はむしろ遅めに走って列車群としての動きが正常に戻る（列車間隔が均等になるなど）ようにするほうが結局遅延回復に要するエネルギーも少なくてすむので，そのような制御が考えられる。

また，上記のように当面回復運転が不必要な列車が回復運転でむだに大きなパワーを消費することがないようにし，特に回復運転が必要な少数の列車には必要な電力を供給するように工夫すると，電力設備の容量の制約のなかで回復余力を増すことも可能になろう。

あるいは，列車運行乱れ時は回生率を犠牲にしても電圧を上げ，列車性能を上昇させて回復力を高める，なども可能になろう。

このほか，IC カードによる個別案内方式をうまく利用し，駅における乗客流の制御を行い，列車の運行乱れの拡大要因をとりのぞくことで，回復を容易化することも考えられる。

III

準備:
饋電特性シミュレーションプログラム
“RTSS”

5

シミュレーションモデルの考え方

本研究で筆者が作成，および使用したシミュレーションプログラムは，本研究のみならず直流饋電システムのシミュレーションに一般的に応用可能な汎用性を持たせたもので，**RTSS**[60](Railway Total System Simulator) として学外に紹介され，実際にシミュレーションによる検討にも用いられている[71][72][73][74]。

この種のプログラムは，直流饋電システムの**運転電力シミュレーションプログラム**または**饋電特性シミュレーションプログラム**と呼ばれる。本論文では，以下後者の呼び名を用いる。なお,「シミュレーションプログラム」というかわりに**シミュレータ**と呼ぶこともある。

饋電特性シミュレータ RTSS では，列車の性能が電車線電圧によって変化するモデルを取り込みながら，従来の同種のプログラムでは実現していなかった駅間走行時分一定化シミュレーションモデルをはじめて盛り込んだ。この他の部分も，過去の論文[1][2][3][4][5] において得られた知見，およびこのプログラムによる研究の過程で得られた数多くのノウハウが生かされ，安定して動くプログラムに育てられている。このプログラムによって，直流饋電システムのシミュレーションの信頼度は大きく向上させることができ，精度の高い議論が可能となったと考えている。

なお，以下において，従来のプログラムとは基本的に文献 [19] に紹介されている方法によるシミュレーションプログラムのことを指す。

⟨5.1⟩　正しいシミュレーションモデルの必要性

⟨5.1.1⟩　饋電特性シミュレーションプログラムとは

4章にて述べた多くのアイディアの検証には，実験などの手段は利用できず，シミュレーションが不可欠である。電力系統のシミュレーションと似ているところがあるが，これと決定的に違うのは，列車群の動きのシミュレーションを電力計算と同時に行わなければならないこと，また列車群の動きにともなって饋電等価回路そのものが変化することである。

饋電システムのシミュレーションにおいて，与える条件は

(1) 路線条件（勾配・速度制限・駅配置・変電所配置など）
(2) 列車性能条件
(3) 列車ダイヤ条件

である。また，出力されるのはさまざまな**饋電特性評価量**である。それらは次のようなものである。

(a) 変電所入力エネルギー
(b) パンタ点入力エネルギー・パンタ点回生エネルギー・列車消費エネルギー
(c) 饋電線損失
(d) 列車回生率（回生率）・回生失効率
(e) 変電所ピーク電流・変電所最高電圧・変電所最低電圧
(f) 変電所電流ヒストグラム・パンタ点電圧ヒストグラム
(g) 総力行状態時間・総加速時間・回生失効時間
(h) 変電所RMS電流

饋電特性評価量のうち(a)・(b)は，饋電系統の等価回路の演算結果のうち，電力を時間で積分することによって得られる。(c)は(a)と(b)の差として求めることができる。(d)も(b)から計算で求める。

また，(e)はシミュレーションを行う時間内で計算された電圧・電流の最大・最小値をその値とする。(f)・(g)は，時間を積分することによって得られる。

最後に，(c)は電流の2乗を時間で積分し，その値を処理することによって得られる。

⟨5.1.2⟩ 饋電特性評価量の定義

ここで，いくつかの饋電特性評価量を定義しておこう。

⟨5.1.2.1⟩ 変電所入力エネルギー 変電所入力エネルギーとは変電所からみた饋電システム全体の消費エネルギーであり，饋電システムの評価の基本となる。以下では，特に断らない限り「変電所入力エネルギー」といえばこの定義に従うことにする。

ただし，細かくいうならば，変電所入力エネルギーといった場合，路線に複数存在する変電所の入力エネルギーの総和を指す場合と，1つの変電所あたりの入力エネルギーを特に全体の総和と区別していう場合とがある。後者の場合，全体の総和を**全変電所入力エネルギー**などと呼んで区別する。また，回生インバータがあり，直流饋電システムから交流側への電力の逆流ができるようになっている場合，以下に定義するパンタ点入力／回生エネルギーのように，変電所入力エネルギーと**変電所回生エネルギー**とを区別して呼ぶことがある。その場合，両者の差，すなわち全体として饋電システムが消費するエネルギーを**変電所総合入力エネルギー**などと呼ぶ。

⟨5.1.2.2⟩ パンタ点入力／回生・列車消費エネルギー パンタ点入力エネルギーとは列車のパンタ点を通じて饋電システムから列車に与えられたエネルギーの総計である。補機電流がゼロの場合，このエネルギーは列車の力行時に饋電システムから列車が受けとるエネルギーに等しい。

逆に，パンタ点回生エネルギーとは列車のパンタ点を通じて列車から饋電システムに戻されたエネルギーの総計である。補機電流がゼロの場合，このエネルギーは列車が回生時に饋電システムに返却するエネルギーに等しい。

そして，列車消費エネルギーとはパンタ点入力エネルギーとパンタ点回生エネルギーとの差に相当する。

列車消費エネルギーについては，すべての列車についての合計で議論することが多い。この合計値を1列車ごとの列車消費エネルギーと特に区別する場合，**総列車消費エネルギー**と称する。

⟨5.1.2.3⟩ 饋電線損失 饋電線損失は，文字通り饋電線における送電損失（オーム損）のことである。シミュレーション上は，全変電所入力エネルギーと総列車消費エネルギーとの差として求めることができる。

⟨5.1.2.4⟩ 回生率 （列車）回生率とは

$$(\text{回生率}) = \frac{(\text{パンタ点回生エネルギー})}{(\text{パンタ点入力エネルギー})} \times 100[\%] \tag{5.1}$$

で定義される量である。

ちなみに，変電所に回生インバータがおかれている場合，**変電所回生率**を

$$(\text{回生率}) = \frac{(\text{変電所回生エネルギー})}{(\text{変電所入力エネルギー})} \times 100[\%] \tag{5.2}$$

で定義することもある。

⟨5.1.2.5⟩　回生失効，回生失効時間，回生失効率　回生失効という言葉は，一般に回生絞り込み終了電圧以上に列車のパンタ点電圧（正確にはフィルタコンデンサ電圧）が上昇し，回生中に主回路が開かれる（回生ブレーキが完全に無効になる）こと，またはその状態をいう。本論文では，これを後述の広義の回生失効と区別するために**狭義の回生失効**と呼ぶことにする。

当然狭義の回生失効は少ない方がよい。そのような観点から，回生失効をおこしていなければ回生ブレーキがかかったはずである，と期待される時間の総計を回生失効時間と定義し，評価量として用いる。

一方，狭義の回生失効，すなわち回生ブレーキ力が完全に zero の状態にはならなくとも，回生絞り込みにより回生ブレーキが「半開き」になる，つまり列車の電力回生能力を100%まで生かしきれなくなるケースがある。このようなケースを含めて回生失効と呼ぶ考え方を，本論文では特に**広義の回生失効**と呼んで区別する。この場合，回生失効は時間ではなく回生失効率という比率で評価する。

回生失効率の定義のしかたはいろいろだが，列車が走行する全区間にわたって電力を時間で積分してエネルギーとし，

$$(\text{回生失効率}) = 1 - \frac{(\text{パンタ点回生エネルギー})}{(\text{回生可能エネルギー})} \tag{5.3}$$

とするのが一般的である。回生可能エネルギー回生可能エネルギーとは，回生絞り込みがなかったならば架線に返っていた「はず」の回生エネルギーのことを指す。

⟨5.1.2.6⟩　変電所ピーク電流，変電所最高／最低電圧　変電所ピーク電流とは，シミュレーションを行っている間に，変電所に流れた電流の最大値をいう。最高電圧・最低電圧も同様に定義される。

⟨5.1.2.7⟩　変電所電流／パンタ点電圧ヒストグラム　シミュレーションを行っている間について統計をとれば，変電所電流，またはパンタ点電圧がある範囲の値になる確率を求めることができる。これをグラフ化すればヒストグラムになるが，このヒストグラムを作成するためのデータを出力する機能も備えている。データは時間を積分することによって容易に得ることができる。

⟨5.1.2.8⟩　変電所RMS電流　変電所電流の2乗平均平方根（Root Mean Square）値である。変電所の機器容量を決定するのに用いることができる。

⟨5.1.2.9⟩　総力行状態時間・総加速時間　力行性能が低下すれば，同一の駅間走行時分を維持するためには力行する時間を長くしなければならない。このことを評価するための評価量が総加速時間と総力行状態時間である。総加速時間は「定速走行」も含め列車が正の加速力を出している時間の総計である。⟨9.2.2⟩（58ページ）に述べる**フルノッチ比**を用いると，フルノッチ比が正である時間の総計と定義することもできる。これに対し，総力行状態時間とは力行状態の時間の総計である。フルノッチ比で表現するなら，フルノッチ比が1である時間の総計であるということもできる。

⟨5.1.3⟩　饋電特性シミュレーションプログラムのおおまかな構造

RTSS も既存のシミュレータも，ごく基本的でおおまかな考え方および構造については共通である。

ある瞬間に饋電回路に流れる電流を求めるためには[19]，

1. 所定のダイヤに従って列車群の配置や，速度・状態を求める。
2. 饋電用変電所・電車線路からなる饋電系統の対応する位置に，上記の電気車を配置して等価回路を作成する。
3. この等価回路を解いて，電圧・電流分布および各種電力を求める。

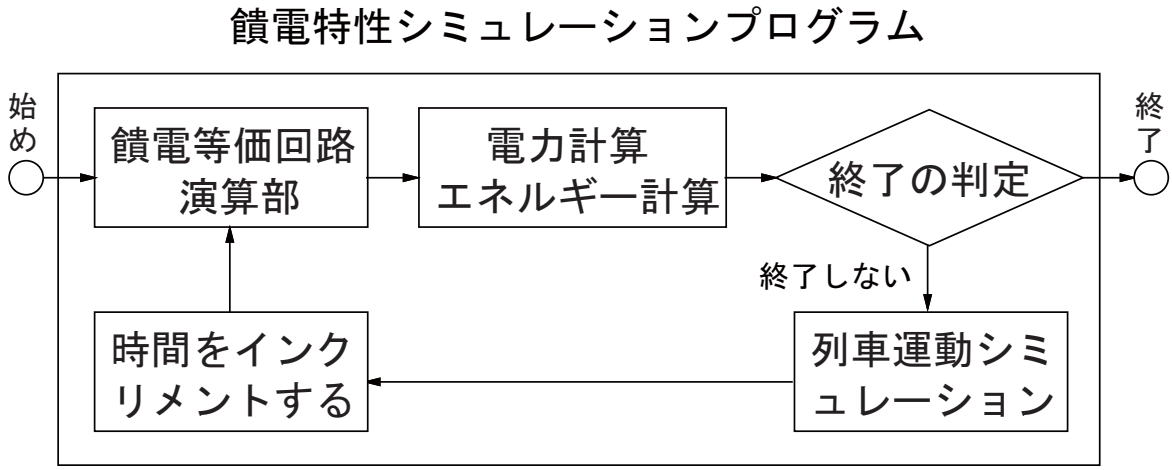


図 5.1: 饋電特性シミュレーションプログラムのおおまかな構造

という手順をとればよい。饋電特性シミュレーションプログラムは，上記の手順1〜3を Δt 時間間隔ごとに，連続的に繰り返し行う必要がある。おおまかな饋電特性シミュレーションプログラムのフローを図5.1に示す。

⟨5.1.4⟩　従来のシミュレーションモデルの問題点

従来のシミュレーションモデルでは次のような問題があった。

⟨5.1.4.1⟩　列車ダイヤ条件が精密に実現できない　⟨5.1.1⟩において，シミュレータに与えるべき条件を述べた。このうち条件(2)については，なるべく実際の列車の特性に近いモデルを入れようとするならば，列車の動力性能は電車線電圧に依存するモデルとしなければならない。

ところが，そのような列車モデルをインプリメントすると，条件(3)を正確に実現することが困難になる。電車線電圧はシミュレーションを実行してみなければわからないし，次数の大きな行列を多数回演算する必要もあるため，簡単に予測することも難しい。すなわち，列車の力行性能は走ってみるまでわからなくなるのだ。力行性能が予めわからないと，駅間のどこで列車をノッチオフさせれば目的駅に時間通り着くのかを知ることはできなくなる。

このため，従来のシミュレーションプログラムではこの条件（とりわけ駅間走行時分条件）を厳密に実現することは実質的に放棄してしまっている。プログラムによっては，⟨5.1.1⟩の条件(2)について列車の動力性能が電車線電圧に依存しないモデルとすることによって条件(3)を実現しているものもあるが，これも同様である。

この列車の動力性能の電圧依存性による影響は，日本の大都市の一般的な通勤鉄道モデルにおいては，駅間走行時分の誤差でいって1〜数秒オーダの小さなものだ。しかし，このオーダの走行時分の変化であっても，列車消費エネルギーはかなり大きく変化することが知られている[28]。

チョッパ車の計算例を見ると，オフブレーキ運転時の駅間走行時分が97秒，これに対して比消費電力量は約56Wh/t/kmである。ところが，駅間走行時分を100 秒に約3%伸ばすだけで，比消費電力量は約40Wh/t/kmへと激減（−29%）する。オフブレーキ運転に近い駅間走行時分では，このように走行時分のエネルギーに対するパラメータ感度が非常に高いのである。

同じ計算例で駅間走行時分を100秒から110秒へ10%伸ばした場合，比消費電力量は40Wh/t/kmから28Wh/t/kmへと約30%減少する。97→100秒への変化ほどは減少は劇的ではないが，まだパラメータ感度は高い。しかし，110 秒から120秒へ約9%伸ばした場合，比消費電力量は 28→25Wh/t/km と約 11% 減少するにすぎなくなる。

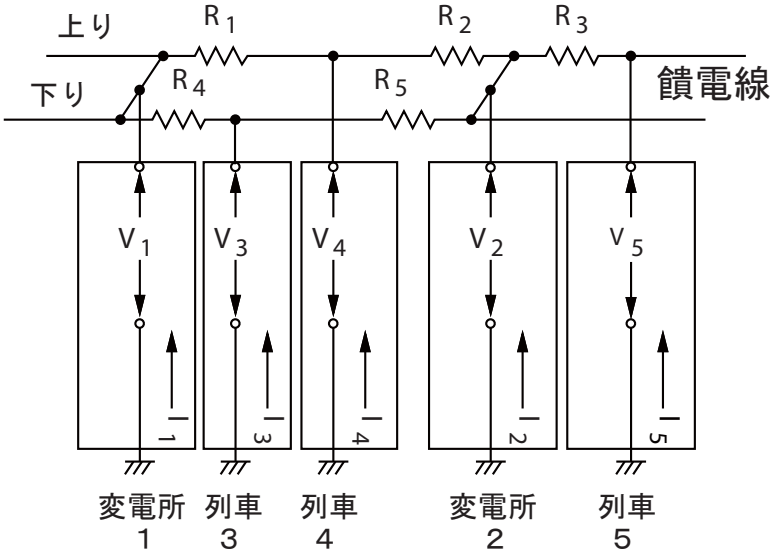


図 5.2: 饋電等価回路の一例

このように，オフブレーキ運転に近い走行時分になればなるほど，わずかの走行時分の減少で大きなエネルギー低減効果が現れることがわかる。通常のダイヤでは最短運転時間より10%程度の余裕を見ているから，駅間走行時分のパラメータ感度は比較的大きな領域で使われていることが，このデータからもわかる。

列車消費エネルギーが大きく変化すれば，最終的に出力される饋電特性評価量のうち(a)や(b)などにかなり大きな影響を及ぼすとみなければならない。そこで，電車線電圧依存の動力特性を正確に模擬しつつ，駅間走行時分を高精度に一定に保つシミュレーション技術を開発する必要がある，と考えられた。

⟨5.1.4.2⟩ 等価回路演算の問題 直流電気鉄道の饋電システムを詳細に見ると，図3.1・3.2（10ページ）のようになっている。饋電線と電車線（架線）とは別になっており，ざっと200〜300メートル間隔で「饋電分岐線」によって結ばれている。RTSS で使用する等価回路上ではこれらのことは無視し，

- 饋電線と電車線は一体であると仮定
- レールの抵抗値と饋電線の抵抗値の和の値となる抵抗を饋電線側に置くだけとし，等価回路上は変電所・列車とも地上側はすべて接地する
- 列車の長さは考慮しない
- 電圧・電流は定常解だけを求めるようにし，過渡解析および高調波解析は行わない

などの簡略化を行う。こうすると，等価回路は図5.2に示すように簡略化される。列車・変電所以外はすべて抵抗だけのネットワークである。この簡略化によって饋電等価回路の演算はだいぶ容易になる。なお，この等価回路は鉄道の現場で計算に用いられているものと基本的に同一である。もちろん，この程度の簡略化で特に正確さに欠けたものになるなどということはない。

ここまで簡略化しても，饋電等価回路の演算は相当な苦労が伴う。これは，変電所が電流の逆流を許さない，または列車が複雑な電流–電圧特性を持つ，などといった非線形性が存在するためだ。これらの非線形性のため，収束演算がうまくゆかない，という問題は頻繁に発生していた。

この他，路線によって饋電回路の形状がいろいろで，シミュレーションを行う際に取り扱いにくい，という問題も残っていた。

RTSS ではこの部分を改善した。まず，収束演算には Newton-Raphson 法を用いることにして，計算の収束性を抜本的に改善した。また，さまざまな饋電回路の形状を取り扱えるようデータ形式を工夫した。このほか，列車モデルも実際の列車の特性になるべく近いものをインプリメントするなどの細かい改良を加えた。

これらの改良の結果，饋電等価回路演算は精度面での不安を大幅に取り除き，シミュレーション結果の信頼性をさらに高めている。

⟨5.2⟩ RTSS の特徴

RTSS は C++ 言語[32][34][35][36]で記述され，ソースコードの行数が1万行以上にのぼるかなり巨大なプログラムである。ここですべての機能について言及することはできないため，プログラムのマニュアルを本論文の付録として添付する。ここでは，特に本文に記述すべきと思われる特徴を述べる。

- 列車の加速性能の電圧に応じた変化によって駅間走行時分が変化しないように，列車が加速をやめる位置を条件に応じて変化させるモデルを開発した。
- 複雑な形態の饋電系統も，データの変更のみでシミュレートできるように配慮した。
- 饋電等価回路の計算には Newton-Raphson 法を用い，計算の収束を改善し，計算不能（収束しない）となる頻度を従来のプログラムより格段に少なくすることに成功した。
- 文献 [39] に提案した，電気車の主回路電力制御によるピークカットの効果を求めるためのルーチンが付加されている。また，変電所のリアルタイム送出電圧制御の効果を求めるためのルーチンも付加されている。

⟨5.3⟩ 饋電等価回路とその演算法

饋電回路の等価回路としては，図5.2のようなものが使われる。この直流饋電等価回路は，変電所・列車をのぞけば抵抗だけを含む回路である。変電所は電圧源＋直列内部抵抗で，列車は電流源でそれぞれ近似することが多いが，このような近似が成立していればこの回路は行列演算1回で解くことができる。

この回路を解く具体的な操作は次の通りだ。まず，変電所および列車の数の合計を N_{CSS} とする。回路のノードアドミタンス行列を Y とすると，これは N_{CSS} 次の正方行列となる。また，変電所および列車の電流・電圧ベクトルをそれぞれ $\boldsymbol{I}, \boldsymbol{V}$ とする。これらはそれぞれ

$$\boldsymbol{I} \equiv \{I_0, I_1, \ldots, I_i, \ldots\}^T \tag{5.4}$$

$$\boldsymbol{V} \equiv \{V_0, V_1, \ldots, V_i, \ldots\}^T \tag{5.5}$$

（ただし i は変電所／列車の一連番号で，$0 \leq i < N_{\mathrm{CSS}}$）と表せる。このとき，回路演算とは

$$\boldsymbol{I} = Y \cdot \boldsymbol{V} \tag{5.6}$$

という式を解くことである。

しかし，列車・変電所は非線形性を持っている。例えば列車は速度が低いか電圧が高いときは電力一定負荷に近くなり，逆に速度が高いか電圧が低いときは純抵抗に近くなる。最近はブレーキ時に電動機で発電して電力を架線に返す電力回生ブレーキ付き電気車が一般的になっているが，この回生中電圧が上がると「絞り込み」という制御をかけて電流を抑制するようにしている。一方，変電所は，負荷である饋電システム側から電源側への逆流はできないのがふつうである。

これらの非線形性の存在により，従来のプログラムでは

(1) ある $\boldsymbol{V}_0$ を仮定し，$i = 0$ として (2) へ
(2) $\boldsymbol{V}_i$ と式 (5.6) から $\boldsymbol{I}_i$ を求める

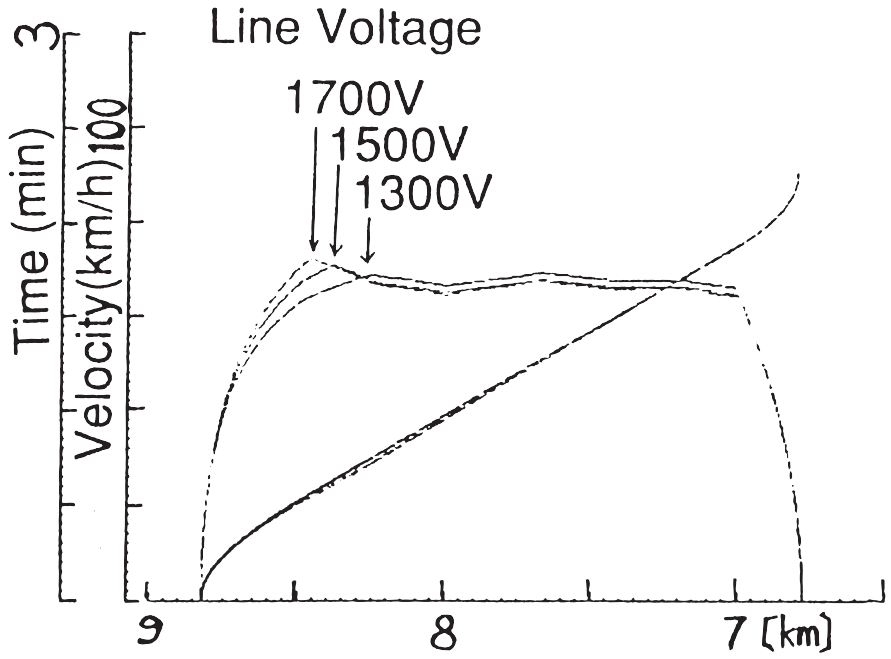


図 5.3: 駅間走行時分一定のシミュレーション例

(3) $\boldsymbol{I}_i$ と列車・変電所の特性から $\boldsymbol{V}_{i+1}$ を求める
(4) $\boldsymbol{V}_i$ と $\boldsymbol{V}_{i+1}$ との差が収束判定限界内なら終了，そうでなければ i を一つ増やして (2) へ

という手順を踏んでいた。しかし，この方法では収束がきわめて悪く，解けないケースも多かった。

RTSS では，この部分には多変数の Newton-Raphson 法を用いている。まず，それぞれの列車・変電所の特性は V–I ないし I–V 平面上で 1 本の曲線として表示されると仮定しよう。このとき，曲線は 1 つの媒介変数を用いて記述できるはずである。1 列車または 1 変電所あたり 1 つの媒介変数が必要である。列車または変電所 No. i（$0 \leq i < N_{\mathrm{CSS}}$）の特性を表す媒介変数を θ_i，それを集めて $\boldsymbol{V}, \boldsymbol{I}$ と同様にベクトルとしたものを $\boldsymbol{\theta}$ とすると，

$$\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{\theta}) \tag{5.7}$$

$$\boldsymbol{I} = \boldsymbol{I}(\boldsymbol{\theta}) \tag{5.8}$$

と書き表せる。こうすると，式 (5.6) を満たす $\boldsymbol{I}, \boldsymbol{V}$ を求める問題は

$$\boldsymbol{I}(\boldsymbol{\theta}) - Y\,\boldsymbol{V}(\boldsymbol{\theta}) \equiv F(\boldsymbol{\theta}) = 0 \tag{5.9}$$

を解いて $\boldsymbol{\theta}$ を求める問題に帰着できる。これを Newton-Raphson 法により解けば解が求まる。

この方法でも計算不能となる場合がまだ残るが，従来のプログラムよりは格段に少なく，また繰り返し計算の回数も減少し，計算時間の短縮が図られた。

⟨5.4⟩ 駅間走行時分を高精度に一定とするシミュレーション技法[40][45]

従来のプログラムは，列車の運転曲線（速度—位置曲線）そのものをデータとして与えて列車の駅間走行パターンを指定している。この場合，例えば列車が加速をやめる位置ないし速度が指定されることになる。この方法でも列車の性能が一定であれば問題はないが，現実の列車はパンタ点の電圧によって性能が変化する。これをモデルに取り込むと列車の駅間走行時分が条件に合わなくなってしまう。

これを防いで駅間走行時分を高精度に一定化するためには，加速をやめる位置・速度を条件に応じて求めるようにすればよい。このモデルを取り込んだ RTSS のシミュレーションモデルを，特に駅間走行時分一定化シミュレーションモデルと呼ぶ。

駅間走行時分一定化シミュレーションの方法は簡単である。パンタ点電圧は前もって予測することが難しいが，列車が加速していないとき（惰行・ブレーキ）には列車の運動はパンタ点電圧と無関係である。特にブレーキ時の性能は，回生ブレーキ性能が饋電システムの状態に強く依存するが，回生ブレーキ力が不足した場合空気ブレーキで自動的に補足されて合計のブレーキ力は一定に制御される。

このことを利用すれば，加速中のある瞬間に列車が加速をやめ，あとは目標地点まで加速せずにゆくと仮定した場合の，目標地点までの到着時刻は計算できる。そこで，この時間が所定の時間より短くなるまで加速を続けるようにプログラムすればよい。

電車線電圧が変化し，列車の性能が変化しても，この方法で駅間走行時分を高精度に一定に保った場合のシミュレーションが可能である。列車の性能変化に応じて加速をやめる点が変化するようすを図5.3に示す。

6

新しいシミュレーションモデルの評価

⟨4.1.2⟩にて述べた通り，回生車を含む直流饋電システムでは，力行車負荷だけだった場合に比べれば**無負荷時送出電圧**を下げることで省エネルギー化を図ることができる。このことに着目し，数多くの研究がこれまでに発表されてきた[19]。

ところが，過去の研究では，⟨5.1.4⟩（22ページ）にて指摘した問題のあるシミュレーションツールを使用していた。特に**駅間走行時分**の条件を必要な精度で与えないまま議論を行っていた。このため，電圧の低下による列車性能の低下の影響，すなわち列車が遅く走ることによる変電所入力エネルギーの低下と，無負荷時送出電圧の最適化による省エネルギー効果とが分離できず，シミュレーション結果の信頼性を著しく下げていた。このことに注目し,「実は無負荷時送出電圧の最適化による効果というのはすべて列車が遅く走る効果なのであり，まやかしではないか」という批判もあったが，従来のモデルでは十分な説得力をもってこれに反論できなかった。

駅間走行時分一定化シミュレーションと，その他の機構を盛り込んだ RTSS は，初めてこのような批判・疑問に対し十分な説得力をもった反論の材料を与え得たシミュレーションプログラムであるといってよい。

ここでは，RTSS に盛り込まれた新しいシミュレーションモデルの妥当性の評価を，いろいろな角度から行う。

⟨6.1⟩　実際のシステムとの比較

5章で述べたプログラム RTSS を，実際の饋電システムと比較・評価する機会を得た[73][74]。ここで比較した実際のシステムは，自動運転となっているが，まだ部分開業であって乗客数が少ない，地下通勤線区 A線 である。路線長は約6.5km，駅は6箇所，変電所は2箇所あり，変電所のうち1つに回生電力吸収装置として回生インバータが設置されている。

このような路線について，測定結果とシミュレーション結果を比較した。表6.1は，変電所特性を所定の条件にした場合について，全変電所入力エネルギー（全変電所のコンバータを通じ饋電システムに供給されたエネルギー），全変電所回生エネルギー（回生インバータを通じ饋電システムから高配負荷に供給されたエネルギー），それらの差となる全変電所総合入力エネルギー，路線の末端に当たるC駅構内の電車線電圧最高値・最低値について，シミュレーション結果と実測値を比較したものである。

測定値はある日1日のみの測定値を比較しているが，測定値には測定日によって変わる停車時分のばらつきなど，大きな誤差要因が入っているため，細かな数字まで合わせることは不可能だと考えられる。ただし，この路線は部分開業状態であり，乗客が非常に少ないため幸いにしてこの影響も他線に比べると相当少ないと考えられる。このことを踏まえつつ比較すると，電力関係の絶対値に違いがあるものの，無負

荷時送出電圧やインバータのオン・オフと各評価量との関係など，大まかな傾向は正しいと評価できる。

ただし，最高電圧だけは**狭義の回生失効**時に過渡的に出る高い値をシミュレーションモデルが再現できないため，実測定との違いが明確になっている。⟨5.1.4.2⟩（23ページ）で述べたように，回路の過渡解析を行わないモデルであるため，このこと自体は問題ではない。実際のシステムにおいては，車両はフィルタコンデンサの電圧が急上昇するのを検知して主回路を開き，回生失効状態になるが，このさい遮断器の動作遅れの影響でこのような高い電圧が出るらしい。回生インバータが動作しているケースにおいては，狭義の回生失効はおきないので，最高電圧の計算結果と測定値とはよい一致を見ている。

また，同じ測定・シミュレーションにおいて列車消費エネルギー関係の評価量，すなわちパンタ点入力エネルギー，パンタ点回生エネルギーおよび回生率の比較を行ったものを表6.2に示す。

測定値は測定日の特定1個編成のエネルギーを集計したもの，シミュレーションは全編成の平均をとったもの，をそれぞれ路線1往復分について比較している。細かな数字の差があるものの，大まかな傾向は正しいと評価できる。

変電所特性が所定のケースのシミュレーションと測定値の比較では，以上の各評価量はおおむねよい一

表 6.1: 実測値とシミュレーションとの比較(1)

No.	無負荷送出	インバータ	変電所入力	変電所回生	総合入力	C駅最高	C駅最低
シミュレーション結果							
S1	1590.0 V	on	1092.2 kWh	175.9 kWh	916.3 kWh	1646.3 V	1438.8 V
S2	1521.0 V	on	1090.7 kWh	183.2 kWh	907.5 kWh	1577.5 V	1360.7 V
S3	1590.0 V	off	1133.3 kWh	—	1133.3 kWh	1674.4 V	1438.8 V
S4	1521.0 V	off	1129.7 kWh	—	1129.7 kWh	1674.4 V	1356.0 V
測定値							
M1	1590.0 V	on	1230 kWh	160 kWh	1070 kWh	1640 V	1432 V
M2	1521.0 V	on	1170 kWh	175 kWh	995 kWh	1588 V	1295 V
M3	1590.0 V	off	1200 kWh	—	1200 kWh	1895 V	1391 V
M4	1521.0 V	off	1210 kWh	—	1210 kWh	1902 V	1340 V

表 6.2: 実測値とシミュレーションとの比較(2)

No.	無負荷送出	インバータ	列車力行	列車回生	回生率
シミュレーション結果					
S1	1590.0 V	on	107.1 kWh	32.7 kWh	30.5 %
S2	1521.0 V	on	106.7 kWh	33.2 kWh	31.1 %
S3	1590.0 V	off	107.8 kWh	15.1 kWh	14.0 %
S4	1521.0 V	off	107.1 kWh	15.0 kWh	14.0 %
測定値					
M1	1590.0 V	on	99 kWh	31 kWh	31 %
M2	1521.0 V	on	104 kWh	36 kWh	35 %
M3	1590.0 V	off	102 kWh	17 kWh	17 %
M4	1521.0 V	off	101 kWh	12 kWh	12 %

表 6.3: 実測値とシミュレーションとの比較(3)

No.	無負荷送出	インバータ	変電所入力	変電所回生	A 変電所	B 変電所
B 変電所無負荷時送出 +30V, 4% 電圧変動率						
V1	1590.0 V	on	1105.7 kWh	185.2 kWh	381.3 kWh	724.4 kWh
V2	1521.0 V	on	1108.8 kWh	192.8 kWh	389.9 kWh	718.9 kWh
ノミナル条件（A・B変電所とも無負荷時送出電圧同一，6% 電圧変動率）						
S1	1590.0 V	on	1092.2 kWh	176.2 kWh	638.8 kWh	453.3 kWh
S2	1521.0 V	on	1090.7 kWh	183.2 kWh	640.4 kWh	450.3 kWh
測定値						
M1	1590.0 V	on	1230 kWh	160 kWh	430 kWh	800 kWh
M2	1521.0 V	on	1170 kWh	175 kWh	410 kWh	760 kWh

致をみていたが，ふたつの変電所（A・Bとする）ごとのエネルギー分担は大小関係が逆になっていた。表6.3は，このことを詳細に検討したものである。

ダイオード変電所では，コンバータに電圧調整能力がないため，コンバータ入力電圧が何らかの原因で高いとそれがそのまま変電所送出電圧の差となって表れる。測定結果などから，B 変電所は A 変電所よりも常に 30V （定格電圧に対し 2%）ほど電圧が高く，電圧変動率も少なめになっていることがわかったので，この条件を入れてシミュレーションを再び行った。この結果，電力分担もシミュレーションによってほぼ再現できることがわかった。シミュレーションでは変電所の特性として所定のものを入れて行ったが，測定したシステムの変電所特性が所定のものからごくわずかながら外れていたために，電力分担のシミュレーションによる再現ができなかったものと判断できる。

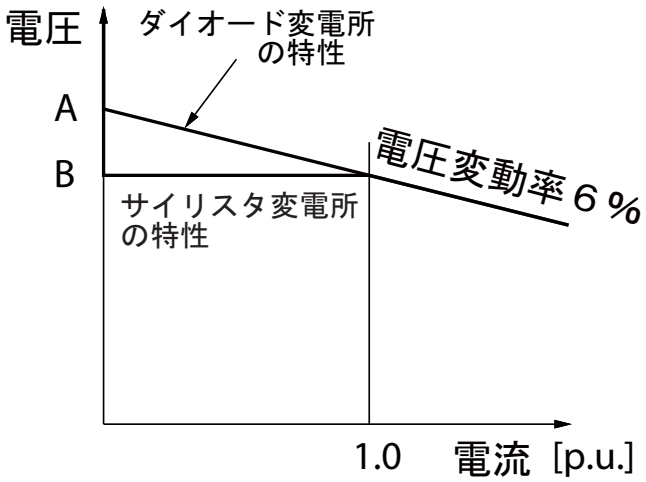
⟨6.2⟩ 既存のシミュレーションプログラムとの比較

5章で述べたプログラム RTSS を，コンセプトの異なる饋電特性シミュレーションプログラムと比較・評価する機会を得た[73][74]。比較したシミュレーションプログラムは，メーカA社で開発・維持されているシミュレーションプログラムである。現在のところ RTSS は直流饋電システムの評価専用だが，A社のプログラムは交流饋電システムもシミュレートできるもので，RTSS より汎用性は高い。このプログラムには，RTSS が持つような駅間走行時分を一定化する機能は含まれていないため，列車は電圧の変化にともなって所定ダイヤより遅れたり進んだりする。このA社のプログラムと RTSS とでシミュレーション結果の比較を行った。

取り上げた路線は，⟨6.1⟩（27ページ）で検討したのと同じ地下通勤線区 A線 である。ただし，⟨6.1⟩では現状の部分開業時を想定したモデルでシミュレーションを行ったが，ここでは全線開業時を想定したモデルにてシミュレーションを行った。路線長は約21.3km，駅数は 19，変電所数は6箇所（部分開業時2）で，このうち3箇所（部分開業時1）に回生インバータが設置されている。なお，このモデルについてのより詳しい記述は⟨7.1⟩（35ページ）にもあるので参照されたい。

このモデルにおいて，変電所の饋電用変圧器の1次側タップを3点選び（無負荷時送出電圧を3点選んだのと等価），変電所入力エネルギーの変化を比較した。

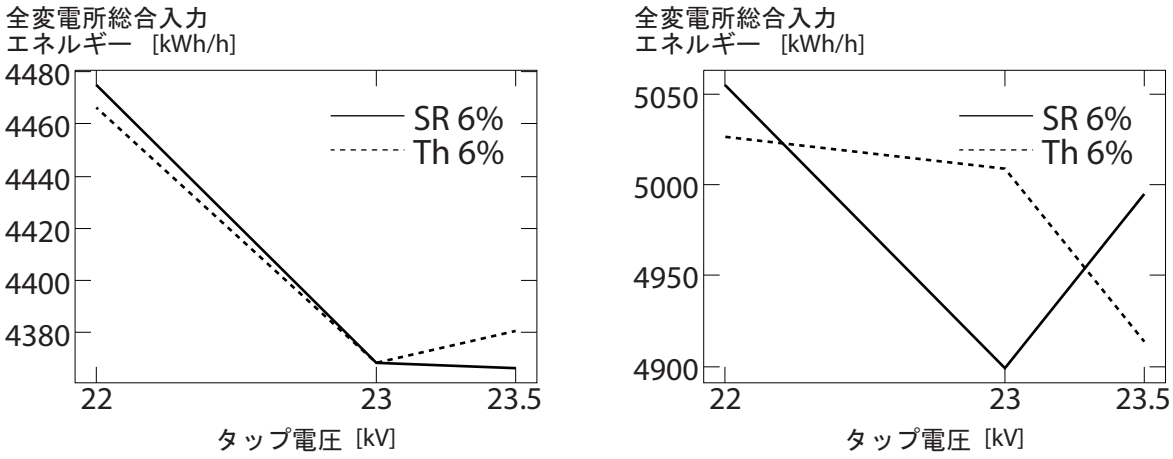
検討においては，全変電所をダイオード変電所とした場合のほか，新規開業となる区間に設置する4変電所についてV–I特性の異なるサイリスタ変電所を混在させるケースも同時に検討した。ダイオード変電所，サイリスタ変電所の V–I特性の比較を図6.1に示す。ダイオード変電所では無負荷時送出電圧が図中 A に示した電圧値となり，電流 0p.u. から電圧変動率一定の直線の特性となるが，サイリスタ変電所では



電圧 A・B の値：

変電所1次側 タップ	A 電圧値	B 電圧値
22 kV	1590 V	1500 V
23 kV	1521 V	1435 V
23.5 kV	1489 V	1404 V

図 6.1: ダイオード変電所・サイリスタ変電所の V–I 特性比較



(1) RTSS による結果　　(2) A 社シミュレータによる結果

図 6.2: 全変電所総合入力エネルギーのシミュレーション結果比較

1p.u. 以下では制御をかけて電圧変動率ゼロ，すなわち定電圧の特性となる。定電圧領域の電圧は図中 B となる。電流が 1p.u. 以上ならばダイオード変電所とサイリスタ変電所の特性に差はなくなる。

このような特性をいれてシミュレーションを行った結果のうち，全変電所総合入力エネルギー（コンバータを通り饋電システムに供給されたエネルギーと，インバータを通り饋電システムから高配負荷に供給されたエネルギーとの差）を図6.2に示す。グラフは横軸が変電所1次側タップ電圧で表示されているので，横軸に 22kV とある点が送出電圧のもっとも高いケースに相当する。また，図中 SR6% とあるのはすべてダイオード変電所とした場合，Th6% とあるのは新設変電所をサイリスタ変電所としてダイオード変電所とサイリスタ変電所とを混在させた場合を表す。

定性的には，送出電圧を下げて行くと，列車の回生絞り込みにかかることなく回生電力を遠方の負荷に供給できるようになるため，回生失効率が下がり回生率は上がる。一方，力行時間が長くなるため力行に要するエネルギーが増大し，同一の電力であれば電流が増加するため饋電線損失も増大する。これらのことから，変電所総合入力エネルギー–送出電圧曲線を描くと下に凸の曲線となるはずだ。

RTSS によるシミュレーション結果は，変電所がサイリスタかダイオードかによらずほぼ同じような傾向を示しており，定性的な説明にもよくあっている。一方，A 社のシミュレーション結果は両者の傾向が

かなり異なっており，定性的な説明もしにくいものになっている。これは，A 社のシミュレーションプログラムが駅間走行時分一定化機能を持っておらず，列車の駅間走行時分や列車群の位相がシミュレーションケースによって変動するためである，と解釈できる。

A 社のプログラムの饋電回路の計算などに問題があるわけではなく，例えば変電所ごとの出力分担の結果は RTSS と極めて似通った結果になる。したがって，大きなバグがこのプログラムに存在するとは考えにくいので，結果の違いは駅間走行時分一定化というシミュレーションの基本原理の差に帰着するのが妥当と考えられる。

このように，RTSS が列車の駅間走行時分の変動をなくすことによって，信頼できる饋電特性評価量を出力できていることがわかる。

⟨6.3⟩　駅間走行時分と変電所入力エネルギーの関係

以上のようにシミュレーションプログラム同士を比較することでも RTSS のこの種の検討における優位性が明らかになった。しかし，駅間走行時分のわずかの変化が列車消費エネルギーに大きな影響を及ぼすことはわかっている[28]ものの，駅間走行時分と変電所入力エネルギーとの関係は実際どのようになるのだろうか。その関係の検討を行った。

ここでは，⟨6.2⟩と同一の路線モデルを用いて検討を行った。その 1 例を図 6.3に示す。全駅間について一律に，駅間走行時分を ± 0 秒（所定と同じ）〜 − 3 秒までの範囲で変化（短縮）させ，**全変電所入力エネルギー**の変化を見たものだ。それぞれの駅間走行時分について，3 つの無負荷時送出電圧値についてシミュレーションを行った。変電所はすべてダイオード変電所を仮定している。駅間走行時分は駅間によりばらつきがあるが，所定（± 0 秒変化）のケースでは平均 93.2 秒となっている。同図 (6) の横軸は駅間走行時分の全線平均値をとっているが，± 0 秒変化（同図 (1)）が 93.2 秒，− 1 秒変化（同図 (2)）が 92.2 秒，などと対応する。

これでみると，同図 (6) にあるようにわずか 3 秒駅間走行時分を早めるだけで 1 割以上も入力エネルギーが変化していることがわかる。しかも，駅間走行時分を短くすればするほど，すなわちオフブレーキ運転の時分に近づけば近づくほど，走行時分の変電所入力エネルギーに対するパラメータ感度が高くなっている。

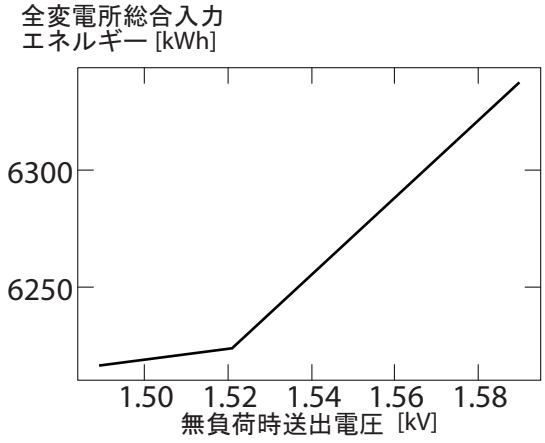
駅間走行時分と列車消費エネルギーの関係（⟨5.1.4.1⟩, 22ページ，または文献 [28] 参照のこと）ほど劇的な変化があらわれなかったのは，全駅間一律に駅間走行時分を短縮させているためである。同じ 1 秒短縮であっても，駅間ごとに与えられている余裕時分が異なるなどの要因があり，短縮するための力行時間の増加割合などまちまちになるためだ。それにしても，この変化は大幅であることは間違いない。

従来のプログラムでは駅間走行時分が 1 秒程度狂うことはしかたがないと考えられてきた。しかし，このように駅間走行時分のわずかな変化がエネルギー評価量に大きな影響を及ぼすことから，走行時分条件を高精度に一定化しないとシミュレーションによる饋電システム最適化は行い得ないことが納得されよう。

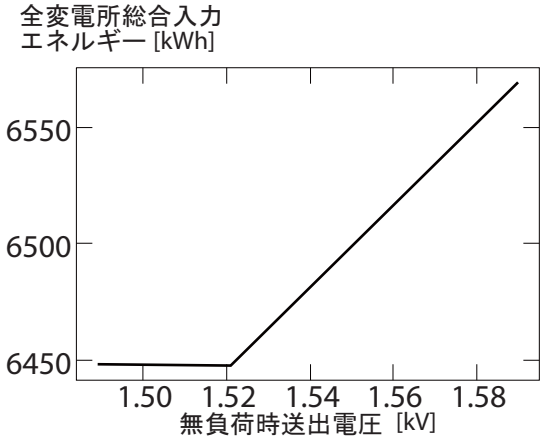
また，駅間走行時分を何秒短縮しても，図 6.3(1)〜(5) のように無負荷時送出電圧とエネルギーの関係の概形に大きな変化がないことに注意しよう。このことは，駅間走行時分の変化による影響を正確に分離しても，なお送出電圧の低下による効果が存在することを明確に示している。

⟨6.4⟩　その他の研究状況

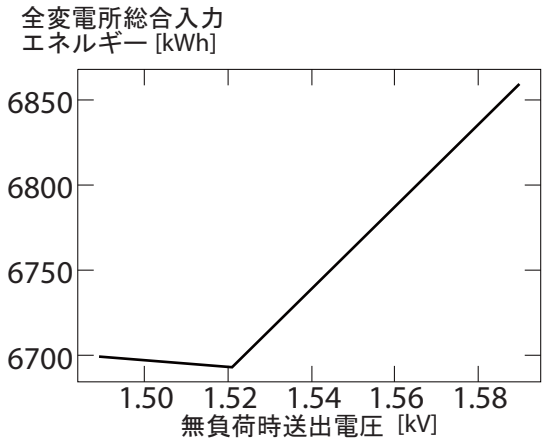
回生車を含む饋電システムにおいて，饋電電圧最適化による省エネルギーは 1980 年代より研究が進められてきた古いテーマである。しかし，つい最近まで筆者の属する東京大学工学部電気工学科・曽根研究室のグループ以外で駅間走行時分一定化シミュレーションによるシミュレーションを行ったグループはなかった。曽根研グループでは，1984 年の論文において松富がノッチオフ速度の指定値を変化させて何ケー



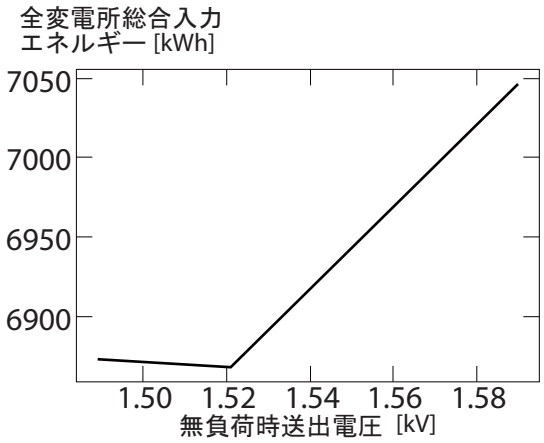
(1) 駅間走行時分 ±0秒



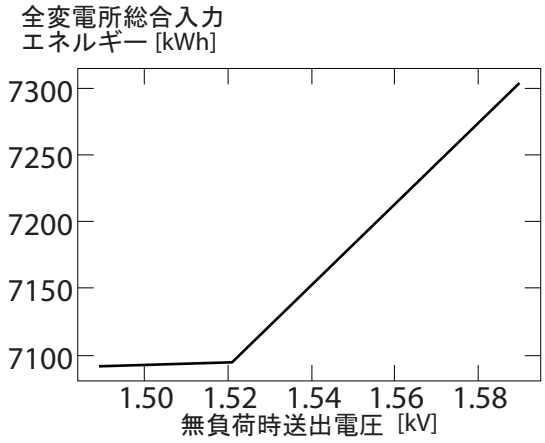
(2) 駅間走行時分 −1秒



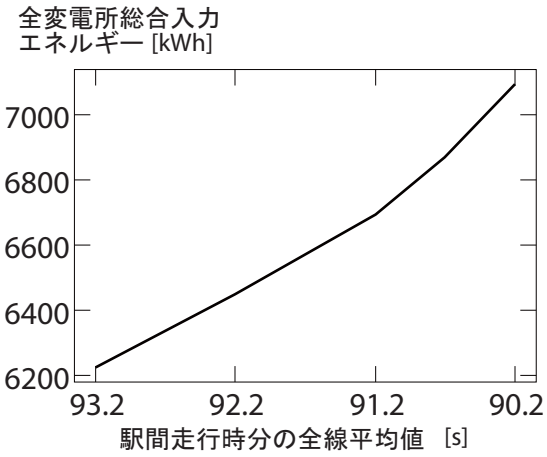
(3) 駅間走行時分 −2秒



(4) 駅間走行時分 −2.5秒



(5) 駅間走行時分 −3秒



(6) 駅間走行時分とエネルギーの関係
(ダイオード変電所の送出電圧 1,521V)

図 6.3: 全変電所総合入力エネルギーと駅間走行時分

スかのシミュレーションを行い，エネルギー評価量の平均速度に対するパラメータ感度を求め，補正する方法による近似的な駅間走行時分一定化シミュレーション手法を開発している[1]。

最近，幸いにして駅間走行時分一定化シミュレーションの必要性およびアイディアが広く認識されるようになり，RTSS で採用した一定化の方法（⟨5.4⟩, 25ページ）ではなくこの松富による方法ながら，駅間走行時分の補正を入れたシミュレーションの報告も現れるようになった[16][17]。饋電システムの最適化の重要性と効果をアピールするために，このようなシミュレーションプログラムが増えることは大変よろこばしいことである。こうして開発されたツールを利用し，数多くの路線についてシミュレーションを進めることで，回生車を含む饋電システムの最適化が進むことを期待したい。

IV

統合化鉄道電力システムにおける
省エネルギー化・設備利用率向上の可能性

7

変電所 V–I 特性の最適化

電力回生ブレーキを常用する電気車，すなわち回生車が増加する以前には，変電所の送出電圧は高めにするのが一般的であった。これは，電車線電圧を高めに維持することによって，列車の性能が維持されること，および饋電線損失が小さくなることによる効果である。しかし，⟨4.1.2⟩（13ページ）で述べたように，変電所の無負荷時送出電圧，あるいは広くいうならば変電所の V–I 特性を最適化することによって，変電所入力エネルギーを最小化する余地が残っている。

回生車がほとんどとなった現状の饋電システムにおいては，回生車の電力が遠くまで到達しうるように電圧は低めにするとよい。しかし，変電所の送出電圧を下げると，列車の性能低下（力行時間の増大）による所要エネルギー増，電流増大による饋電線損失増大などのエネルギー面での欠点が支配的となる。従って，変電所送出電圧には最適値が存在する。

さらに，従来のダイオード変電所に代えてサイリスタ変電所を利用し，図6.1（30ページ）にあるように定電圧領域を持たせる V–I 特性を実現すれば，無負荷時送出電圧は下がるが，重負荷時の送出電圧は高いままに保つことが可能になる。このほか，定電圧領域の電圧を全変電所について揃えておけば，低負荷時に横流が減少し，饋電線損失を減らすことができる。ただし，定電圧領域を極端に広くとると変電所のピーク電流値が上昇するため注意を要する。

サイリスタ変電所の場合はピークカットを V–I 特性変更によって行うことも可能だ。すなわち，V–I 特性に定電流領域を設け，その電流値以上はそもそも流れないようにすることも考えられる。

これらは統合インテリジェント化以前の話であり，これに関する論文なども多い[1][2][3][4][5][19][37]。だが，駅間走行時分一定化機能をもったシミュレーションモデルによって，あらためて饋電特性をより詳細に把握し，改善策を理解し，これらを最適化しておくことも重要であると考える。なお，ピークカットも可能ではあり，その効果も少なくないのであるが，以下では省エネルギー化の分野についてのみ検討を行った。同じような手法でピークカットに関する検討も可能であるが，これは将来への課題として残されている。

本章では，ダイオード変電所とサイリスタ変電所の比較を中心に，消費エネルギーで評価したとき最適な変電所 V–I 特性を探る。

⟨7.1⟩　本章で用いるモデル

ここでは，⟨6.2⟩・⟨6.3⟩ で用いた路線モデルを基本的に用いる。地下通勤線区である A 線，全線開業時モデルである。路線長は約 21.3km であるが，これは比較的短めのモデルといえる。地上側には変電所6箇所があるほか，回生電力吸収装置として回生インバータが3台設置されており，すべて稼働している。なお，比較のためにこれらがまったく稼働していないモデルについても計算した。

列車はVVVFインバータ制御車8両編成である。運転時隔は朝ラッシュ時が5分間隔，閑散時が7分間隔で一定である。大都市の通勤路線としては比較的運転間隔が長めのモデルといえる。

回生インバータの動作開始電圧は，全変電所の無負荷時送出電圧の最高値より 30V 高い値としている。したがって，自変電所がサイリスタ変電所であっても，ダイオード変電所が別な場所に接続されていれば回生インバータはダイオード変電所の場合の特性（動作開始電圧 $B + 30$[V]）で動かすことになる。回生インバータも含めた変電所のV–I特性は図7.1に示す通りである。回生インバータは定格電流値の5倍（この場合 3,333A）以上は吸収しない。

列車の回生絞り込み特性回生絞り込み開始電圧は1,650V，満車時の回生絞り込み終了電圧（絞り切り電圧）は 1,695V である。この電圧は，普通のケースに比べるとかなり低めであるといえる。

変電所は，2箇所が既存変電所であることを考慮し，次の3ケースを取り扱った。

(1) 全変電所ともダイオード変電所
(2) 全変電所ともサイリスタ変電所
(3) 既存2変電所のみダイオード変電所，ほか4箇所サイリスタ変電所

また，サイリスタ変電所においてはコンバータ側の電流 1p.u. 以上での電圧変動率が4%の場合も検討対象にいれた。

ちなみに，電圧変動率とはダイオード変電所の V–I 特性の傾きを表す。定格電流で定格電圧が出るとき，

$$(\text{電圧変動率}) = \frac{(\text{無負荷時送出電圧}) - (\text{定格電圧})}{(\text{定格電圧})} \tag{7.1}$$

で計算される。従って，定格1,500Vで，6% の電圧変動率のダイオード変電所の無負荷時送出電圧は1,590V と計算されることになる。

変電所の変圧器1次側タップ電圧は 22kV，23kV，23.5kV の3つの値を仮定した。ちなみに，22kV タップであるとき定格電流における送出電圧が定格電圧（1,500V）となる。

普通，無負荷時送出電圧に対するエネルギーなどの変化を表す図は横軸に無負荷時送出電圧をとる。しかし，このように種々雑多な変電所特性を比較するのに便利であるため，ここでは特に断らない限り変圧器タップ電圧値を横軸にとって比較する。

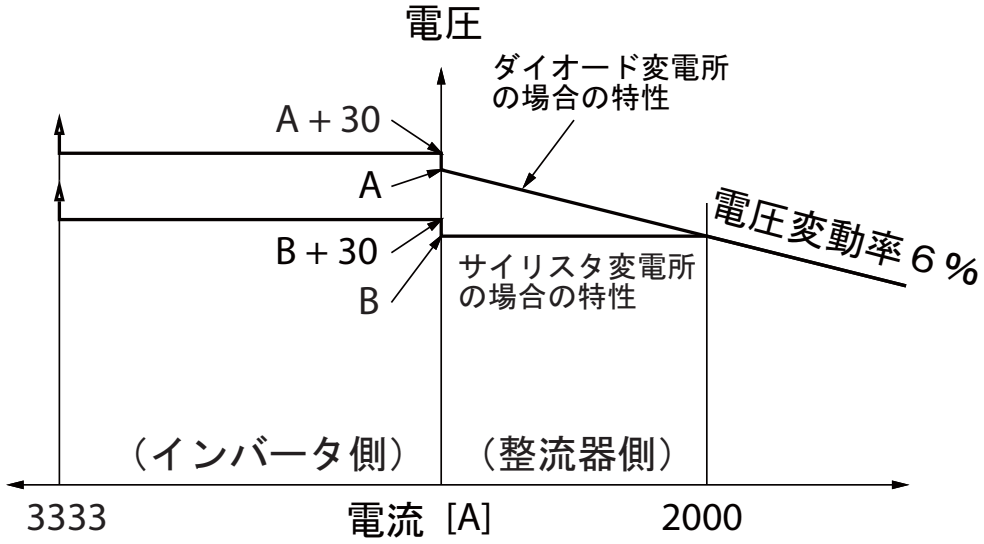


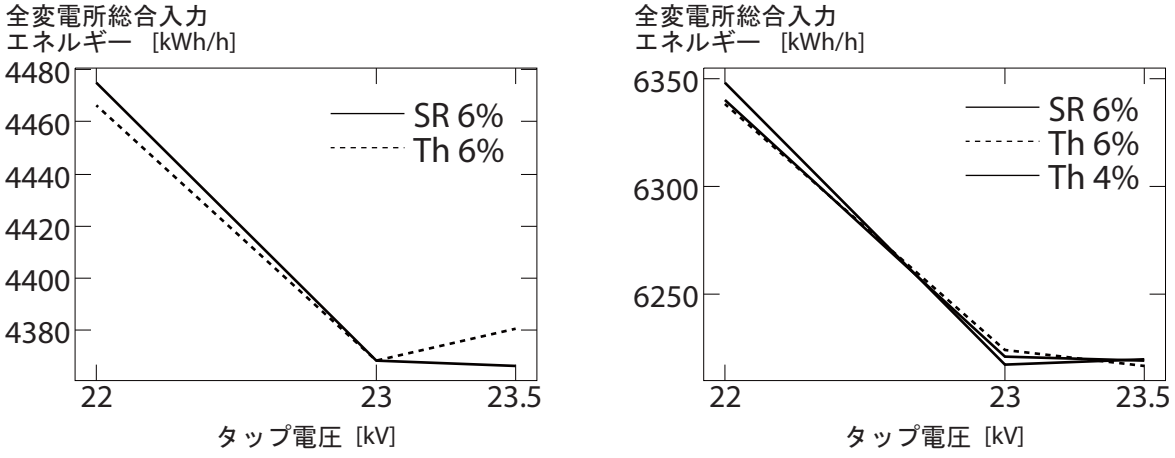
図 7.1: 回生インバータのある変電所の V–I 特性比較. 力行側は変電所容量3,000kW の場合. 図中 A, B の値は図6.1 （30ページ）を参照のこと.

⟨7.2⟩　最適な無負荷時送出電圧の設定

既存2変電所はダイオードのままを前提にし，3つの変電所1次側タップの比較を行った。

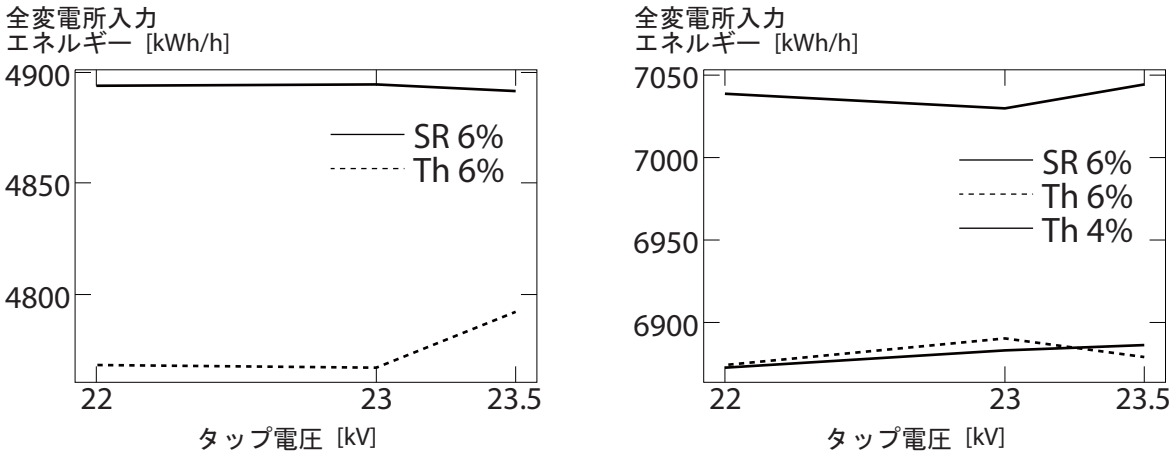
図7.2が，評価の基本となる全変電所総合入力エネルギー，のシミュレーション結果である。7分時隔は閑散時相当，5分間隔は朝ラッシュ時相当である。図中 Th とあるのはサイリスタ変電所とダイオード変電所が混在配置されているケースのため，回生インバータの動作開始電圧はすべてダイオード変電所を基準に決定されている。

図7.3〜7.11によって，詳細にシミュレーション結果を検討しよう。図7.2によれば，時隔や変電所特性によらず変電所1次側タップ 22kV のケースがもっとも不利であるが，これは回生車の回生失効が起こっているためであると推測される。このことを裏付けるデータとして，図7.4から，22kV タップでは列車の全変電所回生エネルギーが大幅に減っていることが読みとれる。同様の傾向は，図7.7に示したパンタ点回生エネルギーについても現れている。



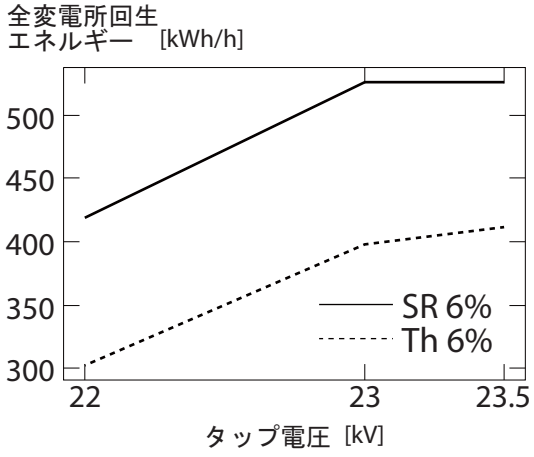
(1) 7分時隔時　　(2) 5分時隔時

図 7.2: 変電所1次側タップ変更時の全変電所総合入力エネルギーの比較. 図中 SR は全ダイオード変電所の場合, Th は既存変電所ダイオード・新設サイリスタの混合のケース. 5分時隔の場合, サイリスタ変電所について 4% 電圧変動率のケースも実施した.

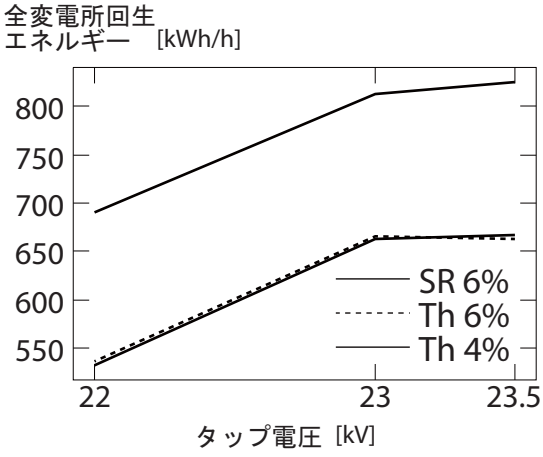


(1) 7分時隔時　　(2) 5分時隔時

図 7.3: 変電所1次側タップ変更時の全変電所入力エネルギーの比較. 図中記号等は図7.2と同様.

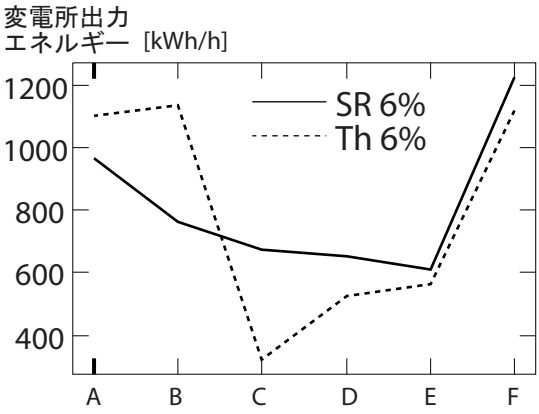


(1) 7分時隔時

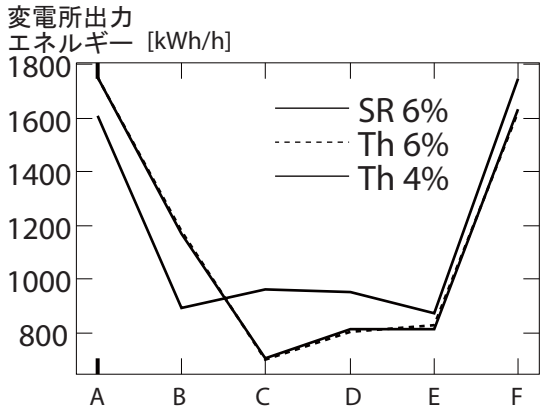


(2) 5分時隔時

図 7.4: 変電所1次側タップ変更時の全変電所回生エネルギーの比較. 図中記号等は図7.2と同様.

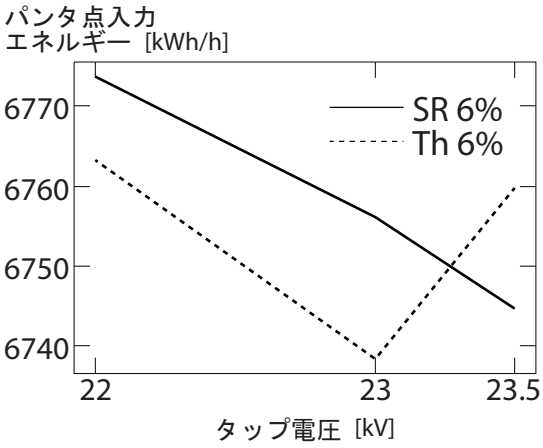


(1) 7分時隔時

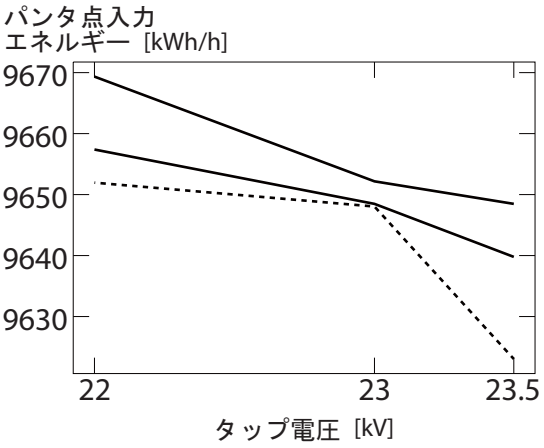


(2) 5分時隔時

図 7.5: 変電所ごとの出力エネルギー分担の比較（変電所1次側タップ 23kV）. 図中記号等は図7.2と同様.

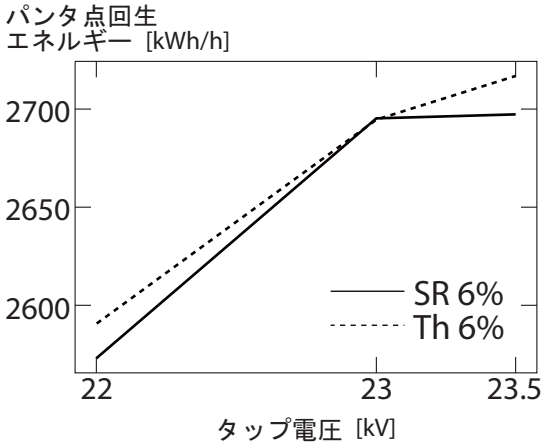


(1) 7分時隔時

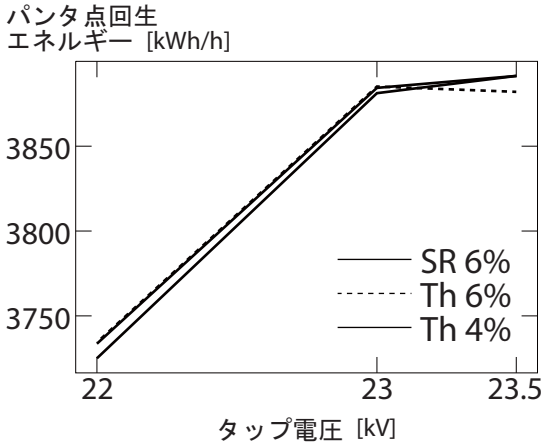


(2) 5分時隔時

図 7.6: 変電所1次側タップ変更時のパンタ点入力エネルギー(補機含む)の比較. 図中記号等は図7.2と同様.

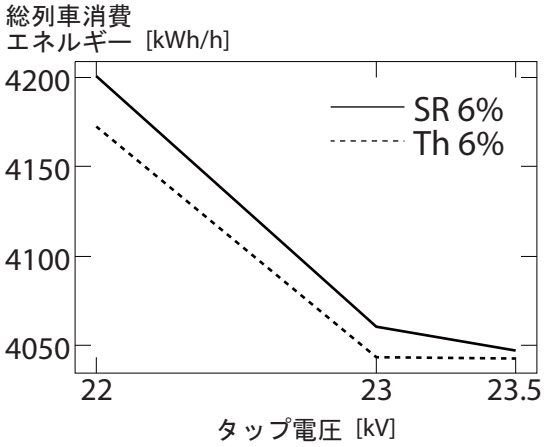


(1) 7分時隔時

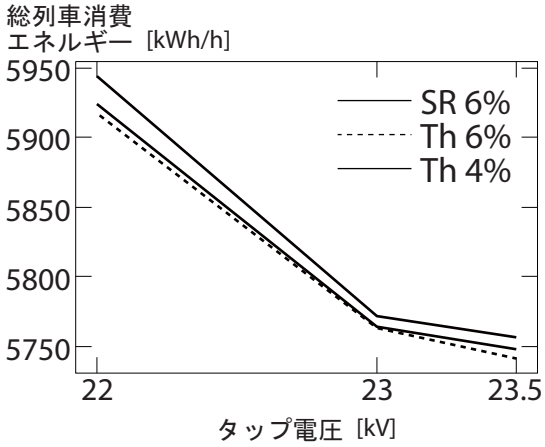


(2) 5分時隔時

図 7.7: 変電所1次側タップ変更時のパンタ点回生エネルギー(補機含む)の比較. 図中記号等は図7.2と同様.

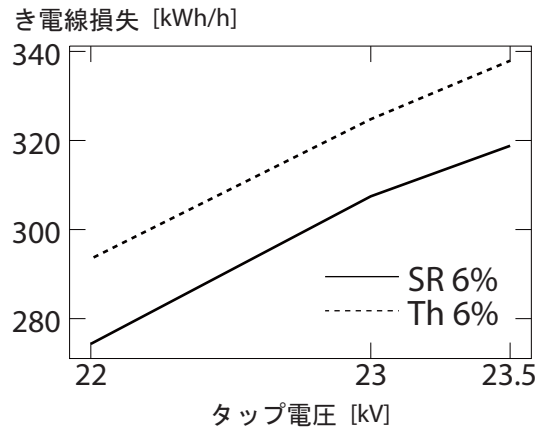


(1) 7分時隔時

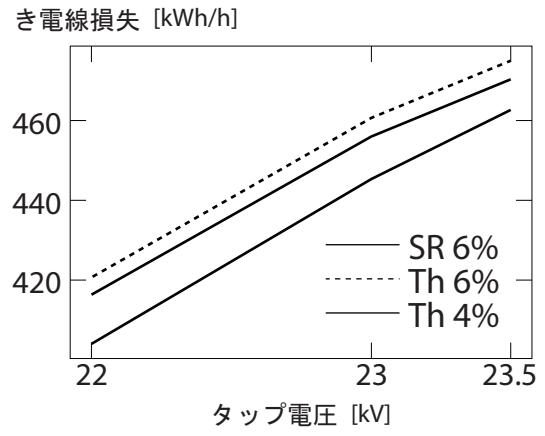


(2) 5分時隔時

図 7.8: 変電所1次側タップ変更時の総列車消費エネルギーの比較. 図中記号等は図7.2と同様.



(1) 7分時隔時



(2) 5分時隔時

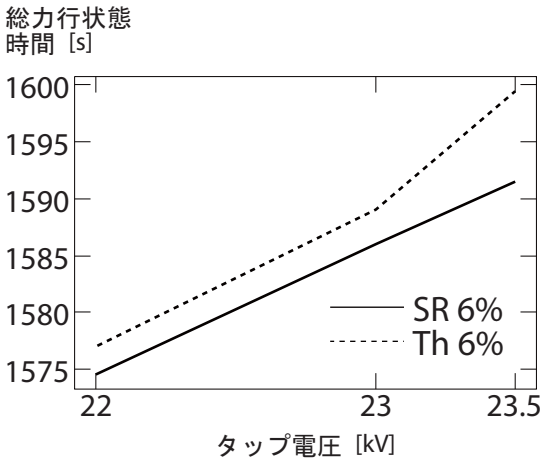
図 7.9: 変電所1次側タップ変更時の饋電線損失の比較. 図中記号等は図7.2と同様.

しかし，23kV タップから 23.5kV タップにする（無負荷時送出電圧をさらに低下させる）と，全変電所総合出力エネルギーが増えるか，または増えない場合でもほぼ横ばいに近くなる。この原因は，電圧低下に伴う列車の性能低下，および電流増加に伴う饋電線損失の増大に求めることができる。

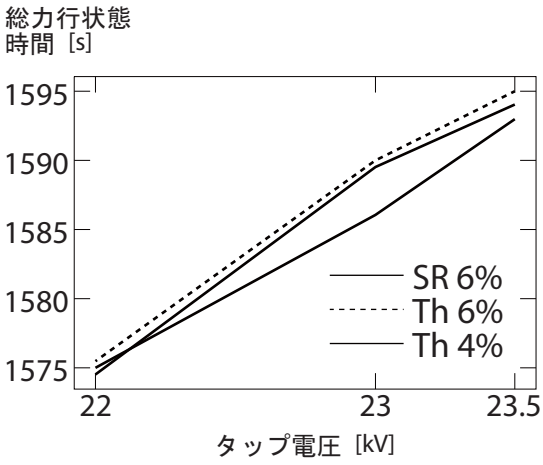
まず，図7.10・7.11にそれぞれ総力行状態時間および総加速時間の変化を示す。参考のため，図7.12にこのシミュレーションにおけるランカーブの例，およびその例における電流カーブを示した。図7.12からわかるように，列車は勾配・速度制限の存在する条件のもと，複雑にいろいろな状態を切替えつつ進むため，図7.10・7.11から傾向をつかみやすいとは必ずしもいえない。それでも，図7.10・7.11から列車の性能低下によりこれらの時間が増大している傾向は読みとれる。ただし，その増加はわずかであり，ダイヤに大きな影響を与えるまでには至らないこともわかる。

次いで，図7.9に饋電線損失の変化を示す。電圧低下に伴い電流も増えるため，損失が増大していることが読みとれる。

これらの結果から3つの変圧器1次側タップ電圧のうち最適なものを選択するとすれば，回生インバータが動作していることも考慮した場合23kV タップがコンバータの種類によらず最適といえるだろう。

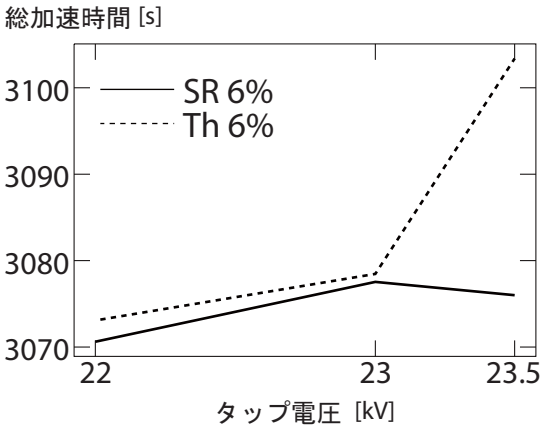


(1) 7分時隔時

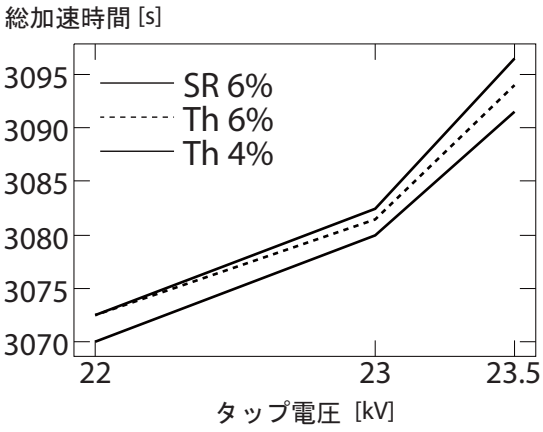


(2) 5分時隔時

図 7.10: 変電所1次側タップ変更時の総力行状態時間の比較. 図中記号等は図7.2と同様.



(1) 7分時隔時



(2) 5分時隔時

図 7.11: 変電所1次側タップ変更時の総加速時間の比較. 図中記号等は図7.2と同様.

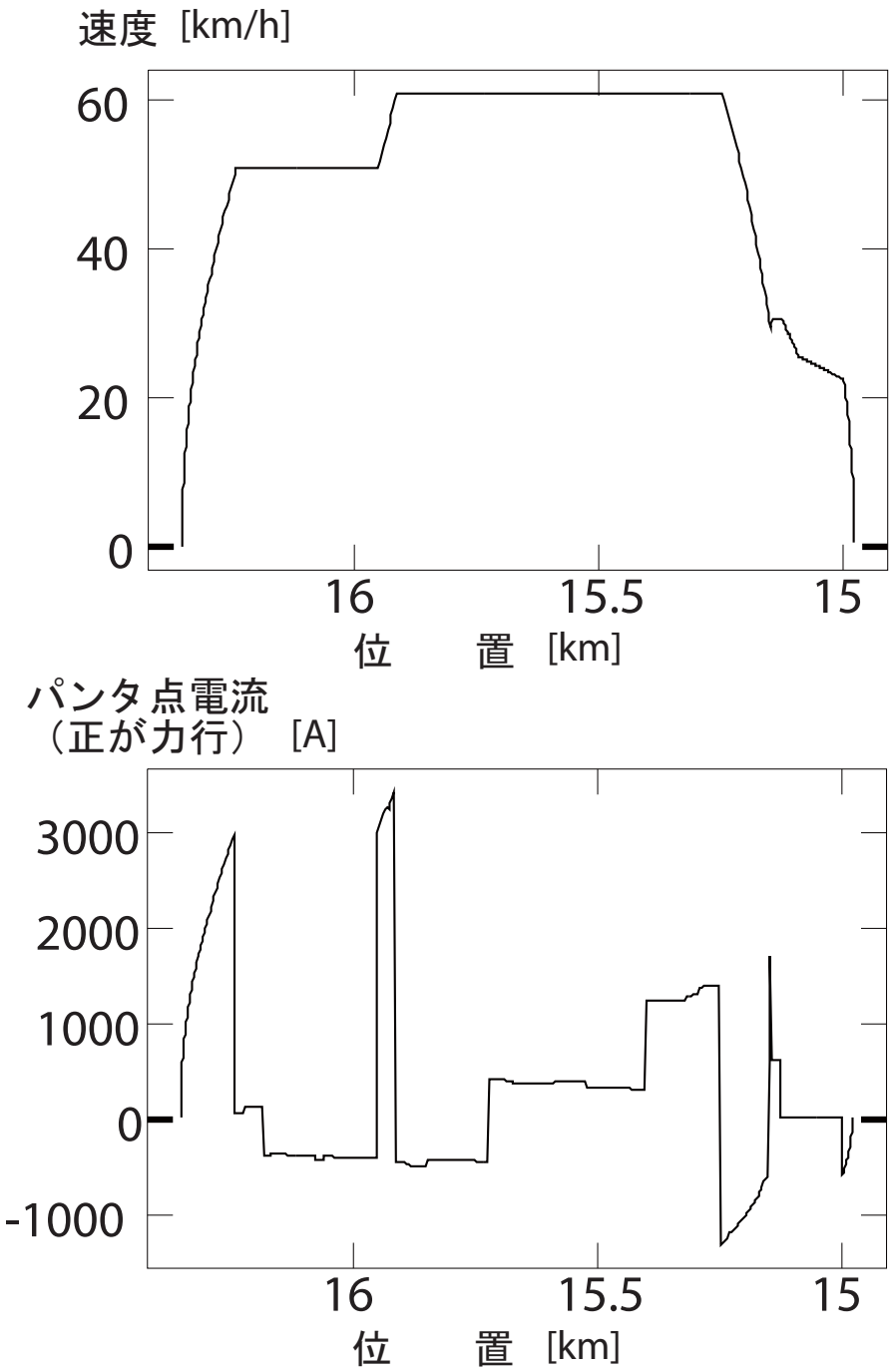
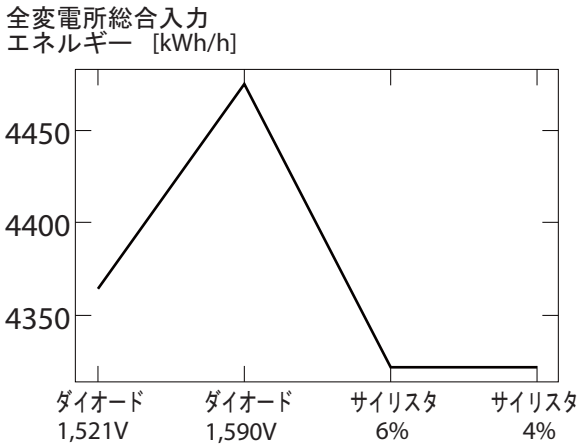
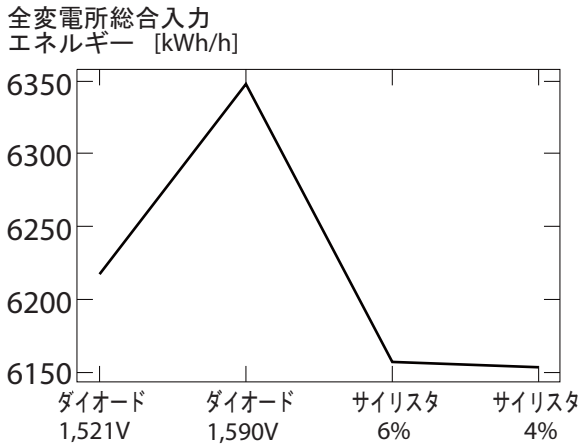


図 7.12: 駅間のランカーブのシミュレーション結果例（上）と，それにおける電流カーブ（下）

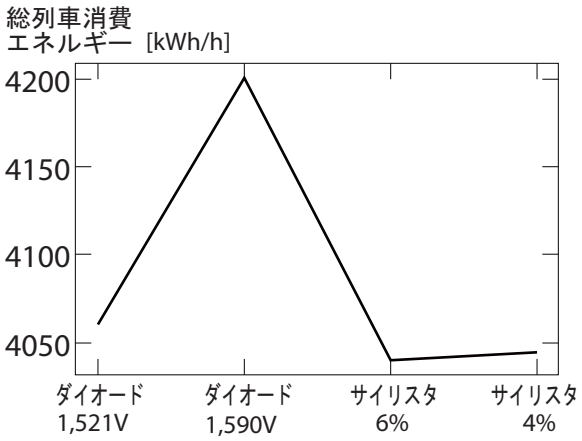


(1) 7分時隔時

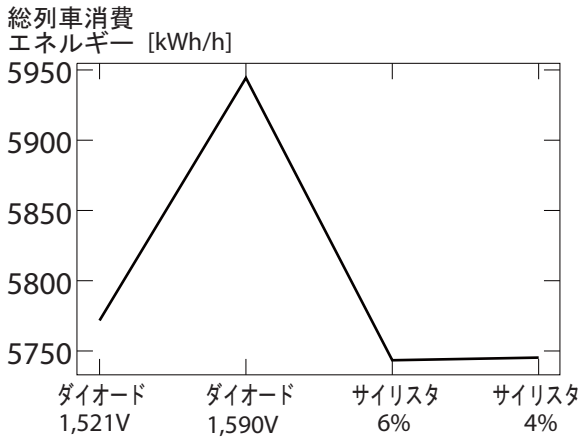


(2) 5分時隔時

図 7.13: サイリスタ変電所とダイオード変電所の全変電所総合入力エネルギーの比較

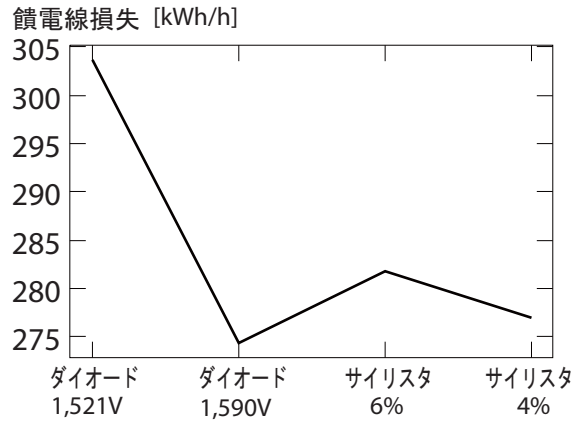


(1) 7分時隔時

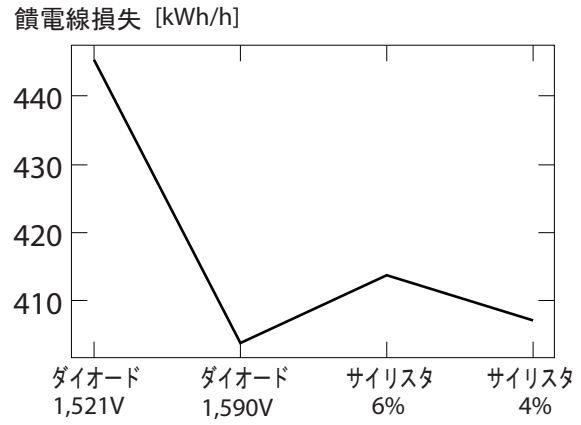


(2) 5分時隔時

図 7.14: サイリスタ変電所とダイオード変電所の総列車消費エネルギーの比較



(1) 7分時隔時



(2) 5分時隔時

図 7.15: サイリスタ変電所とダイオード変電所の饋電線損失の比較

なお，サイリスタ変電所とダイオード変電所が混在した場合では，サイリスタ変電所の方がダイオード変電所より無負荷時送出電圧が低いために**横流**（変電所間を流れる電流）が発生し，饋電線損失が大きくなる。この効果のため，最適な変圧器1次側タップである 23kV タップにおける変電所入力エネルギーは，サイリスタ変電所を混在させた方が増大してしまっていることに注意すべきである。この現象は，図7.5によっても明らかである。図7.18（45ページ）にあるように，混在するケースでは A, B がダイオード変電所である。このとき，ダイオード変電所が多くの負荷を分担している様子が図7.5より歴然としている。

⟨7.3⟩　回生インバータが存在する場合のダイオード変電所とサイリスタ変電所の比較

変電所1次側 23kV タップ（無負荷時送出電圧はダイオード変電所 1,521V，サイリスタ変電所 1,435V）にて，全変電所をサイリスタ変電所とした場合と全変電所をダイオード変電所とした場合とを比較した。このモデルでは，回生インバータの動作開始電圧はダイオード変電所のケースとサイリスタ変電所のケースで異なる値（ダイオード：1,551V，サイリスタ：1,465V）になっている。なお，念のためダイオード変電所で 22kV タップ（無負荷時送出電圧 1,590V）のケースも比較してある。

全変電所総合入力エネルギー，列車消費エネルギー，饋電線損失についての評価結果をそれぞれ図7.13, 図7.14, 図7.15 に示した。図7.13よりサイリスタ変電所のケースのほうがダイオード変電所 23kV （無負荷時送出 1,521V）のケースよりさらに 1% 程度省エネルギーになっていることがわかる。

これらを図7.14と図7.15によって分析すると，サイリスタ変電所の方が無負荷時送出電圧および回生インバータの動作開始電圧が低いことから，総列車消費エネルギーが減少する。しかし，これによる減少幅は全体の半分程度であり，残りは饋電線損失の減少であることが読みとれる。サイリスタ変電所は定電圧領域を持つため，横流が少なく抑えられるために饋電線損失が減少すると考えられる。電圧が低い，すなわち電流が増加するにもかかわらず，饋電線損失は電圧の高いダイオード変電所 22kV タップのケースに匹敵する低さとなっている。

⟨7.4⟩　回生インバータまたはサイリスタ変電所のみ導入時の効果

⟨7.3⟩を見ると，通常の論文に比べてサイリスタ変電所の導入効果が小さく見積もられている。これは，回生インバータがすでに導入された路線にさらにサイリスタ変電所を導入するモデルであることが原因と考えられる。

では，回生インバータなしでサイリスタ変電所のみすべての変電所に入れた場合はどうだろう。結果は図7.16の通りである。横軸は1590V が 22kV タップ，1521V が 23kV タップ，1489V が23.5kV タップにそれぞれ相当する。このシミュレーションケースにおいては，全変電所をサイリスタ化すればわずかながら省エネルギー化が図られるが，その効果は必ずしも顕著とはいえない結果になっている。

サイリスタ変電所とダイオード変電所が混在した場合では，⟨7.2⟩（37ページ）とまったく同じ理由で横流が発生し饋電線損失が大きくなる。この効果のため，最適な変圧器1次側タップ（23kV）における変電所入力エネルギーは，サイリスタ変電所を混在させた方が増大してしまっている。

混在させていないケースでも，サイリスタ変電所化の省エネルギー効果は顕著ではなく，0.5%程度またはそれ未満という低いレベルにとどまっている。しかも，省エネルギー効果を⟨7.2⟩（37ページ）と同様のやり方で分析してみると，列車の回生エネルギーの増大ではなく饋電線損失の低下によるものであることもわかる。

このように，このモデルではサイリスタ変電所化のメリットは小さいことがわかる。この理由の主なものは，シミュレーションを行った路線が短距離で，列車密度も比較的低い路線であることだろう[1)]。そう

[1)] この他に，電圧変動率が6〜4%と小さめに設定されているために，定電圧饋電方式の利点が見えにくくなっていることも考えられる。無負荷時送出電圧を低めに設定し，定電圧領域を 1p.u. より拡大すると，より大きな省エネルギー効果が得られる可能性がある。

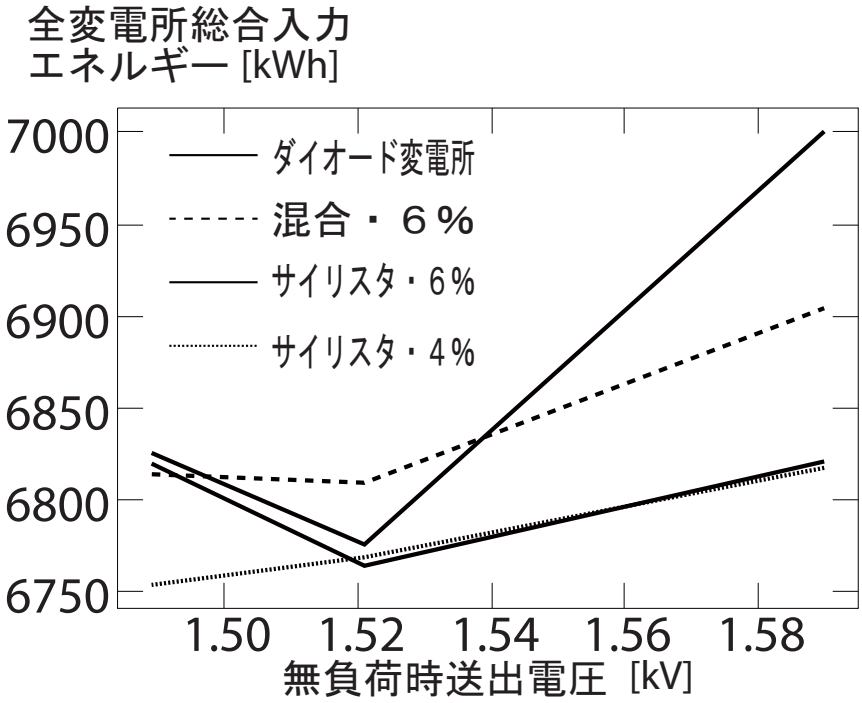


図 7.16: サイリスタ変電所のみ全変電所に入れた場合の導入効果

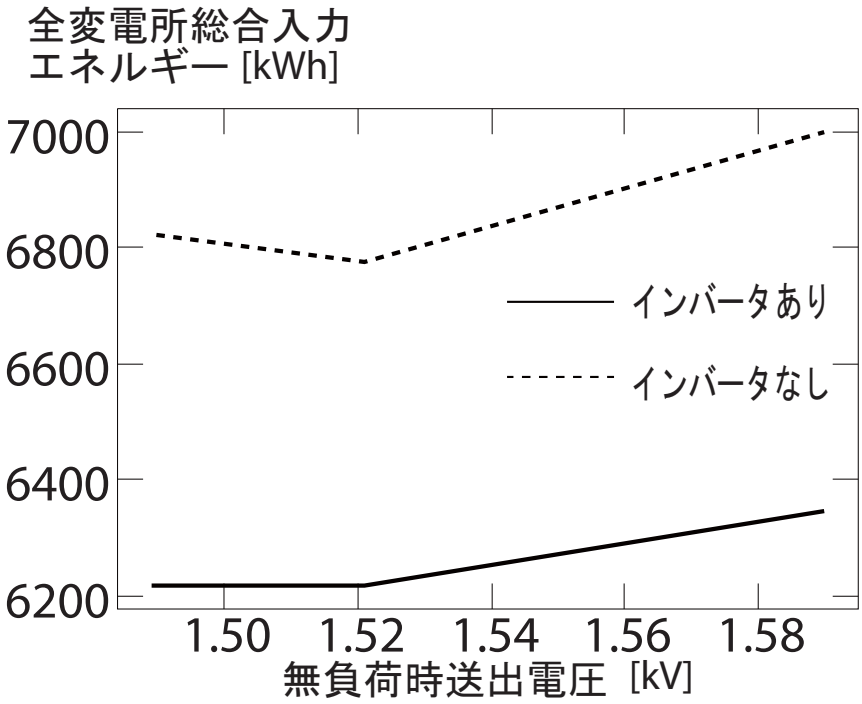


図 7.17: 回生インバータのみの導入効果

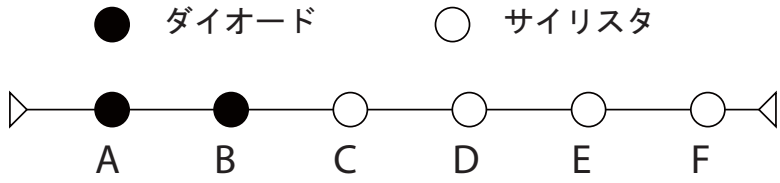
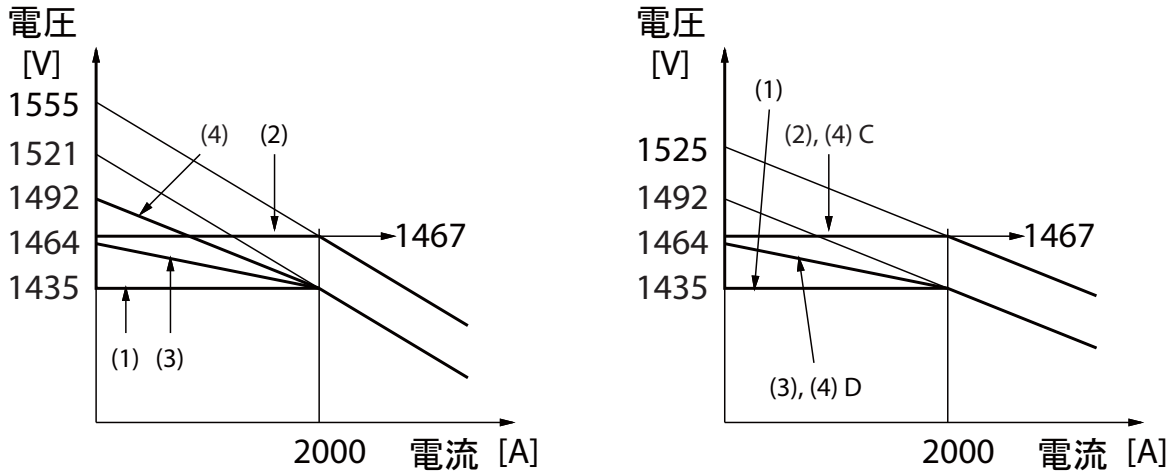


図 7.18: A線モデルにおけるサイリスタ・ダイオード変電所混在時の変電所配置



A. サイリスタ変電所の電圧変動率 6%　　B. サイリスタ変電所の電圧変動率 4%

図 7.19: 混在時の境界変電所に与えた特性

だとすれば，力行車と回生車が饋電システム内に同時に存在する確率それ自体がもともと低いことになる。従って，いくら無負荷時送出電圧低下によって回生電力が遠くまで到達できるようにしても，回生率が上昇せず，省エネルギー効果も得られない。

このような条件の路線では回生インバータの効果が大きくなる。試しに，全ダイオード変電所の条件で回生インバータなしの場合と，3台稼働の場合とを，5分時隔の場合について比較したのが図7.17である。回生インバータ設置により10%程度の省エネルギー化が図られていることがわかる。この省エネルギー効果は，ほとんどが列車の回生エネルギーの増大（回生失効率の低減）によってもたらされたものである。

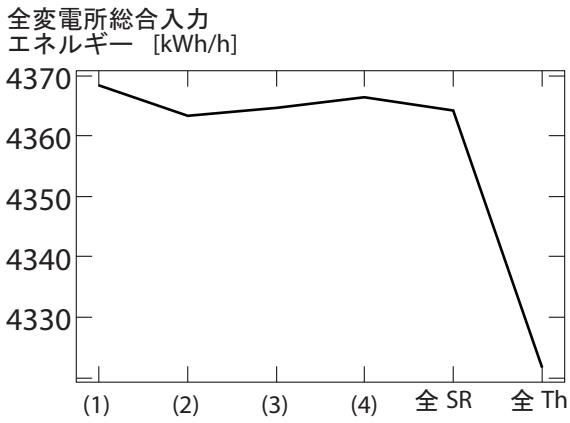
このように，この路線モデルにあっては回生インバータの導入効果がサイリスタ変電所導入による回生電力の有効活用の効果よりはるかに大きいことがわかる。

⟨7.5⟩　サイリスタ・ダイオード変電所混在時の問題

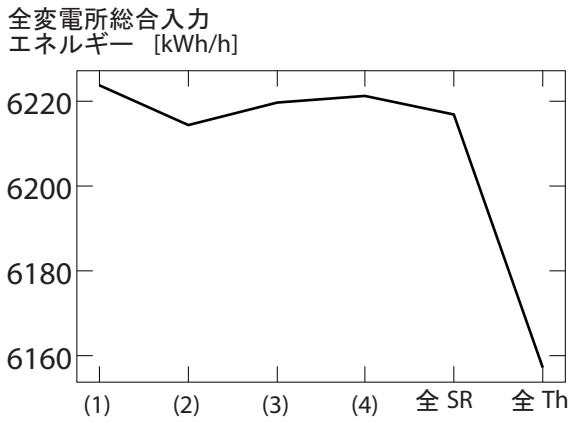
全変電所をサイリスタ変電所化することができれば，比較的大きな省エネルギー効果を得ることができるが，全変電所を一度にサイリスタ化することは困難であり，混在する状態をどうしても相当な期間にわたって経なければならない。

A線において，既設2変電所をダイオードのままとした場合，図7.2 （37ページ）でみればわかるように最適タップとなる 23kV タップ付近では，サイリスタ変電所があるほうがかえってダイオードだけのケースより悪い結果にすらなっている。これは，主に既存の変電所はサイリスタ化された変電所より無負荷時送出電圧が約90V程度高くなるため，変電所間の負荷分担のバランスが崩れており，横流が多くなって饋電線損失を押し上げたためである。

これを緩和するため，サイリスタ変電所とダイオード変電所の境界となる変電所の特性を変化させて，



(1) 7分時隔時



(2) 5分時隔時

図 7.20: サイリスタ・ダイオード変電所混在時の全変電所総合入力エネルギーの比較（A. サイリスタ変電所の電圧変動率 6% 時）

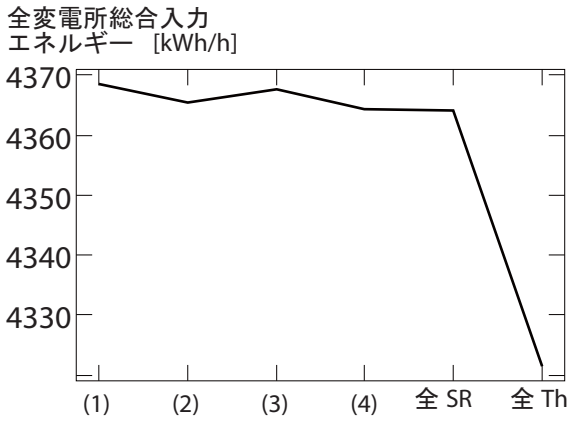
隣接変電所の特性差を少なくし，負荷分担を平均化するのがのぞましいと考えられる。この観点から，つぎの8ケースを考慮し，シミュレーションを行った。

A. サイリスタ変電所の電圧変動率6% （図7.19 A.）

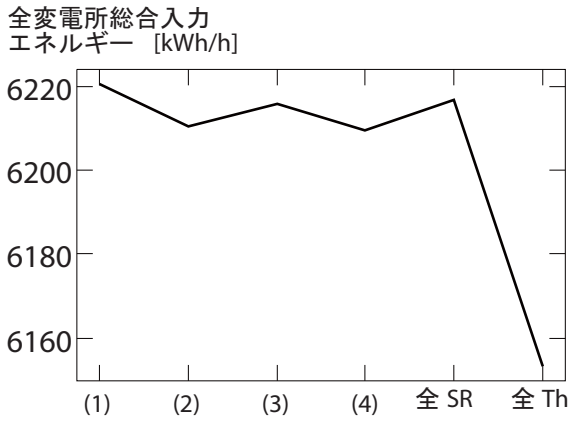
(1) サイリスタ変電所の特性がすべて同一のケース
(2) C変電所を22.5kVタップとしたケース
(3) C変電所の100%負荷までの低負荷時の電圧変動率を2%としたケース
(4) C変電所の100%負荷までの低負荷時の電圧変動率を4%としたケース

B. サイリスタ変電所の電圧変動率4%（図7.19 B.）

(1) サイリスタ変電所の特性がすべて同一のケース
(2) C変電所を22.5kVタップとしたケース
(3) C変電所の100%負荷までの低負荷時の電圧変動率を2%としたケース

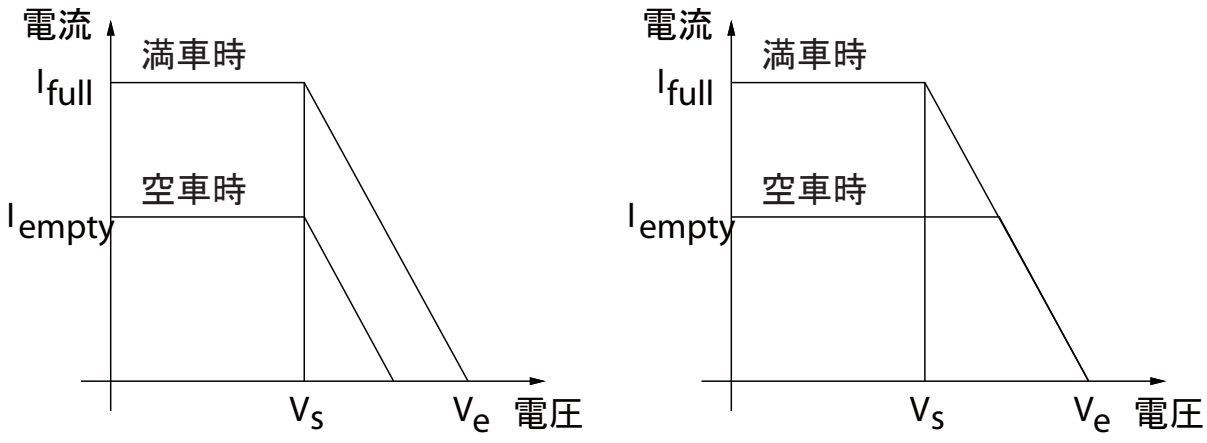


(1) 7分時隔時



(2) 5分時隔時

図 7.21: サイリスタ・ダイオード変電所混在時の全変電所総合入力エネルギーの比較（B. サイリスタ変電所の電圧変動率 4% 時）



(1) 回生率が悪い特性　　(2) 回生率がよい特性

図 7.22: 列車の回生絞り込み特性

(4) C変電所を22.5kVタップとし，D変電所の100%負荷までの低負荷時の電圧変動率を2%としたケース

これらのシミュレーション結果を図7.20〜7.21に示す。これらからわかるように，B.（サイリスタ変電所の電圧変動率4%）の (4) のケースがもっとも変電所入力エネルギーが少ないが，これでも全サイリスタのケースに比べるとだいぶ悪くなっている。見方を変えていうと，サイリスタ変電所の電圧変動率を4%とすることは，全変電所の電圧設定を混在を考慮して変更したのと同じことになる。

従って，このモデルにおいては次のようなことがいえる。

- 境界にある1変電所の特性を変更しただけでは望む効果が得られないこと
- 路線距離が短いため，実質的にすべての変電所の特性を変化させなければ望む効果が得られない，

⟨7.6⟩　列車の回生絞り込み特性と饋電特性

シミュレーション上も，また実際のシステムにおいても，列車の回生絞り込み特性は回生率に少なからぬ影響を与える。例えば，図7.22 の2つの特性では，満車時以外は (2) のほうが回生率がよくなる。このシミュレーションではモデルとした車両が (1) の特性だったために (1) をとっているが，このようなものは改善すべきだ。

また，図7.22の縦軸の「電流」というのがどこの電流のことをいうのかも問題になる。図7.23 に示す**モータ電流**を見て絞り込みをかけるのが通例であり，シミュレーションもそれに合わせて行っている。しかし，速度の低い領域ではモータ電流が大きくても**主回路回生電流**が小さいことがあり，このような場合モータ電流を見ていれば不必要な絞り込みを行ってしまう。結果的に回生率は下がるだろう。

回生絞り込み開始・終了電圧の値は，もっと大きな影響がある。これらの電圧は高ければ高いほど回生率が上がることはよく知られている。そこで，回生インバータが動作していない条件のもとで，回生絞り込み特性電圧を開始電圧，終了電圧とも 100V 上げた場合のシミュレーション結果を図7.24に示す。これ以外の対策を何もしなくても，電圧を上げなかった場合に比べて入力エネルギーが約1% 低下していることがわかる。

このように，饋電システムの最適化に入る前に，列車の特性を最適化しておく必要性があるといえる。

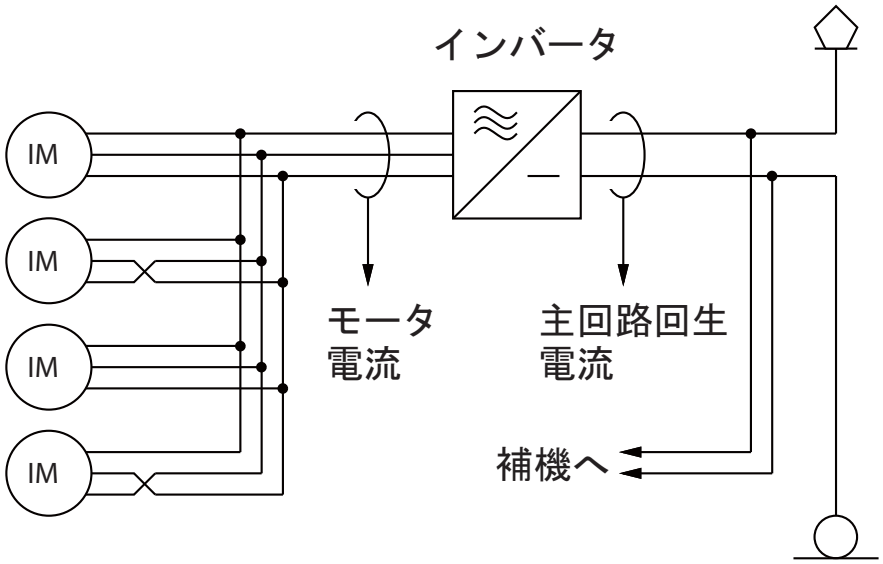


図 7.23: 絞り込み特性の「電流」とはどこのことか？

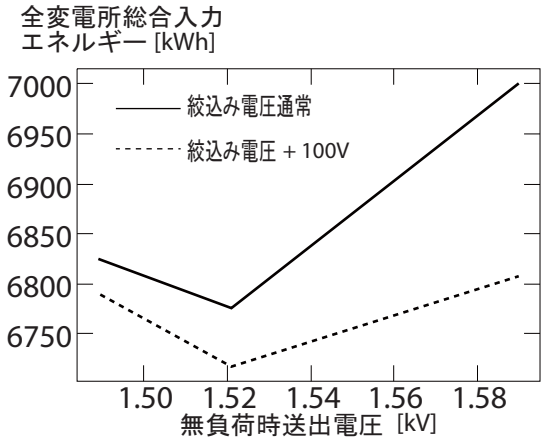


図 7.24: 回生絞り込み電圧と変電所入力エネルギー

⟨7.7⟩ まとめ

まず，A 線モデルにおいて，変電所無負荷時送出電圧の最適化を行った。特性の違ういろいろな変電所特性において，22kV，23kV，23.5kV の 3 種類の変圧器 1 次側タップを比較したところ，ほぼどのケースも 23kV が最適になることがわかった。（⟨7.2⟩, 37ページ）

次いで，A 線モデルにおいて，回生インバータおよびサイリスタ変電所のエネルギー消費量の上での優位性を明らかにできた。回生インバータがない場合に比べ，ある場合は列車の回生率の向上により 10%程度の省エネルギー化が図られる。一方，回生インバータがない場合，全変電所をダイオード変電所からサイリスタ変電所化すると，饋電線損失の低減によって 0.5% 程度の省エネルギー化が図られる。（⟨7.4⟩, 43ページ）また，回生インバータがある場合について全変電所をダイオード変電所からサイリスタ変電所とすると，饋電線損失の低減と回生率の向上によって 1% 程度の省エネルギー化が図られる。（⟨7.3⟩, 43ページ）

A 線モデルにおいては，ダイオード変電所とサイリスタ変電所を混在させると省エネルギー効果を殺してしまうこと（⟨7.2⟩, 37ページ・⟨7.4⟩, 43ページ），またダイオード変電所を設置した領域とサイリスタ変電所を設置した領域の境界にあたる変電所の V–I 特性を調整し，隣接変電所の特性が大きく変化しないようにすることによって省エネルギー効果をわずかに取り戻すことができるが，境界の 1 変電所のみでは十分な効果が期待できないこと，および A 線モデルのように路線距離が短い場合にはほとんどすべての変電所が V–I 特性を調整しなければならないこと（⟨7.5⟩, 45ページ），などを明らかにした。

さらに，饋電システムの最適化の前に列車の回生絞り込み特性を最適化しておくことの重要性も指摘した。

これらについては，本章の冒頭でも述べたように，多くの検討がすでに行われてきた。しかし，駅間走行時分一定化機能を持った RTSS を用いることで，いろいろな効果を詳細に分離して論じることがはじめて可能になったと考える。そして，その結果として,「効果があるかないか」という議論の段階はもはや終わり，効果の定量的把握をこれらツールを使って行うべき時代になったといえる。

ただし，特に注意すべきことは，シミュレーションを行った路線モデルに結果が強く依存することである。ここで挙げられた数字はどの路線についても一般的に成り立つものではない。現段階では，シミュレーションツールとして信頼できるものはできたものの，どんな路線条件の際にどんなことが起きるかという定性的な知識はまだ不足している。そこで，さまざまな条件の路線について，今後も数多くのシミュレーションを繰り返す必要があると考えられる。

8

変電所送出電圧のリアルタイム制御

7章での検討結果から，電力回生ブレーキを常用する列車が大部分を占める電気鉄道にあっては饋電電圧は比較的低めの値にするのがよいことがわかった。これは，主として電圧が高いために回生電力が遠方にある負荷まで到達せず，列車の電力回生能力が生かされないことによるものだ。しかし，電圧低下は列車の性能低下につながるほか，電流の増大により饋電線損失を増すデメリットがある。

地上側において，饋電システム内にあるすべての列車の位置・速度・状態が把握できるなら，饋電電圧低下のメリットが出る場面にのみ電圧を低くする，変電所電圧リアルタイム制御が考えられる。本節では，列車の回生能力向上を意図したリアルタイム制御について基礎的な検討を行う。

⟨8.1⟩　変電所送出電圧リアルタイム制御の導入によって期待される効果

饋電電圧低下によって生じるエネルギー的なメリット・デメリットを挙げると，主なものは次のようになる。

(1) メリット

a. 回生絞り込み量の低減による省エネルギー化

(2) デメリット

a. 列車の性能低下によるブレーキ初速の上昇（力行時間の増加）
b. 電流増大による饋電線損失の増加

メリットは (1) a. だけだが，これが出てくるのは力行車と回生車が共存する場合である。従ってこのような場合については電圧を絞り込めばよい。このケース以外の場合には，電圧はつねに高めの方がよいこともわかる。

7章で議論した送出電圧設定の議論では，変電所の V–I (電圧—電流) 特性は列車の位置・速度・状態によらず一定で議論していた。したがって，ひとたび送出電圧を低め設定すれば，力行車と回生車が共存している瞬間はもちろん，そうでない瞬間にも送出電圧は低いままである。また，これ以上電圧を下げればさらに回生絞り込み量を減らす余地がまだある場合にも，その余地を残したままである。

これに対し，変電所送出電圧のリアルタイム制御では電圧を低めにする必要のない瞬間には高い送出電圧で饋電することができる。力行車と回生車が共存していて，送出電圧を下げれば回生絞り込みを防止できる瞬間には，送出電圧を下げて回生絞り込みを防止することができる。当然ながら，これ以上電圧を下げても回生絞り込み量を減らす余地がないところまで電圧下げが可能となる。

このように，変電所送出電圧のリアルタイム制御によって，饋電電圧低下のデメリットを抑えつつ，メリットをより大きく得ることが可能になると考えられる。

⟨8.2⟩　変電所送出電圧リアルタイム制御のアルゴリズム

ここでは，変電所送出電圧リアルタイム制御のアルゴリズムとして，過去の研究による定性的な結論を紹介したのち，ここでシミュレーションを行うアルゴリズムについて紹介する。

⟨8.2.1⟩　電圧制御アルゴリズムの過去の研究における基本的な考え方

リアルタイム変電所送出電圧制御のアルゴリズムに関する過去の研究[1][3]では，列車位置の検出方法から議論し始めている。列車位置の検出は実際にこのようなシステムを構成する上で重要な課題であり，⟨8.4⟩（54ページ）で議論する。

これらの研究における変電所送出電圧制御の考え方はつぎのように記述できる。すなわち，

(1) （力行車の）パンタ点最低電圧を守る
(2) 回生車の回生に支障がない程度に電圧を下げる。
(3) なるべく電圧は高くする。

を，上のルールほど優先順位を高くして制御し，変電所入力エネルギーを最小化しようというものである。

また，これらの研究における特徴は自律分散システムを指向したアルゴリズムを考えている点が挙げられる。これは，このような付加的制御機構が故障した結果としてシステム全体が混乱に陥るのは実用上まずいと考えているためだ。

⟨8.2.2⟩　今回検討したアルゴリズム

これまで，この種のアルゴリズムを本格的にインプリメントしたシミュレーションプログラムはなかった。RTSS についても，このような機能がもともとあったわけではない。これは，リアルタイム制御のアルゴリズムとして確立されたものがまだないからである。

そこで，今回は，実用的なアルゴリズムの構築に先駆け，シミュレーション上でリアルタイム制御の効果のデモンストレーションを行うことを主な目的として検討を行った[74]。そこで，RTSS を用い，プログラムの改造が最小限で済むよう，次のような簡易なアルゴリズムで ⟨8.2.1⟩ 節で述べた3つのルールに基づいた制御を実現した。実用システムでは自律分散システムを指向したアルゴリズムの採用が望ましかろうが，今回はそれも無視している。さらに，変電所ごとに制御する方法も採らず，全変電所一斉に同一の無負荷時送出電圧値に制御することを考えた。

(1) 変電所はすべてサイリスタ変電所とし，データで指定したある範囲内の無負荷時送出電圧を出せるものとする。変電所の電圧—電流特性は無負荷時送出電圧の移動にともなって電圧—電流平面上を平行移動するものとする。また，変電所を制御するコントロールセンタですべての列車の位置・速度・状態，およびすべての変電所の電圧・電流などが瞬時にかつ完全に把握できていると仮定する。
(2) まず，送出電圧がもっとも高い状態で饋電回路を解く。
(3) 回生状態（フルノッチ比が負）であって，回生絞り込みを行っている電車のうちもっとも回生失効率が高いものを探す。
　該当する列車がない場合はこれで回路演算終了。
(4) (3)で，該当する列車があった場合は，その列車の電圧を回生絞り込みが起きない電圧まで下げる。
　この際の電圧の下げ方は次のようにする。その列車の回生失効率を R_i, 変電所の無負荷時送出電圧を V_S （仮定(1)により全変電所同一）とする。このとき，

$$V_D = \frac{\partial R_i(V_S)}{\partial V_S} \tag{8.1}$$

で与えられる V_D を用いて，R_i が 0 となるように

$$k = \frac{R_i}{V_D} \tag{8.2}$$

で定めた定数 k (> 0) を用い，

$$V_S \leftarrow V_S - k \cdot V_D \tag{8.3}$$

のようにして V_S を更新する。

(5) 回生状態（フルノッチ比が負）であって，回生絞り込みを行っている電車のうちもっとも回生失効率の高いものを探す。

該当する列車があった場合は，(4) に戻り，その列車について電圧を下げる操作を繰り返す。ただし，いくら電圧を下げても回生失効が防げないケースを考慮し，繰り返し回数に適当な制限を設ける。

(6) パンタ点電圧が最低電圧 V_{MIN}（例えば1,200V）に満たない列車のうち，もっとも電圧の低いものを探す。該当する列車がない場合はここで計算終了。

(7) (6)で該当する列車があった場合は，その列車の電圧を最低電圧より上げる。

この際の電圧の上げ方は次のようにする。その列車の電圧を V_{Tx}, 変電所の無負荷時送出電圧を V_S （仮定(1)により全変電所同一）とする。このとき，

$$V_D = \frac{\partial V_{Tx}(V_S)}{\partial V_S} \tag{8.4}$$

で与えられる V_D を用いて，V_{Tx} が V_{MIN} となるように

$$k = \frac{V_D}{V_{MIN} - V_{Tx}} \tag{8.5}$$

で定めた定数 k (> 0) を用い，

$$V_S \leftarrow V_S + k \cdot V_D \tag{8.6}$$

のようにして V_S を更新する。

(8) (6)に戻る。（こちらは，全列車について最低電圧確保ができるまで何度でも繰り返す）

ここで，**回生失効率**とは通常

$$(\text{回生失効率}) = 1 - \frac{(\text{回生エネルギー})}{(\text{回生可能エネルギー})} \tag{5.3}$$

という量をさすが，ここでは瞬時瞬時の計算をしなければならないので，

$$(\text{回生失効率}) = 1 - \frac{(\text{回生電力})}{(\text{回生可能電力})} \tag{8.7}$$

という定義式になることに注意すべきだ。

⟨8.3⟩ シミュレーションとその結果

⟨8.2.2⟩にて示したアルゴリズムを，RTSS 上に実装し，シミュレーションを行った。

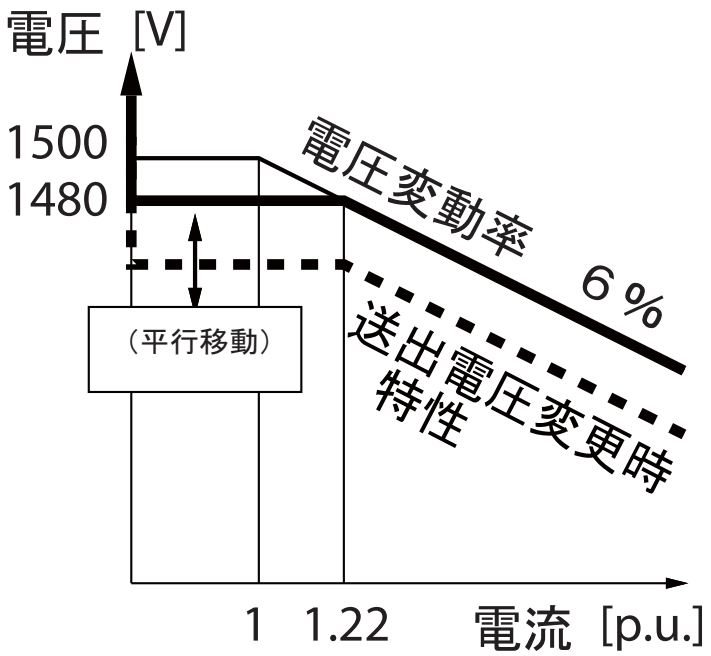


図 8.1: リアルタイム制御の特性

⟨8.3.1⟩ 条件

⟨7.1⟩（35ページ）のモデルを用いた。ただし，列車の間隔は7分時隔時のモデルを用いている。また，回生インバータ設備はないものと仮定している。

比較のため，すべての変電所がサイリスタ変電所であり，無負荷時送出電圧が1,435V，電圧変動率 6% で電流が 100% まで定電圧となる特性（⟨7.1⟩において，サイリスタ変電所 23kV タップ相当の特性）でリアルタイム制御は行わないケースを合わせてシミュレートした。このケースは，リアルタイム制御なしのケースの中では変電所総合入力エネルギーがほぼ最小のケースとされたものである。これに対し，リアルタイム制御では，基本となる電圧—電流特性として電圧変動率 6%，無負荷時送出電圧 1,500V，電圧変動率 6% で 100% 電流まで定電圧となるサイリスタ変電所の特性（⟨7.1⟩のサイリスタ変電所 22kV タップ相当の特性）のうち，定電圧領域の電圧を 20V 下げた特性（無負荷時送出 1,480V で，約 122% 電流まで定電圧となる）を入れた（図8.1で太い実線がこの特性である）。電圧制御で無負荷時送出電圧を下げた場合，全変電所の電圧—電流特性はこの特性曲線を当該無負荷時送出電圧のところまで平行移動した特性（図8.1で太い破線）となる。

⟨8.3.2⟩ 結果

シミュレーション結果を表8.1に示す。列車のパンタ点電圧が平均で 40V 程度も上昇しているのに，回生失効率・回生率はほぼかわらず，饋電線損失の低減によって変電所総合入力エネルギーが極めてわずかながら減少していることがわかる。列車の総加速時間（列車が力行状態にある時間，および列車が定速走行状態にあり走行抵抗が正である時間の総和）では長くなっているが，列車の力行状態時間は減少していることから，ダイヤ上の余裕は増していると考えてよかろう。このように，送出電圧を高く保つことのデメリットをリアルタイム制御によって緩和した，と見ることができる。

この結果はアルゴリズムの改善が十分でないまま出されたものであり，性能向上の余地もまだ相当残されていることが考えられる。このことも考慮に入れれば，この結果から，変電所送出電圧リアルタイム制御によって，饋電特性改善の可能性がまだ見込まれる。しかし，残念ながらこのモデルでは省エネルギー

表 8.1: シミュレーション結果

項　　目	無制御 (A)	制御 (B)	差 [%] $((B-A)/A \times 100)$
力行車パンタ点平均電圧 [V]	1420.7	1462.2	2.921
総加速時間 [s]	3081.0	3086.5	0.179
総力行状態時間 [s]	1590.5	1582.0	−0.534
列車総力行エネルギー [kWh/h]	6752.8	6766.1	0.198
列車総回生エネルギー [kWh/h]	2353.1	2357.9	0.207
総列車消費エネルギー [kWh/h]	4399.7	4408.2	0.193
列車回生率 [%]	34.8	34.8	—
回生失効率 [%]	12.7	12.4	—
全変電所総合入力エネルギー [kWh/h]	4701.3	4694.7	−0.142
饋電線損失 [kWh/h]	301.6	286.5	−5.016

化に関していえばあまり大きな効果は期待できない，という見方もできる。

⟨8.4⟩　変電所送出電圧リアルタイム制御の実現のための課題

このような変電所送出電圧制御を実現するためには，次のようなことに考慮を払う必要がある。

⟨8.4.1⟩　地上側で列車の状態を知る方法の確立

アルゴリズムの種類 — 変電所が制御する自律分散タイプ，およびコントロールセンタが変電所に対し制御指令を行う集中タイプ — によらず，このような制御を実現するためには何らかの形で地上側が列車の状態を知る必要がある。このさい，列車の位置・速度のみならず，列車のパンタ点電流の値がわからなければならない。

これらを推定するために利用する情報として，次のようなものが考えられている。

(1) 変電所の電圧情報
(2) 変電所の方面別電流情報
(3) 饋電定数
(4) 変電所接続点直下を列車が通過したことの検知情報
(5) 標準運転曲線
(6) 変電所電流の変化の検知

隣接変電所間に 1 列車だけが存在する場合，列車の位置・電流は隣接する 2 つの変電所の (1)・(2)，および (3)の 3 種類の情報から容易に求められる[1]。これが列車状態推定の基本である。しかし，隣接変電所間に複数列車が存在する場合にはこれだけでは正確な位置が求められない。過去の研究では，これにいくつかの情報を付加してやることで列車位置を求めようとしている。

例えば，文献 [3] では，基本の情報に加えて (4)(5)を用いる方法を考察している[21]。(4)から隣接変電所間にいる列車の数を推定するほか，他の情報によって列車の位置が決定できない場合に標準運転曲線を用いるものだ。また，最近発表された文献 [22] では，基本の情報に加えて (6)を用いる推定法が提案されている。大きな電流変化が観測された場合には列車の状態変化（ノッチイン，ノッチオフ，ブレーキイン，ブレーキオフなど）が起きたものと仮定して列車状態を推定するものだ。これ以外に，駅からの情報，軌道回路からの情報も有効だろう。

しかし，**列車・地上間通信**によって列車から地上側に位置・速度・状態が送信できれば，そもそも推定する必要がない。送信すべき情報量は，電話線1本分を利用し，モデムを介して送信できる約9600bpsに比べわずかであり，送信それ自体には技術的な問題はほとんどない（⟨9.4.1⟩, 66ページまたは文献[56]を参照のこと）。

⟨8.4.2⟩ アルゴリズムの検討

今回のアルゴリズムは，⟨8.3.2⟩（53ページ）で述べたようにまだ消費エネルギー削減や回生失効率低下などの面で改善の余地があるものと考えてよい。また，自律分散的システムを考慮したアルゴリズムも開発されるべきだ。特に，回生インバータ設備をどのように制御したらよいのかについてはいまだ明確な方針は確立されていないため，今後の研究成果が待たれる。

また，実際のシステムに適用する場合，ここでの検討では考慮されていなかった問題，例えば：

(1) 計算処理に要する時間
(2) 通信に要する時間
(3) 制御に用いる各種データ（列車位置・速度など）の精度
(4) 送出電圧変化速度の上限

などをもアルゴリズム上考慮する必要があろうが，これらもすべて今後の課題として残されている。

一方，回生ブレーキの有効性と列車の走行パターンとの関係も重要である。この観点からは，変電所のみならず列車に対するリアルタイム制御を行い，列車の運行上調整可能な範囲内で意図的に力行車と回生車を共存させるように走行パターンを変更することも考えられてよいと思われる。**9章**で検討する列車主回路電力制御による回生失効防止も，そのようなアイディアのひとつである。

なお，本論文では取り扱わなかったが，**ピークカット**制御にリアルタイム制御を応用することも考えられており，いくつかの研究成果の発表もなされている[1][24]。リアルタイム制御によるピークカットでは，重負荷時にも変電所 V–I 特性の変更だけによるピークカットより質の高い電力供給ができる可能性がある。ただし，従来の研究では変電所 V–I 特性の変更だけによるピークカットとの優劣などの比較はきちんと行われておらず，リアルタイム・ピークカット制御アルゴリズムそのものの研究とあわせて将来の課題として残されている。

9

列車主回路電力制御による ピークカット・回生失効防止

⟨4.2⟩（15ページ）で述べたように，電気鉄道の負荷電流は，その最大値と平均値の比が非常に大きい性質を持つ。このため地上に置かれる電力設備の稼働率は低くならざるを得ない。電力回生ブレーキを持つ電気車の大量導入は，この傾向に拍車をかけたうえに，車両の持つ電力回生能力が十分生かされない，いわゆる回生失効という問題も浮かび上がらせた。そこで，電力システムをインテリジェント化し，これらの問題を解決することが考えられている[13]。例えば，何らかの制御により負荷電流のピークを抑制すれば，電力設備の機器利用率は向上する。また，直流饋電システムでは回生失効は回生電力が付近に存在する別な負荷によって消費されないと起こるので，回生電力がより遠くの負荷まで届くようにすれば回生失効は防止できる。

饋電システムインテリジェント化に関する従来の研究では，地上の電力設備に対して制御をかける方法—変電所の電圧制御など—が専ら研究されてきた[1][19]。これに対して，本章では列車主回路電力制御による新しい方法を提案する。この方法では，ある変電所の電流が大きい場合に付近を力行中の列車が主回路電力を抑制することで変電所電流の抑制を図る。これによれば，変電所電流のピークは従来の手法では考えられないほど（最大約31%）激減し，電鉄負荷の様相は一変する。また，主回路電力を制御することにより生じる列車の運行乱れは無視できるほど少ない。また，この手法を応用して回生失効についてもほぼなくすことが可能である。

本章では，まず列車主回路電力制御の概念を述べる。次いで，これを応用して変電所電流のピーク抑制および回生失効防止を行うシステムを列車・地上間通信を行わずに実現する方法を述べ，制御の効果をシミュレーションによって明らかにする。その後，列車・地上間通信を行なった場合のシステムの改良可能性についても検討する。

なお，以下の議論では，駅間での運転パターンは「加速（力行） → 惰行 → ブレーキ → 停車」という簡単なものを仮定する。

⟨9.1⟩ 列車主回路電力制御の概念

⟨9.1.1⟩ 列車力行電力の制限と列車遅れ[46]

例題として電鉄用変電所の負荷電流のピークカット制御について検討しよう。ある変電所の負荷電流のピークを減少させるには，

1. 変電所の電圧制御により付近の変電所に救済させる
2. 変電所付近の列車の力行電力を減少させる

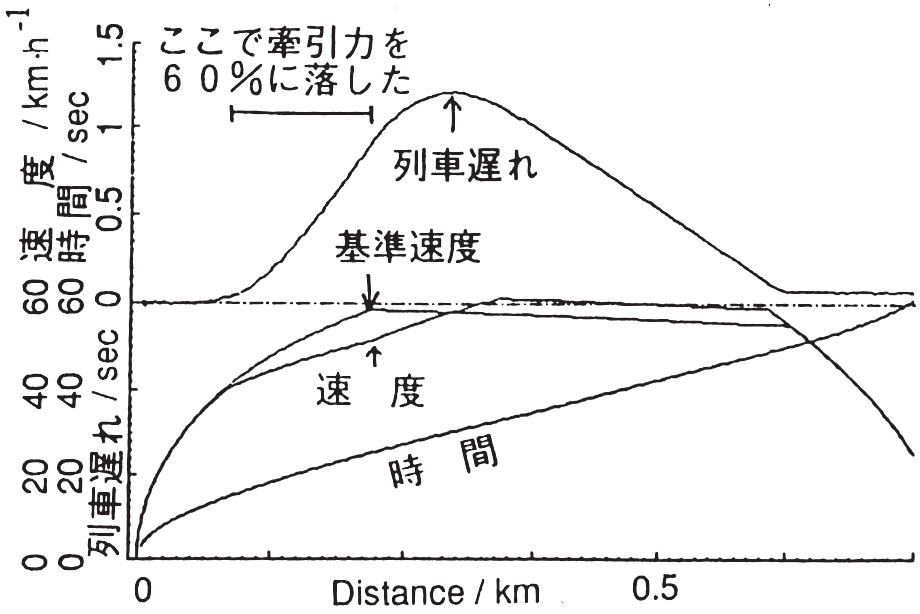


図 9.1: 主回路電力制御と列車遅れ

3. 変電所付近の列車の運動エネルギーを，減速が必要ない場合であっても回生ブレーキにより饋電システムに返させる

という3つの方法が考えられる。従来の研究では1のみが専ら研究されてきたのだが，この方法によって得られる効果はそれほど顕著ではなかった[1]。

もし，2および3が使えるならば，ピークカットをより効果的に実現できる。しかし，これを行なえば当然列車は遅れると懸念される。そこで，列車の力行引張力を一定時間減少させたときの列車遅れを計算した。結果を図9.1に示す。加速開始後15秒から25秒までの10秒間，力行引張力および電流を最大加速時の60%に落としたケースである。絞り込み量が大きいにもかかわらず，遅れは最大1.3秒程度と小さく，通常の列車運行においては無視できる。

このことから，比較的高速域にある列車ならば，主回路電力の絞り込みによる遅れは通常の列車運行における乱れと比べて非常に小さく，効果的かつ大幅なピークカットを行なう可能性があることがわかる。なお，より低速域での絞り込みは大きな遅れにつながるおそれがあるが，低速では主回路電力も小さく，主回路電力制御の効果も小さくなるので，問題は少ない。

⟨9.1.2⟩　列車の状態遷移による救済

⟨9.1.1⟩の結果から，低速域以外であれば惰行中の列車が一時的に電力を回生して減速したり電力を消費して力行することも同様に可能と考えることができる。これを列車状態遷移と呼ぶ。この方法を用いることによって，フライホイールのように列車それ自体の運動エネルギーを電力の融通に使うことが可能となる。

これを実現するときには，効率を不必要に下げないように列車が「回生のみ」なる状態をあらたに持つ必要がある。通常の「ブレーキ」状態では回生ブレーキ力が不足すれば空気ブレーキによる補足が自動的に行なわれる。そこで,「回生のみ」状態では回生ブレーキ力のみでブレーキをかけるものとすれば，空気ブレーキによる運動エネルギーの損失を防ぐことができる。

また，列車状態遷移を回生失効防止制御に応用することもできる。回生失効が起きそうな場合には惰行

中の列車を力行状態に転じさせて負荷を増やせば，回生失効を防止できる。

⟨9.2⟩　列車・地上間通信を行わない場合の制御法

⟨9.2.1⟩　定性的な制御ルールの記述

⟨9.1⟩にて述べた列車主回路電力制御の考え方を，変電所ピークカット制御および回生失効防止制御に応用した場合の制御ルールを検討する。

まず，変電所電流のピークカットを行うためには，列車は次のようなルールに従って行動すればよい。

1. 変電所電流が過大ならば，速度の高い力行列車は力行電流を絞る。
2. 変電所電流が過大ならば，速度の高い惰行列車は回生状態に転移する。

また，回生失効を防止するためには，列車は次のようなルールに従って行動すればよい。

3. 回生車が多ければ，惰行列車は力行状態に転移する。

なお，このモデルでは列車が力行ないしは惰行状態からブレーキ状態に遷移する条件は「一定減速度で減速して所定の位置に停止できること」である。従って，ブレーキ状態の列車は必ず駅の停止目標を狙ってブレーキをかけているので，列車がブレーキ状態にある間はこのような制御は行わない。

⟨9.2.2⟩　フルノッチ比

まず，力行車を考えてみよう。

インバータ制御またはチョッパ制御（界磁チョッパを除く）の電気車であれば，出しうる最大の牽引力より小さければどんな値でも自由に好きな牽引力を出せる。また，パンタ点から見た効率が牽引力の値によらず一定と仮定する。このとき，列車速度 v のときに，列車がその速度における最大牽引力 $T_{max}(v)$ を発揮している（これを「フルノッチで加速している」と呼ぶ）ときの電流を $I_{max}(v)$ とすれば，牽引力 T を出しているときの電流 I は

$$I = \left(\frac{T}{T_{max}}\right) \cdot I_{max} \tag{9.1}$$

で与えられる。ここで，$T/T_{max} = r$ とすれば，

$$T = r \cdot T_{max} \tag{9.2}$$

$$I = r \cdot I_{max} \tag{9.3}$$

と与えられる。r は牽引力および電流のフルノッチ時に対する比率を表すことになるので，これをフルノッチ比と定義する。

フルノッチ比は力行時には 0 〜 1，回生時は 0 〜 −1 の範囲の値をとるものとし，回生車も力行車のケースと同様に定義する。

従来の走行パターンにおいては，フルノッチ比は力行時には 1，惰行時には 0，最大減速度でのブレーキ時には −1 となる。

以下では，この量を用いて制御アルゴリズムを記述する。

⟨9.2.3⟩　フルノッチ比を用いた制御方針の記述

まず，列車・地上間通信は行わないから,「変電所電流が過大」「回生車が多い」という饋電システム側の状態は，饋電線電圧から推定する。列車が知ることができる饋電線電圧はパンタ点のものだけなので，けっきょくパンタ点電圧をもとに推定することになる。

変電所電流が大きくなれば一般に送出電圧が下がるので，パンタ点電圧が「低い」ときには変電所電流が「過大である」と，パンタ点電圧が「低くない」ときには変電所電流は「過大でない」と，それぞれ判

断する。また，回生車が多ければ饋電線電圧が上昇するので，パンタ点電圧が「高い」ときには「回生車が多すぎる」と，パンタ点電圧が「高くない」ときには「回生車は多すぎない」と，それぞれ判断する。

また，力行電流を絞ることはフルノッチ比を下げることである。同様に，惰行列車が回生状態に転移するのはフルノッチ比を下げることであり，惰行車両が力行状態に転移するのはフルノッチ比を上げることである。

これらから，⟨9.2.1⟩の制御ルールはフルノッチ比およびパンタ点電圧を用いて次のように書き換えることができる。

1. パンタ点電圧が低くなったら，速度の高い力行・惰行列車はフルノッチ比を下げる。
2. パンタ点電圧が高くなったら，惰行列車はフルノッチ比を上げる。

⟨9.2.4⟩ フルノッチ比決定アルゴリズム

⟨9.2.3⟩の制御ルールに従った制御を実現するためには，力行列車・惰行列車についてフルノッチ比を従来の走行パターンとは違う値にする必要がある。そのためのアルゴリズムとして，次のようなものを提案する。

⟨**9.2.4.1**⟩ **力行列車** 力行中の列車は，フルノッチ比を図9.2もしくは式(9.4)のように，パンタ点電圧によって変更する。

$$r = \begin{cases} r_{min} & \text{for} \quad V \leq V_1 \\ r_{min} + (1 - r_{min}) \cdot \dfrac{V - V_1}{V_2 - V_1} & \text{for} \quad V_1 < V < V_2 \\ 1 & \text{for} \quad V \geq V_2 \end{cases} \tag{9.4}$$

ここに，V はパンタ点電圧，V_1 および V_2 は「電圧が低い」領域の上限を表わす電圧定数。

式(9.4)中の変数 r_{min} は**最小フルノッチ比**と呼ぶ。これが1であれば「電圧が低い」領域でも力行電力の絞り込みは行われない。また，これが0であれば絞り込みがフルに行われる。そこで，この最小フルノッチ比を速度 v の関数として図9.3もしくは式(9.5)にて与えることにより,「速度の高い」列車に対して

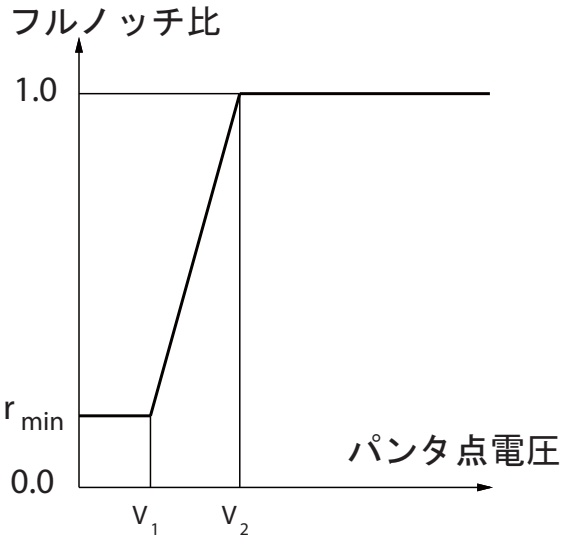


図 9.2: 提案方式における力行車のフルノッチ比 – パンタ点電圧特性

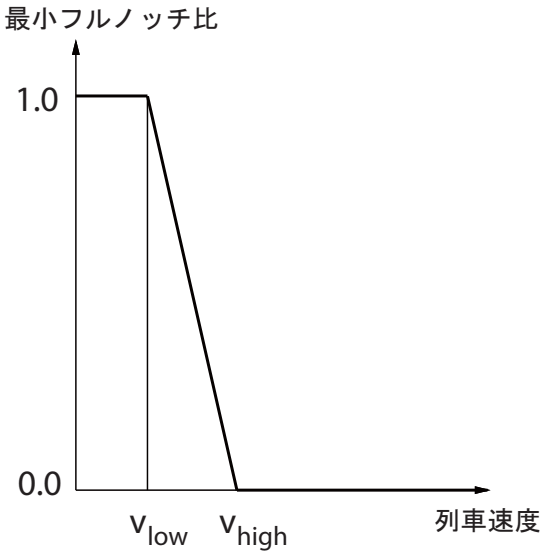


図 9.3: 提案方式における最小フルノッチ比–速度特性

だけ絞り込みを行うようにすることができる。

$$r_{min} = \begin{cases} 1 & \text{for} \quad v \leq v_1 \\ 1 - \dfrac{v - v_1}{v_2 - v_1} & \text{for} \quad v_1 < v < v_2 \\ 0 & \text{for} \quad v \geq v_2 \end{cases} \tag{9.5}$$

ここに，v_1 および v_2 は「速度が低い」領域の上限を表わす速度定数。

⟨9.2.4.2⟩　惰行列車　惰行中の列車は，フルノッチ比を図9.4もしくは式(9.6)のように，パンタ点電圧によって変更する。「惰行」状態は，従来はまったく牽引力を発揮しない状態であったが，提案するシステムでは「電圧が低いと弱い回生ブレーキをかけ，電圧が高いと弱い力行をし，電圧が低くも高くもないと何もしない」状態と再定義される。

$$r = \begin{cases} -r_{max} & \text{for} \quad V \leq V_1 \\ -r_{max} \cdot \left(1 - \dfrac{V - V_1}{V_2 - V_1}\right) & \text{for} \quad V_1 < V < V_2 \\ 0 & \text{for} \quad V_2 \leq V \leq V_3 \\ r_{max} \cdot \dfrac{V - V_3}{V_4 - V_3} & \text{for} \quad V_3 < V < V_4 \\ r_{max} & \text{for} \quad V \geq V_4 \end{cases} \tag{9.6}$$

ここに，V, V_1 および V_2 は式(9.4)に同じ，V_3 および V_4 は「電圧が高い」領域の下限を表わす電圧定数。

式(9.6)における r_{max} は**最大フルノッチ比**であり，式(9.5)の r_{min} を用いて次式のように与えられる。

$$r_{max} = 1 - r_{min} \tag{9.7}$$

⟨9.3⟩　列車・地上間通信なしの場合のシミュレーション

⟨9.2.3⟩で述べたように列車の主回路電力を制御した場合と，しない場合とについて，シミュレーションを行い結果を比較した[47]。

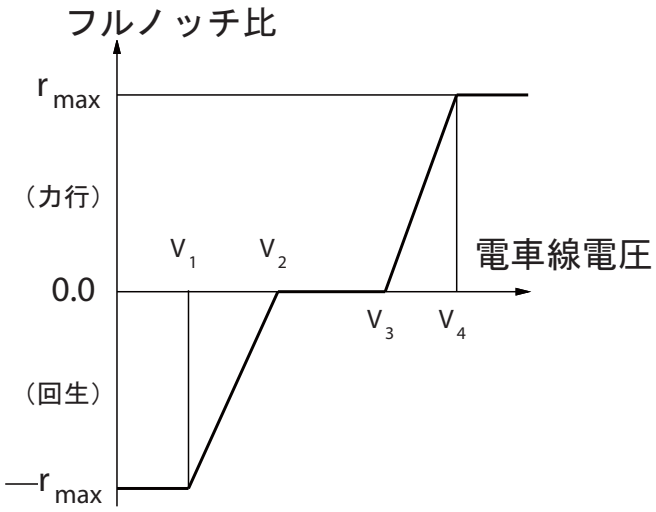


図 9.4: 提案方式における惰行車のフルノッチ比 – パンタ点電圧特性

⟨9.3.1⟩ シミュレーション条件

シミュレーションには **RTSS** を使用した。RTSS については，**5章**，または付録を参照されたい。シミュレーション条件は以下の通りとした。

1. 路線: JR山手線
2. ダイヤ: 144秒周期，完全に周期的なダイヤ
3. 車両: 小田急1000系インバータ電車
    - 5M5T 高加速 最大力行電流 4400A
    - 本数 50本
4. 変電所: ダイオード変電所
    - 等価内部抵抗 0.025Ω (7200kW相当)
    - 無負荷時送出電圧 1600V
5. 饋電方式: 全線並列饋電，変電所母線で上下線を接続
6. 饋電定数: 0.0327Ω/km
7. その他:
    - 山手線は独立な系とする
    - 列車表定速度は⟨5.4⟩（25ページ）の方法により一定とする

⟨9.3.2⟩ 変電所数通常時

まず，変電所数が山手線全線で11（現状と同じ）の場合について，⟨9.2.4⟩のアルゴリズムにおける電圧定数V_1, V_2 の値を変更してシミュレーションを行った。その結果，変電所電流の1ダイヤ周期（144秒）あたりの2乗平均平方根（以下これを変電所RMS電流と呼ぶ）・変電所入力エネルギーの評価量が同時に最小となる値が存在したので，その点(V_1 =1350V, V_2 = 1450V)を最適な点として，制御を行わなかった場合と比較した。(図9.5〜9.8，表9.1)

ピークカット制御の効果は，ピーク電流値の減少で最大約31%と，非常に顕著である。RMS電流についても最大約10%減少している。このようにピークを大幅に抑制することができると，変電所などの機器設計の考え方が変更でき，容量低減ないしはコストダウンが可能である。また，ピーク率を低く抑制でき

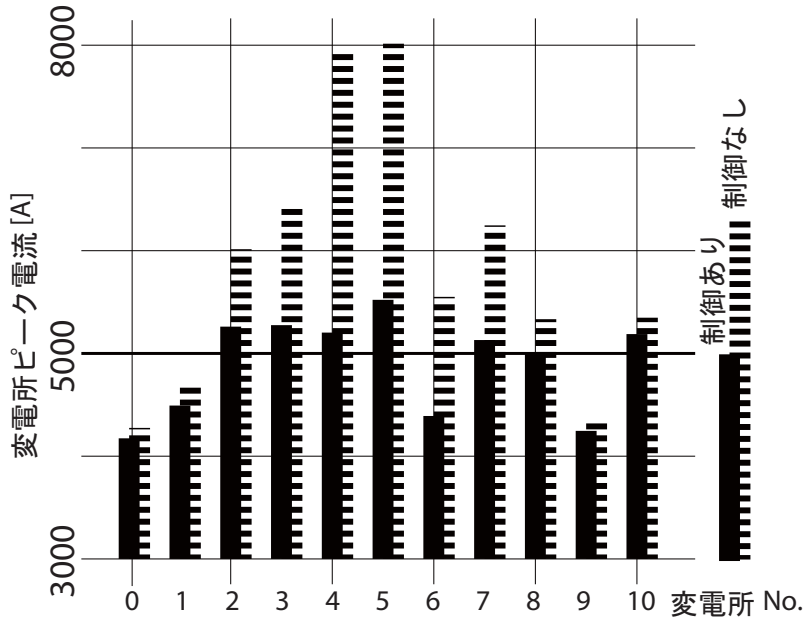


図 9.5: 主回路電力制御による変電所ピーク電流の変化のようす (変電所数 11)

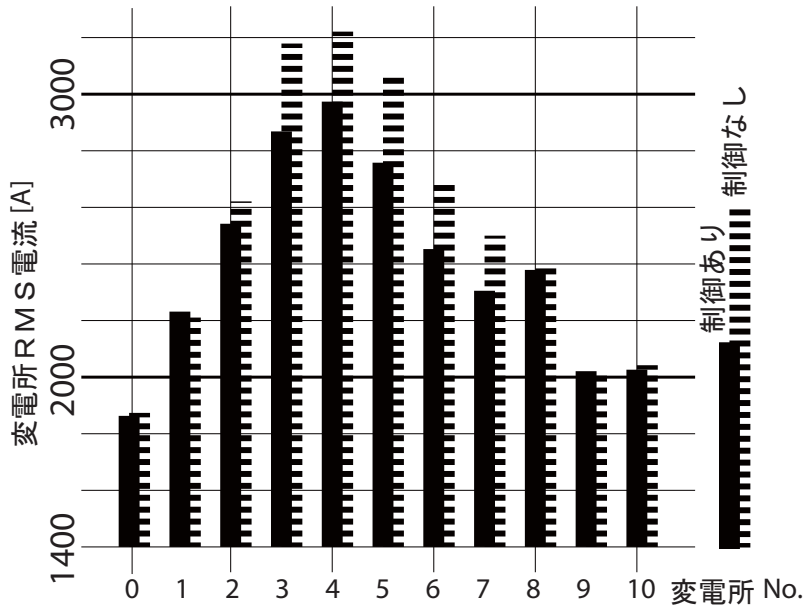


図 9.6: 主回路電力制御による変電所 RMS 電流の変化のようす (変電所数 11)

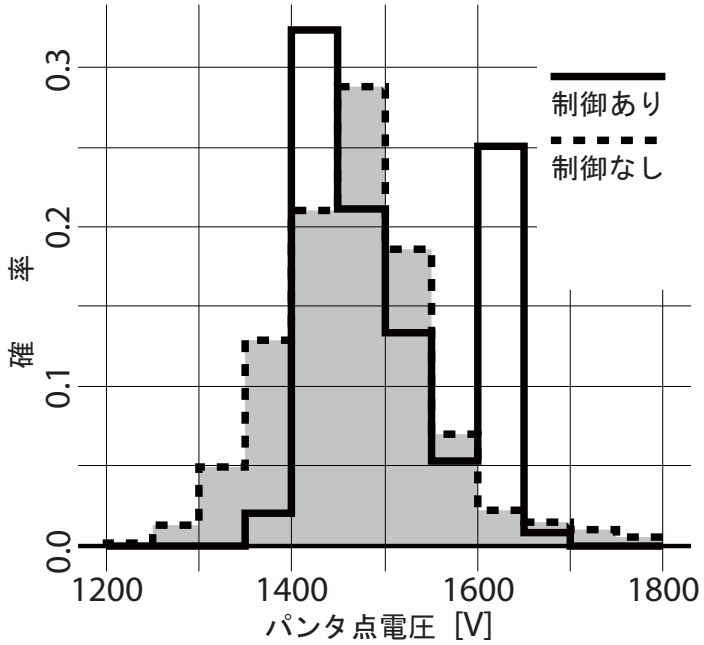


図 9.7: 主回路電力制御による力行車パンタ点電圧ヒストグラムの変化のようす (変電所数11)

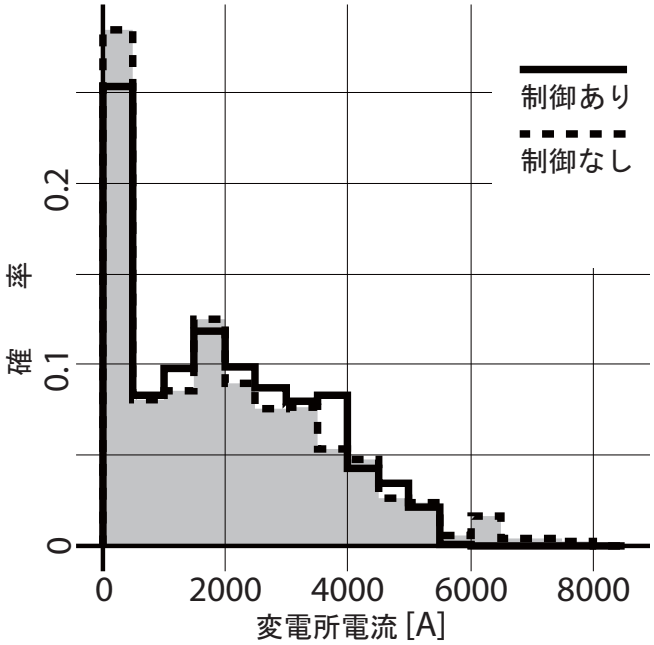


図 9.8: 主回路電力制御による変電所電流ヒストグラムの変化のようす (変電所数11)

ると電気鉄道負荷の電力系統からみた性質が従来よりよくなるから，電源が弱い地域での鉄道電化に対する障害が少なくなる。

また，従来のピークカット制御では，救援される変電所の電圧を周囲より下げるために力行車パンタ点電圧が低くなる確率が増す。しかし，このシステムでは惰行車の回生状態への遷移による救援も加わるために，電圧がごく低くなる確率はむしろ減少する。一方，ピークカット制御の結果変電所入力エネルギー

表 9.1: 主回路電力制御とその他の評価量の変化 (変電所数11)

項　　目	制御あり	制御なし
変電所入力エネルギー [MW]	31.917	31.948
総力行時間 [s]	4052	2793
回生失効率 [%]	0.761	8.80
回生率 [%]	37.6	33.4

- エネルギーは山手線1周分（1時間）の平均とし，単位MWで表示した。この値はエネルギーを単位MWhで表わした場合と数字の上では同一になる。
- 総力行時間は内外回り各2列車が1周したときの力行時間の合計。
- 力行時間とはフルノッチ比が0より大である時間とした。

は上昇するのがふつうだが，ここで提案する列車主回路電力制御によるピークカットでは，後述する回生失効防止制御の効果もあって変電所入力エネルギーはわずかながら減少している。

総力行時間が非常に大きいのは回生失効防止制御を同時に行ったためである。この回生失効防止制御の効果で，回生失効率はほとんど0になっている。回生失効防止のため加速した列車は，走行時分合わせのためあとで回生のみ状態に転じるので，回生失効率低下分は一部をのぞいて変電所入力エネルギーの低下にも寄与している。

⟨9.3.3⟩　変電所数減少時

変電所数が6と，通常より少ない場合についても，同じ手法でシミュレーションを行なった。ほぼ等間隔に従来と同一容量の変電所を配置して，⟨9.3.2⟩と同じく V_1, V_2 の最適値を探した。その最適値 ($V_1 = 1260\mathrm{V}, V_2 = 1360\mathrm{V}$) でのシミュレーション結果を図9.9, 9.10に示す。

このケースでも，ピーク電流値の抑制割合は最大約28%と大幅である。RMS電流値も最大約9%減少し

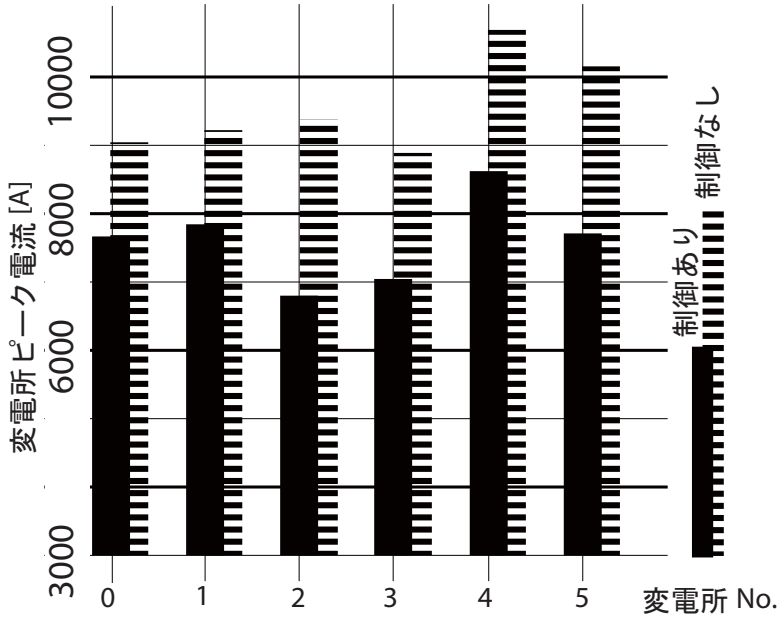


図 9.9: 主回路電力制御による変電所ピーク電流の変化のようす (変電所数6)

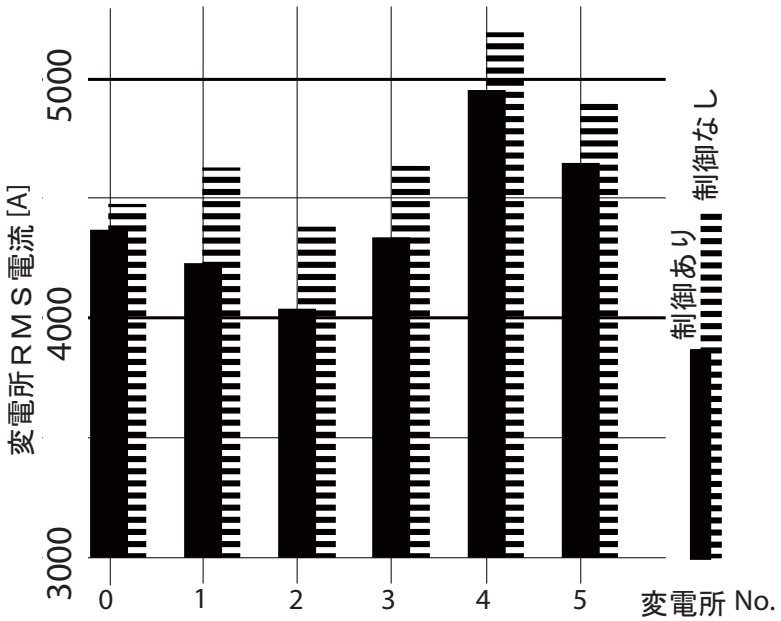


図 9.10: 主回路電力制御による変電所RMS電流の変化のようす (変電所数6)

ているが，変電所の連続定格以上の電流が流れている変電所もある。シミュレーション条件は現状のラッシュ時を想定しているが，一日すべてがこのように重負荷であるわけではない。変電所の遮断器が過電流を検知して動作するほどに大きな負荷電流は流れないことが保証される。そこで，このシミュレーション結果から短時間ならばこの方法によって変電所半減でも運転可能ということができる。

また，熱時定数が小さい半導体ダイオードなどは，ピーク電流値が大幅に減少することから実質的に容量を低減できる可能性がある。熱時定数が大きい変圧器などについても，定格をE種などのピーク率の高いものからピーク率の低いものへ変更できることによるコスト低減や，容量低減の可能性があるといえる。

⟨9.4⟩　列車・地上間通信による定数 V_1, V_2 の自動調整

このシステムに**列車・地上間通信**を付加することにより，システムのさらなる改善が期待できる。特に，列車群制御システムとのトータル化が実現できれば，非常に大きな効果が得られる可能性がある。しかし，ここではより簡単な改善の可能性として，定数を路線条件に合わせて自動的に調節することを試みる。

⟨9.2.4⟩に示したアルゴリズムでは，パンタ点電圧のみを見て制御を行なうが，これでは変電所側で絞り込みの必要なほど大きな電流が流れていないときに列車が電流を減らす行動に出る，またはその逆で絞り込みが必要なときに列車が行動を起こさない，ということがあり得る。その確率が小さくなるように式 (9.4) および (9.6) の定数 V_1, V_2 を選ぶべきだが，変電所の配置は路線の全長にわたって等間隔・等容量とは限らないことから，定数値は列車の位置により変えたほうがよい。また，地上変電所の容量も常時一定とは限らず，工事や故障などにより容量が変化するから，定数もそれに合わせて変化させる機構がある方がよい。

一方，エネルギー消費の面からは無駄な絞り込みは行なわず，なるべく変電所電流のピークは高い状態で使う方が，エネルギー消費は少なくなる。電流絞り込みは列車の性能を等価的に下げるため，ノッチオフが遅く，ブレーキ初速が上がるため、エネルギー消費が多くなる。

そこで，本節では列車・地上間通信を導入し，列車がそれから得られる情報により定数を簡単なアルゴリズムで自動調整することを考慮し，そのための制御ルールおよび制御の実現手法を検討する。

⟨9.4.1⟩ 通信すべきデータと通信量

変電所から列車に対して，変電所の全電流を送信すればよい。通信量は，山手線のケースについて考えると，全列車に対して全11変電所の電流値を0.25秒に1回の割で送信するとき

$$4[\text{回/s}] \times 11 \times 16[\text{bit}] = 704[\text{bit/s}] \tag{9.8}$$

となり，電話線1本分の容量とされる 9600bit/s に遠く及ばないので，特に問題はないと考える。

⟨9.4.2⟩ 制御ルール

定数V_1, V_2が過大であると，変電所の電流が過大でないのに列車が電流を絞り込む確率が多くなる。一方，定数が過小であると電流が過大なのに列車が電流を絞り込まない確率が多くなる。また，省エネルギーの観点からは，絞り込みはなるべく行なわないのがよい。

そこで，次のようなルールが考えられる:

1. 列車の周囲の変電所電流が過大でなければ，その列車の定数V_1, V_2を下げてフルノッチ比を上げる。
2. 列車の周囲の変電所電流が過大ならば，その列車の定数V_1, V_2を上げてフルノッチ比を下げる。

⟨9.4.3⟩ 定数調整アルゴリズム

⟨9.4.2⟩で述べた制御ルールに従った制御を具体化するために，次のようなアルゴリズムを考えた。時刻 t における変電所電流を $I_{Sp}(t)$，その変電所の許容最大電流を I_{Sm}，電流が過大である領域の下限値を I_{Sc}，電流が過大とはなりえない領域の上限値を I_{Sl}，と書くことにする。このとき次式が成り立つ。

$$0 < I_{Sl} < I_{Sc} < I_{Sm} \tag{9.9}$$

力行車だけが存在するシステムについて考える。ある時刻 t における列車配置において，変電所電流が $I_{sp}(t)$ である。列車は電流源に近く，変電所は電圧源に近いから，列車配置が時刻 t と同一で，すべての列車の電流が t における値の

$$C_c = \frac{I_{Sc}}{I_{Sp}(t)} \tag{9.10}$$

倍である場合を考えると，変電所電流はおおよそ I_{Sc} となろう。同様に

$$C_l = \frac{I_{Sl}}{I_{Sp}(t)} \tag{9.11}$$

倍である場合，変電所電流はおおよそ I_{Sl} となろう。

通信は Δt 秒ごと（シミュレーションでは0.25秒とした）に行うものとする。力行車は，ある時刻 t における C_c, C_l の値から，次の時刻 $t + \Delta t$ には定数V_1, V_2を下のルールに従って変更する。

なお，絞り込みが問題になるほど負荷が大きいときには，力行車負荷の影響が支配的であろう。また，定数の変更アルゴリズムを力行車と惰行車で別に持つことも合理的ではない。そこで，惰行車についても，下の同じルールを適用する。

- $V_2 - V_1 = V_d$ は，一定の値とする。
- $I_{Sl} < I_{Sp}(t) < I_{Sc}$ の場合は定数調節は行なわない。
- ある列車が仮に力行しているとしたときのフルノッチ比 r_p は，その列車の時刻 t におけるパンタ点電圧を V_p とする。また，その列車が時刻 t において保持している電圧定数 V_1，V_2 の値をそれぞれをV_{1p}，V_{2p} とする。r_p は式(9.4), (9.5)より

$$r_p = \begin{cases} r_{min} & \text{for} \quad V_p \le V_{1p} \\ r_{min} + (1 - r_{min}) \cdot \dfrac{V_p - V_{1p}}{V_{2p} - V_{1p}} & \\ & \text{for} \quad V_{1p} < V_p < V_{2p} \\ 1 & \text{for} \quad V_p \ge V_{2p} \end{cases} \tag{9.12}$$

ただし, r_{min}は式(9.5)で与えられる定数

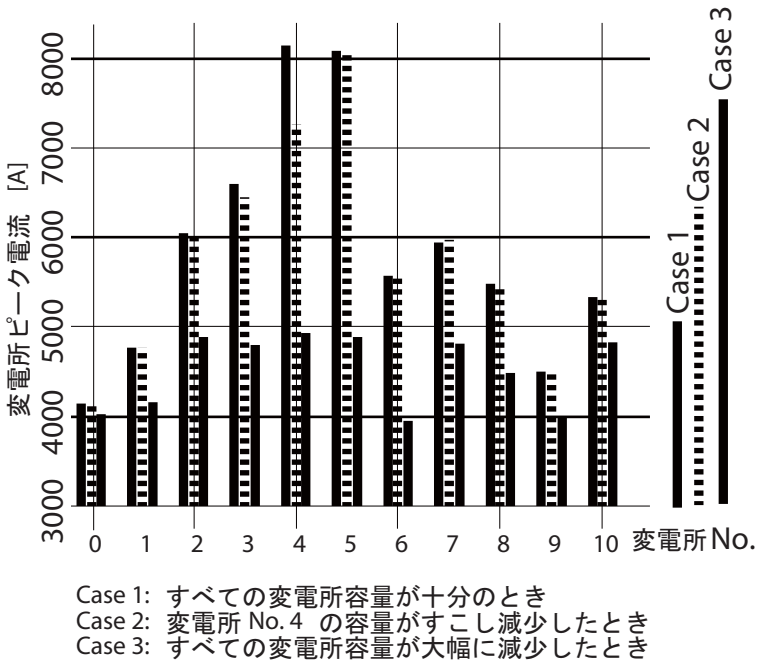


図 9.11: 電圧定数の調整と変電所ピーク電流

と与えられる。

- $I_{Sp}(t) > I_{Sc}$ の場合，フルノッチ比 r_p を $C_c(<1)$ 倍に減らすことを目標とする。式 (9.12) から，定数を調節してもフルノッチ比は r_{min} を下回ることはないことを考慮し，定数調節の目標とするフルノッチ比 r_t を

$$r_t = \begin{cases} C_c r_p & \text{for} \quad C_c r_p >= r_{min} \\ r_{min} & \text{for} \quad C_c r_p < r_{min} \end{cases} \tag{9.13}$$

とする。

- $I_{Sp}(t) < I_{Sl}$ の場合，フルノッチ比 r_p を $C_l(>1)$ 倍に減らすことを目標とする。式 (9.12) から，定数を調節してもフルノッチ比は 1 を上回ることはないことを考慮し，定数調節の目標とするフルノッチ比 r_t を，

$$r_t = \begin{cases} 1 & \text{for} \quad C_l r_p > 1 \\ C_l r_p & \text{for} \quad C_l r_p <= 1 \end{cases} \tag{9.14}$$

とする。

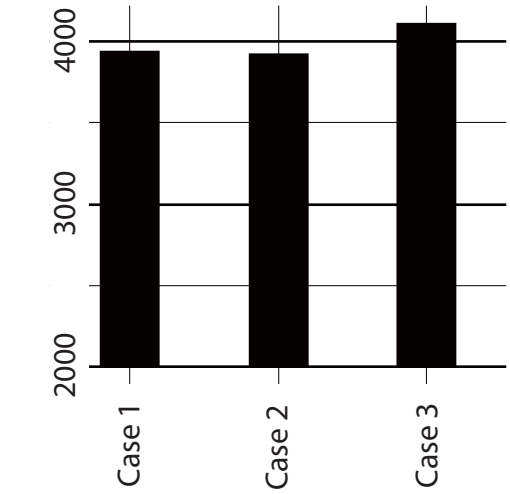
- 列車が時刻 t 現在のその列車の速度およびパンタ点電圧のもとで力行していると仮定したとき，フルノッチ比が，式 (9.13),(9.14) で定めた値になるように，定数 V_{1p}, V_{2p} を変化させる。変化後の値 V_{1t}, V_{2t} は下の式で定められる。この値を，時刻 $(t + \Delta t)$ におけるその列車の V_1, V_2 として定める。

$$V_{1t} = V_p - V_d \cdot (r - r_{min}) \tag{9.15}$$

$$V_{2t} = V_{1t} + V_d \tag{9.16}$$

⟨9.4.4⟩ シミュレーション結果

シミュレーションは，⟨9.3.1⟩（61ページ）にて示した条件で，変電所数11 の場合について行なった。結果を図9.11, 9.12に示す。



Case 1: すべての変電所容量が十分のとき
Case 2: 変電所 No. 4 の容量がすこし減少したとき
Case 3: すべての変電所容量が大幅に減少したとき

図 9.12: 電圧定数の調節と総力行時間

ケース1はI_{sl} =9600A, I_{sc} =12000Aの場合で，容量が十分のためほとんど絞り込みが行なわれていない。

ケース2は変電所No. 4のみI_{sl} =4800A, I_{sc} =7200Aとした場合で，この変電所だけ電流ピークが抑えられており，他の変電所が影響をうけてピーク電流値が大きくなるような現象も現れていない。このように，変電所の特性のある程度の変化に対しても，列車・地上間通信を付加すれば簡単なアルゴリズムで柔軟に対処が可能であることがわかる。

ケース3は全変電所についてI_{sl} =3600A, I_{sc} =4800Aとした場合で，この場合でもピーク電流は抑制されている。しかし，列車の総力行時間が増大し，列車は遅れ，エネルギー消費が増加している。このように，大幅に変電所の容量が減少した場合には，この制御ルールでは変電所ピーク電流の抑制については有効であるが，それ以外の評価量を最適化することができないことがわかる。

列車・地上間通信の存在を前提とした列車群制御システムと電力システムの協調により，より大幅な改善の可能性が開けてくる。ケース3のような場合にも対応してエネルギー消費などの異常な増加を防ぐには，余裕時分再配分制御や適切な遅れの指示など，列車群制御システムとの連係がどうしても必要である。一方，列車群制御を適切に行えば，列車の性能が等価的に上がったような効果が得られ，生み出された余裕を電力システム最適化のためにまわすことも可能になる。

⟨9.5⟩ まとめ

本章では，

1. 列車主回路電力制御による饋電システムのインテリジェント化の可能性を示した。次いで，変電所電流ピークカットおよび回生失効防止制御にこれを応用したシステムを提案し，このシステム導入による効果は非常に大きいことを示した。また，副作用としての列車の運行乱れが無視できるほど小さいことを同時に明らかにした。
2. このシステムに列車・地上間通信を組み合わせ，列車が持つ制御用のパラメータを自動調整するアルゴリズムを提案し，変電所容量が変化した場合でも，このアルゴリズムによればピーク電流を確実に抑制することが可能であることを示した。しかし，列車群制御システムとの協調がないと，

ピーク電流を抑制すること以外のパラメータを最適にすることは難しいことも述べた。

通常，変電所容量が不足している場合には，列車の持っている性能をかなり犠牲にした使い方で切り抜け，変電所容量の増強を待つことが多い。列車の持っている性能をかなり犠牲にした使い方とは，例えば4ノッチまであるのに3ノッチまでしか使わないとか，機関車の牽引トン数を抑制するとかいったことである。また，変電所事故の場合は，復旧までピーク時の輸送力を半減させるなどの対策で切り抜ける。これは，折しも1994年12月10日のJR東日本・山手線・新宿変電所の事故で，事故後実際に行われた運転方法である。

本章の成果は，インバータ制御またはチョッパ制御（界磁チョッパを除く）の電気車が走るほとんどの直流電気鉄道に応用可能であり，その効果も大きい。変電所容量が不足して鉄道の輸送力増強を阻んでいるケースは多く，それらへの適用により輸送力不足を一日もはやく解消することが期待される。また，変電所事故で変電所の一部が使えなくなったような場合にも，この手法を応用してピークカット・負荷平準化を行うことにより，事故後の輸送力をより多く確保することができるだろう。

ただし，特に変電所事故のような大きな事故の場合，ここで述べたピークカット制御による効果でもまだ不足で，さらにピークカット・負荷平準化を行いたいケースが出てくる。その場合，2の末尾で述べたように，列車群制御システムとの協調のもとに列車主回路電力制御を行うことによってより大きな効果が得られる可能性が残されている。この議論については，**１１章**を参照されたい。

１０

エネルギーと経済効果

７章～**９章**にて，いろいろな省エネルギー化・設備利用率向上のアイディアを示してきた。鉄道を運営するのに要するエネルギーの減少は，社会的に見れば環境への影響の軽減やエネルギー資源の枯渇防止に役立つことになろう。この観点は，いうまでもなく重要である。

一方，鉄道を実際に運営している立場からすれば，エネルギーが少なくて済むことにより運営コストの低下が図れる，という経営上のメリットがある。このメリットが大きければ，省エネルギー化の直接の動機となり得る。設備利用率向上も，設備にかかるコストの削減という意味で同様である。

本章では，省エネルギー化・設備利用率の経済効果を簡単に議論することにする。

⟨10.1⟩　省エネルギー化

変電所入力エネルギーが減ることは，いうまでもなく電力会社に支払うべき電力料金が減ることを意味する。それだけではなく，例えば広義の回生失効がなくなった結果省エネルギー化が図られるのならば，回生失効がおきていた場合にはブレーキシューで止めていたものが摩耗部分のない電気ブレーキで止めることができるようになったことを意味する。このことは，ブレーキシューの摩耗が軽減されるため，取り替え費用の減少という形で経済効果が現れるはずである。このように，回生失効はエネルギーの浪費とブレーキシュー摩耗の増加という2つの無駄を生むことになる。

ここでは，この2つを考えて省エネルギー化の経済評価を行う。

⟨10.1.1⟩　ブレーキシュー摩耗

ブレーキシュー1つあたりの値段，および取り替え費用（取り替えのための人件費等）は一定と仮定しよう。ブレーキシュー1つが吸収可能なエネルギーは一定である。岩下[4]によれば，ブレーキシュー1個が吸収可能なエネルギーは約 2×10^9 [J] であり，1個あたり取り替え費用は約 8000[円] と見積もられる。岩下論文のデータはいささか古いが，人件費の上昇とブレーキシュー単価の低減など考え合わせて，ここでは同じデータを使うことにする。

1kWh は $1[\mathrm{kJ/s}] \times 3600[\mathrm{s}] = 3.6 \times 10^6$ [J] なので，

$$8000[\text{円/個}] \times \frac{3.6 \times 10^6[\mathrm{J/kWh}]}{2 \times 10^9[\text{J/個}]} = 14.4[\text{円/kWh}] \tag{10.1}$$

となる。すなわち，1kWh をブレーキシューに吸収させると，14.4円のコストがかかる。

⟨10.1.2⟩　電力料金

電気運転の統計資料[25]によれば，1993年度の全国の民鉄が支払った電力料金の平均値は，14.31[円/kWh] であった。岩下論文では 18 [円/kWh] という数字を採用しているが，その後の外国為替市場における円高ドル安を反映し，差益還元が行われたために料金が安くなってきている。

このように，現在の電力料金水準では，電力料金とブレーキシュー摩耗費用とがほぼ 1:1 となっている。

⟨10.1.3⟩　A 線モデル ⟨7.1⟩（35ページ）における経済効果の試算

さて，この数字を使って A 線モデルにおける省エネルギー化の経済効果を試算してみよう。

A 線の場合で，ダイオード変電所で変圧器 1 次側タップ電圧が 22kV というのが最悪のケースであった。このとき，回生インバータ稼働時の変電所総合入力エネルギーのシミュレーション結果は 5 分時隔時で 5950kWh/h, 7 分時隔時 4200kWh/h となっている。あまり精密な議論はここではしないので，とりあえずこの数字を使うことにしよう。

列車の運転は，平日は 1 日あたり 5 分時隔が 5 時間，7 分時隔が 13 時間と仮定する。休日は 7 分時隔が 18 時間としよう。また，年間 365 日のうち平日が 220 日程度，と見積もることにする。

こうすると，

$$\begin{aligned}&(5950[\mathrm{kWh/h}]\times 5[\mathrm{h/日}]+4200[\mathrm{kWh/h}]\times 13[\mathrm{h/日}])\times 220[\mathrm{日/年}]\\&\quad+4200[\mathrm{kWh/h}]\times 18[\mathrm{h/日}]\times(365-220)[\mathrm{日/年}]\\&\quad=29.5\times 10^{6}[\mathrm{kWh/年}]\end{aligned}\tag{10.2}$$

のようになり，年間の電力は 29,500 MWh，年間電力費用 422 百万円 と見積もられる。もちろん，このほか定常的にブレーキシュー交換費用がかかっているが，それはこの金額の中には入っていない。

ここで，回生失効が減少した結果としてエネルギーの 1%，すなわち 295000[kWh/年] のエネルギーが削減されたとすると，

$$295000[\mathrm{kWh/年}]\times(14.31+14.4)[\mathrm{円/kWh}]=847\times 10^{4}[\mathrm{円/年}]\tag{10.3}$$

のコストダウンとなる。1% のエネルギー削減で年間約 1 千万円のコストダウンであり，これはかなり大きいと見るべきだろう。

ただし，回生失効率の低減によらない省エネルギー，すなわち定電圧領域の拡大による饋電線損失低減によるものなどでは，ブレーキシュー摩耗の低減が期待できないことに注意すべきだろう。

⟨10.2⟩　設備のコスト

では，地上側電力設備のコストはどのくらいのものだろう。残念なことに，電力料金などと異なり電力設備の設置コストには土地代金やコンバータの値段などが含まれるのだが，この値段は正確なところを知るのが難しい。そこで，この節の議論は「だいたいこれくらい」というレベルに留まらざるを得ない。

さて，土地代金を除くと，屋内変電所 1 ヶ所あたりの設置コストはだいたい 10 億円といわれる。最近は更地に建設する場合でも，周囲への騒音など考慮して変電所建屋を建設し，その中に機器を収納することが多い。したがって，電力設備の利用率向上により変電所の建設そのものをやめられれば，コストダウンの効果が比較的大きいことが納得されよう。

一方，容量 6000kW のもので比較すると，変電所用ダイオードコンバータはざっと 1 億円，サイリスタコンバータはざっと 2 億円で，後者の方が 1 億円ほど高くなる。なお，サイリスタコンバータの方が設置スペースが 5 割ほどよけいに必要になるという。回生インバータについてはサイリスタコンバータとほぼ同程度と考えてみよう。

7 章の結果では，回生インバータの設置で 10%もの省エネルギー化が図られることになっているから，年間 8470 万円ものコスト削減効果が得られることになる。このことと，地上設備の寿命が比較的長いことを考慮すれば，このケースでは回生インバータ（A 線で 3 台）の設備は，1 台 2 億円として 6 億円程度であるから，経済的に引き合う投資であるといってよいだろう。一方，サイリスタ変電所化は 1% 程度の省エネルギー化であるので，これがすべて回生失効によるものであるとして年間 847 万円のコスト削減効果

ということになる。1変電所あたり1億円，6変電所で6億円の投資増に対して年間847万円というのはいかにも少ないといわざるを得ない。

なお，この結論自体は必ずしも一般的とはいえない。もう少し路線長が長く，列車密度も高いところでの検討では，回生インバータよりサイリスタ変電所の方が効果が高くなる可能性もある。しかし，議論の方法は同じものがそのまま適用できるはずである。

また，電力料金・ブレーキシュー摩耗費用の低減以外にもメリットが見い出せるなら，総合的な判断によって回生インバータやサイリスタ変電所などの設備を投入することが考えられよう。例えば，実際のA線は自動運転であり，ホームドアがあるため高い定点停止精度が要求された。ところが，回生失効があると電気ブレーキから空気ブレーキへの切り替わりの遅れにより定点停止精度を守れなくなることがわかっている。従って，エネルギーの議論からすれば回生インバータの設置台数を減らす可能性もあるのだが，台数減に踏み切っていない。また，サイリスタ変電所の場合変電所母線容量の増大が図れるメリットやV–I特性変更によるピークカット制御が可能などのメリットもあり，これらが生かせる場合には積極的に使うという判断もできよう。このように，実際の投資に当たっては，いろいろなメリット・デメリットを総合的に判断し，最終的な決断をすることが求められる。

V

統合化鉄道電力システムにおける列車群制御の可能性

１１

ダイヤ小変更によるピークカット

鉄道システムに見込まれているさまざまな「余裕」は，システムを構成する各サブシステムごとにとられるのが普通である。サブシステム相互を統合する統合インテリジェント化の狙いのひとつとして，この「余裕」を統合によって減少させ，生み出された余力をサービス改善に振り向けることがあげられる。

本章では，主としてこの観点から，現状では独立なサブシステムである饋電システム・列車群制御システムの統合化による鉄道サービスの改善可能性を論じる。

⟨11.1⟩　「余裕」減少の可能性

これらの「余裕」を減らすことができれば，鉄道サービスの大幅な改善が見込めるのだが,「余裕」はゼロにすることは一般的に難しい。例えばダイヤの余裕について考えてみよう。余裕ゼロのダイヤを作ったとすると，速達性が改善したり，大幅な輸送力増強が図られたりするかもしれないが，そのダイヤで運行中にちょっとした外乱が入るだけで，システムは大混乱に至ってしまうはずである。このように，余裕は通常は想定されるシステムのトラブルに対応し，輸送の信頼度を確保するための余力であるから，きちんと設計されたシステムであればシステムのトラブルの確率自体が減少しなければその余裕も減少させることはできないはずである。

ところで，もともとサブシステムは単独に存在しているわけではない。例えば，電力システムの容量に余裕が多ければ，ダイヤの余裕も多くなる。また，ダイヤの余裕が多ければ，電力システムへの負担も少なくてすむ。このように，あるサブシステムの余裕を増せば，別なサブシステムにも余裕が生まれることがある。しかし，サブシステムごとに余裕を設ける従来の考え方ではこのようなことは考慮されず，

- あるサブシステムAを除く他のサブシステムのいわば「最悪」な動作状態を仮定する
- その仮定のもとでも運用可能な態勢をAの基準とする
- A内部で想定されるトラブルが生じてもその基準となる態勢をみたせるようにAを冗長設計する

としているものが多く，結果的に余裕が過大となることが指摘されてきた。

仮に，複数のサブシステム間で余裕を「融通」することが可能であれば，各サブシステムの余裕を現在の設計の考え方より減少させることが可能になるはずである。余裕を減らした状態で，異なるサブシステムにおけるトラブルが競合すれば，もちろん現在のシステムよりも混乱が長引くことはある程度やむを得なかろう。しかし，トラブル競合の確率はじゅうぶん低くできると考えられるので，トラブルの競合に対して備えがあるメリットを捨てたとしても，サービス改善によるより大きなメリットが得られると考えられる。

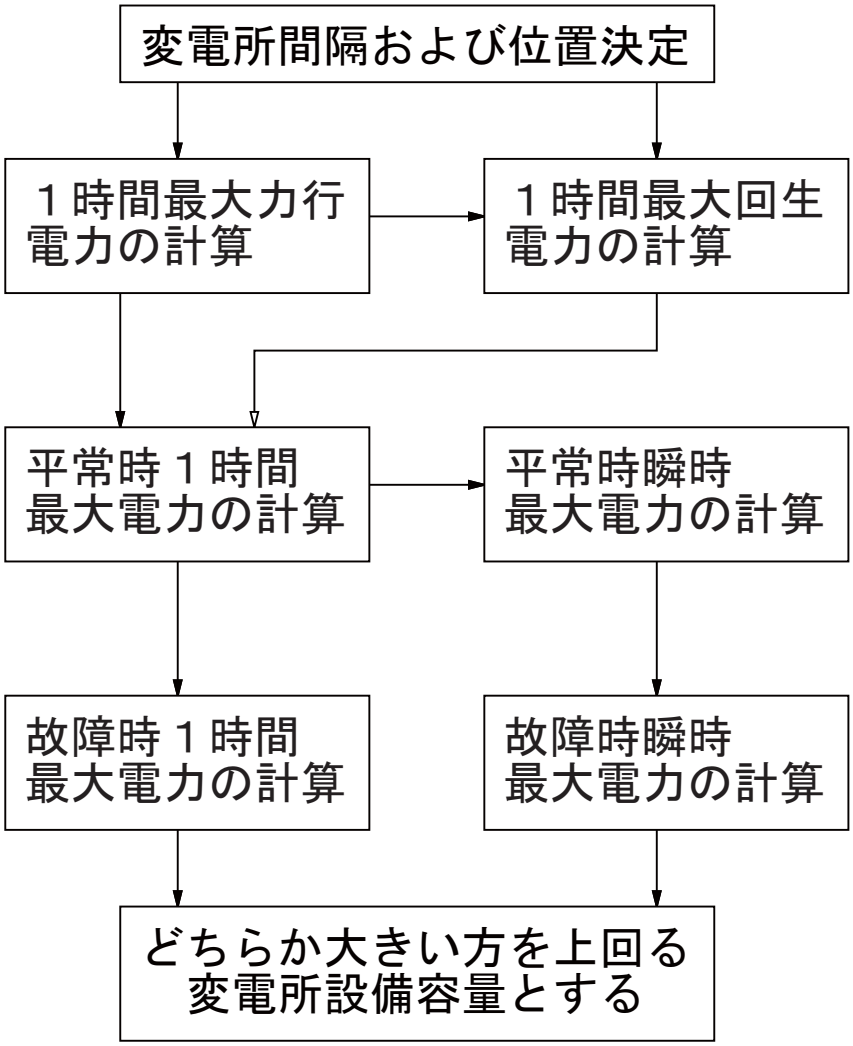


図 11.1: 直流変電所の容量決定フローの1例

⟨11.2⟩　直流鉄道電力システムの設計の現状

さて，その一例として直流鉄道電力システムの変電所容量について見よう。例えば，わかりやすいケースとしてJR東日本における変電所容量の決定フローを文献[20]によって紹介しよう。

決定フローは 図11.1 のようになっている。まず想定される列車負荷から通常時の1時間最大電力（1時間の平均電力の最大値）Y_n および瞬時最大電力 Z_n が求められる。これらから仮に変電所容量を決めた後で，隣接変電所の1組の変成器が脱落した場合の1時間最大電力 Y_a と瞬時最大電力 Z_a の 1/2.5 とを求め，これらのうちでいずれか大きいほうを選ぶ。

これは，変電所に予備の変成器を持たずに，変成器故障時には隣接変電所が負荷を分担する方式である。このほかに，予備の変成器を持って，切替え使用するケースもある。

このように，隣接変電所の機器ダウン時にも運転可能な容量を持たせるのは，変電所機器ダウンにより電源分布が異なっても，列車は電力システムの状況と無関係にダイヤ通りの運転を行ない，結果的に隣接している健全な変電所の負荷が極端に増えてしまうという問題があるからである。そこで，もし仮にこのような負荷の急増を抑制することが可能ならば，このような冗長設計は不要，ということになるはずである。

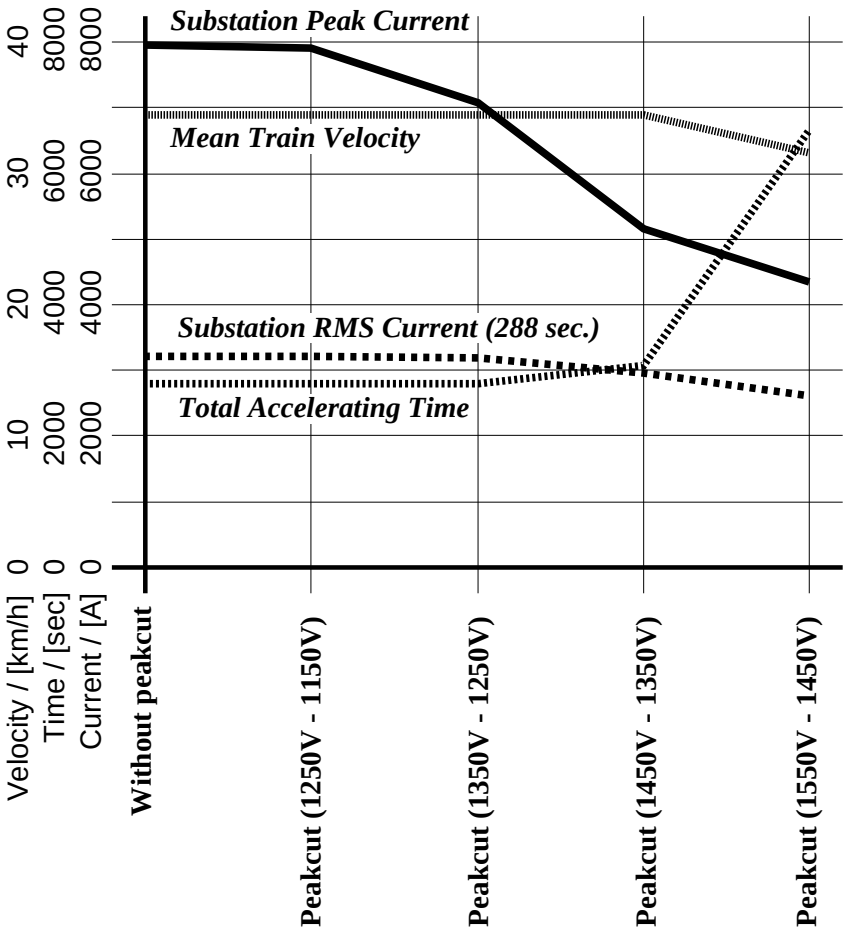


図 11.2: ピークカット制御と列車総加速時間の変化の関係. ピークカット制御は**9章**（56ページ）にて提案した方法で行った.

⟨11.3⟩ ピーク抑制手法としての列車主回路電力制御

我々は，列車の主回路電力の制御という手法で変電所負荷のピークを大幅に抑制することが可能なことを**9章**にてすでに示している[39]。その手法とは，簡単にいうならばある変電所の電流が過大であるとき，その変電所の近傍にいる

- 速度の高い力行列車は力行電流を絞る。
- 速度の高い惰行列車は回生状態に転移する。

という制御をかけることによりピークを抑制しようとするものである。

この手法は，変電所が苦しいときには列車の運動エネルギーをいったん饋電システム側に返却させ，あるいは性能を等価的に下げて，変電所を救済しようとする手法である，といいかえることもできる。もう少し簡単にいえば「列車ダイヤの余裕をピークカットにふりむけている」という見方もできよう。そこで，ピークカットにより列車はどの程度遅れているのかを，シミュレーション結果から見ることにする。

図11.2は，この手法（**9章**において,「列車・地上間通信なしのケース」とあるものを仮定した）によりピークカットを行なった場合，変電所ピーク電流・RMS電流（山手線の11変電所のうちの一つを取り出した)・列車総力行時間（山手線を内・外回り2周ずつしたときの列車の力行時間の総和)・列車平均速度（停車時間を含めた平均速度）をプロットしたものである。シミュレーション前提条件はすべて**9章**と

同一とした。ただし，**9章**ではピークカット制御の他に回生失効防止制御も同時に行っているが，ここでは回生失効防止は行わず，ピーク抑制のみを行っている。図中横軸の Peakcut の4ケースに (1250V - 1150V) などの電圧が記してあるが，これらは提案方式におけるピークカット制御のパラメータ V_1 および V_2 で，数字が大きいほど変電所の電流ピークが低くなる，つまりピークカットの幅が大きいことを示す。

これでみると，ピークカット制御なしのケースと比較して，(1450V - 1350V) のケースで30%以上のピークカット効果が得られていることが読みとれる。このような大幅なピークカット効果が得られたとき，列車の総加速時間（*Total Accelerating Time* とある破線）は増加してはいるが，その増加幅は1割に満たないレベルである。列車平均速度は変化していない。列車平均速度および駅停車時分が同一であれば，総加速時間が短いほどダイヤに余力が多いといえるが，30%以上のピーク抑制にもかかわらず，ダイヤの余裕はあまり減少していないことがこれよりわかる。

このように，ダイヤの余裕のうちわずかの部分を饋電システムのために割くことで，この例のピークカット制御のように饋電システムの大幅な合理化が達成できるのである。

なお，このシミュレーションは全変電所の容量を同一として行なっているが，列車位置に応じて列車が持つ制御用パラメータを変更させることによって，多数の変電所のうちのいくつかの容量が減少した場合にも対応でき，隣接する健全な変電所に負荷の増大が見られないようにすることができることも明らかにしている（⟨9.4⟩, 65ページ，または文献 [39] 参照のこと）。

⟨11.4⟩ ダイヤ小変更によるこれ以上の抑制の可能性

一方，列車側のパラメータのみを変更し，これよりなお大きなピークカットを行わせようとしたものが，図11.2の (1550V - 1450V) のケースである。ピークは確かに減少しているものの，列車総力行時間がピークカット制御なしの場合の2倍以上に増えているにもかかわらず，結果的に列車平均速度が下がり，列車が遅れてしまっていることがわかる。

このことは，変電所容量が当該レベルの列車本数および速度でサービスを継続するのに「そもそも不足」であるところまで下がってしまった場合には，列車の主回路電力制御によってピークカットを行なっても列車は正常に走れない，ということを意味している。したがって，もしも「何らかの手段で列車の速度を下げることが可能」ならば，さらにピークカットを行なう余地が出てくると考えることができる。

ところで，変電所容量が「下がった」場合を考えてみると，全変電所がいっせいにダウンすることは考えにくく，変電所の一部が局所的にダウンした場合を考慮すれば十分であろう。このときに，通常のピークカット制御だけでは間に合わないならば，局所的に苦しくなった区間の余裕時分を増加させて，その分を，他の比較的「苦しくない」区間に割り振る余裕時分再配分制御が考えられる。このように制御すれば，電源ダウンにより苦しくなった区間の周辺では列車速度が下がったような効果が得られ，変電所ピークを余裕時分の再配分前よりさらに効果的に抑制することが可能となる。しかも，列車群の動きは正常な状態とは異なるもののほぼまともであり，完全なシステムダウンには至らないことになる。

⟨11.5⟩ 駅間走行時分と列車消費エネルギー

ここで，駅間走行時分と列車消費エネルギーの関係をもう一度見てみよう[28]。チョッパ車の計算例を見ると，オフブレーキ運転時の駅間走行時分が97秒，これに対して比消費電力量は約56Wh/t/kmである。ところが，駅間走行時分を100秒に約3%伸ばすだけで，比消費電力量は約40Wh/t/kmへと激減（−29%）する。このように，オフブレーキ運転に近い駅間走行時分では，走行時分のエネルギーに対するパラメータ感度は非常に高い。

同じ計算例で，駅間走行時分を100秒から110秒へ10%伸ばすと，比消費電力量は40Wh/t/kmから28Wh/t/kmへと30%減少するが，97→100秒への変化ほどは減少は劇的ではない。さらに110秒から120秒へ約9%伸ばしても，28→25Wh/t/km と約11%減少するにすぎない。

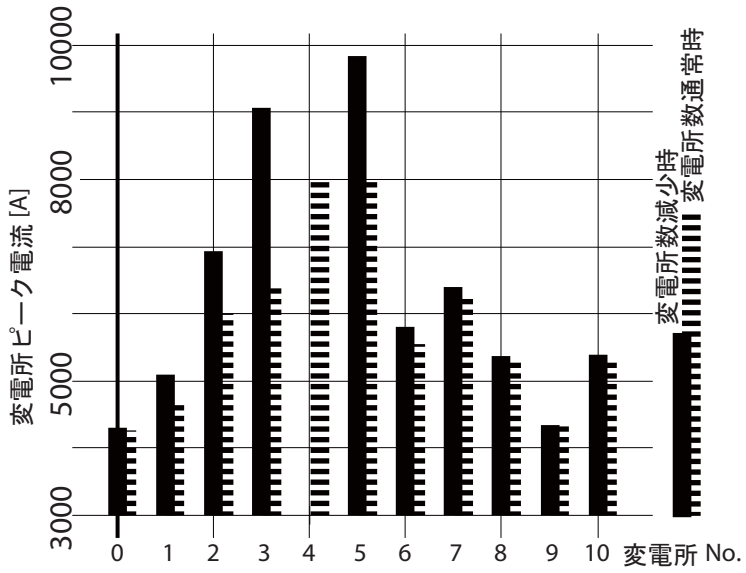


図 11.3: 1変電所脱落時のシミュレーション (1)無理にダイヤ通り走らせたとき．変電所 No. 4 が脱落したものと，そうでないものを比較．ピークカット制御はいずれも行っていない．

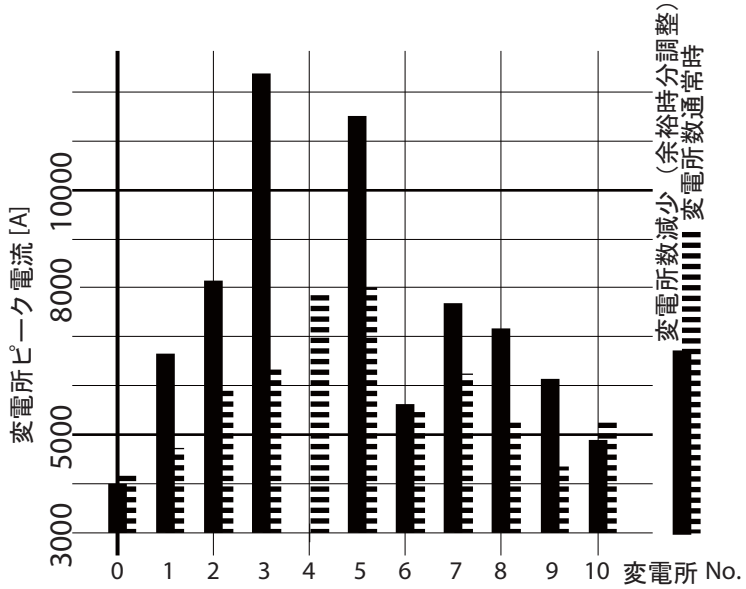


図 11.4: 1変電所脱落時のシミュレーション (2) 脱落変電所付近に余裕時分を再配分．変電所 No. 4 が脱落した場合について，余裕時分を脱落変電所付近に再配分 (最大1駅間あたり 40 秒程度)．ピークカット制御はいずれも行っていない．

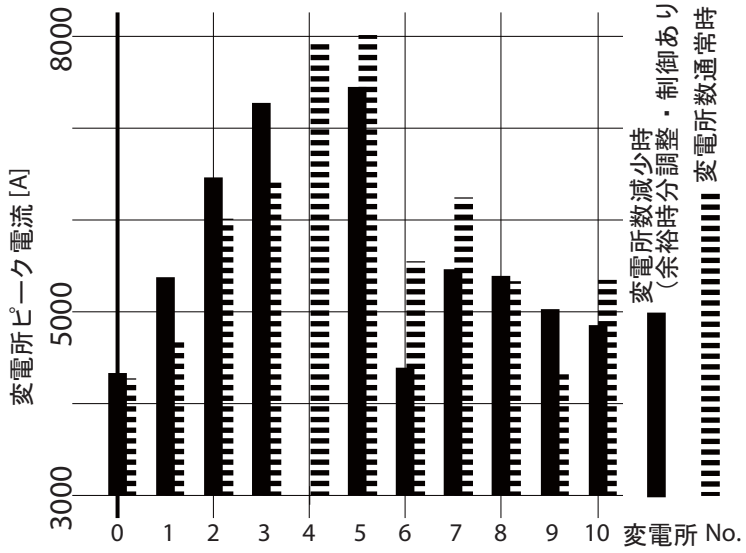


図 11.5: 1変電所脱落時のシミュレーション (3) 余裕再配分＋ピークカット制御. 変電所 No. 4 が脱落した場合について，余裕時分を脱落変電所付近に再配分 (最大1駅間あたり 40 秒程度)，さらにピークカットを行ったものと，事故前でピークカット制御は行っていないものとを比較.

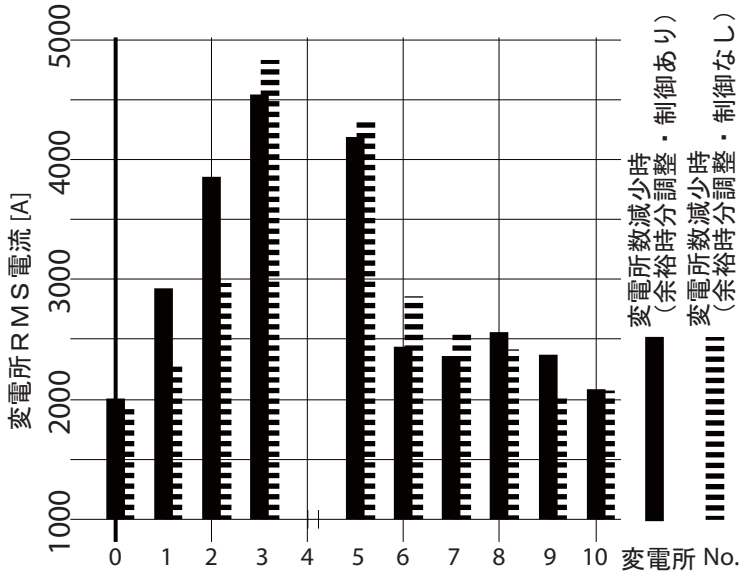


図 11.6: 1変電所脱落時のシミュレーション (4) 余裕再配分＋ピークカット制御 (RMS 電流). 条件は図 11.5 と同一.

通常のダイヤでは，最短運転時間より10%程度の余裕を見ているから，駅間走行時分のパラメータ感度は比較的大きな領域で使われていることが，このデータからもわかる。そして，大幅なピークカットが行なわれているときは列車性能が等価的に下がっていると考えられるので，駅間走行時分の調整，すなわち余裕時分再配分制御は，全体の余裕時分のうち大きくない部分を再配分するだけでかなりな効果を期待できるといえよう。

⟨11.6⟩　ダイヤ小変更によるピークカットのシミュレーション

この効果をみるため，**RTSS**を利用して，JR山手線の変電所（11箇所）のうち1箇所を脱落させた場合のシミュレーションを行った。RTSS については，**5章**，または付録を参照されたい。

シミュレーションモデルは（⟨9.3.1⟩, 61ページ）に示したものと同じである。ただし，変電所は11ヶ所のうち負荷の重い No. 4 変電所をダウンさせるシミュレーションとした。

結果を図11.3〜11.6に示す。図11.3のように単に無理にダイヤ通り走らせると脱落した変電所の前後の変電所のピーク負荷が急増している様子がわかる。ピークカット制御による列車の遅れは1区間あたり最大で40秒程度に達したので，図11.4〜図11.6では，その分の余裕時分を他の多くの区間から少しずつ集めて再配分することにより，山手線1周の時間は変化がないようにしている。

余裕時分再配分だけでは，デマンド管理ができないため，ピーク電流値は図11.4のようにかえって上がる場合もある。ピークカット制御と組み合わせて初めて効果が出る。余裕時分の再配分とピークカット制御を組み合わせることにより，前後の変電所のピークが抑制でき，ほぼ正常な運転ができるようになることが図11.5・11.6よりわかる。

⟨11.7⟩　まとめ

このように，列車が積極的に饋電システムの状況を考慮しつつ動くことの可能性の一つとして，列車群制御システムにおいてダイヤの余裕再配分を行う制御により変成器容量の低減が可能であることを示した。変電所1つがダウンしても正常に近い運行が保てることが，シミュレーションにより明らかにされた。電力システムの冗長設計で考慮することが普通になっている変成器ダウンの予備容量は，ダイヤの余裕を融通してやることで完全に代替可能であろう。

ちょうど，折しも1994年12月10日（土曜日）の深夜に，JR東日本・山手線・新宿変電所で変電所火災があった。山手線以外にも，中央線快速電車，中央・総武緩行線，埼京線（山手貨物線）などが集中する場所での火災であり，直流変電所だけでなくJR東日本の自営交流電力系統の変電所もある場所だったらしく，非常に多数の乗客に影響が出る災害となった。中央線・山手線は電車が動き出したのが翌日（日曜日）の昼であり，しかも山手線は12月13日（火曜日）まで朝ラッシュピーク時の運転間隔を3〜4分（通常2分30秒）に延伸する措置をとった。

大変残念な事故であるが，このような事故は統合インテリジェント化では防ぐことができないのはいうまでもない。しかし，今回の事故では事故後の影響が相当日数尾を引いている問題もある。そして，その影響はこの技術を使えばほとんど回避することができたはずである。特に，山手線のピーク時にこれほどの間引き運転をする必要はなくなるだろう。

もちろん，このようなきわめて稀な事故に備えた冗長設計は冗長度が高過ぎると考えられる。従って，通常はこの余裕を列車の増発に当て，混雑の激しい通勤鉄道の輸送サービス改善に努めることが求められよう。

１２

列車運行乱れ時の省エネルギー

朝ラッシュ時の列車運行はいつも分単位で運行にさまざまの乱れがあるのが普通である。頻繁に起きる現象のひとつに，停車時分が延びて，駅間で列車が信号に従って停止してしまう「駅間停止」がある。これは，それでなくても混雑した状態で乗せられている乗客にとってみればイライラの原因以外の何者でもないが，饋電システムの立場からしても，発進・停止を繰り返すことによりシステムに不要な負担がかけられてしまうことになる。

平行ダイヤの場合には，駅の最小発着時隔の制約だけを考慮すればよいが，追い越し／待避を伴うダイヤ（例えば緩急結合ダイヤ）を採用しているところでは，追い越し／待避駅が特定されているために，追い抜く列車と追い抜かれる列車のペアのうち一方が遅れるともう一方に影響を与えることがある。

本章では，まず駅間停止を防止した場合の効果を論じる。次いで，緩急結合ダイヤにおける運行乱れ時の省エネルギー化についても論じる。

⟨12.1⟩　駅間停止の防止による省エネルギー

駅間停止をシミュレートする場合，RTSS に信号系のインプリメンテーションが現在のところないため，駅間走行時分一定化機能を活用してデータを作ってやることにより，シミュレーションを行った。

⟨12.1.1⟩　駅間停止のシミュレーションモデル

まず，駅間停止がどのようにして起きるかを考えてみよう。

当然のことながら，列車ダイヤを作成する際，ラッシュ時の停車時分および駅間走行時分はある値が仮定される。仮にこの停車時分および駅間走行時分が完全に守られ，ダイヤ通りの運転が行われるなら，列車は常にG現示の信号を見ながら走ることができるようになっている。

ところが，実際には停車時分は混雑により所定の値より10秒〜20秒ほども長くなっている。例えば停車時間40秒のときに信号システムの制約から求められる最小可能運転時隔が T_S[秒] の信号システムが設備されている場合，停車時分が所定40秒のところ50秒へ10秒延伸すれば，最小可能運転時隔は $T_S + 10$[秒] とならざるを得ない。

停車時間が列車によって大きくばらつかない場合を考えよう。もしも，駅停車時分の延びが T_{Dl} であり，ダイヤ上の計画時隔が T_h であるとするとき，

$$T_S + T_{Dl} > T_h \tag{12.1}$$

または，$T_S + T_{Dl} - T_h \equiv T_{\mathrm{delay}}$ とするならば

$$T_{\mathrm{delay}} > 0 \tag{12.2}$$

という不等式が成り立つ場合には，時間間隔 T_h で次々と到着する列車群は1列車あたり $T_{\rm delay}$ 秒ずつ次第に遅れて行き，最終的には駅間停止に至ると予想される。山崎[6]によれば，打子式ATSを設置して，日本最短の1分50秒時隔の列車運行をしている現在の交通営団・丸ノ内線では，A線（池袋→新宿・荻窪・方南町方面）の御茶ノ水駅で実際にこのような状態が現れており，定常的な列車遅れ・駅間停止の原因となっている。

このような現象のシミュレーションを行うためには，簡単なものであっても信号システムのインプリメンテーションがシミュレーションプログラムに必要であるが，RTSS は現在その機能を持っていない。

ところが，実際には通常の運行時における平均的な T_{Dl} が，式(12.2)が成り立つほどには大きくないと思われる場所でも，駅間停止がおきることがある。これは，列車の停車時分に大きなばらつきがあるからである。

例えば，ダイヤ上の停車時分 40秒 に対し，平均停車時分が50秒程度となる駅を考えよう。$T_S + T_{Dl}$ と T_h がほぼ等しいか，わずかに T_h が小さい場合でも，停車時分のばらつきがなければ原理的には駅間停止は起きないはずである。ところが，乗客としての体験があれば誰でも知っていることだろうが，停車時分は確率的な要因で20〜30秒といった長時間延びることがよくある。このように極端な停車時分の延びがあると，1列車あたり $T_{\rm delay}$ 秒ずつの遅れの蓄積を待たず，即座に駅間停車に至る。

もう一つ重要な要因は，駅間停止を伴う運転パターンにおいては T_S 自体も大きくなることだ。駅間停車を伴うケースでは，先行列車が駅を発車してから次列車が到着するまでの時隔（発着時隔）は駅間停車なしのケースより長くならざるを得ないことが知られている。$T_{\rm delay}$ が負であるとき，いったんこのような要因で生じた遅れは1列車あたり $-T_{\rm delay}$ 秒ずつ取り返してゆくことになるが，T_S が大きくなると $T_{\rm delay}$ 自体も大きくなってしまう。すなわち回復余力が減ってしまう。従って，このような駅停車時分の大きな遅延がひとたび発生すると，どうしてもラッシュ時間帯にはその後到着するすべての列車が駅間停止することになる。

このような現象を理解した上で，駅間停止による饋電システムへの影響を明らかにするシミュレーションをもっとも簡単に行うには，駅間に仮想的な「駅」を追加したデータを RTSS に与えて，計算すればよいだろう，ということがわかる。

⟨12.1.2⟩　駅間停止のシミュレーションとその結果

ここでは、JR山手線（全29駅）のうちわずか2駅（新宿・渋谷）の駅停車時分が内・外回りとも延びた条件で、駅間停止を起こした場合と起こさなかった場合についてシミュレーションを行った。シミュレーションでは、⟨12.1.1⟩で述べたように，駅間停止がある場合は駅が増える（全列車，駅間の同一位置に停止する）ように、また駅間停止がない場合は駅間を非常にゆっくり走るように，それぞれデータを与えた。

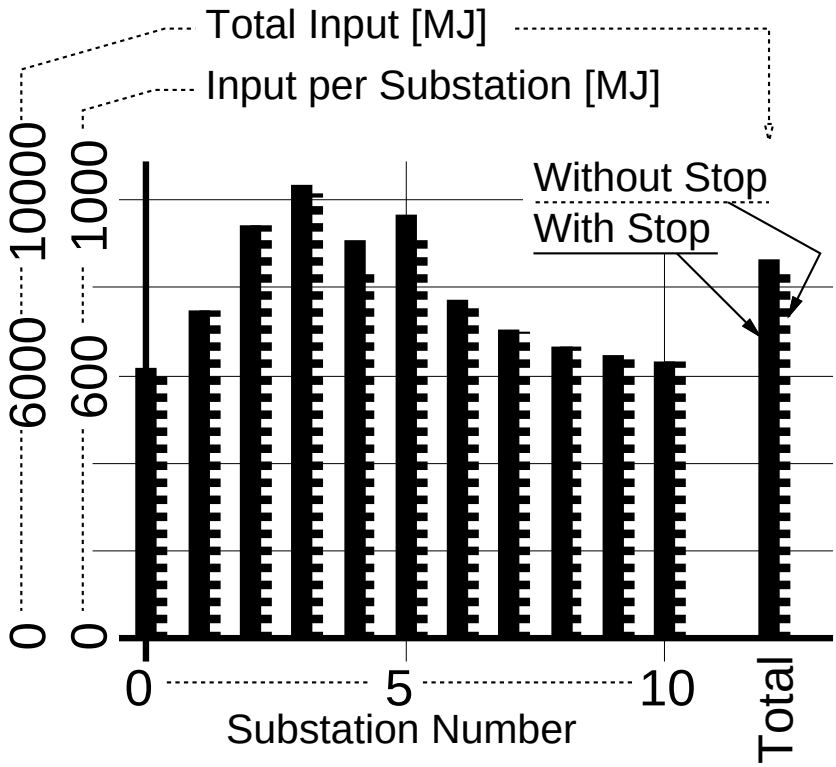
図12.1 は駅間停止を起こした場合と起こさなかった場合の比較である。変電所 No. 4 および No. 5 付近で駅停止時分が増える駅が2駅存在するモデルとなっている。駅間停止すると発着回数が増し、全体の消費エネルギーが2% ほど増える。駅間停止を2駅についてのみ防止しただけでの2%の減少ということは，多数駅について同じような対策をとればかなり大きな効果を期待できることを意味する。

変電所のピーク電流についてもわずかながら増えていることがわかる。ただし，ここで用いているモデルは**インバータ制御電気車**のモデルなので，起動時の電流が**抵抗制御・界磁制御**の電車より少ないことに注意すべきだ。従って，起動回数が増えてもピーク電流値に影響があまり出ないことになる。現在の山手線において使用されている電車は，界磁制御電気車である。

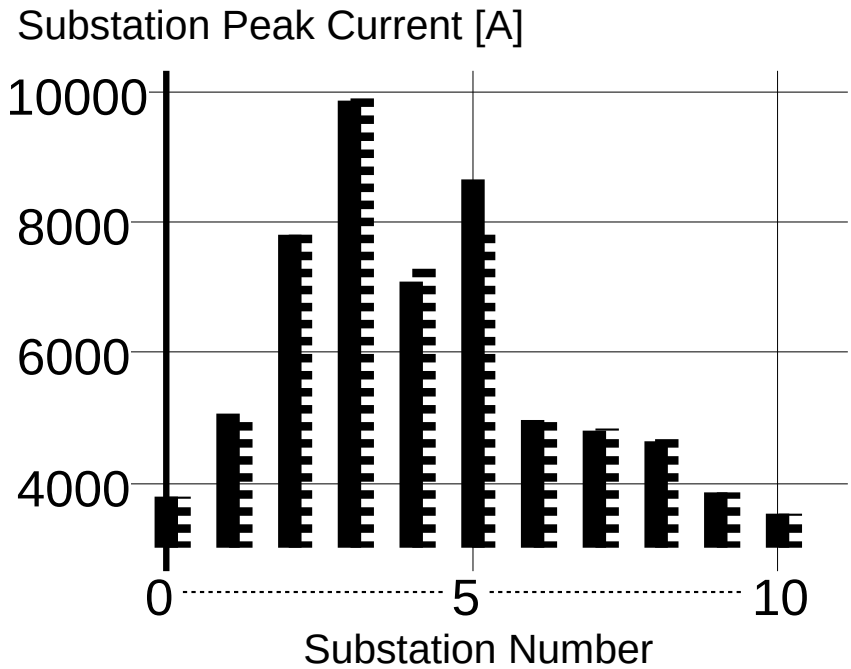
⟨12.1.3⟩　まとめと今後の課題

本節では，駅間停止の防止により，消費エネルギーを相当大きく低減できる可能性を示した。

実際に駅間停車を防止するには，列車・地上間通信を積極的に行い，駅間停止がおきそうな場合には予測によってあらかじめ減速運転を行うなどの対策が必要になる。このこと自体はそう困難な課題とは思われない。ただし，駅停車時分をある程度精密に予測する必要はあろう。



(1) エネルギー消費 (300 秒)



(2) 変電所ピーク電流値

図 12.1: 駅間停止がある場合とない場合の比較

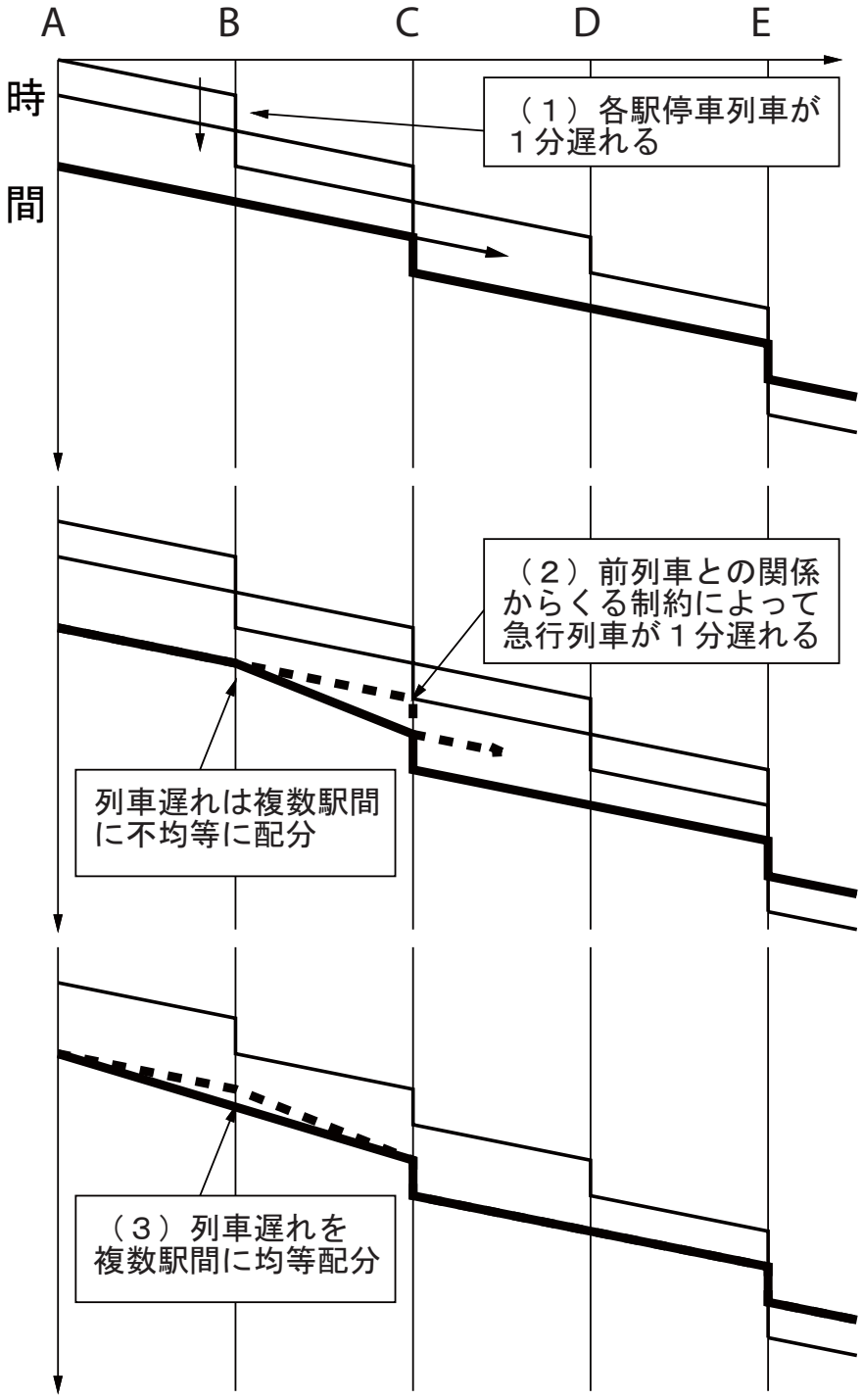


図 12.2: 通過列車の遅れ時分を通過駅間に均等配分する

仮に予測ができたとしても，後続列車の走行パターンによって駅における発着時隔が変化するため、実際にはここで述べたように単純に遅く走ればよいというわけにはゆかない。エネルギー最小化という評価量を用いた走行パターンの最適化の結果と，列車運転時隔最小化という評価量を用いた最適化の結果とで齟齬が見い出される部分でもあろう。しかし、現実の朝ラッシュ時の列車運行においてはかなり多数の駅で駅間停止が常態化しているから、この点を考慮しても駅間停止防止による省エネルギー・ピーク低減などの効果は大きいと期待される。

さらに，混雑した列車の駅間停止は乗客のイライラの原因にもなり得る。このイライラが定量化され，駅間停止防止によってそれが軽減されることが証明できるなら，省エネルギーだけではなく，旅客サービスの向上も期待できる技術であるということができるだろう。

⟨12.2⟩　緩急結合輸送の場合

列車運転に乱れ（遅れ）が生じると，列車は当然回復運転を行わなければならない。しかし，頑張って走っても前方に遅延した列車があって意味がないことがある。ここでは，緩急結合輸送の例について検討してみよう。

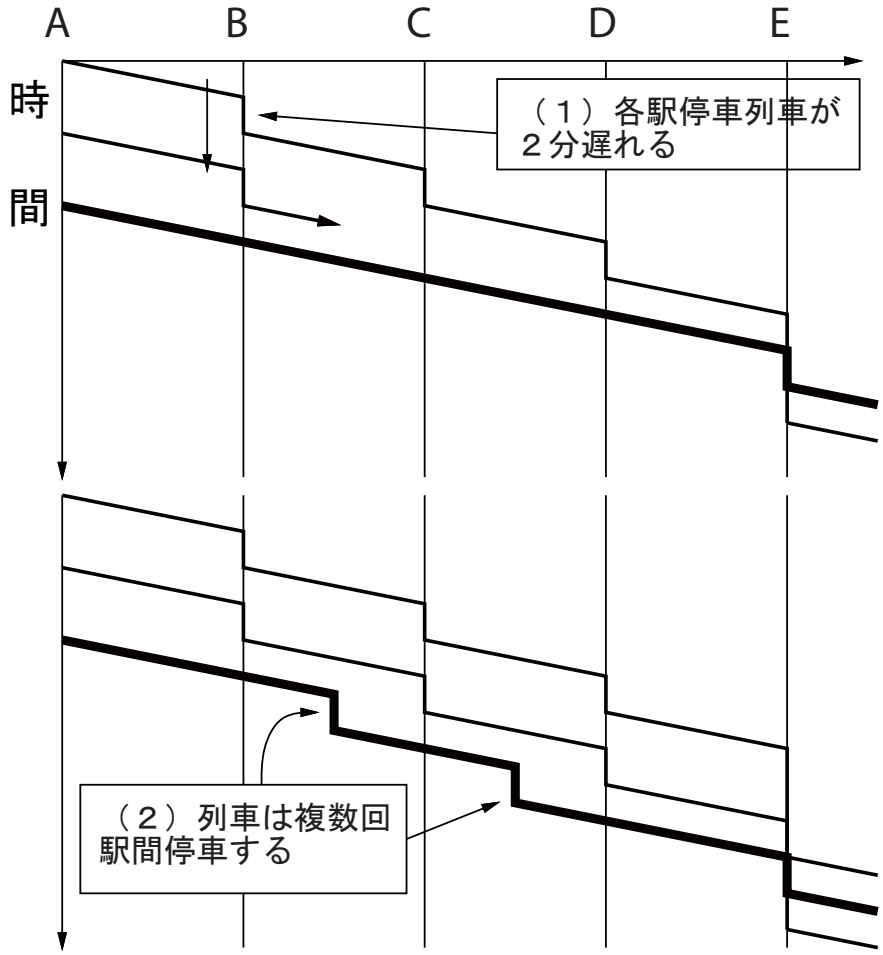


図 12.3: 多数駅間を通過する場合の例

⟨12.2.1⟩　追い越し／待避と列車運行乱れ

2つの例をあげよう。まず図12.2は，(1) のようにまずA駅発の各駅停車列車が遅れたケースである。そうすると，後続の急行列車はB駅までは普通に走ってくるが，C駅で遅れた各駅停車列車に近づき，(2) のように A–B 駅間と B–C駅間とで遅れ時分に差が生じる。これを (3) のように均等配分すれば，省エネルギー化が図れるはずである。

なお，この図では後続の急行列車は影響を受けるのが2駅間のみで，しかも駅間停止はない。実際には図12.3のように前方列車の遅れによって急行列車が多数回の駅間停止を繰り返すようになってしまうことが多い。このような運転になれば所要エネルギーも大幅に増加するだろう。図12.2のケースでも，C 駅付近で後続の急行列車の駅間停止がおきても不思議ではないケースだ。しかし，駅間停止の評価はすでに前節で行ったから，ここでは駅間停止なしのモデルを用いて検討する。

もう1例，こんどは図12.4で，(1) のように急行列車のA駅発が遅れたケースである。ここでは，先行する各駅停車列車はE駅まで普通に走ってゆくが，E駅で自分を抜いて先行するはずの後続列車が遅れているため，(2) のようにE駅での停車時分が長くなる。この長くなった停車時分の分だけ，(3) のようにその前の各駅間の走行時分を延伸してやれば，省エネルギー化が図れるはずである。

なお，あまり極端な遅れを生じた場合，待避駅変更というオプションが存在する。しかし，現在実現している列車群制御システムにおいては，最終的に指令員が指示を出すまでは待避駅変更は行わないのが普通だろう。そこで，本節では待避駅変更までは考えないことでシミュレーションを行った。

⟨12.2.2⟩　緩急結合ダイヤにおける列車運行乱れシミュレーションとその結果

ここでは，通過運転を行う列車のうち，1列車のみが始発駅2分遅延した場合についてシミュレーションを行った。

路線は19駅，6変電所モデルで，列車数は上下線合わせて32列車ある。列車は5 分あたり2本が走る。通過列車と各駅停車列車が交互に走り，路線途中の2駅で緩急接続が行なわれるダイヤとしてある。このダイヤにおいて，上り線の通過列車が1列車のみ始発駅を2分遅延するものとした。各駅停車列車が通過列車を退避する駅は変更しないこと，影響は下り線には波及しないことを前提に，次の3ケースについてシミュレーションを行った。

(1) 後続の列車は定刻に出発，または定刻より遅れざるを得ない場合はなるべくはやい時刻に出発

(2) (1)で設定した通過列車のうち，通過区間となる駅間ごとに遅れ時分が均等配分されていない場合，遅れ時分の均等配分を行う（図12.2）

　このケースではすべての乗客に対し悪い影響は出ない。調整可能な列車は全列車（32運用）のうち2列車5駅間だけであった。

(3) (2)で設定した各駅停車列車が，退避駅で後続の通過列車遅れのために長時間の待ちを余儀なくされている場合，この余分の待ち時間を退避駅より前の駅間の走行時分に加える（図12.4）

　このケースでは，各駅停車列車に乗って退避駅まで（複数駅間にわたって遅らせる場合には関係する駅間すべて含む）ゆく乗客（**OD**(Origin–Destination)**需要**でいうならDが関係する駅に該当する乗客）にとっては列車が遅く着くための不利益を生ずる可能性があるが，それ以外の乗客には不利益がない。調整可能な列車は全列車のうち2列車だけであった。

なお，いずれのケースでも駅間停止はすでに防止の措置がとられたものとしてシミュレーションを行った。特に図12.2のケースでは C 駅付近で後続の急行列車の駅間停止がおきても不思議ではないケースだが，⟨12.2.1⟩で述べたように駅間停止はしないモデルにした。

シミュレーション結果は表12.1に示す。ケース (1) とケース (2) との差が約 0.1% あるのは，関係する列車が2列車・5駅間だけしかない（全体としてみると32列車，のべ576駅間）こと，および駅間停止をあえて避けていることを考えれば大きな効果と考えられる。同様にケース (2) とケース (3) についても同じことがいえる。

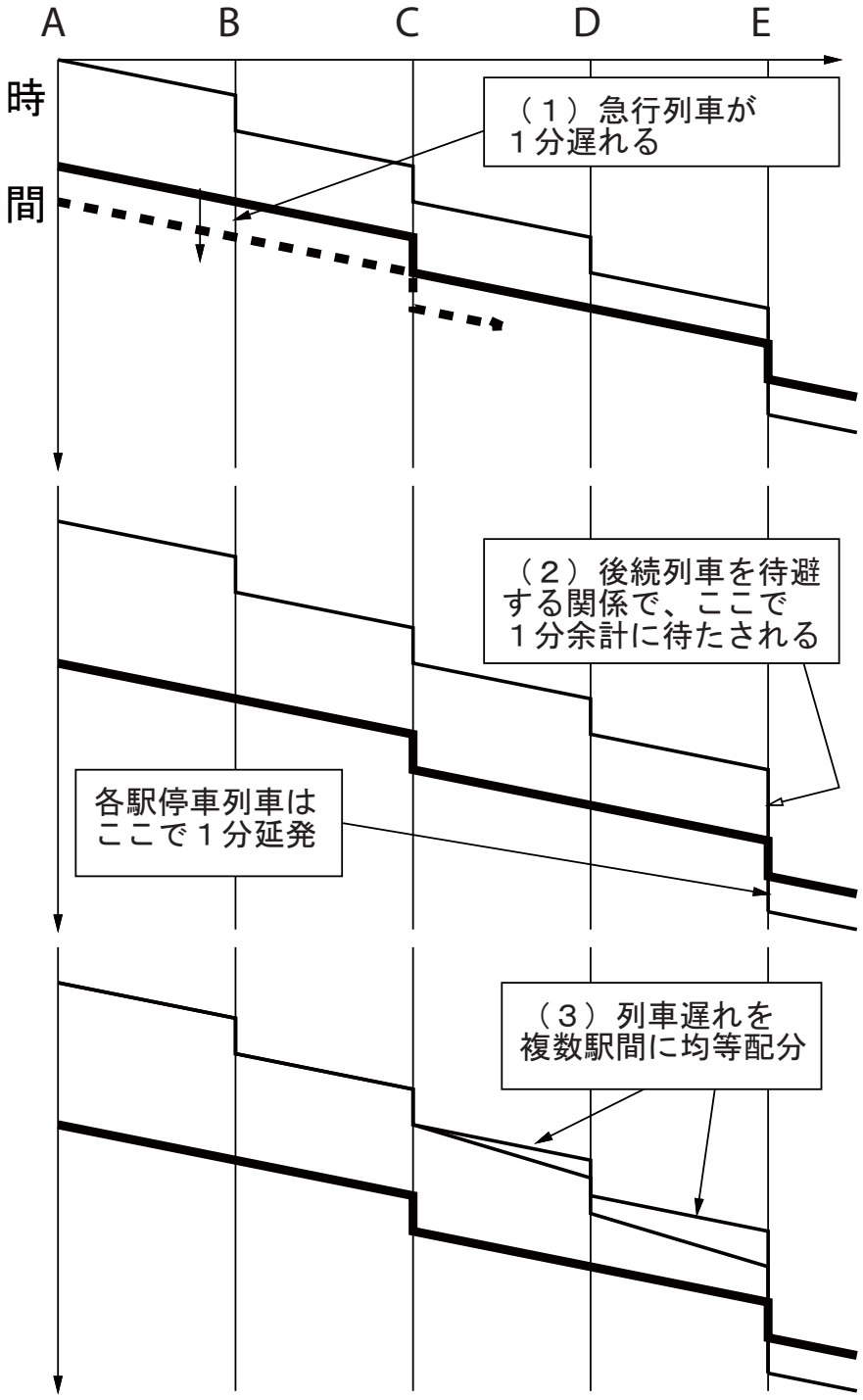


図 12.4: 各駅停車列車の遅れ時分を駅間に均等配分する

表 12.1: 緩急結合輸送における列車運行乱れ時のシミュレーション結果

項　　目	ケース (1)	ケース (2)	ケース (3)
全変電所総合入力エネルギー [kWh/h]	10554.2	10544.7	10503.7
同上：比（ケース (1) を 100 とした）	100.00	99.91	99.52

⟨12.2.3⟩　まとめと今後の課題

始発駅を出発し，途中まで高速で走った急行列車が，先行列車に頭を抑えられて途中から非常なノロノロ運転になる，といった現象は，緩急列車を混在運転している路線においては，朝ラッシュ時頻繁にみられる。この運行乱れの結果「待たされざるを得ない」時間が増大するのだが，この時間を利用することによってこのような省エネルギー化が図れることを述べた。⟨12.1⟩で述べた駅間停止防止の対策と合わせて，比較的列車運行乱れの大きな朝ラッシュ時の省エネルギー化に貢献すること大であると考える。

なお，例えば，⟨12.2.2⟩のケース (2)（図 12.2，84ページ）およびケース (3)（図 12.4, 87ページ）では，複数駅間に遅れを配分する場合，エネルギー的に最適な配分方法があるはずである。その配分方法についてはいくつかの論文がある（例えば，金の博士論文[2]など）。ここでもこれらと同じ方法で議論が可能と思われるが，今回はそのようなことは何も考慮せず，駅間走行時分に対する割合で見て各駅間とも均等になるように，新たな余裕時分を配分した。この検討は今後の課題として残されており，結果次第ではこのシミュレーションよりなお大きな省エネルギー効果を得ることができるようになるかも知れない。

ただし，ケース (3) または図 12.4については，一部の乗客に不利益が生じている。例えば，図 12.4において，先行する各駅停車列車で E 駅までゆき，そこで下車する乗客にとっては，同図 (3) の運行では各駅停車列車が E 駅に 1 分延着する不利益を被ることになる。仮に E 駅が緩急間の乗換より地元降車がはるかに多い駅であったなら，図 12.4(3) のようなパターンはとらないほうがよい。しかし，E 駅が緩急間の乗換がほとんど，というような駅なら，このパターンで問題なかろう。

このように，乗客の被る不利益と省エネルギー効果との関係を把握しておく必要がある。乗客の被る不利益がわずかで省エネルギー効果が大きいなら，それは統合化鉄道電力システムにおける制御戦略として有効といえよう。

１３

列車群の最適走行パターン問題とその数値求解

饋電システムと列車群制御システムとを統合インテリジェント化することにより，必要な冗長度を低減し，既存設備を活用したサービス改善を行う可能性があることを，これまでに多くの例を通して述べてきた。しかし，当然ながらこのような既存設備の活用には限度があろう。この限度を見い出すためには，何らかの最適化手法を応用する必要があると考える。

鉄道の運転において，古くから興味を持たれてきた問題に**列車最適走行パターン問題**がある。この問題は通常1個の単独列車について，所要エネルギーが最小となる走行パターンを探索するものだった。筆者は，この問題を列車群の走行パターン問題に拡張することを提唱した[50]。直流饋電・列車群制御の統合システムにまつわる問題の多くは，**列車群最適走行パターン問題**に帰着させることが可能である。そして，列車群最適走行パターン問題は，次数の大きさを除けば最適列車走行パターン問題と同じアルゴリズムで解くことができる。

本章では，列車群最適走行パターン問題の定式化を行い，この定式化が直流饋電・列車群制御の統合システムにまつわる問題の多くに適用可能であることをいくつかの例によって示す。次いで，解法アルゴリズムを紹介し，このアルゴリズムが列車最適走行パターン問題と列車群最適走行パターン問題のどちらにも適用可能であることを示す。次いで，簡略列車モデルを用い，列車最適走行パターン問題の数値解を示し，その解が定性的に見て妥当な性質を持っていることを検証する。

⟨13.1⟩　定式化

⟨13.1.1⟩　列車最適走行パターン問題の定式化

列車最適走行パターン問題は，次のように書くことができる[49]:

状態変数が $x(t)$(列車位置), $v(t)$(列車速度) で与えられる系の運動方程式が

$$\dot{x} = v \tag{13.1}$$

$$\dot{v} = \frac{1}{M}\{\, T(v,r) - R(v) - G(x) \,\} \tag{13.2}$$

ただし，

$T(v,r)$: 牽引／ブレーキ力
$R(v)$: 走行抵抗
$G(x)$: 勾配抵抗 (曲線抵抗含む)
M : 列車質量等から定まる定数

で与えられている。与えられた時刻 t_0, t_f に対し

$$x(t_0) = x_0, \qquad v(t_0) = v_0 \tag{13.3}$$

$$x(t_f) = x_f, \qquad v(t_f) = v_f \tag{13.4}$$

が指定されている。 このとき，(t_0, x_0, v_0), (t_f, x_f, v_f) を通り，評価関数（パンタ点入力エネルギ）

$$E(r) = \int_{t_0}^{t_f} V I_P(v, r) \tag{13.5}$$

ただし，

V : パンタ点電圧 (一定と仮定)

$I_P(v, r)$: パンタ点電流

を最小化するような牽引力フルノッチ比の関数 $r(t)$ を定めよ。ただし，入力 r は

$$|r| \leq 1 \tag{13.6}$$

という不等式拘束条件を満たさなければならない。また，列車速度 v は

$$v \leq v_{limit}(x) \tag{13.7}$$

という不等式拘束条件を満たさなければならない。

ここで牽引力フルノッチ比とは

$$\begin{cases} T(v, r) = r \times T_{max}(v) & (r \geq 0) \\ T(v, r) = r \times T_{Bmax}(v) & (r < 0) \end{cases} \tag{13.8}$$

で定義される r をいうものとする。$T_{max}(v)$ (> 0) は速度 v で走行中の列車が出し得る最大の牽引力，$T_{Bmax}(v)$ (> 0) は同じく最大のブレーキ力である。

⟨13.1.2⟩ 列車群最適走行パターン問題への拡張

⟨13.1.1⟩で定式化した問題では，回生ブレーキ時の挙動を正しく最適化することができない[50]。例えば，図13.1 のように2列車が存在している場合を考えよう。図中 (1) において，2つの列車の走行パターンは最適列車走行パターンであるとしよう。この場合，力行列車と回生列車が同時に存在することはなく，この間回生失効が起きることになろう。それよりは，パターンを最適なものからずらしてでも (2) のように力行列車と回生列車が同時に存在する時間帯を作る方が省エネルギーになると考えられる。

このように，饋電システムと列車群制御システムとを統合化したシステムにおいては列車群の振舞いを饋電システムを考慮しつつ最適化することが重要である。また、評価関数もパンタ点から列車に入力されるエネルギではなく、より基本的な，変電所での入力エネルギとして考慮すべきであると考えられる。そこで，このような統合システムの検討をする場合には，単独列車の走行パターン最適化ではなく，列車群最適走行パターン問題を解く必要がある。

⟨13.1.1⟩で定式化した問題をわずかに修正すれば列車群最適走行パターン問題の定式化が出来上がる。まず，評価関数 (13.5) はパンタ点入力エネルギーであったが，これを饋電システムの存在を考慮して変電所入力エネルギーとする。あとは状態変数，初期・終端条件，運動方程式，不等式制約条件式を列車の数だけ用意すれば，列車群最適走行パターン問題となる。すなわち，状態変数ベクトルは

$$\boldsymbol{x} \equiv \{x_0(t), x_1(t), \ldots, x_{N_T}(t)\}^T \tag{13.9}$$

$$\boldsymbol{v} \equiv \{v_0(t), v_1(t), \ldots, v_{N_T}(t)\}^T \tag{13.10}$$

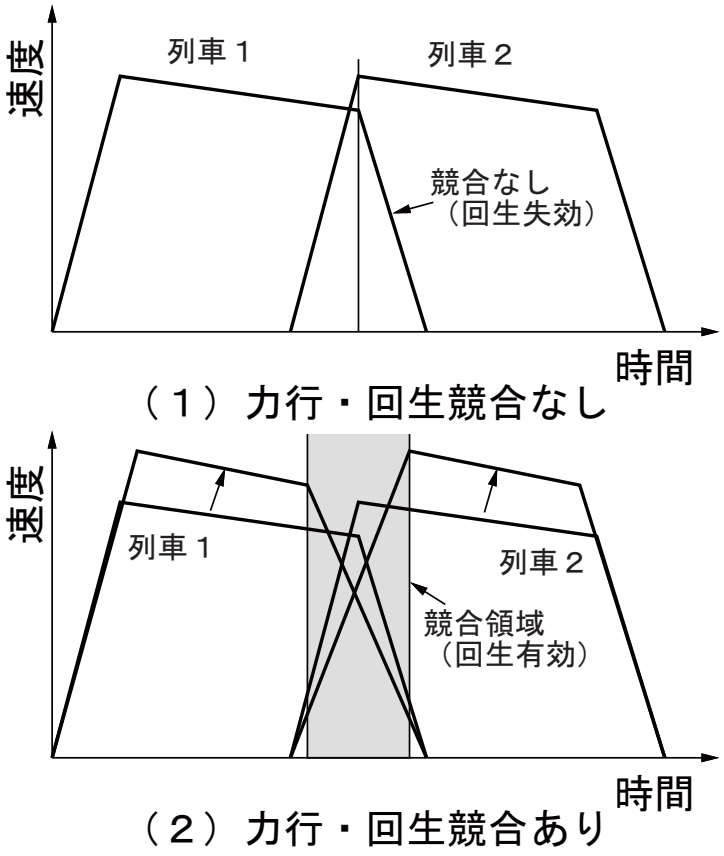


図 13.1: 2列車の走行パターンと力行・回生の競合

（ただし N_T は列車数）であり，x_i および v_i $(i = 1, \ldots N_T)$ について式(13.1)・(13.2)・(13.7)が成立する。また，フルノッチ比ベクトルを

$$\boldsymbol{r} \equiv \{r_0(t), r_1(t), \ldots, r_{N_T}(t)\}^T \tag{13.11}$$

とするとき，r_i について 式(13.6)が成立する。さらに評価関数は

$$E(\boldsymbol{r}) = \int_{t_0}^{t_f} \sum_j V_j(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{r}) I_j(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{r}) \tag{13.12}$$

（ただし $j = 1, \ldots, N_S$, N_S は変電所数，V_j, I_j は変電所 No. j の電圧・電流）と書ける。

この問題と⟨13.1.1⟩の問題とを比較すると，変数や制約条件式が増えただけであり，問題の構造は変化していないことがわかる。従って，⟨13.1.1⟩ と基本的に同一のアルゴリズムで解けるはずである。

⟨13.1.3⟩　列車群最適走行パターン問題の統合システムへの応用

⟨13.1.2⟩で定式化した問題に適当な変数や制約条件をいくつか加えることによって，饋電・列車群制御統合システムにまつわるいろいろな問題が記述できることを，2つの例によって示そう。

⟨13.1.3.1⟩　変電所のピークカット　変電所電流のピークが平均値に比べ高いことから電力機器の利用率が低いことはすでに述べた（⟨4.2⟩, 15ページを参照のこと）が，ピーク電流は抑えれば抑えるほどよいということではなく，機器の定格による制約の範囲内ならピークは高くてもよいということにすれば，⟨13.1.2⟩で定式化した問題において

$$I_j(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{r}) < I_{j,max} \tag{13.13}$$

という制約を加えればよい。制約条件式が変電所数だけ増えたことになるが，数学的な問題の構造はなお変わらず，同一の解法アルゴリズムが使える。

⟨13.1.3.2⟩　変電所の最適電圧制御　変電所がサイリスタコンバータなどのように電圧–電流特性を自由に選べるものなら，列車の位置・状態に応じて最適な電圧を出させることが考えられる[23, 21]。このようなものも列車群最適走行パターン問題に定式化可能である。ここで，変電所を制御する場合，直流饋電システムでは電気車の特性が電車線電圧に依存することから，走行パターンも変化してしまう。このことが無視できないため，変電所の制御に関する問題であっても結果的に走行パターン問題にならざるを得ないのである。

この場合，

$$V_j \equiv V_j(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{r}, \boldsymbol{s}) \tag{13.14}$$

（ただし $\boldsymbol{s} = \{s_1, s_2, \ldots, s_{N_S}\}$ は変電所制御パラメータ）のように変電所電圧 V_j を書き換えればよい。入力変数が変電所数だけ増えたことになるが，数学的な問題の構造はなお変わらず，同一の解法アルゴリズムが使える。

なお，列車側については積極的な制御を行わないのならば，フルノッチ比 $\boldsymbol{r}$ が入力変数ではなく，位置・速度・電圧・時刻などによって定まるものとすればよい。

⟨13.2⟩　解法アルゴリズム

これらの問題は一般的に制約つき2点境界値問題と呼ばれる。この最適化問題を解くアルゴリズムのクラスとして，Sequential Gradient–Restoration Algorithm が知られている。この一種で，最大勾配法の代わりに共役勾配法を使う **SCGRA**[29]を利用して，数値解を求めることを考えた。

⟨13.2.1⟩　不等式拘束の等式制約化

このアルゴリズムをそのまま適用するには，不等式拘束の等式制約化[30]が必要だが，これは次のようにして可能である。以下，列車最適運転パターン問題の場合について述べる。

入力 r に関する不等式拘束条件式 (13.6) は次のように等式制約に変換できる:

$$C \equiv r - \sin r_d = 0 \tag{13.15}$$

ここで，ダミー入力 r_d を入力変数の一つとして新たに加える。

一方，状態 v に関する不等式拘束条件式 (13.7) は一般に取り扱いが困難であるとされている。しかし，スラック変数法[31]を用いると，次のように等式制約に変換できる[48]:

$$\begin{aligned} C_s &\equiv \frac{1}{M}\{T(v,r) - R(v) - G(x)\} \\ &+ v_d s - v\frac{dv_{limit}(x)}{dx} \\ &= 0 \end{aligned} \tag{13.16}$$

ここで，スラック状態変数 v_d およびスラック入力変数 s を状態および入力変数に新たに加える。

⟨13.2.2⟩　SCGRA の概要

SCGRA は，初期条件のみを満たすものの等式拘束・運動方程式・終端条件は必ずしも満たさない初期解を与え，そこから計算をはじめる。等式拘束・運動方程式・終端条件を満たさない場合は Restoration Phase（修正フェーズ…… R フェーズ）を走らせ，これらが満たされたなら最適性条件を満たす解を探索する Conjugate Gradient Phase（共役勾配法フェーズ……CG フェーズ）を走らせる。CG フェーズの結果等式拘束・運動方程式・終端条件に対する違反が再び生じたなら，R フェーズをその都度走らせる。このことを繰り返して行くと，最適解にいつかたどり着く，というのがこのアルゴリズムの概要である。

⟨13.3⟩ 数値解の例

このアルゴリズムによる数値解の例を提示する。列車群最適走行パターン問題と列車最適走行パターン問題とは同じアルゴリズムで解くことができるので，ここではプログラムの確からしさを見る目的も兼ねて，列車最適走行パターン問題の数値解を提示する。

⟨13.3.1⟩ 簡略化列車モデル

ここでは，プログラムのディバッグを兼ねるために，次のような簡略化列車モデルを使用している。ここで，初期条件（t_0, x_0, v_0）はすべてゼロ，終端条件を t_f, x_f とする（$v_f = 0$ である）。速度制限および線路勾配は考慮しないモデルとしてある。

- 状態変数ベクトル

$$\boldsymbol{x} = \{x, v\}^T \tag{13.17}$$

の微分方程式は以下のようにした。

$$\dot{\boldsymbol{x}} = \begin{pmatrix} v \\ T - R \end{pmatrix} \tag{13.18}$$

- 列車の牽引力 T は，フルノッチ比 r $(0 \le r \le 1)$ に対し次元を加速度として次のように決めた。

$$T = r \ [\mathrm{m/s^2}] \tag{13.19}$$

- 走行抵抗 R は次元を加速度として，次のように表した。

$$R = 0.05 + 0.005v + 0.0003v^2 \ [\mathrm{m/s^2}] \tag{13.20}$$

- 初期解は次のように与えた。

$$r = \begin{cases} 0.1, & \text{for } 0 \le t < \dfrac{1}{3}t_f \\ 0, & \text{for } \dfrac{1}{3}t_f \le t < \dfrac{2}{3}t_f \\ -0.1, & \text{for } \dfrac{2}{3}t_f \le t \le t_f \end{cases} \tag{13.21}$$

$$r_d = 0 \tag{13.22}$$

$$x = x_f \times \frac{t}{t_f} \tag{13.23}$$

$$v = \begin{cases} \dfrac{2x_f}{t_f} \cdot \dfrac{3t}{t_f}, & \text{for } 0 \le t < \dfrac{1}{3}t_f \\ \dfrac{2x_f}{t_f}, & \text{for } \dfrac{1}{3}t_f \le t < \dfrac{2}{3}t_f \\ \dfrac{2x_f}{t_f} \cdot \dfrac{3(t_f - t)}{t_f}, & \text{for } \dfrac{2}{3}t_f \le t \le t_f \end{cases} \tag{13.24}$$

- 評価関数は本来パンタ点入力エネルギーとすべきであるが，動輪周効率を η として考慮し次のように定めた。

$$I = \int_{t_0}^{t_f} \eta v T dt \equiv \int_{t_0}^{t_f} L(v, r) dt \tag{13.25}$$

ただし

$$\eta = \begin{cases} 1.25 & \text{for } T \ge \varepsilon \\ 0.8 & \text{for } T < -\varepsilon \end{cases} \tag{13.26}$$

とし，$-\varepsilon \le T < \varepsilon$ の間は T についての多項式とし，$T = 0$ で $\eta = 1$ であり，微係数などが連続になるように関数をつなげてある。ε は適当に定める小さい数である。

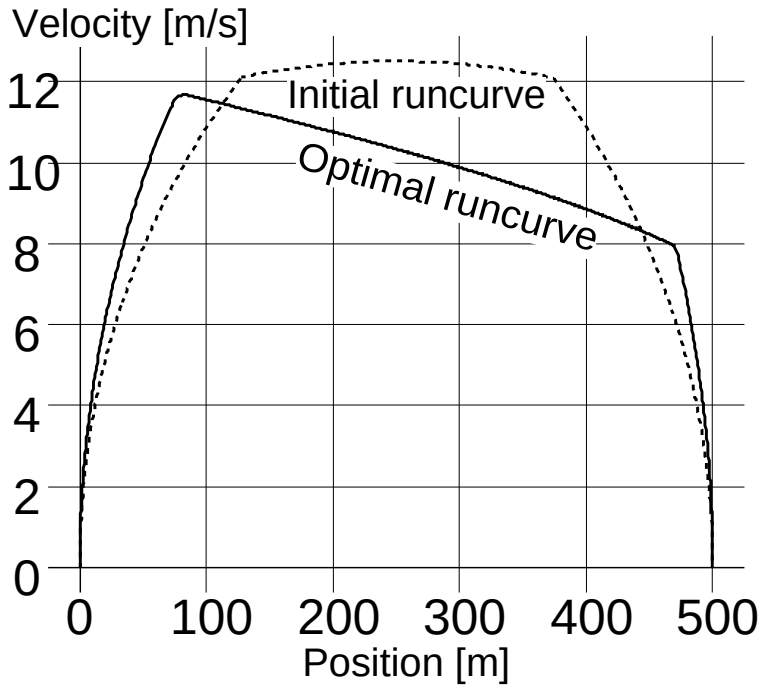


図 13.2: 数値的最適化の結果例 (1) $L = 0.5$ km. ランカーブ.

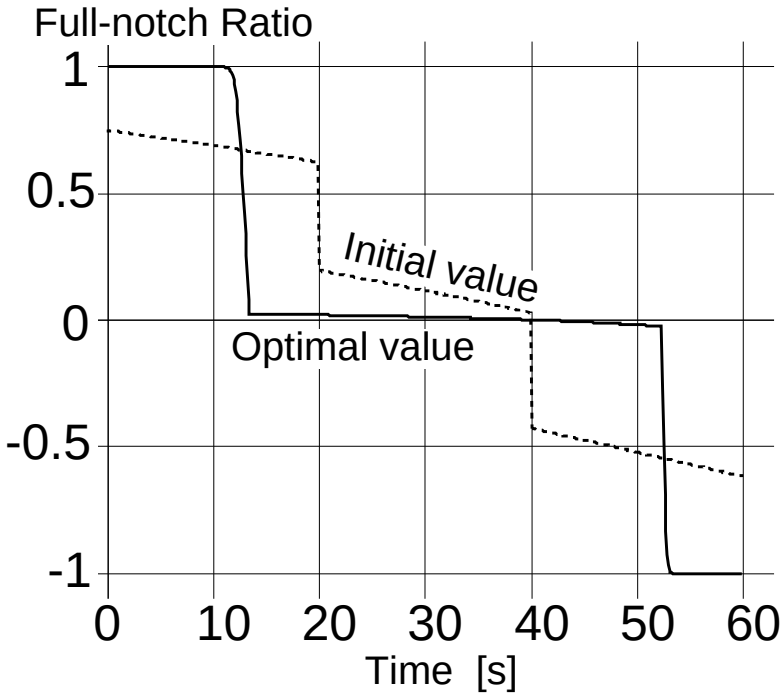


図 13.3: 数値的最適化の結果例 (1) $L = 0.5$ km. フルノッチ比.

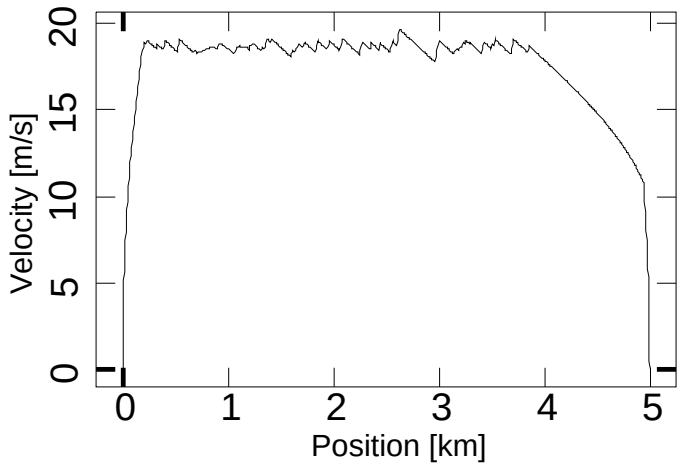


図 13.4: 数値的最適化の結果例 (2) $L = 5$ km. ランカーブ.

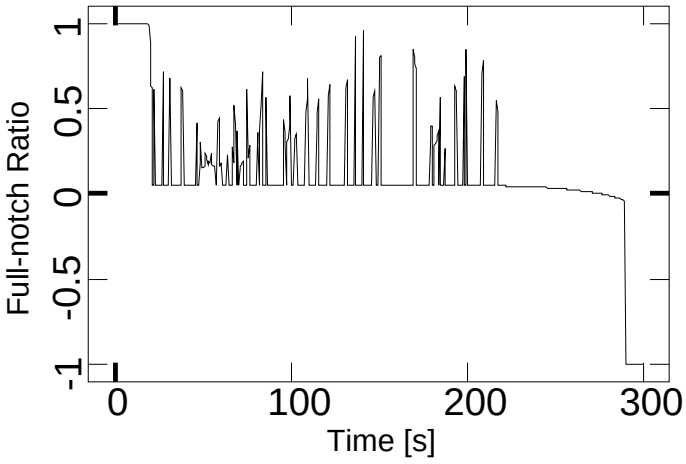


図 13.5: 数値的最適化の結果例 (2) $L = 5$ km. フルノッチ比.

⟨13.3.2⟩　数値解の例

数値解の例をいくつか示す。

図13.2・13.3は駅間距離 L が短い 500m で，駅間走行時分が60秒，$\varepsilon = 0.05$ の場合である。また，図13.4・13.5は駅間距離 L が長い 5km で，駅間走行時分が300秒，$\varepsilon = 0.15$ の場合である。さらに，図13.6・13.7は駅間距離 L が 500m で，駅間走行時分 60秒，$\eta \equiv 1$ とした場合である。

最適走行パターンに関する従来の研究[2, 7]によれば，駅間に下り勾配・速度制限がない場合この問題の解は《加速 → 定速走行 → 惰行　→ ブレーキ》という走行パターンになることが知られている。これを定性的に説明すると以下のようになる[2]。

- 列車の性能が高いほどブレーキ初速が低くできる → 最大加速・最大減速がよい
- 列車の走行抵抗は下に凸の単調増加関数 → 再力行・惰行の繰返しより定速走行が得
- 力行とブレーキは隣接しない方が得 → 定速走行（＝弱力行）とブレーキの間に惰行をはさむ方が得

また，駅間距離が短く，平均速度が高い場合には，定速走行がないパターンが最適となる，ということも記されている。図13.2〜図13.5の結果は，このような定性的な検討とよくあうものといえる。図13.2・

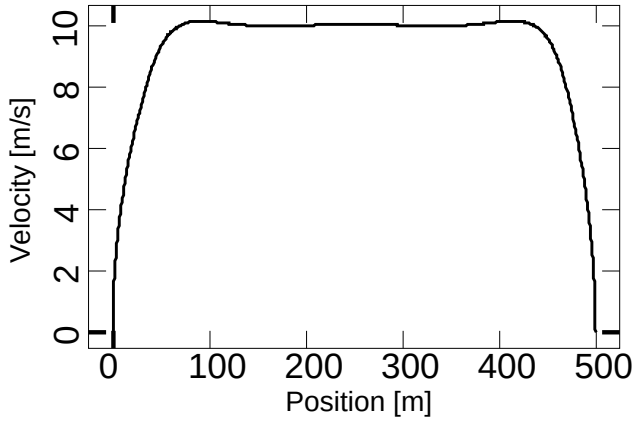


図 13.6: 数値的最適化の結果例 (3) 主回路損失なし．ランカーブ.

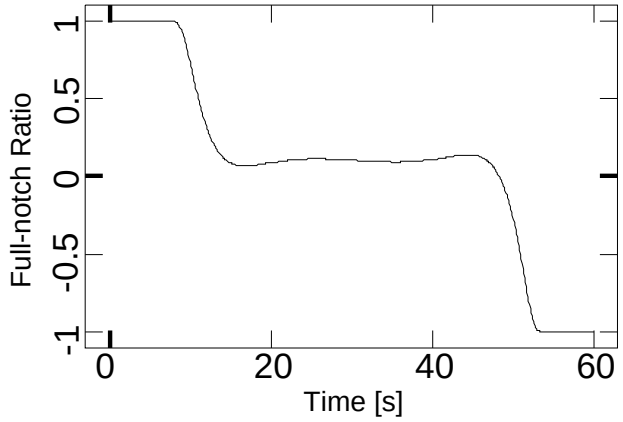


図 13.7: 数値的最適化の結果例 (3) 主回路損失なし．フルノッチ比.

13.3は駅間の短いケースで，力行からほぼ即座に定速走行なしで惰行に移っている。また，図13.4・13.5は駅間の比較的長いケースで，力行からややノコギリ運転状ではあるものの定速走行を経て惰行・ブレーキと状態が遷移している。この定速運転モードがノコギリ運転状になっているのは，式 (13.26) においてフルノッチ比ゼロの近傍では主回路の効率が1に近くなっているため，なるべくフルノッチ比ゼロの近傍で力行・回生を行っていた方が効率がよいが，その範囲内では過剰な速度低下を食い止めることができないため，いっきに力行して速度を元のレベルまで回復させるためと考えられる。

また，図13.6・13.7は，$\eta = 1$ つまり主回路の損失がない場合を計算したものである。この場合，余分な運動エネルギーはすべて回生ブレーキによって無駄なく架線側に返されるのだから，計算結果は走行抵抗による損失が最小となる走行パターンを示している。この場合，ランカーブ（速度—位置曲線）は位置に関して対称となるはずであり，図13.6 はそれにあったパターンとなっていることが読みとれる。

⟨13.4⟩　列車走行パターン問題の数値的最適化にまつわる今後の課題

列車群最適走行パターン問題の定式化を行い，数値的解法アルゴリズムを数値解のサンプルとともに示した。また，サンプルとして示した数値解が定性的に不自然でない性質を持っていることも簡単に考察した。

しかし，大規模な問題になればなるほど，〈13.2.1〉（92ページ）で述べた不等式制約条件の等式制約化は弱点が見えてくる。文献[30]でも言及があるが，数学的にはいささか「強引」の感を免れない。状態変数に対する不等式制約を入力に関する等式制約に変換するスラック変数法[31]はこれよりさらに問題が多いと見られる。

また，定式化で走行時分一定としているのも，状態変数制約を増やしたのと等価であるという見方もでき，数値的最適化にはかなり厳しい条件を与えていると見られる。現実に，図13.4・13.5では定性的考察において「定速走行」となるべき部分に不規則な走行パターンが現れているが，これはこの走行時分一定条件を与えたためではないかと見られる。

列車群の問題に関して数値的最適化を行うには，これらの問題をひとつひとつ解決しなければならない。システムの規模が大きいだけに難しい問題になるだろう。

このようにして，数値的に最適化ができたとき，その結果はひとつの極限を表すものとなる。しかし，その解はそのまま実際のシステムに適用できるわけではない。最適化の際に考慮していない停車時分外乱などが存在するためだ。このような外乱を考慮するために，数値的最適化の結果から制御規則を抽出して定性的に記述し，この規則をベースとしたシステムを構築することが考えられる。最適化結果を得たあとは，この規則をどう抽出し，システムとしてまとめてゆくかが課題となろう。

VI

結論

１４

本論文のまとめと残された課題

⟨14.1⟩　本論文の成果のまとめ

本論文の成果をまとめると次のようになる。

⟨14.1.1⟩　饋電システムと列車群制御システム（第II部）

統合化鉄道電力システム（**４章**にて定義）による改善可能性を次のようにまとめ，体系化した。

- 省エネルギー化 （⟨4.1⟩, 13ページ）

  列車運転パターンの最適化による省エネルギー化 （⟨4.1.1⟩, 13ページ）

  定められた駅間走行時分で走る駅間走行パターンのうちもっとも列車消費エネルギーの少ないものを選ぶ最適化。

  過去に金[2]・保川[7]他による検討がある。

  饋電システム（変電所）の制御による省エネルギー化 （⟨4.1.2⟩, 13ページ）

  変電所送出電圧を最適値に設定，または制御することにより，回生電力を遠くの力行列車負荷まで届かせること，その他による省エネルギー化。

  松富[1]や，筆者自身による検討[37]の他にも多くの検討例がある。本論文でも，**７章**にて改めて検討した。

  列車主回路電力制御による回生失効防止 （⟨4.1.3⟩, 14ページ）

  回生車の多いときに惰行車を力行させることで回生失効防止を図る。

  本論文で取り扱った新しい提案（**９章**）で，効果的であることを論証した。

- 地上電力設備の機器利用率向上 （⟨4.2⟩, 15ページ）

  変電所の制御によるピークカット （⟨4.2.1⟩, 15ページ）

  サイリスタ変電所であれば，変電所 V–I 特性を変更し，ある電流値に達したら定電流制御を行うことにより，その電流値以上の電流が流れないようにコンバータを制御することでピークカット。

  既知技術の応用であり，筆者が卒論[37]にて検証したこともある。

  列車主回路電力制御によるピークカット （⟨4.2.2⟩, 15ページ）

  変電所電流が過大な場合には力行車が電力を一時的に絞る，および惰行車が一時的に電力回生ブレーキで自車両の運動エネルギーを饋電システムに返却する制御を行うことでピークカットを図る。制御によるダイヤ乱れは1秒程度かそれ以内で小さい。

本論文で取り扱った新しい提案（**９章**）で，非常に有効である。ただし，これに類似した研究は他にもいくつか存在する[26][27]。

- 列車群制御における饋電システムの制約の考慮（⟨4.3⟩, 16ページ）

  列車ダイヤ上の余裕の融通による饋電系の救済　（⟨4.3.1⟩, 16ページ）

  ダイヤ余裕の利用による電力システムの救済。例えば，変電所容量が変電所事故などにより局所的に不足している場合には，列車のダイヤ上の余裕時分を再配分して変電所容量が足りない区間では列車速度を抑制する制御による大幅なピークカットができる。

  本論文で取り扱った新しい提案（**１１章**）である。

  列車運行乱れ時の制御の工夫　（⟨4.3.2⟩, 16ページ）

  運行乱れが発生した結果として前方で大幅な減速を余儀なくされる場合，そのことを予め考慮して減速運転をする。運行乱れ時にはすべての列車が回復につとめるよりは，特に遅れている1列車が回復につとめ，他の列車はむしろ遅めに走って列車群としての動きが正常（列車間隔が均等）に戻るようにする，など。

  本論文で取り扱った新しい提案（**１２章**）である。

⟨14.1.2⟩　饋電特性シミュレーションプログラム “RTSS” の開発と評価（第III部）

直流饋電システムの饋電特性シミュレーションプログラム RTSS の開発，およびシミュレーションモデルの評価を行った。

シミュレータに与えるべき条件（⟨5.1.1⟩, 19ページに列挙）のうち，列車性能条件のモデルを実際の列車に近づけるためには，列車動力性能が電車線電圧に依存するモデルとしなければならない。ところが，そのような列車モデルをインプリメントすると，列車ダイヤ条件，なかでも駅間走行時分条件を正確に実現することが困難になる。

この条件を高精度に実現できないシミュレーションモデルを利用すると，変電所送出電圧の最適化による省エネルギー化効果と，送出電圧低下に伴って列車の性能が低下するための走行時分増大による所要エネルギー減効果とが分離できず，議論の信頼性そのものを下げてしまう。ところが，RTSS 開発以前には，駅間走行時分一定条件と列車動力性能の電車線電圧依存性を同時に考慮したのは松富[1]のみで，しかも議論上十分な精度にまでは考慮されていなかった。

これらの問題点に対し，RTSS は

- 列車の加速性能の電圧に応じた変化によっても駅間走行時分が変化しないように，列車が加速をやめる位置を条件に応じて変化させるモデルを開発した。
- 饋電等価回路の計算の収束を改善し，計算不能（収束しない）となる頻度を従来のプログラムより格段に少なくすることに成功した。複雑な形態の饋電系統も，データの変更のみでシミュレートできるように配慮した。

などの特徴を持つ。（⟨5.2⟩, 24ページ）

まず，RTSS のシミュレーション結果と実際の饋電システムの実測値とを比較し，妥当性が検証された。（⟨6.1⟩, 27ページ） また，RTSS と駅間走行時分一定化シミュレーションモデルを持たない従来のシミュレータとを比較し，同モデルによって RTSS が定性的な考察とよく合う結果を出力していることが確かめられた。（⟨6.2⟩, 29ページ） さらに，駅間走行時分と変電所入力エネルギーの関係を RTSS により検討したところ，平均駅間走行時分が約93秒の路線で駅間走行時分を3秒縮めるだけで，1割以上の入力エネルギー変化が起こることがわかった。このことから，駅間走行時分条件を高精度に実現した RTSS のようなシミュレーションプログラムを用いなければ，変電所制御による省エネルギー化の議論は不可能であることが，明らかにされた。（⟨6.3⟩, 31ページ）

このように，RTSS のような信頼できるシミュレーションツールが完成した現在,「効果があるかないか」という議論の段階はもはや終わり，効果の定量的把握をこれらツールを使って行うべき時代になったといえる。

⟨14.1.3⟩ 統合化鉄道電力システムにおける省エネルギー化・設備利用率向上の可能性（第IV部）

⟨14.1.3.1⟩ 変電所 V–I 特性の最適化（**７章**, 35ページ） 路線モデルとして地下通勤線区である A 線を取り上げ，次のような検討を行った。一般的に得られた知見のみを記すが，この路線モデルにおいては具体的な最適値を RTSS によって求めることができた。

- 変電所無負荷時送出電圧の最適化（⟨7.2⟩, 37ページ）
- ダイオード変電所を前提とした場合の回生インバータの導入による省エネルギー効果の検討（⟨7.4⟩, 43ページ）
- 回生インバータなしの場合のサイリスタ変電所の導入による省エネルギー効果の検討（⟨7.3⟩, 43ページ）
- 回生インバータを導入した場合のサイリスタ変電所の導入による省エネルギー効果の検討（⟨7.3⟩, 43ページ）
- ダイオード変電所とサイリスタ変電所を混在させると省エネルギー効果を殺してしまうことがわかった（⟨7.2⟩, 37ページ・⟨7.4⟩, 43ページ）が，これに対処するための制御法の検討（⟨7.5⟩, 45ページ）

一般論として，回生車を導入した場合には導入しない場合より変電所の無負荷時送出電圧を低めに設定するのがよいことが確かめられた。また，回生インバータの設置により列車のパンタ点回生エネルギーが増大し，省エネルギー化が図られることが確かめられた。サイリスタ変電所の導入により無負荷時送出電圧がダイオード変電所より低くでき，回生電力の有効利用が図られる効果も確かめられた。このモデルでの検討においては，サイリスタ変電所の V–I特性に定電圧領域があるため，サイリスタ変電所導入により変電所間の横流が抑制され，省エネルギー化が図られる効果が見られた。

ここで求められた結果は路線モデルに強く依存するため，出来上がったシミュレーションツールを使って今後もシミュレーションを続けることが重要であることを示した（⟨7.7⟩, 48ページ）。

さらに，饋電システムの最適化の前に列車の回生絞り込み特性を最適化しておくことも重要である。（⟨7.6⟩, 47ページ）ここでは，

- 回生絞り込み開始・終了電圧はなるべく高くするほうがよいこと（これ自体は既知の事実）
- 列車の荷重に応じて絞り込み特性を変化させる場合，空車時には不必要な絞り込みをさせないような特性とすべきであること
- 回生絞り込みを行う際，パワーは低いがトルクが大きい低速域を考慮すれば，モータ電流ではなく主回路回生電流を基準として絞り込み特性を与えた方が回生率が高いこと

を指摘した。

⟨14.1.3.2⟩ 変電所送出電圧のリアルタイム制御（**８章**, 50ページ） **７章**（35ページ）での検討結果から，回生車が大部分を占める電気鉄道にあっては饋電電圧は比較的低めの値にするのがよいことがわかった。しかし，電圧低下は列車の性能低下につながるほか，電流の増大により饋電線損失を増すデメリットがある。饋電電圧低下のメリットが出る場面は力行車と回生車が共存する場合のみである。そこで，この場合にのみ電圧を低くする変電所電圧リアルタイム制御の効果を見るための簡易なシミュレーションを行った。

シミュレーション結果から，リアルタイム制御により送出電圧を高く保つことのデメリットである回生失効率の上昇，および回生率の低下を防ぎ，同時に饋電線損失の低減によって変電所総合入力エネルギー

を減少させることができることがわかった。しかし，リアルタイム制御による省エネルギー化効果はあまり大きく期待できないことも明らかにした（⟨8.3.2⟩, 53ページ）。

⟨14.1.3.3⟩　列車主回路電力制御によるピークカット・回生失効防止（９章, 56ページ）　以上の方法は変電所の電圧設定の最適化や電圧制御を行う方法であり，本研究以前に発表された饋電システム最適化の研究もすべてこの方法によっている。これに対し，本論文では列車主回路電力制御による新しい方法を提案し，その効果が大きいことを示した。

まず，変電所電流のピークを抑制するための列車の行動ルールを考察した（⟨9.2⟩, 58ページ）。その結果，

1. 変電所電流が過大ならば，その変電所の近傍にいる速度の高い力行列車は力行電流を絞る。
2. 変電所電流が過大ならば，その変電所の近傍にいる速度の高い惰行列車は回生状態に転移する。

という行動ルールにしたがって列車主回路電力制御を行えば，変電所の負荷が減少することになり，ピークカットができることを示した。この制御により列車の遅れが懸念されるが，列車速度が高い領域であればそれらは少ないレベルに抑制し得ることも示した（⟨9.1.1⟩, 56ページ）。

また，回生失効を防止するための列車の行動ルールについても考察した（⟨9.2⟩, 58ページ）。その結果，

3. 回生車が多ければ，回生車の近傍にいる惰行列車は力行状態に転移する。

という行動ルールに従って列車主回路電力制御を行えば，回生車の近傍に負荷がふえたことになり，回生失効防止ができることを示した。

さらに，列車・地上間通信なしのシステムにおいても，電車線電圧の高低によって「変電所電流が過大である」または「回生車が多い」という状況が判断できることを示した。

JR山手線でのシミュレーションにより，列車・地上間通信のないシステムにおいて，この手法を応用すれば，最大約31%の大幅なピークカットができること，回生失効がほとんどなくせること，を確認した（⟨9.3⟩, 60ページ）。

また，列車・地上間通信を追加することにより，変電所のピーク電流値をほぼ思い通りの値に制御する可能性も示した（⟨9.4⟩, 65ページ）。

⟨14.1.3.4⟩　エネルギーと経済効果（１０章, 70ページ）　回生失効による損失には電力料金の増加とブレーキシュー摩耗費用の増加の2 重のコスト的損失があることを示し，省エネルギー化によるコスト削減効果の金額換算を試みた。また，変電所の建設コストのおおまかな値から，回生インバータ，サイリスタ変電所導入の経済的な合理性を評価した。

ただし，ここでの結果は路線モデルに強く依存するため，議論の結果ではなく方法が重要であることを強調した。また，ここでは電力料金・ブレーキシュー摩耗費用の低減以外考慮していないが，これ以外にもメリットが見い出せるなら，総合的な判断によって回生インバータやサイリスタ変電所などの設備を投入することが考えられることも示した。

⟨14.1.4⟩　統合化鉄道電力システムにおける列車群制御の可能性（第Ⅴ部）

列車が積極的に電力システムの状況を考慮しつつ動くことの可能性の一つとして、列車群制御システムにおいてダイヤの余裕再配分を行う制御により変成器容量の低減が可能であることを示した。主回路電力制御によるピークカット制御と余裕時分再配分制御により，変電所1つがダウンしても正常に近い運行が保てることが，山手線におけるシミュレーションにより明らかにされた（**１１章**, 74ページ）。

朝ラッシュ時に定常的に見られる現象として，乗降人員の多い駅の手前での列車の駅間停止がある。この場合，列車の到着時遅れが同一なら，駅間停止を防止することによってきわめて大きな省エネルギー化が図れることを，シミュレーションにより示した（⟨12.1⟩, 81ページ）。

緩急結合輸送においては，緩急どちらかの列車が運行乱れを起こした場合，追い越し／待避駅前後で緩急相互の干渉が起きて大きな駅間走行時分の増大や駅停車時間の増大が局所的に発生する場合がある。こ

の場合，これらの時間を複数駅間に均等に配分し，駅間走行時分を延伸してやることにより，省エネルギー化の余地があることを，シミュレーションにより示した（⟨12.2⟩, 85ページ）。

最後に，インテリジェント化の極限を見るための最適制御問題への定式化を提案し，数値解の例を示した（**１３章**, 89ページ）。

⟨14.2⟩　今後の課題

残された課題について述べる。

⟨14.2.1⟩　饋電特性シミュレーションプログラム “RTSS” の開発と評価（第III部）

RTSS の現在インプリメントされている機能については，プログラムのさらなる安定化や汎用化を図りながら，使い込んで行くことが求められる。

RTSS の弱点は信号システムのインプリメンテーションがないこと，およびグラフィカルインタフェースが皆無に等しいことである。これらは早急の改善が求められる。逆に，これらを改善できたなら，RTSS の使い勝手は非常に改善されることであろう。

⟨14.2.2⟩　統合化鉄道電力システムにおける省エネルギー化・設備利用率向上の可能性（第IV部）

⟨14.2.2.1⟩　変電所 V–I 特性の最適化（**７章**, 35ページ）　すでに述べたように，変電所V–I特性の最適化の効果を定量的に把握すべきである。そのため，⟨14.2.1⟩で述べたように，これらツールを使い込んで，大量のシミュレーションを行い，定性的な知識を蓄えて行くことが期待される。

⟨14.2.2.2⟩　変電所送出電圧のリアルタイム制御（**８章**, 50ページ）　変電所電圧のリアルタイム制御を実現するためには，主な課題として

(a) 地上側で列車の状態を知る方法の確立

(b) アルゴリズムそのものの検討

が残されている。(a)についてはデータを列車・地上間通信でやりとりするのがよいだろう。通信すべき情報量もわずかである。一方，アルゴリズムの検討はこれよりは困難な課題である。

(1) 計算処理に要する時間
(2) 通信に要する時間
(3) 制御に用いる各種データ（列車位置・速度など）の精度
(4) 送出電圧変化速度の上限
(5) 自律分散的システムの考慮

自律分散的システムを考慮したアルゴリズムも開発されるべきだ。特に，回生インバータ設備をどのように制御したらよいのかについては，いまだ明確な方針は確立されていないため，今後の研究成果が待たれる。

⟨14.2.2.3⟩　エネルギーと経済効果（**１０章**, 70ページ）　RTSS には，ブレーキシュー摩耗費用の評価を行うルーチンがないので，このような観点からの評価を頻繁に行うならばこれを追加する必要があろう。

ただし，回生失効率が式 (5.3)（21ページ）で定義される場合,「回生可能エネルギー」の定義が問題になろう。電圧が低くなると，回生可能エネルギーも低くなるためだ。高速域では，回生失効率0 でも回生ブレーキ力が不足するため，空気ブレーキで補足しなければならない。電圧が低い場合，回生失効率は0 に近くなるが，空気ブレーキで補足すべき領域も増えることになる。この事実の評価をシミュレーションモデルに組み込む必要もあろう。

⟨14.2.3⟩　統合化鉄道電力システムにおける列車群制御の可能性（第Ⅴ部）

この部は全体に提案のみにとどまっており，具体的に制御を自動で行う場合のアルゴリズムの開発が待たれる。

最適制御問題は解をきちんと求めるところまで進んでおらず，今後も計算を進めることが求められる。また，仮に数値的に最適化ができた場合にも，その解がそのまま実際のシステムに適用できるわけではないため，数値的最適化の結果から制御規則を抽出して定性的に記述し，この規則をベースとしたシステムを構築することが考えられる。

謝辞

本研究は，以下に述べる多くの方々のご助言，ご助力がなければ到底成立しなかっただろう。

指導教官の**曽根　悟教授**の導きがあればこそ，筆者は常に進むべき適切な道を見い出し，安心して歩むことができた。常に問題の本質を突く曽根先生のご指摘に学ばせていただいたことは数知れない。また，いくつか大学外での活動の機会も与えていただき，大変貴重な経験を得ることができた。

古関 隆章講師は，1992年に曽根研のメンバになられ，現在研究室は「曽根・古関研」と呼ばれるようになっている。若い古関先生の，まさに現役の研究者らしい鋭いご指摘・ご批判は，研究を進める上で大いに参考になった。研究以外の面でも，よき先輩としていろいろな相談にのっていただいた。

曽根研究室の**笠井 啓一助手**には，筆者が1989年に卒論生としてお世話になるようになってから，一貫してまったく理想的な研究環境を常に整備していただいた。曽根教授秘書の**南　佳子**さん，**田中 靖子**さんにもいろいろお世話になった。また，1994年度卒論生の**植田 直季**くんには，付録の RTSS マニュアルの整備を手伝ってもらった。

電気・電子・電子情報工学科内でもいろいろな先生方のお世話になった。特に**堀　洋一助教授**には，公私にわたりいろいろ貴重なご意見を賜わった。**茅　陽一教授**，**正田 英介教授**には，最適制御問題の解法に関し有益なアドバイスをいただいた。

大学外の多くの方々とのディスカッションからも非常に多くのものを得た。特に，(社) 日本鉄道電気技術協会で1993〜94年度にわたって設置された「最適き電電圧調査研究委員会」におけるディスカッションは，**7章**の議論の中核をなすものである。この委員会では，委員長の曽根教授はもとより**水間　毅**氏（運輸省 交通安全公害研究所 交通安全部 主任研究官，工博），**伊東 利勝**氏（(財) 鉄道総合技術研究所 伊藤研究室 研究員），**田代 維史**氏（(株) 日立製作所 日立研究所 交通システム研究室 主任研究員），**小山 忠雄**氏（帝都高速度交通営団 電気部 電力課 課長補佐）をはじめ，多くの方々に有益なご教示・ご助言をいただいた。ここに関係するすべての方々のお名前を挙げることができないのが非常に残念である。

筆者が参加した学会などでも，いろいろなご意見をいただいた。中でも**杉本　健**氏（(財) 鉄道総合技術研究所 伊藤研究室 研究員）には，数少ない直流饋電システムの研究者として，いろいろな意見交換をさせていただいた。

この他，いちいち名前は記さないが，ほか多くの方々……博士課程修了を忍耐強く待ってくれている両親……尊敬すべき，愛すべき博士課程の同期の学生の面々……フリーの有益なソフトウェアや研究室の計算機システムの管理方法の提供などでお世話になった多くのインターネット・コミュニティの方々……研究室の卒論・修士・受託研究員の後輩諸君……にも，本研究は多くを負うている。

これら，お世話になった方々に，筆者の心からなる感謝をささげたい。

1994年12月20日

高木　亮

参考文献

(1)　本学卒業論文・修士論文・学位論文・研究論文

[1] 松富: 「回生車を含む饋電システムの最適化」, 東京大学（修士論文）, 1984.

[2] 金: 「省エネルギーを考慮した列車運転制御」, 東京大学（学位論文）, 1985.

[3] 清水: 「変電所における列車状態の推定と送り出し電圧制御」, 東京大学（卒業論文）, 1985.

[4] 岩下: 「直流回生車の能力向上のためのき電システム」, 東京大学（卒業論文）, 1986.

[5] 尾原: 「直流電気鉄道のき電システム」, 東京大学（研究論文）, 1985.

[6] 山崎: 「選択停車による通勤ダイヤの改善」, 東京大学（卒業論文）, 1990.

[7] 保川: 「新幹線列車の省エネルギー運転に関する研究」, 東京大学 (学位論文), 1989.

(2)　インテリジェント化に関して

[8] 新村: 「広辞苑」, 第三版, 岩波書店, 1983.

[9] Wolfram: "*Mathematica* — A System for Doing Mathematics by Computer (Second Ed.)" (日本語版: 白水訳), アジソン・ウェスレイ・パブリッシャーズ・ジャパン, 1992.

[10] 曽根: 「鉄道のインテリジェント化と車両」, 電気車の科学, **41**-1, 電気車研究会, 1988.

[11] Kataoka, K. and Komaya, K.: "Computer-aided Railway Scheduling System for High-density Train Operations", *Proc. of 4th International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Mass Transit Systems (COMPRAIL 94)*, Vol. 2, Madrid, Spain, 1994.

[12] Schaefer, H. and Pferdmenges, S.: "An Expert System for Real-time Train Dispatching", *ibid.*

[13] 「電気鉄道のインテリジェント化」, 電気学会技術報告 (Ⅱ部) 第 341 号, 1990.

(3)　饋電特性シミュレータに関して

[14] 松富, 曽根: 「回生車能力の有効利用について」, 日本シミュレーション学会研究発表会 第4回, 1983.

[15] 灘友, 乙黒: 「パソコン版直流電鉄き電システムの動的シミュレーションプログラムの開発」, 平成3年電気学会全国大会, No. 1003, 金沢, 1991.

[16] 杉本: 「VVVF 車投入にともなうき電システムについての考察」, 電気学会 交通・電気鉄道研究会資料, TER-94-29, 1994.

[17] 杉本: 「直流変電所の電圧制御の一考察」, 日本機械学会 第3回交通・物流部門大会併催 鉄道技術連合シンポジウム（J–RAIL 94）, 4206R, 川崎, 1994.

[18] 門馬, 戸次, 高野, 小熊, 川島, 岩崎, 秋山, 藤原: 「鉄道総合シミュレータ New Jumps」, 第30回鉄道におけるサイバネティクス利用国内シンポジウム論文集, 313, 1993.

(4) 回生車を含んだ饋電システムのあり方に関して

[19] 「回生車を含むき電システムの現状とあり方」, 電気学会技術報告(Ⅱ部)第 296 号, 1989.

[20] 「回生車両に対応した直流変電所容量設計法」, 電気学会技術報告(Ⅱ部)第 360 号, 1991.

[21] 曽根, 清水: 「変電所における送り出し電圧制御のための列車情報推定法」, 昭和60年電気学会全国大会, 889, p.1092, 1985.

[22] 阿部, 伊東, 井上: 「変電所回線電流による最低パンタ点電圧の推定法」, 電気学会 交通・電気鉄道研究会資料, TER-92-41, pp. 41-50, 1992.

[23] 曽根, 松富: 「変電所電圧制御による回生車能力有効利用」, 昭和59年電気学会全国大会, 816, pp.1018–1019, 1984.

[24] 本間: 「鉄道システムと分散制御」, 電気学会交通・電気鉄道研究会資料, TER-94-1, pp.1–10, 仙台, 1994.

[25] 小山: 「鉄道の電力需給」, 鉄道と電気技術, **5**-11, 1994.

[26] 杉本, 保川: 「大出力電気機関車出力適正化制御システムに関する研究」, 平成6年電気学会産業応用部門全国大会, 185, pp.773–778, 1994.

[27] 門馬他: 「直流き電系における省電力シミュレーション」, 電気学会交通・電気鉄道研究会資料, TER-91-27, 1991.

(5) 列車の運転パターン，およびその最適化に関して

[28] 「電気車の運転経済と駆動系の仕様」, 電気学会技術報告(Ⅱ部)第 112 号, 1981.

[29] Wu, A. K. and Miele, A.: "Sequential conjugate gradient – restoration algorithm for optimal control problems with non-differential constraints and general boundary conditions, part 1", *Optimal Control Applications & Methods*, **1**, 1980.

[30] 加藤: 「工学的最適制御～非線形へのアプローチ」, 東京大学出版会, 1988.

[31] Jacobson, D. H. and Lele, M. M.: "A transformation technique for optimal control problems with a state variable inequality constraint", *IEEE Trans. Automatic Control*, **AC-14**-5, 1969.

(6) プログラミング言語解説書

[32] Stallman, R. M.: "Using and Porting GNU CC", Free Software Foundation, 1993.

[33] カーニハン, リッチー (石田訳): 「プログラミング言語C 第2版」, 共立出版, 1989.

[34] 門内, 赤堀: 「C++プログラミング」, 日本ソフトバンク出版事業部, 1989.

[35] Dewhurst, Stark (小山監訳): 「C++言語入門」, アスキー出版局, 1990.

[36] Ellis, Stroustrup (足立・小山訳): 「注解 C++リファレンスマニュアル」, アジソン ウェスレイ・トッパン, 1992.

発表一覧

(1) 卒業論文・修士論文

[37] 高木: 「電鉄用直流サイリスタ変電所の最適なV–I特性」, 東京大学工学部電気工学科 (卒業論文), 1990.

[38] 高木: 「直流鉄道電力システムのインテリジェント化」, 東京大学大学院工学系研究科電気工学専攻 (修士論文), 1992.

(2) 学会誌論文

[39] 高木, 曽根: 「列車の主回路電力制御を応用したインテリジェント化直流鉄道電力システム」, 電学論 D, **113**-6, pp.808–816, 1993.

[40] 高木, 曽根: 「直流き電システムの駅間走行時分一定化シミュレーション」, 電学論C（投稿中）.

(3) 国際会議

[41] TAKAGI R. and SONE S.: "The Development of Intelligent Power Systems for Railways", *Proc. of the 3rd International Conference on Computer Aided Design, Manufacture and Operation in the Railway and other Advanced Mass Transit Systems (COMPRAIL 92)*, Vol. 2, pp.39–49, Washington DC, USA, 1992.

[42] TAKAGI R. and SONE S.: "Integration of Power Feeding and Train Dispatching Subsystems to Increase Railway Service Capability", *Proc. of the 4th International Conference on Computer Aided Design, Manufacture and Operation in the Railway and other Advanced Mass Transit Systems (COMPRAIL 94)*, Madrid, Spain, 1994.

[43] TAKAGI R., SONE S. and MIZUMA T.: "On Introduction of Substations with Thyristor Rectifiers in DC Railway Power Feeding Systems", *1995 International Power Electronics Conference (IPEC-Yokohama 95)*, Yokohama, Japan, 1995 (to appear).

(4) その他雑誌

[44] 高木, 曽根: 「列車の主回路電力制御によるき電諸特性の改善」, 電気車の科学, **45**-8, pp.21–24, 電気車研究会, 1992.

(5) 国内・研究会論文

[45] 高木, 曽根: 「インテリジェントな饋電システムの構成における諸問題」, 電気学会交通・電気鉄道研究会資料, TER-90-41, pp.85–94, 東京, 1990.

[46] 高木, 曽根: 「インテリジェンスの導入による電鉄負荷のピークカット」, 電気学会交通・電気鉄道/道路交通合同研究会資料, TER-91-1/RTA-91-1, pp.1–9, 東京, 1991.

[47] 高木, 曽根: 「鉄道電力システムのインテリジェント化(その2)」, 電気学会交通・電気鉄道研究会資料, TER-91-36, pp.27–36, 東京, 1991.

[48] 高木, 曽根: 「列車群制御のための省エネルギ運転パターンに関する研究」, 電気学会交通・電気鉄道研究会資料, TER-92-43, pp.61–69, 名古屋, 1992.

[49] 高木, 曽根: 「非線形最適化手法を用いた列車の省エネルギー運転パターンの検討」, 電気学会システム・制御研究会資料, SC-93-7, pp.21–29, 福岡, 1993.

[50] 高木, 曽根: 「鉄道電力システムと列車群制御システムとの統合化に関する基礎検討」, 電気学会交通・電気鉄道研究会資料, TER-93-36, pp.41–49, 大阪, 1993.

[51] 高木, 曽根: 「シミュレーションによるサイリスタ変電所導入効果の検討」, 電気学会交通・電気鉄道研究会資料, TER-94-34, pp.19–28, 東京, 1994.

(6) 国内・大会論文

[52] 高木, 曽根: 「電鉄用直流サイリスタ変電所の最適なV–I特性」, 平成2年電気学会全国大会, 934, p.(8-179), 東京, 1990.

[53] 高木, 曽根: 「駅間走行時分補正機能付き直流饋電システムシミュレーションプログラム」, 平成2年電気学会産業応用部門全国大会, 14, pp.65–68, 大阪, 1990.

[54] 高木, 曽根: 「電気車の主回路電力制御による電鉄負荷のピークカット」, 平成3年電気学会全国大会, 1002, pp.(8-170)–(8-171), 金沢, 1991.

[55] 高木, 曽根: 「鉄道電力システムのインテリジェント化」, 平成3年電気学会産業応用部門全国大会, 13, pp.70–73, 札幌, 1991.

[56] 高木, 曽根: 「鉄道電力システムのインテリジェント化(その3)」, 平成4年電気学会全国大会, 962, pp.(8-167)–(8-168), 千葉, 1992.

[57] 高木, 曽根: 「列車の主回路電力制御を応用したインテリジェント化直流鉄道電力システム」, 平成4年電気学会産業応用部門全国大会, 8, pp.35–40, 名古屋, 1992.

[58] 高木, 曽根: 「列車の省エネルギ運転パターン問題への非線形最適化手法の適用」, 平成5年電気学会全国大会, 1079, pp.(8-217)–(8-218), 熊本, 1993.

[59] 藤原, 曽根, 高木: 「先行列車出発時刻の予測に基づく短時隔運転の実現手法」, 平成5年電気学会全国大会, 737, p.(6-185), 熊本, 1993.

[60] 高木, 曽根: 「直流鉄道電力システムシミュレーションプログラム RTSS」, 日本シミュレーション学会 第12回 シミュレーション・テクノロジー・コンファレンス, 4-3, pp.121–124, 東京, 1993.

[61] 曽根, 高木: 「省エネルギー運転の研究」, 平成5年電気学会産業応用部門全国大会シンポジウム, S5-4, pp.(S-136)–(S-139), 東京, 1993.

[62] 高木, 曽根: 「鉄道電力システムと列車群制御システムとの統合の可能性」, 平成5年電気学会産業応用部門全国大会, 116, pp.491–494, 東京, 1993.

[63] 高木, 曽根: 「饋電系と列車群制御の統合システム化の基礎検討 — 停車時分外乱の考慮 —」, 平成6年電気学会全国大会, 1081, pp.(8-166)–(8-167), 東京, 1994.

[64] 高木, 曽根: 「列車群の最適走行パターン問題の数値解」, 平成6年電気学会産業応用部門全国大会, 1994.

[65] 高木, 曽根: 「列車群制御とき電系の統合化の可能性」, 日本機械学会 第3回交通・物流部門大会併催鉄道技術連合シンポジウム（J–RAIL 94）, 4218R, pp.395–398, 川崎, 1994.

(7) 学内：大学院論文輪講資料

[66] 高木: 「鉄道における情報の処理と利用」, 大学院論文輪講資料 (M1), 1990.

[67] 高木: 「鉄道電力システムのインテリジェント化」, 大学院論文輪講資料 (M2), 1991.

[68] 高木: 「電気自動車」, 大学院論文輪講資料 (D1), 1992.

[69] 高木: 「鉄道電力システムと列車群制御システムとの統合の可能性」, 大学院論文輪講資料 (D2), 1993.

[70] 高木: 「最近の鉄道高速化の技術〜 12 年前から今までの間に何があったのか？ 〜」, 大学院論文輪講資料 (D3), 1994.

(8) 筆者が参加した学外委員会・検討会などの報告書

[71] 「平成 3 年度 常磐新線建設に伴う地磁気擾乱に関する対策設備の調査検討報告書」, (社) 日本鉄道電気技術協会, 1992.
（筆者の執筆担当部分：7.3.1 節 (pp.85–91)）

[72] 「平成 4 年度 常磐新線建設に伴う地磁気擾乱に関する対策設備の調査検討報告書」, (社) 日本鉄道電気技術協会, 1993.
（筆者の執筆担当部分：4.1 節 (pp.34–36), 5.1.1 節 (2) イ (イ)(pp.58–65)）

[73] 「地下通勤線区における最適き電電圧設定に関する調査・研究報告書 (中間報告)」, (社) 日本鉄道電気技術協会, 1994.
（筆者の執筆担当部分：5.2.2 節 (pp.51–55), 付属資料 5(pp. 付 31–付 35)）

[74] 「地下通勤線区における最適き電電圧の設定に関する調査・研究報告書」, (社) 日本鉄道電気技術協会, 1994.
（筆者の執筆担当部分：5.2.2 節 (pp.45–49), 5.3.4 節 (2)(pp.56–57), 6 章 (6.2.2 節除く, pp.58–62, pp.69–94), 9 章 (pp.117–129)）

[75] 「常磐新線に導入する車両に関する調査報告書」, (社) 日本鉄道車輌工業会, 1994.
（筆者の執筆担当部分なし）

(9) 本研究に関連して受けた補助金

[76] 「乗客からみた魅力向上のための鉄道の統合インテリジェントシステム化」, (財) 安藤記念奨学財団 120 万円 (1994 年度).

概念索引

索引中，ページ数が **11** などと太字であるものは，そこが定義などのメインの説明になっていることを示す。同様に *12* などとイタリック体であるものは，それに関する説明ではなくそれが使われたり応用されたりしているページを示す。

か

が

き

く

さ

し

ぶ

へ

む

も

よ

り

れ

記号索引

索引中，ページ数が **11** などと太字であるものは，そこが定義などのメインの説明になっていることを示す。同様に *12* などとイタリック体であるものは，それに関する説明ではなくそれが使われたり応用されたりしているページを示す。

C

D

E

G

I

M

N

R

S

T

V

X

Y

付録 I

RTSS マニュアル・序論

A

はじめに

やや散文的な記述を許していただこうと思う。

1992年11月に曽根研ワークステーションがディスクトラブルに見舞われたときのことが忘れられない。筆者は当時システム管理者だったから，貴重な1週間をシステムのリストアに費やさざるを得なかった。当時筆者はこのプログラム RTSS（当時はまだこの名前で呼ばれてはいなかったが）を用いたシミュレーションの仕事をしていたが，当然その仕事も滞る。その仕事の関係でお世話になっていた鉄道総研の**田中裕**さんにそのいきさつをお話ししたところ,「我々が master で，ワークステーションが slave なはずなのだけれど，実際には slave に支配されてしまうことが多いですね」といったことをおっしゃった。それに対して,「シミュレーションだけでなく，文書の処理も TEX ですからワークステーション任せなのです」といったら,「それじゃあ，高木さんの全存在じゃないですか」。

ワークステーションに依存しなければならなかった理由は明確であった。当時曽根研にあったパソコンは V30 CPU という貧弱なものが主流で，メモリ空間が 640kBytes しかなかった。このプログラムが行っているような饋電等価回路演算は，大きな行列の演算を必要としたが，それには能力不足が明確だった。これらの制約から当初メインフレームで行われていたプログラム開発が，ワークステーション導入に伴いこちらに移行したのはきわめて自然な成行きだったと思う。そして，ワークステーションで計算を行うのなら，結果の処理や文書の管理などもそこで行うのがやりやすいということになった。

結局，RTSS が走りやすい環境に，他のすべてを合わせていった，というのが実感だ。そういう意味で，もしワークステーションが筆者の研究室における全存在なのであれば，その本当の意味は「RTSS が全存在」ということなのである。

じっさい，RTSS は曽根研究室における筆者の全存在と呼ぶにふさわしい。卒業論文[37]・修士論文[38]はいずれもこのプログラムによって出された結果を用いて書かれている。筆者のこの何年間だかの研究のすべての成果やノウハウがここに凝縮しているというのは驚くべきことだ。逆にいうと，このプログラム程度のものが全存在ということは，筆者の力なんてちっぽけなものだ，ということをも痛感させられる。あるいは，人間一人がなしうる貢献というのは，この程度のものなのかもしれない。

いま，せめて願うことは，このコードがいろいろな場所で生き，活用され続けることだ。そのために，このマニュアルがすこしでも役に立てばよいと考えている。

⟨A.1⟩　沿革

このプログラムは，標準的な直流饋電システムのシミュレーションを汎用的に行えるシミュレーションプログラムとして，東京大学工学部電気工学科曽根研究室において，**曽根　悟教授**のご指導のもと筆者が開発・維持しているものである。筆者が卒論生として曽根研にお世話になるようになった1989 年から開

発を始め，現在に至るまで継続的に改良が続けられている。

⟨A.1.1⟩ ver.1.0 — FORTRAN プログラム

NEC の PC-9801 が日本で手に入る唯一の「マトモなパソコン」だった時代から一転して，IBM PC 系の殴り込みなどから最近のパソコンの性能向上は目を見張るものがあるが，このプログラムを開発した当時はパソコンで饋電特性の計算をするためには（主に MS–DOS の制約から）相当な工夫をしなければならないことが明白だった。このため，開発当初の1989年10月より1990年12月にかけては，このプログラムは東京大学工学部オンライン計算機センターの M880 計算機上で FORTRAN 77 言語によって記述されたプログラムとして動いていた。なお，行列演算のサブルーチンとしては数値演算ライブラリ MSL2 を用いていた。この FORTRAN プログラムは，当研究室における筆者の卒業研究[37, 52]のためのツールとして開発され，卒業研究に引続き筆者の修士課程での研究の初期の段階においても活用された[53][45]。当時はその名はなかったものの，このプログラムが RTSS ver.1.0 と呼ぶべきものだと考えている。

このプログラム（当時は `rt` = Ryo Takagi が作った `kiden` プログラムということで `kidenrt` などと呼んでいた）の基本部分には，過去に曽根研究室で開発されたさまざまなシミュレーションプログラムの経験が反映されている。しかし，**橋本 樹明博士**（現・文部省宇宙科学研究所助手，当時曽根研究室の博士課程の学生だった）をはじめとする研究室の諸先輩から，松富[1]などが残しているプログラムにはバグがあり,「そのコードを見てバグを発見して直すよりは自分で書いた方が早いだろう」と勧められたため，従来のプログラムのコードは全く参照せずに自分ですべての部分を書き下ろした。

この ver.1.0 の段階で導入された新しい機能の最たるものが，駅間走行時分一定化シミュレーション技術のインプリメンテーションである。このシミュレーションプログラムが他に例を見ないものとして評価されるゆえんでもある。この駅間走行時分一定化のソースコードは，現在のプログラムにもかなりの部分がほぼそのままの形で反映されており，RTSS の「いちばん古い部分」でもある。この他の部分は，例えば等価回路演算のアルゴリズムには松富論文[1]とか電気学会技術報告[19]などで紹介されるコンベンショナルなものが用いられている。ただし，等価回路計算に時間がかかることを見越し，回路の演算頻度を列車運動シミュレーションの4 分の1 に減らす手法を採り入れ，演算時間の短縮化をはかっている。

⟨A.1.2⟩ ver.2.0 の開発

1990年度はじめに当研究室にもワークステーションが導入され，本プログラムも計算に課金のかかる M880 計算機から移植することとなった。当初は C 言語[33]での書換えを考えていたが，**入内島 健博士**（現・(株) 東芝）および**八杉 昌宏博士**（現・東大理学部情報科学科）という二人の先輩から強力に C++ を勧められたため，最終的には C++言語[32, 34, 35, 36]へと移植することになった。

お二人のお勧めの中には，自動的に FORTRAN 言語を C 言語に「翻訳」する `f2c` なるコマンドの利用もあった。しかし，こちらは「人間に可読なコード」を生成するコマンドではあり得ず，むしろ C コンパイラしかない環境のもとで FORTRAN を利用可能にするためのツールと考えるべきだったようだ。従って，移植はすべて人間の手による翻訳で行った。当然ながら C++ の特徴である「クラス」機能を利用するためにデータ構造などは大幅な変更を行っている。

1991年5月には C++ 言語によるプログラムとして全面的な書換えが完了し，ワークステーション上で動くユーティリティとして動き始めた。その後，新しいデータ・関数・機能の付加，修士論文[38]のための研究の進捗，常磐新線でのシミュレーション・プロジェクトへの本シミュレータの適用[71, 72]などに伴い，さまざまな改造を実施した。

ver.2.0 は，処理系の変更と同時に相当大がかりな書換えを行ったため，ver.1.0 のコードが受け継がれている部分の方がはるかに少ないのだが，例外的に駅間走行時分一定化の部分だけは ver.1.0 のコードがほぼそのままのかたちで受け継がれている。ただし，ver.1.0 では別に用意した近似計算ルーチンで計算していたため誤差が大きかった。これを，通常の列車運動の計算に用いている関数による計算ルーチンを加え，精度をあげた。また，データ形式の大幅な変更により，さまざまな形態の路線のシミュレーション

に応用できる汎用性を備えることができた。速度制限を取り扱うこともできるようになった。また，等価回路演算の高速化のため，変電所および列車の電気的特性を媒介変数表示するという tricky な方法を用いて，Newton-Raphson 法を応用した計算スピードアップをはかった。このかわりに回路計算の頻度を下げるのはやめてしまった。

この他，列車の主回路電力制御によるピークカット[38][39]のシミュレーションのため，フルノッチ比なる概念を導入した。この概念の導入は，研究上のメリットもさることながら，さまざまなコードをわかりやすく記述できるというプログラム上のメリットも同時にもたらした。

⟨A.1.3⟩ ver.2.1 および ver.2.2 の開発

その後，筆者はそのまま博士課程に進学したので，RTSS もさらに開発が続けられている。

まず，1993 年から (社) 日本鉄道電気技術協会に委員会が設けられ，そこで RTSS を再びシミュレーション[73][74] に用いることになった。

そのためにかなり大規模な改良が行われた。RTSS のコードのかなりの部分は実は構造体型データの配列を管理する部分になっているが，構造体の種類がいろいろであるためその都度同じようなコードが書かれていた。ver.2.1 ではこれらの配列コードの整理が大規模に行われ，ほとんどの配列が `template` なる機能[36]を用いて書き直された。この `template` 機能は 1990 年になって ANSI 標準に加えられた新しい機能であり，`gcc` 2.3.X あたりでは安定して動いていなかった。最近の `gcc`（2.6.0 を現段階では利用中）ではほぼバグもとれて安定しており，安心して使えるようになった。このテンプレート機能によって減ったコードは全体で 200〜300 行くらいに及ぶものと思うが，機能追加のためにコードはそれ以上に増殖してしまった。最近，RTSS のソースファイルの総行数は 12000 行を越えているようだ。

このプロジェクトにおいては，実測定とシミュレーションとの整合性が問題になっており，実測定データとシミュレーション結果との比較を行っている。この比較には RTSS の他にメーカのシミュレーションプログラムも参加した。幸いにしてどの比較でも悪くない一致をみたといえる。

このプロジェクトに参加したため，RTSS には実際のシステムに合わせるための「さまざまな，細かい改良」が目だっている。シミュレーションの命といえる等価回路演算ルーチンは，このバージョンで大きなバグがとれ，安定性も飛躍的に向上している。

シミュレーションプログラムの結果と実測値を，あるいはシミュレーションプログラムの結果同士を突き合わせるというのは，シミュレーションをする立場からすれば大変な仕事だ。このプロジェクトでお会いしたメーカ側の方々も大変苦労されたようで，メーカ側の研究員（女性）が「どんどん歳をとってしまう感じがしますね」といっていたのが思い出される。同業者としてそういう神経のすり減るような気持ちはよくわかるが，これがシミュレーションという研究手法の一つの宿命なのかも知れない。

1994 年 8 月まで続いたこのプロジェクトでは，さらに進んで変電所の送出電圧をリアルタイム制御する場合の検討も行った。それにも RTSS を利用することになったのだが，これは運輸省・交通安全公害研究所の主任研究官，**水間　毅博士**がおっしゃったひとことによって筆者の仕事となったものである。十分なアルゴリズム上の検討を行うことはできなかったが，いちおうデモンストレーションのためには十分と思われる結果が出た。そこで，報告書にリアルタイム制御のための 1 章を設けて記述を行うことができた。

⟨A.1.4⟩ それ以降の開発，および総合シミュレータ化

筆者の博士論文のプロジェクトでは最適化シミュレーションルーチンを組み込んだ RTSS ver.3 の開発にも着手した。ところが，これがなかなかに険しい道であった。まず，最適化手法のためのプログラミングに手間取った。最適化手法としては Sequential Gradient-Restoration Algorithm の一種に分類される SCGRA[29]を用いているが，すでに完成しているものから SCGRA に関係するコードだけを拾い出してみると，すでに 2000 行を越えている。しかも，問題の特殊性（パラメータなし，終端条件固定など）を最大限利用した結果としてのコードなので，先行きが思いやられる。

そのような大規模なプログラムであることに加え，関係する変数が非線形なだけでなく，微係数まで含めて不連続であるようだと解けないらしく,「丸め」る操作が必要らしい。特に，フルノッチ比と電流（評価関数……エネルギーに密接に関係する）との関係は特に考慮しておかなければならない点の一つだということもわかっている。

一方，数値的最適化を行わないプログラムについては，順次いろいろな機能の追加が進み，現在 RTSS の最新バージョン番号は 2.4.0 となっているはずである。

なお，1994年度から2年間,「鉄道総合シミュレータ」とかいう題目で，シミュレーションプログラムそのものを作る研究に科学研究費による助成が出た。この関係で，将来的に総合シミュレータとして必要なダイヤ評価などのコードも取り込んでゆく可能性がある。

⟨A.1.5⟩　RTSS というプログラム名の由来

RTSS とは

Railway Total System Simulator

の略であるが,

Ryo TAKAGI and Satoru SONE

の略，すなわちコードを書いた**高木　亮**と指導教官の**曽根　悟教授**の頭文字の略，という2つの意味を持たせている。この名前が初めて世に出たのは文献 [60] でだった。その後かなりいろいろな場所で，この名前でプログラムを紹介している。

ところで，英語名のなかにある Total という言葉には，このプログラムの目指す方向が示されている。つまり鉄道システム全体のシミュレーションに使える汎用プログラムを目指そう，という意思表示でもある。

⟨A.2⟩　C++言語

本プログラムは，いわゆる「オブジェクト指向言語」の一つである C++を用いて記述されている。

C++言語は C 言語にオブジェクト指向プログラミングのための仕様を追加した，C 言語のスーパーセットである。オブジェクト指向プログラミングの狙いは，大規模なプログラムを「楽で間違いを少なく」開発できる，というのが基本的な狙いだろう。しかし,「楽で間違いが少ない」という説は少々疑いたくなることも多い。オブジェクトをきちんと設計できなければ，かえってオブジェクトという存在が障害になることも多い。オブジェクトの設計いかんにかかってくるから，新しいルーチンを書こうとするとオブジェクトの設計で時間の大半が過ぎてしまうこともしばしばある。

C++言語自体にも，若い言語ゆえの問題点が少なくはない。例えば，C++の言語仕様は急速に標準化が進んでいるとはいえまだ十分ではない。なにより，現在の問題点は C++ の言語仕様が年を追うごとに拡張されていることだ。そのため，C++ の本は分厚くて読みにくいものになってしまっている。幸い，コンパイルに利用している GNU C++[32]が，文献 [36] で紹介されている標準化案にある言語仕様のほとんどをサポートするようになっているようなので，UNIX システムで利用する限りにおいては心配は少ない。

UNIX では世界的に標準言語の座を占めている C 言語のスーパーセットとして考えられている C++言語は，オブジェクト指向言語の中ではもっとも明るい将来を持っている言語でもある。オブジェクト指向言語としては，C 言語との互換性を考えたため不完全な部分が多いともいわれるが，そのような背景から比較的長い将来にわたって比較的ポピュラーな言語であってくれる可能性が大きい。

ただ，C や C++を FORTRAN に比べたとき，科学技術計算に使用するに当たっての弱点の一つが「ライブラリ資源の蓄積がまだ小さいこと」であろう。C 言語に比べると，C++言語では複素数クラスがすでにライブラリの形で与えられているし，高水準入出力制御法としての「ストリーマ」なども同様に与えら

れている。しかし，このプログラムではこの「ストリーマ」機能は使わなかった。もったいないとは思ったが，C の標準入出力関数で書くのがどうもやりやすい感じに思えたのだ。じっさいデータファイルの出力などの局面では，いちいちストリーマクラスを使う必然性は感じられない。それに，例えば行列演算のようなものにしても，FORTRANでは既存のものが使用できたのに，C++では自作しなければならなかった。

⟨A.3⟩ 仕様

主な仕様を示す。

処理系 C++ 言語にて記述。 GNU C++ Compiler (g++), GNU C++ Class Library (libg++) を使用してコンパイル。現在は GNU C++ 2.6.0 を利用している。SunOS 4.1.3 の走る Sun SPARC Station 10 および SPARC Station 5, 2, ELC, SLC で計算を行なう。なお，g++ の走るマシンであれば，ソースコードを移植すれば走るはずであるし，SPARC チップを CPU に用いたワークステーションならばバイナリの実行形式ファイルを移植するだけで動作するだろう。

列車数，特性 メモリの許容範囲内でいくつでも可能。特性はインバータ制御電気車および4象限チョッパ制御電気車のみに対応している。現在のところ，全列車の特性は同一である必要がある。

駅数，列車ダイヤ 駅数はメモリの許容範囲内でいくつでも可能である。次駅までの線路条件，走行時分，停車時分などに関するデータはすべて「次駅データ」にまとめられる。これをダイヤ１周期分だけ集めたものをダイヤパターンデータと称し，メモリの許容範囲内でいくつでもこのダイヤパターンを持つことが可能。それぞれのダイヤパターンにそって列車は等間隔配置される。

変電所数，特性 メモリの許容範囲内でいくつでも可能。折れ線V–I特性を持つ変電所ならばどんな特性のものもシミュレートできる。変電所毎に特性を与えることが可能。変電所母線から饋電線接続点までのインピーダンスを与えることができる。

饋電線数，形態，饋電定数 メモリの許容範囲内でいくつでも可能。形態は,「線分（端点あり）」「円（端点なし）」の2種類が選べる。これらを組み合わせることによって，あらゆる形態の路線がシミュレートできる。饋電定数は全線一定。原則として，全線並列饋電，変電所 Busbar で上下タイを行なうことを前提としているが，データの与え方で相当いろいろなことができるはずである。

距離・位置の表現法 饋電線ごとに距離原点をセットできる。例えば山手線と京浜東北線を同時にシミュレートするような場合は，山手線の2本の饋電線は「円形」で東京が原点，京浜東北線の2本は「線分」で大宮原点，などと表現が可能。変電所位置は饋電線接続点の位置を饋電線ごとに与える。従って，上の例に関していえば東京にある同じ変電所でも山手線と京浜東北線では位置が違う数字になる。列車は「どの饋電線の下を走るか」を決めて，その饋電線の座標で位置を表現。

付録 II

RTSS マニュアル・プログラムの概要

B

シミュレーションの方法の概要

直流饋電システムのシミュレーションプログラム（饋電特性シミュレータ）は，従来の曽根研における研究でも作成されている[1, 2, 5, 14]。曽根研以外でも，新たにシミュレータを開発したという報告がある[15, 18]。本研究において開発したシミュレーションプログラムもこれら既存のシミュレータも，ごく基本的でおおまかな考え方および構造については共通である。

⟨B.1⟩　プログラムのおおまかな構造

直流饋電回路に流れる電流を求めるためには[19]，

1. 所定のダイヤに従って列車群の配置や，速度・状態を求める。
2. 饋電用変電所・電車線路からなる饋電系統の対応する位置に，上記の電気車を配置して等価回路を作成する。
3. この等価回路を解いて，電圧・電流分布および各種電力を求める。

という手順をとればよい。これは，ある瞬間における饋電回路の状態を解析したものである。

しかし，一般には饋電特性とは「饋電系統内に点存する多数の電気車が連続走行するときの饋電系統の電圧・電流特性，電力特性，エネルギー特性，電気車の走行特性などの諸特性を総称したもの」である。特にエネルギーは電力を時間積分しないと求まらない。従って，饋電特性シミュレーションプログラムは，上記の手順1〜3を連続的に，Δt 時間間隔ごとに，繰り返し行う必要がある。

そこで，饋電特性シミュレーションプログラムは必然的に，図5.1（22ページ）のような構造を持つことになる。つまり，プログラムは大きく分けて「列車を動かす部分」（列車運動シミュレーション部）と「饋電等価回路演算」部分とにわかれるのである。

⟨B.2⟩　列車運動シミュレーション部と饋電等価回路演算部との関係

これらの二つの部分は，以上に述べたようにかなり独立した機能を持っている。しかし一つのシミュレーションプログラムの中で動作している場合，完全に独立ではなく，次のように相互にデータをやりとりすることになる。

まず，饋電等価回路演算部は当然ながら列車の位置・速度・状態を列車運動シミュレーション部から受け取らなければ計算ができないはずである。このうち，列車の状態については後に詳しく説明するが，ここでは「力行（加速のこと）中」「惰行中」「ブレーキ中」などといったことがらを列車の状態と呼んでいると理解していただければ十分であろう。

一方，列車の性能は電車線電圧に強く依存することが知られている。このことから，列車運動シミュレーション部も饋電等価回路演算部から列車のパンタ点電圧のフィードバックを受けないと正しい計算ができないことがわかる。このことは，饋電回路の条件を変えると列車の速度が変化することがあることを意味する。

C

クラス・オブジェクトの大まかな構成

この章ではクラス・オブジェクトの概要を述べる。

⟨C.1⟩ 配列クラステンプレート table その他

C++ のテンプレート機能（パラメトライズド・クラス）を使った配列クラスである。なにぶん新しい言語のため，このテンプレート機能のきちんとした説明が載っている参考書がなかなかなく，苦労した（もっとも，コンパイレーションに使用している GNU C++ コンパイラも，バージョンが 2.3.X だったころはテンプレート機能がうまく働いていなかったようだから，よかったのかもしれない）。

パラメトライズド・クラス（クラス・テンプレート）は，型宣言の際に「引数（パラメータ）」としてクラスの型名とか整数のデータなど，いくつかのパラメータを渡すことのできる形式である。実際にコードを書いてみると，int 型の配列であっても double 型の配列であってもほとんど同じ操作になることが多いのだが，肝心の「型」の名前が違うというだけで異なるコードを書かなければならない。このことを奇麗に解決する手法として，1988 年に初めて提案され，1990 年 7 月に ANSI C++ 標準化委員会に採択された新しい機能である[36]。

RTSS で利用するテンプレートクラスとしては，table，reftable，zerotable，zeroreftable の 4 つが定義されている。細かな違いがあるものの基本はどれも同一であって，例えば，構造体 foo の配列を作りたい場合

```
table<foo> x;
```

のように宣言すればそれでよい。例えば，この配列の中身を参照する場合，通常の C 言語でポインタに対してするように x[3] のようにブラケット演算子が使える。ここで，配列として実際に確保した範囲を越える領域をアクセスしようとすると，通常の C なら Segmentation Fault を起こすところだが，それも回避され，エラーとして表示され，プログラムの実行が止まるから安全である。このテンプレートは，以下に述べるさまざまな配列を含むクラスの基底クラスとして用いられている。

⟨C.2⟩ 行列演算クラス matrix

行列演算を行うクラスである。自作であるが、逆行列、行列の 4 分割など必要な機能を一応備えている。

⟨C.3⟩ データリードクラス readdata

データリードを行うためのクラスである。# から行末までをコメント行として扱うルーチン、実数や複素数を読み出すルーチンなど、基本的な関数がいくつか含まれている。

RTSS は C++ のせっかくの「ストリーム・クラス」を使っていないため、入出力部分はいささかエレガントでない。そのうえ、コマンド名の検索に異常な行数を使ってしまっている。このあたりの改善は，今後の課題だろう。

⟨C.4⟩ 変電所クラス elecchar

変電所の特性を表すためのオブジェクトが elecchar である。電気的な特性に関する種々のデータを取り扱う。変電所の電圧—電流特性を模擬するのがこのオブジェクト主要な機能である。この機能については後に詳しく述べることにする。

なお、この elecchar という名称は，このクラスが列車・変電所共通の基本クラスとして，饋電等価回路から見た電気的な特性を模擬するためのオブジェクトとなることから名付けられている。

⟨C.5⟩ 列車クラス train

列車の特性を表すためのオブジェクトが train である。上記の elecchar と同様，電気的な特性に関する種々のデータを取り扱う。それに加え，列車の運動に関するデータを取り扱っている。

変電所は複数の饋電線に同時に接続（母線を通じて饋電線同士のタイを行うことに対応している）できることおよび位置が不変であることを除けば，等価回路上列車との区別がない。このことを利用するため，列車クラスは変電所クラスから派生させることとし，電気的な特性を示す関数について列車クラスと変電所クラスでまったく同様のインタフェースを持たせた。そして，変電所クラスからの拡張部分に上記の列車運動シミュレーション部分を入れることにした。この様子を図C.1に示そう。

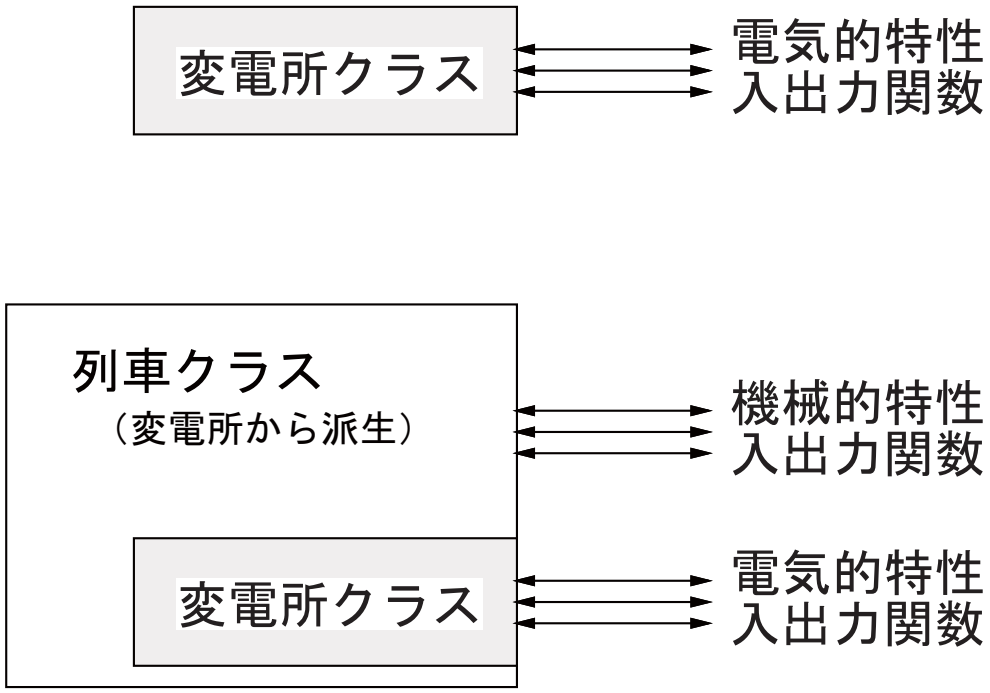
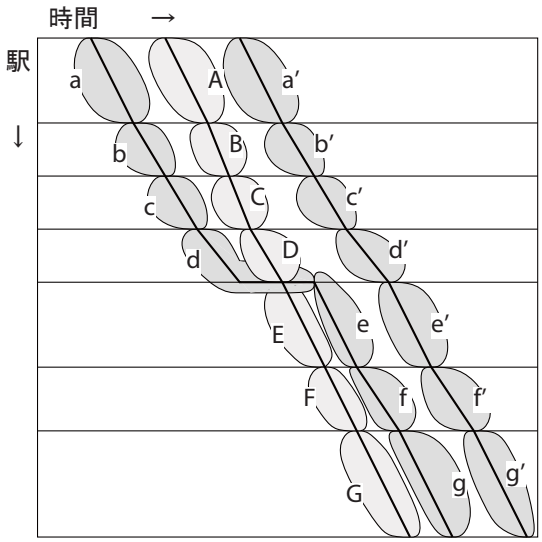


図 C.1: 変電所クラスおよび列車クラス



- **次駅データ**:
  A, B, C, . . ., a, b, c, . . .
- **急行列車のダイヤパターン**:
  { A, B, C, D, E, F, G }
- **待避を行う緩行列車のダイヤパターン**:
  { a, b, c, d, e, f, g }
- **待避を行わない緩行列車のダイヤパターン**:
  { a', b', c', d', e', f', g' }

図 C.2: 次駅データ配列の概念

いっぽう，現在のところ**信号保安設備関係はまったくインプリメントしていない**。このことから，列車の走行は各列車が互いに完全に独立にシミュレート可能である。このため，列車の走行に関するルーチンは列車クラスの中にすべて封じ込めることが可能である。この列車走行に関するデータは後に述べる nextsta クラスとして保存し，これへのポインタを列車データに持たせることで対処している。

これも，詳しいことは後述する。

⟨C.6⟩ 列車ダイヤパターンクラス diapattern，および次駅データクラス nextsta

列車ダイヤのモデルは，列車の路線全体にわたる走らせ方を規定するものであり，diapattern クラスが一つのダイヤパターンを管理する。diapattern クラスは実は nextsta クラス（次駅データクラス）の配列である。さらに nextsta クラスは gradcrv クラス（勾配・曲線・速度制限データクラス）の派生クラスとなっている。

列車ダイヤを表現する最小のデータ単位が次駅データで，それを表現するためのクラスが nextsta クラスである。全体を駅間ごとに切りとって管理し，列車がある駅からその次の駅（場合によっては駅間に特定の地点を設け，その地点までと限定することもある）までの走行に必要な情報を格納する。

勾配・曲線情報などのデータもこの次駅データクラス nextsta によって管理される。ファイルリード時には独立させ，別に gradcrv クラスの配列をつくることとし，データを記述するファイルも別にしてある。しかし，ファイルリードが終われば，nextsta クラスは必要な勾配・曲線情報をも自分で保持する（該当する gradcrv クラスへのポインタとしてではなく，自分で持っている）。この方法はメモリ容量をいささか無駄使いするが，メモリ容量制限の多かったパソコンでの使用は考えていないので，現在までのところ目だった問題にはなっていない。

次駅までの走行に関するデータがすべて nextsta クラスによって管理されるのだから，nextsta オブジェクトを必要数だけ並べれば列車の行路表が描かれたことになる。これが diapattern クラスである。このダイヤパターンの上に等時隔で複数列車を張り付けることもできるようになっている。高木が今までにシミュレーションしたほぼすべてのケースでは，1 ダイヤパターンに複数の列車を張り付けるこの方法でシミュレーションが可能であった。

このことをもう少し詳しく見てみよう。図 C.2 は次駅データ配列，すなわち diapattern クラスの概念的な図を示す。A, B, C, . . . は急行列車のダイヤパターンであり，緩行列車のそれは退避を行うものが a, b, c, . . .，行わないものが a', b', c', . . . となっている。当然ながら同一の駅間を扱う次駅データである A

と a，a' には同一の勾配・曲線データが書かれる（多くの場合速度制限データも同じだろう）。さらに，緩行列車の場合，退避駅を含まない次駅データ，すなわち a ～ c，e ～ g と a' ～ c'，e' ～ g' と（d と d' とを除く）はほとんどのケースでまったく同一になる。しかし，A ～ Gの1組，a ～ g の1組，そして a' ～ g' の1組が，それぞれ1つの行路表に対応するので，同一でもデータは持たなければならない。

列車クラスは初期値としてどれかのダイヤパターン内の次駅データへのポインタを持ち，これを用いて列車走行の計算を行う。次駅データに記述された走行区間が終了したら，列車クラスは現在の次駅データの「次」の次駅データにポインタを切替える。

⟨C.7⟩ 饋電線クラス feedline および Y 行列作成クラス feed_y

饋電線クラスは饋電線に関するデータ（例えば饋電定数……長さ当たりの饋電線の抵抗），および列車・変電所とこの饋電線との接続関係を記述する。この接続関係を記述するためのデータは feedpos 構造体 と呼ばれる。1本の饋電線には通常複数の変電所並びに列車が接続するから，feedline クラスは feedpos 構造体の配列を持っている。ただし，変電所は位置不変だが列車は位置も数も変わるので，これらを分ける必要がある。従って，他と同様な単純な配列というわけにゆかなかったため，このクラスは table クラステンプレートを使っていない。

直流電気鉄道の饋電システムを詳細に見ると，図3.1・3.2（10ページ）のようになっている。これからわかるように1本の饋電線というのは変電所と変電所を結ぶものである。また饋電線と電車線（架線）とは別になっており，ざっと200〜300メートル間隔で「饋電分岐線」によって結ばれている。プログラム，および等価回路上ではこれらのことは無視し，饋電線と電車線は一体であるとして計算を行うほか，饋電線に3つ以上の変電所が取り付くことも可能にする。また，饋電線と饋電線はすべて変電所の母線で接続されることを利用してモデルを作る。帰線としてレールを使っているが，レールの抵抗分も饋電線の抵抗に合算し，変電所・列車とも地上側はすべて接地するように考える。さらに，列車の長さは考慮しない。電圧・電流は定常解だけを求めるようにし，過渡解析および高調波解析は行わない。こうして得られる等価回路は図5.2（23ページ）に示すようになる。列車・変電所以外はすべて抵抗だけのネットワークである。この簡略化によって饋電等価回路の演算はだいぶ容易になる。

さて，この饋電等価回路から Y 行列を作る必要がある。列車・変電所をのぞいた回路網のブランチについて，電流および電圧ベクトルをそれぞれ $\boldsymbol{I}_B$，$\boldsymbol{V}_B$ としよう。図5.2の場合は5つのブランチ（図中 R_1〜R_5）が存在する。ここで，これら5つのブランチのアドミタンスを対角成分に持つ行列 Y_B を定義しよう。図5.2の場合

$$Y_B = \begin{pmatrix} 1/R_1 & & & & 0 \\ & 1/R_2 & & & \\ & & 1/R_3 & & \\ & & & 1/R_4 & \\ 0 & & & & 1/R_5 \end{pmatrix} \tag{C.1}$$

となる。このとき、

$$\boldsymbol{I}_B = Y_B \cdot \boldsymbol{V}_B \tag{C.2}$$

となる。

ここで，節点–枝接続行列 H なるものを導入しよう。節点–枝接続行列とは，ノードとブランチの接続関係を表す行列であり，ブランチ k はノード i が始点でノード j が終点であるとき，その (i,k) 要素は 1,

その (j,k) 要素は -1 となる。1，-1 以外の要素はすべて 0 である。H は，例えば図5.2の場合には

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix} \tag{C.3}$$

のように書ける。

これを利用すると，すべての列車・変電所の集合はこの回路網のカットセットになるから，列車・変電所の電流・電圧ベクトルをそれぞれ $\boldsymbol{I}$，$\boldsymbol{V}$ とすると

$$\boldsymbol{I} = H \cdot \boldsymbol{I}_B \tag{C.4}$$

$$\boldsymbol{V}_B = H^T \cdot \boldsymbol{V} \tag{C.5}$$

となる。

式 (C.2)・(C.4)・(C.5) より $\boldsymbol{V}_B$, $\boldsymbol{I}_B$ を消去すれば，

$$\boldsymbol{I} = H \cdot Y_B \cdot H^T \cdot \boldsymbol{V} \tag{C.6}$$

$H \cdot Y_B \cdot H^T \equiv Y$ とすれば、

$$\boldsymbol{I} - Y \cdot \boldsymbol{V} = 0 \tag{C.7}$$

という式が導かれる。この式，および Y 行列が回路計算の基本となる。

ここで注意したいのは，Y 行列を使っている限りブランチを意識する必要がないことだ。ブランチは列車の位置に依存してその本数や位置関係がかわるため，番号をつけるのが難しい。それを意識しないで済むことで，簡単になる要素が非常に多い。

さて，饋電システム内には複数の饋電線があり，それぞれが `feedpos` 構造体の配列を持っている。その構造体を距離でソートすれば，`feedpos` 構造体の間の距離が求まるだろう。さらに，こうしてソートした `feedpos` 配列を利用して，ブランチの数，および饋電定数と距離とからブランチのアドミタンスを求めるのは容易である。こうして取り出されるブランチに関する情報（始点ノード番号，終点ノード番号，アドミタンスなど）は，ブランチ1本あたり1つの `bran_node` 構造体に格納する。この情報を `bran_node` 構造体の配列の形で1つのオブジェクトに集め，この中で上記の Y 行列を求める操作を行えば楽であろう。これが `feed_y` オブジェクトである。従って，`feed_y` オブジェクトは `bran_node` 構造体の配列である。`table` クラステンプレートを使っている。

各 `feedline` オブジェクトは `feedpos` 構造体のソートを行い，`bran_node` 情報を引き出してそれを小さな `feed_y` オブジェクトに配列として格納する。饋電システム内には全体を統括する `feed_y` オブジェクトが1つあり，各 `feedline` オブジェクトが作成する小さな `feed_y` オブジェクトを統合し，全体の回路について Y 行列作成の作業を行うようになっている（図 C.3）。従って，特に必要がなければ接続行列やブランチアドミタンス行列はこの `feed_y` オブジェクトの中でのみ取り扱われることになる。

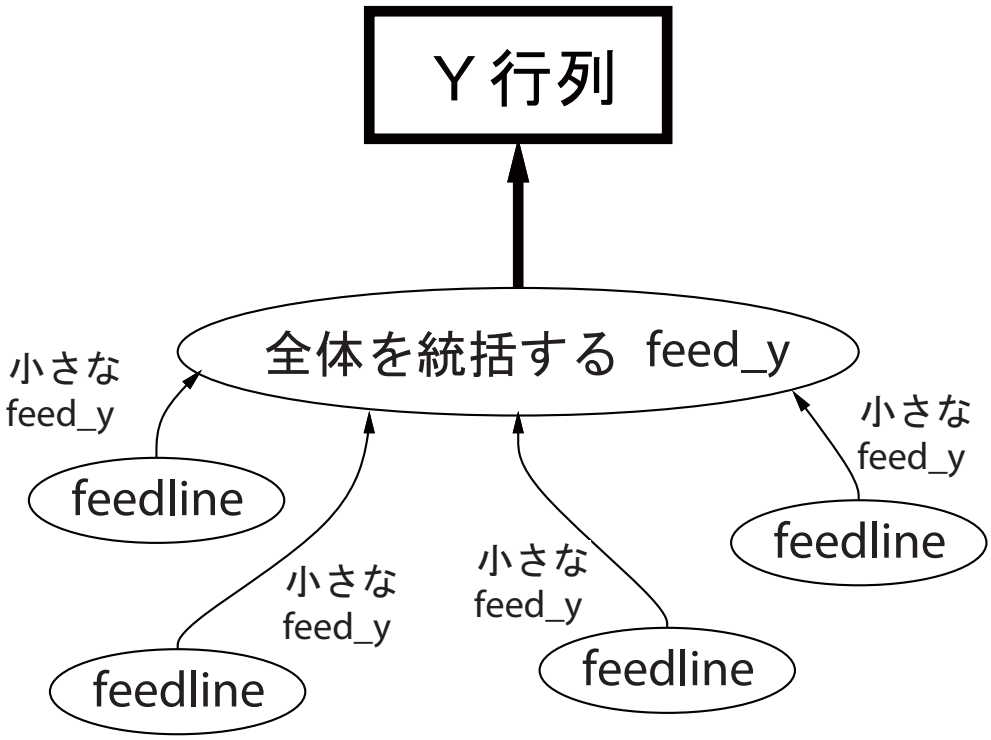


図 C.3: Y 行列の作成にかかわる主なオブジェクト

付録 III

RTSS マニュアル・アルゴリズム詳説

D

配列管理オブジェクト

この章では，いろいろなアルゴリズムを述べる前に，配列管理オブジェクト `table`，`reftable`，`zerotable`，`zeroreftable` について述べる。

⟨D.1⟩　配列管理オブジェクトの目的

あるオブジェクトの配列を手軽に定義したいというのが，このクラスの基本的な目的である。ほとんどのクラスライブラリに似たような機能のものがあるのだろうが，このプログラムでは自分で書いたライブラリを使っている。

配列管理オブジェクトにおいてやりたいことは，以下のようにまとめられる:

(1) 配列を管理する。通常の C のポインタ機能だと「配列の外に飛び出す」エラーが頻繁に起こるが，それを Segmentation Fault エラーで検出しているのは危険（大量のデータを扱っていると，Segmentation Fault が出ないでプログラムが動き続けることすらある！）。これを正確に検出し，エラーを出して止まらせたい。

(2) 配列に手軽に要素を付け加えたい。可変長の配列。配列を一気に消去したい場合にも対応できる。要素の数を知ることも容易にできる。

(3) 配列を手軽に代入したい。

(4) 配列の中身を見るには，従来と同じく `[]` 演算子が使いたい。

(5) いろいろな種類のオブジェクトに，同じコードで対応させたい。

C++ にとっては，これらの機能の実現はいわば「お手のもの」であるといってよい。

⟨D.2⟩　考え方

まず，⟨**D**.1⟩の (5)から，配列は当然のことながらテンプレート機能を使って実現することになる。また，(3)の実現は代入演算子のオーバーロードで，また (4)の実現は `[]` 演算子のオーバーロードでそれぞれ実現できるだろう。`[]` 演算子において，配列範囲外の参照をチェックする機構をつけておけば完璧である。

(2)にあるような要素を付け加えたり削除したりを自在にできるようにする機能の実現に，この種のクラスライブラリのミソがあろう。しかし，RTSS で使っている自作のクラスはこの部分が十分といえないうらみがある。それでも,「1 つ加える」のは `+=` 演算子をオーバーロードすることによって簡単に実現できる。また，配列同士を「くっつける」ことも，`+=` 演算子により簡単に実現できるようになっている。削除の方がいま一つだな，と反省しているところだが，一斉に全部削除なら簡単にできるようにはなっている。

⟨D.3⟩ 関数リファレンス

この table クラス群は，いくつかのクラスの基底クラスとなっている。そのリストを示そう:

- diapattern クラス（nextsta クラスオブジェクトの配列）
- sschar クラス（subchar 構造体オブジェクトの配列）
- sscon クラス（ssfeedconnect 構造体オブジェクトの配列）
- feed_y クラス（bran_node 構造体オブジェクトの配列）
- gradcrv クラス（gcvel 構造体オブジェクトの配列）
- nextsta クラス（gcvel 構造体オブジェクトの配列……gradcrv からの派生クラス）
- gradcrv クラス（gcvel 構造体オブジェクトの配列）
- station_obj クラス（to_station 構造体オブジェクトの配列：現在設計中）

これらが table，reftable，zerotable，zeroreftable のどれかを基底クラスとして持つクラスとなっている（いうまでもないことだが，テンプレートクラスであるから型はすべて異なっており，関数名が同一という以上のことはないのだが)。つまり，以下で述べる関数はすべてこれらのクラスも共通に持っていることに注意してほしい。

⟨D.3.1⟩ 変数

基本的な変数は3つ，num，data，mmax である。

なお，以下で TP 型というのはテンプレート機能で置換される前の型をいう。例えば，宣言で

```
template<class TP> class table {
    TP* data;
}
```

などと書いておいて，さらに

```
table<foo> a;
```

と書かれたならば，a においては TP は foo と置き換わっていることになる。ここには基本的に int でも double でも何でも書くことができる。

data … TP * 型，すなわち TP オブジェクトへのポインタである。これが配列そのものを表す。

mmax … 現在 data ポインタの指している配列が，TP オブジェクトいくつ分のメモリを持っているかを示す。当然のことながら，mmax 個以上のオブジェクトをここに持つことは許されない。

num … 現在配列にいくつのオブジェクト TP があるかを示す。これは mmax の値よりも小さければよい。すなわち，配列の中身を「すべて消す」操作をしたいなら，num の値を単に0にすればよいことになる。

⟨D.3.2⟩ コンストラクタ

どのクラスも，引数なしのデフォルトコンストラクタと，いわゆるX(X&)（参照による初期化用コンストラクタ）しか持っていない。

デフォルトコンストラクタは，単に data にヌルポインタを，num と mmax にゼロを代入するだけである。参照による初期化用コンストラクタは，mmax，num および data の中身を（ポインタをコピーするだけではなく）きちんと1要素ずつコピーする。

⟨D.3.3⟩ 要素を参照する

基本的には，operator[]() 関数を使うことによって可能なのだが，　table，zerotable と reftable，zeroreftable とでは機能に大きな違いがある。すなわち，前2者では返り値が TP オブジェクトそのもの

（のコピー）であるのに対し，後2者では返り値がTP への参照となっている。つまり，ref がついていると配列の中身をあとで変更可能ということになる。従って，この両者は時と場合によって使いわけをしなければならないことになる。特に前2者の場合，この関数はconst メンバ関数である（この辺は C++ の解説書を参照のこと）。なお，後2者において，operator[]() が const でないために困るケースが出てくることがあるため，operator[]() と同一の仕組みながら TP 型（参照型ではなく）を返す関数として element() を用意している。

この点を除けば，関数 operator[]() の機能は基本的に同一である。table<foo> 型のオブジェクト a の持つ配列の3番めのデータにアクセスしたい場合，

```
foo b = a[3];
```

のように，通常のポインタに対するのと同じように書くことで，メモリ上確保された範囲を越えることなくデータにアクセス可能となる。もちろん，

```
foo b = a.operator[](3);
```

でもよい。メンバ関数ならば後者の方法がよいだろう。

⟨D.3.4⟩ データを加える

基本的には，operator+=() 関数を使うことによって可能である。　operator+=() 関数は，引数として TP 型ををとる場合と自分自身の型（table<foo> なら table<foo> など）をとる場合とがある。

前者では，引数に与えたオブジェクトを配列の一番末尾にコピーする。もし，配列がすでに確保したメモリいっぱいまで使われてしまっているならば，配列を大きくする操作を行う（一時的記憶領域に配列の中身をコピーし，data を delete し，より大きい領域を new し，中身を戻す）。ここで，table および reftable クラスは「より大きい値」として，直前のより GETMEMORYNUMBER だけ大きな配列をnew 演算子で得るようにしている。+= 演算子は通常1つのデータのみを与えるが，1つだけならば単に1つだけ大きな配列を new してくれば足りる。しかし，そうせずに GETMEMORYNUMBER （現在 40 に#define されている）だけ大きな領域を確保することにより，高価な操作である delete や new の回数を削減することができる。ただ，場合によっては記憶領域が無駄になることもあるため，zerotable および zeroreftable で「1つだけ」のタイプのものも用意はしてある。

後者の場合は，ふたつの配列をつないで一つにする。もちろん，+= 演算子の左辺が前，右辺が後となる。table・reftable とzerotable・zeroreftable の差異は前者と同じである。

⟨D.3.5⟩ 要素の数を知る

number() 関数を使うことによって可能である。　number() 関数は引数をとらない。int 型であり，単純にnum を返すだけである。

⟨D.3.6⟩ データを消去する

任意のデータを1つだけ消去することは，現時点では不可能である。しかし，データをいっきにすべて消去することなら，renew() 関数を使うことによって可能である。　renew() 関数は void 型である。引数として int 型を1つとる場合と，引数なしの場合がある。引数なしの場合は単純に num を 0 にリセットするだけである。引数をとる場合，将来的にはその数だけのオブジェクトを格納する可能性があることをオブジェクトに知らせる。すなわち，

```
table<foo> a;
a.renew(45);
int b = a.number();     // ここでは b = 0 となる
```

としたばあい，a は少なくとも 45 個の foo オブジェクトを格納することができるだけの記憶領域を動的に確保する。しかし，その後でnumber() 関数を参照してみると，0 となる，すなわちオブジェクトは記憶領域だけは確保したもののオブジェクトは一つも「もっていない」状態になっていることがわかる。

これの応用で，setnum() という関数も用意されている。　これはvoid 型の関数で，引数として int 型を1つとる。renew() したあとで num 変数を希望の値にセットする。例えば，

```
table<foo> a;
a.setnum(45);
int b = a.number();
```

のように操作すると，b は 45 となる。ただし，以前number() で返る値が 20 だったものをこの setnum() 関数で 45 に増やした場合，新しい配列の 0〜19 番めに古い値は保存されないことに注意する必要がある。

⟨D.3.7⟩　その他

このクラスの派生クラスでは，error26_name() なる関数をプライベート関数として用意しておくことが推奨される。　error26_name() 関数は void 型である。引数は何もとらず，operator[]() 関数で添字変域範囲の逸脱を起こした場合に表示すべきエラーメッセージの表示ルーチンを書き込む。こうすることによって，エラーメッセージからのディバッグがより容易になることだろう。

E

等価回路演算の方法

まず、この章では電気的評価の基本となる等価回路演算の方法の概要を述べる。

⟨E.1⟩ Newton-Raphson 法を用いた饋電等価回路演算

⟨E.1.1⟩ 考え方

図5.2（23ページ）の直流饋電等価回路は，変電所・列車をのぞけば抵抗だけを含む回路である。変電所は電圧源＋直列内部抵抗で，列車は電流源でそれぞれ近似することが多いが，このような近似が成立していればこの回路は行列演算1回で解くことができる。すなわち，回路のY行列を用いて，変電所・列車の電流・電圧ベクトルを$\boldsymbol{I}$・Vとするならば、すでに130ページで述べた式

$$\boldsymbol{I} - Y \cdot \boldsymbol{V} = 0 \tag{C.7}$$

を解けば容易である。

しかし，列車・変電所は非線形性を持っている。例えば列車は速度が低いか電圧が高いときは電力一定負荷に近くなり，逆に速度が高いか電圧が低いときは純抵抗に近くなる。最近はブレーキ時に電動機で発電して電力を架線に返す電力回生ブレーキ付き電気車が一般的になっているが，この回生中電圧が上がると「絞り込み」という制御をかけて電流を抑制するようにしている。一方，変電所は，負荷である饋電システム側から電源側への逆流はできないのがふつうである。

これらの非線形性の存在により，従来のプログラムでは

(1). ある $\boldsymbol{V}_0$ を仮定し，$i = 0$ として (2) へ
(2). $\boldsymbol{V}_i$ と式 (C.7) から $\boldsymbol{I}_i$ を求める
(3). $\boldsymbol{I}_i$ と列車・変電所の特性から $\boldsymbol{V}_{i+1}$ を求める
(4). $\boldsymbol{V}_i$ と $\boldsymbol{V}_{i+1}$ との差が収束判定限界内なら終了，そうでなければ i を一つ増やして (2) へ

という手順を踏んでいた。しかし，この方法では収束がきわめて悪く，解けないケースも多かった。

RTSS では，この部分には多変数の Newton-Raphson 法を用いている。まず，それぞれの列車・変電所の特性は V–I ないし I–V 平面上で1本の曲線として表示されると仮定しよう。このとき，曲線は1つの媒介変数を用いて記述できるはずである。1列車または1変電所あたり1つの媒介変数が必要である。列車または変電所 No. i の特性を表す媒介変数を θ_i，それを集めてベクトルとしたものを $\boldsymbol{\theta}$ とすると，

$$\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{\theta}) \tag{E.1}$$

$$\boldsymbol{I} = \boldsymbol{I}(\boldsymbol{\theta}) \tag{E.2}$$

と書き表せる。こうすると，式(C.7)を満たす$\boldsymbol{I}$, $\boldsymbol{V}$ を求める問題は

$$\boldsymbol{I}(\boldsymbol{\theta}) - Y\,\boldsymbol{V}(\boldsymbol{\theta}) \equiv F(\boldsymbol{\theta}) = 0 \tag{E.3}$$

を解いて $\boldsymbol{\theta}$ を求める問題に帰着できる。これをNewton-Raphson 法により解けば解が求まる。

この方法でも計算不能となる場合がまだ残るが，従来のプログラムよりは格段に少なく，また繰り返し計算の回数も減少し，計算時間の短縮が図られた。

⟨E.1.2⟩ 多変数 Newton-Raphson 法

多変数の Newton-Raphson法は，連立方程式を数値的に解く方法として大変ポピュラーな手法であり，収束も速いことが知られている。その方法の概要は以下の通りである。

連立方程式

$$\boldsymbol{F}(\boldsymbol{\theta}) = \boldsymbol{0} \tag{E.4}$$

は，第0近似解 $\boldsymbol{\theta}_0$ が与えられたとき、次の漸化式を用いて解くことができる。

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - J^{-1}\,\boldsymbol{F}(\boldsymbol{\theta}_k) \tag{E.5}$$

ただし，J はヤコビ行列:

$$J = \begin{pmatrix} \dfrac{\partial f_1}{\partial \theta_1} & \cdots & \dfrac{\partial f_1}{\partial \theta_n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \dfrac{\partial f_n}{\partial \theta_1} & \cdots & \dfrac{\partial f_n}{\partial \theta_n} \end{pmatrix} \tag{E.6}$$

である。ここに

$$\boldsymbol{F}(\boldsymbol{\theta}) = (f_1, f_2, \ldots, f_n)^T,$$
$$\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^T,$$

また $\boldsymbol{\theta}_k$ は解くべき方程式の第 k 近似解である。

⟨E.1.3⟩ プログラム

`feeder::ycal()` 関数 で等価回路の Y 行列を作成し，次いで`feeder::ccal()` 関数 で等価回路を数値的に解く操作を行う。`ccal()` 関数の方に，Newton-Raphson 法の基本的な部分が含まれている。また，これらの関数を順次呼び出す関数として，`feeder::simccsub()` 関数 が用意されている。

⟨E.1.3.1⟩ Y **行列の作成** `simccsub()` 関数で，計算の最初にまず `ycal()` 関数を呼び，Y 行列を求める。

この `ycal()` 関数内では，次のようなことを行う。

(1) まず列車オブジェクト `train` から饋電線接続点データとなる `feedpos` 構造体を引き出し，対応する饋電線オブジェクト `feedline` に順次記憶させてゆく。こうすると，`feedline` がもともと持っている変電所と饋電線の接続点情報を合わせ，列車・変電所と饋電線のすべての接続点，すなわちノードに関する情報が，複数の `feedline` オブジェクトに分散して集められたことになる。

(2) このあと，それぞれの `feedline` オブジェクトの中で配列になっている `feedpos` 構造体を距離でソートする。そうすると，隣接する `feedpos` 構造体の間がブランチということになる。`feedline` オブジェクトがデータとして持っている饋電定数値に `feedpos` 同士の距離をかければ、ブランチのインピーダンスが求まる。`feedpos` はその点のノード番号も知っているから，ブランチはどのノードからどのノードまでを結ぶかがわかる。こうすると，すべてのブランチについて，ブランチの起点・終点ノード，インピーダンスが把握できることになる。こうして，1つのブランチに対して1つの `bran_node` 構造体が作られる。

なお，それぞれの feedline オブジェクトには feeder_shape 列挙型定数データが含まれている。これが Long であれば通常の直線上の路線となり，Circle であれば饋電線の末端同士がつながれたモデルになる。Circle は山手線のような環状路線のシミュレーションにおいて用いられる。また，ブランチインピーダンス決定に際して，変電所母線と饋電線接続点が離れているようなケースを扱えるよう，饋電線の長さに比例する項だけでなく饋電線接続点とそのノードとの間に一定のインピーダンスを挿入できるような機能も備えている。さらに，隣あうノードの距離が 1m を下回る場合は 1m として計算する例外処理機能もある。

(3) たくさんの bran_node 構造体がそれぞれの feedline オブジェクト内で生成された段階では，ブランチはまだナンバリングされていない。これは，ブランチの全体数がまだはっきりしないからである。そこで，これらを一箇所に集めることを考える。まずは，それぞれの feedline オブジェクトから bran_node 構造体の配列を feed_y クラスのかたちで返す。すでに ⟨C.7⟩ ないしは図 C.3にて述べた通り，feed_y は bran_node の配列として定義されている。

なお，(2)・(3) の一連の操作は，それぞれの feedline オブジェクトに対して feedline::sortx() 関数 を呼び出すことで行われる。この sortx() 関数は feed_y オブジェクトを返す。

(4) 返された複数の feed_y オブジェクトをまとめ，一つの feed_y オブジェクトとする。この操作は，sortx() 関数が返した feed_y オブジェクトを，feeder クラスのメンバオブジェクトである feedy に += 演算子で加えることによって実現される。

(5) こうすると，すべてのブランチに関する情報，すなわち bran_node 構造体が一つの feed_y オブジェクト feedy の中に集められたことになる。こうなれば，ブランチの番号は feedy における配列の順番として定義すれば容易であり，節点—枝接続行列，ブランチアドミタンス行列，従って求める Y 行列を求めることは，実に容易であることがわかろう。

ところで，Y 行列をこのようにして求めると，変電所と列車が隣接している場合にその両者の間にあるアドミタンスが非常に大きくなる。このときにはごくわずかの列車・変電所の電圧のずれが大きな電流になることとなり，結果として収束が悪くなる。そこで，130ページで述べた式

$$\boldsymbol{I} - Y \cdot \boldsymbol{V} = 0 \tag{C.7}$$

において，列車と変電所とで電圧・電流を逆に見ることにしてある。そのために Y 行列を 4 分割し，書き換えるルーチンも作ってある。例えば，変電所が電圧源として計算され，そのきわめて近傍にある列車が電流源であるなら，その間のアドミタンスが非常に大きくても極端に大きな電流がながれるようなことはなくなり，収束がわずかながら改善される。

⟨E.1.3.2⟩ Newton-Raphson 法による演算 基本的な Newton-Raphson 法の考え方は上記に述べたが，収束「しない」場合の対策としてリミットサイクル発見ルーチンを設けてある。これは，式 (E.3) の左辺の $\boldsymbol{F}(\boldsymbol{\theta})$ のうち最大のものの値が周期的に現れるようになったらそこでリセットをかけるというものである。そのために，ccal() 関数の冒頭にその値を格納する配列 ccbftt が static に定義されている。

リミットサイクルを検出したばあい，まず 1 サイクル分の媒介変数の平均をとり，それから計算をやり直す。それでも失敗する場合は，いったん ccal() 関数を抜け，計算失敗しないように列車の特性を変更してやり直す（この場合は誤差が出る）。このリトライもまた失敗したならば，計算そのものが不能であるといって諦めるプログラムとしてある。

⟨E.2⟩ 変電所における電圧–電流特性の実現

ここでは、変電所における V–I 特性の実現方法を簡単に論じる。列車についても媒介変数を使う部分については同じだが、速度などの要素が複雑に絡むため同一には扱えない。そこで、ここでは論じず、**付録 (III) F**章にて改めて論じる。

⟨E.2.1⟩ 考え方

変電所の特性を表すのに必要なデータは，電圧–電流特性である。変電所の位置と饋電線との接続関係は饋電線データファイルにおいて与えられる。この電圧–電流特性の与え方は次のように至ってシンプルである。

まず，変電所の電圧–電流特性は V–I 平面上の折れ線で与える。折れ線は電流については正側（電力供給側）および負側（電力回生側）のどこを通ってもよいが，あとで饋電等価回路演算の際困らないようなものを与えないと回路演算失敗の原因を作ることになるので注意が必要だ。また，当然のことながら電圧の方は正側のみを通るように特性を与えなければならない（与えようと思えば負電圧も与えられないことはないが)。

特性の折れ曲がる点は，通常3つの成分（媒介変数，電圧，電流）によって表示する。その点は変電所毎に任意の数を設定できる。例えば，折れ曲がる点のうち i 番目のものの3成分がそれぞれ θ_i, V_i, I_i，点 $i+1$ の3成分がそれぞれ $\theta_{i+1}, V_{i+1}, I_{i+1}$ と与えられているとすれば，点 i から点 $i+1$ までの間の特性は媒介変数を用いて

$$V(\theta) = \frac{\theta-\theta_i}{\theta_{i+1}-\theta_i}(V_{i+1}-V_i)+V_i \tag{E.7}$$

$$I(\theta) = \frac{\theta-\theta_i}{\theta_{i+1}-\theta_i}(I_{i+1}-I_i)+I_i \tag{E.8}$$

と表せる。要するに線形に補間しているだけである。

⟨E.2.2⟩ プログラム

特性の折れ曲がる点を与えるのが subchar 構造体である。この構造体を配列にしたものが sschar クラスである。sschar クラスは配列クラステンプレート zeroreftable を用い，これからの派生として定義されている。

変電所は elecchar クラスとして表現される。このクラスのオブジェクトが変電所の数だけ存在することになる。この elecchar オブジェクトは sschar クラスのオブジェクトを一つずつ持ち，subchar オブジェクトを必要な数だけ sschar クラスにデータとして持たせるようにしている。

これらの構造体およびクラスの中身は ‘elecchar.hh’ で定義されている。

subchar 構造体は teta, volt, ampere, controlnext, voltlow の5成分で構成されている。これらのうち controlnext だけが bool 型変数で，他は double 型である。これらのうち teta，volt，ampere はそれぞれ式 (**E**.7)・(**E**.8) における θ_i, V_i, I_i に対応する。これ以外の2つは「変電所電圧リアルタイム制御・タイプA」に対応するもので，後述する。なお，teta は綴りが違う (正しくは theta) が，θ のつもりだ。

各 elecchar オブジェクトは現在の電圧・電流に対応する θ の値を変数 teta として保存している。こうして，回路計算がさきに述べたような極めてシンプルなアルゴリズムでできるようになる。

sschar データから変電所の電圧電流特性を実現する関数は，このアイディアによって極めてシンプルに書けている。elecchar::teta_vi() 関数 がこれを実現する。この関数では，現在 elecchar オブジェクトが持っている teta の値に対応する電圧・電流値を計算する機能を持つ。これとあわせ，Newton-Raphson 法の計算に必要な，その位置での関数の微係数の計算も行っている。

なお，teta は subchar 構造体の指し示す各点に対応する媒介変数を表すだけであるから，自由に選ぶことができる。ただし，elecchar のメンバである subc 配列（sschar 型）の最初の subchar 構造体に記述された値より小さい teta，および最後の subchar に記述されたのより大きい teta は許されないようになっており，この関数で teta の変域も変域内に抑え込まれる。また，subc 配列の前の方ほどデータとして与えられる teta の値も小さいことを仮定してプログラムが書かれている。データファイルのこの点のチェックはしていないので，ユーザはデータファイルの記述においてはきちんと考慮を払う必要がある。なお，電圧・電流の方は，大きい方から書いても小さい方から書いても，順にみてゆくため，特性が正しければ問題はない。

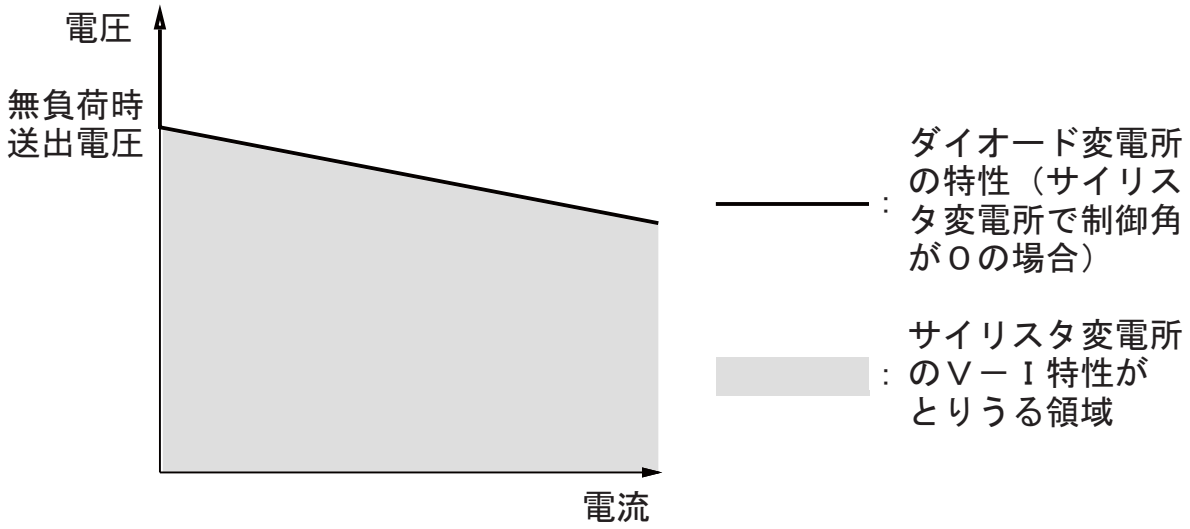


図 **E**.1: ダイオードおよびサイリスタ変電所のV–I特性

⟨E.2.3⟩ ノウハウ

ところで，データをきちんと与えたとしても，計算に失敗することはよくある。折れ線でデータを与えている限りこれは仕方のないことかも知れない（実物は折れ線でも何でもないからだ！）。現在までに高木が経験したこととしては次のようなことがあげられる。

- ふつうのいわゆるシリコン変電所は無負荷時送り出し電圧以上では電流ゼロで，無負荷時送り出し電圧以下の場合電流の増加にともなって電圧が低下していくようにモデル化される（図**E**.1）。しかし，このモデルでは特に負荷が小さい場合に計算失敗を起こしやすい。対策として次のようなことが有効であった。
  - この部分の特性を負側にわずかに傾ける。
  - この部分は媒介変数の変域を多めにとる。
- 複雑な形態の特性も計算失敗の原因になる。特に，インバータつきの変電所をモデル化する場合は要注意だ。このようなばあい，インバータとコンバータを一つの特性にモデル化するのではなく，分けてモデル化することが有効だった。

計算失敗の例を示そう。表**E**.1(1) のようなデータファイルを与え，計算を行った。ところが，この特性を与えたところ，計算失敗がかなりの頻度で（等価回路演算6回に1回程度）起こった。こんなに失敗するのでは計算結果は使い物にならないだろう。ところが，これを訂正し，同一特性ながら表**E**.1(2) のように変更したところ，計算失敗はまったく起きなくなったのである。

このファイルはSファイルであるから，データファイルの意味は⟨**J**.4⟩を参照されたい。また，特に注意していただきたいのは，変更したのは特性上の折れ曲がり点の電圧・電流値ではなく，媒介変数の値のみであることだ。主としていわゆる V–I 特性の「垂直部分」（電流がゼロの領域）に対して，媒介変数の「幅」を大きめに与えることで計算失敗が防止されていることがわかる。

すなわち，問題はV–I特性がタテに垂直に折れ曲がっている部分に集中しておきているようだ。この部分については何らかの特例的なコードを書いてやる必要があるのかも知れない。その場合，後述する、電車のモデルで回生絞り込み電圧以上になった場合のケースが参考になるかも知れないと思っている。

表 **E**.1: 同一変電所特性でもデータの与え方により計算失敗が起こる

```
substations 8

# (0) A SS
subchar 5
-3610.0 1800.0 -3333.333334
-3360.0 1550.0 -3333.333334
-30.0 1550.0 -1.0
0.0 1520.0 0.0
20000.0 1070.0 10000.0
ratedcurrent 4000.0 2.0 2.5 3.0

# (1) B SS
subchar 3
-280.0 1800.0 -1.0
0.0 1520.0 0.0
20000.0 1070.0 10000.0
ratedcurrent 2000.0 2.0 2.5 3.0

# ……以下略
```

(1) 大量のエラーが起きたケース

```
substations 8

# (0) A SS
subchar 5
-43300.0 1800.0 -3333.333334
-23000.0 1550.0 -3333.333334
-20000.0 1550.0 -1.0
0.0 1520.0 0.0
20000.0 1070.0 10000.0
ratedcurrent 4000.0 2.0 2.5 3.0

# (1) B SS
subchar 3
-20000.0 1800.0 -1.0
0.0 1520.0 0.0
20000.0 1070.0 10000.0
ratedcurrent 2000.0 2.0 2.5 3.0

# ……以下略
```

(2) エラーが起きなかったケース

F

列車の電気的モデル

列車モデルは，列車の運動方程式，電流—電圧特性，応荷重特性などを模擬するものである。電気的な，つまり列車の運動そのものに関係ない変数や関数は変電所のものと同じであるので，`train` クラスは `elecchar`（変電所モデルを格納するクラス）からの派生クラスとして定義されている。この章では，主としてこの列車モデルのうち電気的部分のアルゴリズムを詳細に論じる。

RTSS で利用する等価回路表現においては，変電所も列車も同一のノードであり区別がないこと（ただし変電所に限り複数の饋電線と関係を持つことができることだけが違う）から，変電所と列車を同じ変電所クラスへのポインタからアクセスすることにより，オブジェクト指向プログラミングのまねごとができている。いうまでもなく，列車と変電所ではその中身はまるで違うが，電圧と電流が関係を持って変化するという点では同一なのである。その点をある程度（巧みに，と書きたいところだけれど自分で書くと照れくさいので書かない）利用したのが現在のRTSS のクラス定義の形態である。

なお，列車のモデルとしてはインバータ制御のものだけをプログラムしてある。これでインバータ制御および4象限チョッパ制御の電気車はシミュレートできると思うが，他のタイプのものは不可能だ。もっとも，インバータ制御であっても若干の制限やモデル誤差は存在せざるを得ない。この辺の詳細も述べることにしよう。

⟨F.1⟩　基本

列車の電気的特性を，最終的には I–V 平面上の1本の曲線として表し，それを媒介変数表示してやろうというのが基本的な考え方である。ところが，実際には速度により電流が大幅に変わるわけだから，実車と同様に速度に応じて列車の I–V 特性を変更するための機構を用意しておかなければならない。

⟨F.1.1⟩　考え方

まず，電圧が一定とした場合の議論から入る。インバータ制御電車のトルク（引張力）–速度曲線を考えよう。これは例えば図**F**.1のように与えられる。このうち「定トルク領域」とは速度によらずトルクが一定の領域である。また,「定パワー領域」は速度によらずパワーが一定の領域で，トルクは速度に反比例する。最後の「フリーラン領域」は電動機の特性に従って加速して行く領域である。従来の直流電動機とのアナロジーから，この領域ではトルクは速度の2乗に反比例すると考えられる。

パンタ点電流–速度曲線について見ると，図**F**.2のようになる。低速域での損失があるものの，定トルク領域ではほぼ速度に比例したパンタ点電流が流れる。定パワー領域はその名の通りパワー一定なので電流は速度によらず一定となる。またフリーラン領域では電流は速度に反比例ということになる。

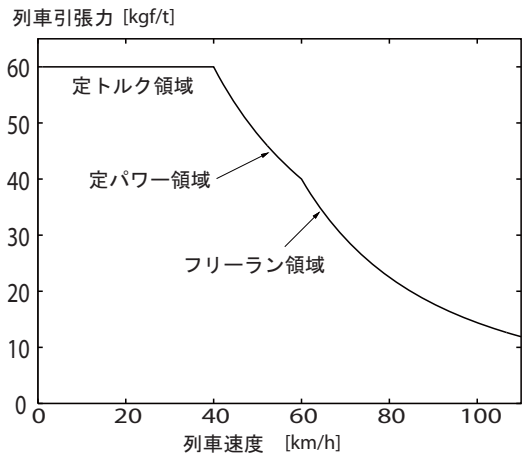


図 **F**.1: 列車の引張力–速度曲線の例

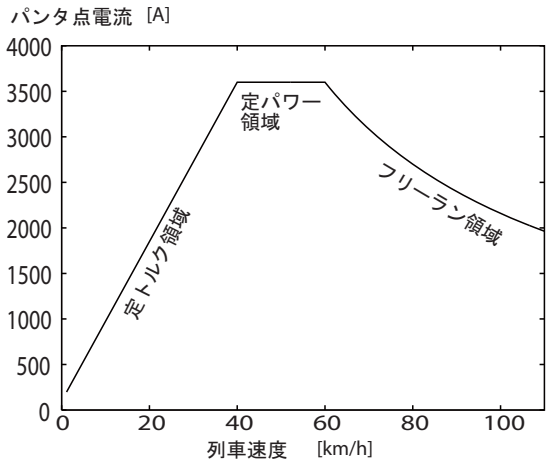


図 **F**.2: 列車のパンタ点電流–速度曲線の例

まず，これらを式にして示そう。列車速度を v とする。定トルク領域では，速度0での電流を I_0，定トルク領域から定パワー領域に移行する速度を v_l，v_l での電流を $I_{\max}$ とするならば

$$I = I_0 + (I_{\max} - I_0) \cdot \frac{v}{v_l} \tag{F.1}$$

となる。次いで，定パワー領域では

$$I = I_{\max} \tag{F.2}$$

となる。さらに，フリーラン領域では，定パワー領域からフリーラン領域に移行する速度を v_h とするならば

$$I = I_{\max} \cdot \frac{v_h}{v} \tag{F.3}$$

で表される。

さて，実際に回路計算をする際には，列車の速度 v は定数である，すなわち回路演算の最中に変化することはないと仮定して計算を行う（**付録 (II) B**参照）。しかし，電圧は変化すると考えなければならない。このとき電流–電圧特性はどのように変化させたらよいのだろうか？ ここで次のような仮定を置く:

- **v_l および v_h はパンタ点電圧に比例する。**

すなわち，列車のパンタ点電圧を V，v_l および v_h のパンタ点ノミナル電圧値 V_N における値をそれぞれ v_{low}，v_{high} とすると

$$v_l = v_{\text{low}} \cdot \frac{V}{V_N} \tag{F.4}$$

$$v_h = v_{\text{high}} \cdot \frac{V}{V_N} \tag{F.5}$$

となる。

この仮定は必ずしも正確とはいえないものの，それほど不自然なものでもない。少なくとも，従来のシミュレーションの経験からすると，大きな誤差がこれを原因として出たことはなかった。

このことを踏まえ，式(**F**.1)〜(**F**.3)の3つの式をじっくり見ると，つぎのようなことがいえる。

(1) 定トルク領域では，電圧と電流が反比例関係にある。
(2) 定パワー領域では，電流は電圧によらず一定である。
(3) フリーラン領域では，電流と電圧は比例関係にある。

ここまで来ると少々わけがわからない話になってしまう。つまり，定トルク領域では電圧と電流が反比例，すなわち定パワーであり，定パワー領域は定電流領域であり，フリーラン領域では純抵抗である，というわけだ。定トルクが定パワー，といったあたりですでに頭が爆発しそうになっている読者が多いのではあるまいか？ ここでの「領域」の呼び名は，あくまでトルク–速度曲線上での性質からとったものであり，電流–電圧曲線上の性質からとったものではないことに，特に注意を払っていただきたい。

このような混乱は常に起きる可能性があり，現に高木が研究室でディスカッションする上にあっても説明に苦慮することが多かった。しかし,「定トルク領域」などという言葉はそれなりに流通している言葉らしいので，他にあえて呼び名を作ることも考えられず，しかたなしにこの言葉で通してきているのが現状だ。

⟨F.1.2⟩ プログラム

`train::teta_vi()` 関数が，基本的にこのあたりのことを取り扱っている。いわゆる仮想関数というものになっており，`train` オブジェクトに `elecchar` へのポインタからアクセスしても `elecchar::teta_vi()` 関数ではなくこちらが呼ばれるようになっている。ここではまだノッチ・荷重の考慮をしていない（⟨F.2⟩参照）が，やっていることの基本は`elecchar::teta_vi()` 関数と同一である。この関数の中で，各変電所・列車オブジェクト毎に与えられた媒介変数（各オブジェクト内では `tt` という変数に保存）から電圧・電流およびそれらの媒介変数による微係数（$dV/d\theta$，$dI/d\theta$）を求める。

まず，列車では定電圧特性は考えられないことを利用し，`tt`（各列車・変電所オブジェクトが持っている媒介変数 θ）からまずパンタ点電圧を決めてしまうようにしている。この場合，`tt` $= 0$ がノミナル電圧値，すなわち 1,500V になるように

$$V_P = \theta + 1500 \tag{F.6}$$

としている。このノミナル電圧値を変更できるようにはしていないが，近い将来したいと思っている。

電圧がまず決まってしまうので，あとは電流関係を決めればよいことになるが，それには⟨F.1.1⟩で述べたいくつかのパラメータを決めてやらなければならない。

力行時については，必要なデータを `powerdata` なる構造体に格納している。それぞれの `train` オブジェクトは，この`powerdata` 構造体を `pdata` というメンバ変数として1つずつ保持している。

列車速度 v … これは現在の状態として与えられ，変数 `vel` として記憶されている。
パンタ点電圧 V … これは媒介変数からすでに式 (F.6) によって求まっている。
起動時電流 I_0 … データ `pdata.zerocur` にて与える。
定トルク・定パワー領域境界速度 v_{low} … データ `pdata` によって計算した値を `vlow_p` として保存し，これを用いる。
定パワー・フリーラン領域境界速度 v_{high} …データ `pdata` によって計算した値を`vhigh_p` として保存し，これを用いる。
力行時最大電流I_{max} … データ `pdata` によって計算した値を`ampmax_p` として保存し，これを用いる。

回生時については基本的に力行時の裏返しなのだが，定トルク領域についてはわずかに考え方が異なっている。力行時は速度0でもトルクが出るかわりに速度0での電流が与えられるが，回生時は速度がある値v_{off} を下回るとトルクが出ないモデルとしている[1)]。v_{off} を用いると，ブレーキ時には式 (F.1) が次のように置き換えられる:

$$I = \begin{cases} I_{\mathrm{max}} \cdot \dfrac{v - v_{\mathrm{off}}}{v_l - v_{\mathrm{off}}} & \text{for } v \geq v_{\mathrm{off}}, \\ 0 & \text{otherwise.} \end{cases} \tag{F.7}$$

[1)] 逆相ブレーキを使うモデルも考えられないことはない。これも，必要ならば将来インプリメントすることは考えられよう

必要なデータは力行時と同じように brakedata なる構造体に格納している。それぞれの train オブジェクトは，この brakedata 構造体を bdata というメンバ変数として1つずつ保持している。当然ながら，力行時と回生時とでは境界速度 v_{low}，v_{high} や最大電流値 $I_{\max}$ も異なるため，ブレーキ用に別なデータを保持することになる。

列車速度 v … これは現在の状態として与えられ，変数 vel として記憶されている。
パンタ点電圧 V_P … これは媒介変数からすでに式 (**F**.6) によって求まっている。
回生オフ速度 v_{off} … データ bdata.regenoff にて与える。
定トルク・定パワー領域境界速度 v_{low} … データ bdata によって計算した値を vlow_b として保存し，これを用いる。
定パワー・フリーラン領域境界速度 v_{high} …データ bdata によって計算した値をvhigh_b として保存し，これを用いる。
回生時最大電流 $I_{\max}$ … データ bdata によって計算した値をampmax_b として保存し，これを用いる。

なお，現在のプログラムでは，せっかく列車毎に性能データ powerdata，brakedata を持っているのに，入力ルーチンの都合ですべての列車が同一性能となってしまっている。

⟨F.2⟩ ノッチ，荷重の考慮

実際には，列車の I–V（電流–電圧）特性曲線は速度以外にもノッチ（加速の強さを表すものと考えればよかろう），そして荷重によって変わってくる。本章ではそれらを考慮するためのモデルを与える。

通常，このような特性を記述するために必要なデータがすべて与えられることはまずない。RTSS が開発・維持されている大学のような環境はもちろん，鉄道総研のような研究所，ないし場合によってはメーカですらこのようなデータの詳細が得られないことがあるようだ。そのため，合理的と思われるある種の仮定をおいて，速度・ノッチ・荷重変化時の特性を近似することになる。RTSS の仮定は，そういう観点からすると決してよい近似になっているとは思えない。

⟨F.2.1⟩ 応荷重装置のモデル化

⟨F.2.1.1⟩ 考え方 o応荷重装置のモデリングは，RTSS ver.2.0 までは2点指示方式を用いていた。これは，Inv_pmode1（定トルク領域），Inv_pmode2（定パワー領域），Inv_pmode3（フリーラン領域）の境界の速度が，混雑率 pbempty.conges のときそれぞれ pbempty.lowvel, pbempty.highvel であり，混雑率が pbfull.conges のときはそれぞれ pbfull.lowvel, pbfull.highvel であったとするとき，lowvel および highvel はこれらの中間の混雑率にあっては混雑率に比例して完全に線形に配分されるとするものだ。詳しくいうと，データとして混雑率 C_l のときの値 K_l, C_h のときの値 K_h が与えられているとき，混雑率 C のときの値は

$$K = \frac{K_h - K_l}{C_h - C_l} \times C + K_l \tag{F.8}$$

で与えられるとするものだ。

このように2つの混雑率において値を指定する場合でも，Inv_pmode2 が十分広い速度域にわたっていればほとんど問題はない（少しある）が，一部の電車ではこの領域が極端に狭いものがあり，場合によってはこの補間法は適用できないと考えられる。

そこで，線形な補間法の他に次のような補間法を用意し，スイッチで切替えられるようにした。

線形補間が適用できないケースとは，図**F**.3のように定パワー領域がないものである。定トルク領域の引張力に関しては従来と同じ線形補間で計算可能であると考えるべきだ。従来の方法で計算された定トルク領域の引張力の値を T としよう。

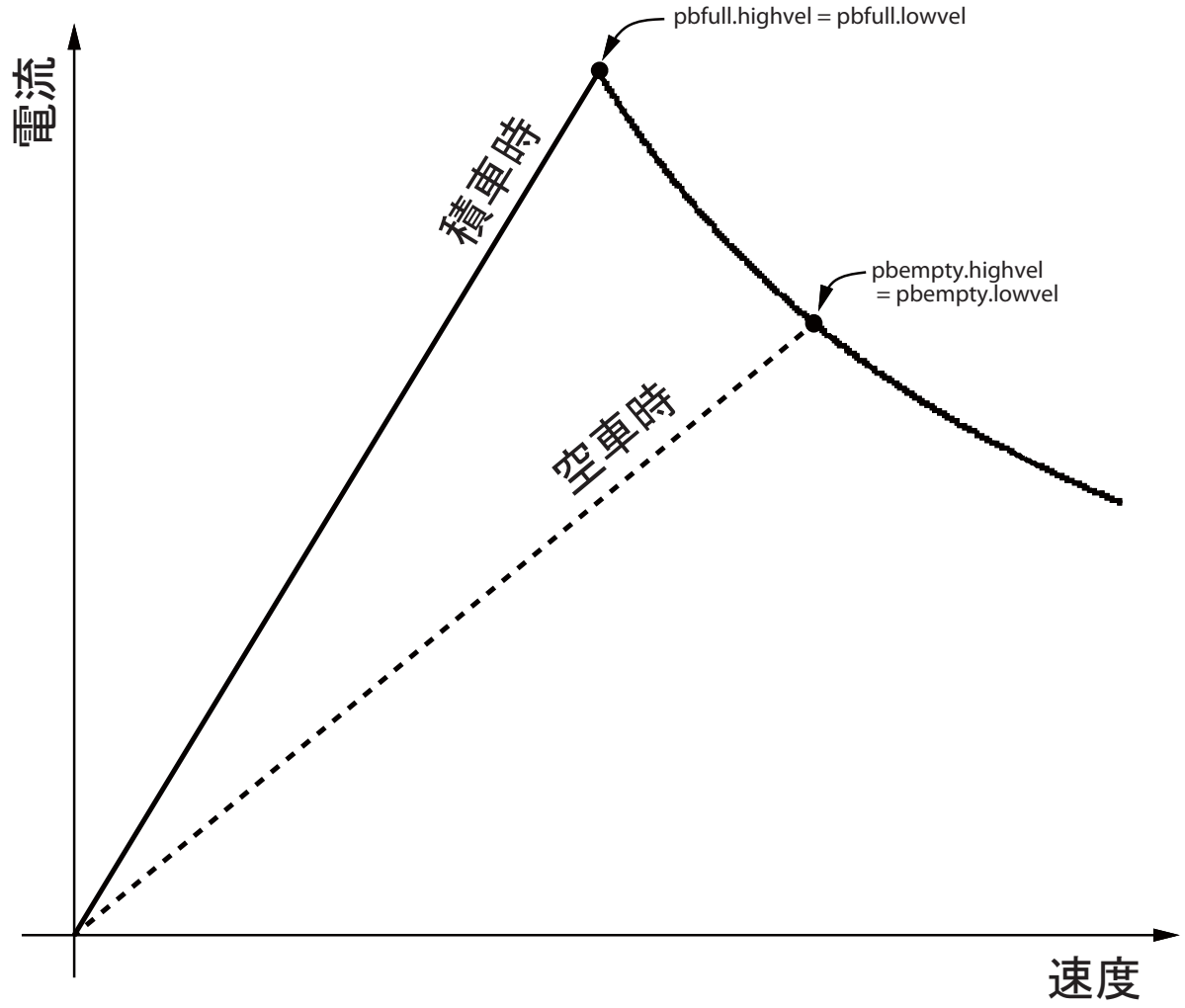


図 **F**.3: 線形補間が適用できないケースとして想定されるもの

フリーラン領域にあっては，引張力は速度の自乗に反比例することが知られている。そこで，混雑率ゼロのときの引張力および定トルク／フリーランモードの境界速度を T_e, v_e，混雑率 C_m のときのそれらを T_m, v_m としよう。このとき，データが正確に

$$T_e v_e^2 = T_m v_m^2 \tag{F.9}$$

と与えられることはないと考えられるので，Tv^2 が混雑率によって線形に変化すると仮定する。このとき，求められた Tv^2 の値を K_{Tvv} としよう。こうすれば，モード境界速度は

$$v = \sqrt{\frac{K_{Tvv}}{T}} \tag{F.10}$$

として容易に求められる。

一方，電流についてはフリーラン領域の電流が速度に反比例することを利用する。混雑率ゼロおよび C_m のときのモード境界速度（それぞれ v_e, v_m）における電流（最大電流）をそれぞれ I_e, I_m とすれば

$$I_e v_e = I_m v_m \tag{F.11}$$

とは正確にならないことから，Iv が混雑率によって線形に変化するものと仮定する。こうして求めた Iv

を K_{Iv} とすれば，最大電流は

$$I = \frac{K_{Iv}}{v} \tag{F.12}$$

と求められる。最大電流は電圧によるモード境界速度の変化にかかわらず一定である。

線形補間でよい混雑率の領域とこの新しい補間によらなければならない混雑率の領域の区分は，ある混雑率 C_m を境にそれより低いところがこの新しい補間方式，それより高いところが従来の線形補間領域と簡単に決められるはずである。そこで，RTSS ver.2.1.0 以降では混雑率に関して3点のデータを与える3点指示方式を採用した。

⟨**F.2.1.2**⟩ **プログラム** 'enumerat.hh' にはこの補間法と従来の線形補間とのどちらを使うかを指定するスイッチが enum で定義される。pbchar_interpolation がそれで，Linear が線形，Inv_mode13 が新しい形式だ。

しかし，どうやら補間方式をスイッチで指定する必要はなさそうである。これは，3つのデータのうちたとえば1つめが空車，2つめが中間，3つめが満車としたとき，1つめと2つめの間が Inv_mode13，2つめと3つめの間が Linear であると考えればよいからだ。

'train.hh' では powerdata, brakedata というふたつの構造体を定義しており，それぞれが力行時の性能，回生時の性能を示す指標を格納してある。どちらも，pbchar 形式のデータ（これが混雑率1点分に対応するデータを格納する構造体）を3つ pbempty, pbmid, pbfull として格納している。pbempty と pbmid の間が Inv_mode13，pbmid と pbfull の間が Linear である。

注意すべきは，データを与える際に pbempty と pbmid の間が有効（つまり，境界混雑率 C_m がゼロ）の場合には，pbempty にしろ pbmid にしろ，二つのモード境界速度 highvel と lowvel は等しい値になることに注意すべきだ。なぜならこの補間方式を利用する際の前提が定トルク領域とフリーラン領域がとなりあい，定パワー領域がないことだからだ。

これらの構造体に格納されたデータは train クラスの関数が利用するが，利用する関数はいろいろなファイルに分散して存在する。利用のしかたは力行／回生問わずほとんど同一だ。共通のルーチンに入れて集中化した方が保守がしやすそうだし，同じ計算を何度もしなくて済むので計算のスピードアップにもつながろう。というわけで，混雑率をセットするために用いる train::setconges() 関数（引数は double）のなかで，混雑率をセットしたのち⟨**F**.1.2⟩にて述べたいくつかの変数，vlow_p，vhigh_p，ampmax_p，vlow_b，vhigh_b，ampmax_b などを計算する関数を用意している。この数字は混雑率を仮定すれば一意に定まる定数であるため，この関数の中で計算すれば他で計算する必要はない。

このほか，応荷重補間に関係する記述がある電気的特性関数は以下の通り：

- train::tracf() 関数。引数は double, double, double で，順に列車速度，パンタ点電圧，フルノッチ比（⟨**F**.2.2⟩に後述）。牽引力を定める関数。
- train::nr_constv() 関数。フルノッチ比を決める関数の定速走行モード用。
- train::ttvi_power() 関数。媒介変数 θ から電圧・電流とその媒介変数による微分値を求めるルーチン。モード境界速度の利用は前の二つと同一だ。
- train::cgdef() 関数。引数は double，double。現在の混雑率から cg という定数を決める。この定数は次の関数で使われる。従来の線形補間のための関数であり，最終的には setconges() 関数において利用されるサブルーチンだ。
- train::cgcal() 関数。引数は double，double，double。前述の cgdef 関数で決めた cg なる定数を利用し，さまざまな線形補間計算を行なう。これも setconges() において利用されるサブルーチンだ。
- train::velcal_mid() 関数。引数は double × 6。cgdef 関数で得た定数 cg および，境界混雑率 C_m 時および空車時の引張力・モード境界速度，現在の引張力を引数に与える。非線形補間方式に

おいてモード境界速度を返す関数だが，このモード境界速度は与えるものも得られるものも2つの速度 highvel と lowvel が同一の値であることに注意しよう。プログラムもそのように組み立てられる。これもサブルーチン。

- train::ampcal_mid() 関数。引数は double × 6。非線形補間方式において最大電流を返すサブルーチン。なお，定トルク領域におけるトルクを返す関数は用意しなかったが，これはトルクのみは混雑率からの線形補間で求めうるものだからだ。

⟨F.2.2⟩ フルノッチ比

実際の電車では，ある速度において列車がとり得る牽引力を比較的少数の不連続な値のみとすることが多い（最近のインバータ電車などでもそうなっているものがある）。しかし，インバータ電車であれば望み通りの牽引力を常に発揮することができるし，そうであるべきだと思われる。このため，以下で定義するような「フルノッチ比」を考え，プログラムの各所で利用することにしている。

⟨F.2.2.1⟩ 定義[39] まず，インバータ制御またはチョッパ制御（界磁チョッパを除く）の電気車であれば，出しうる最大の牽引力より小さければどんな値でも自由に好きな牽引力を出せるものと考えよう。また，速度一定と仮定した場合，パンタ点から見た効率が牽引力の値によらず一定と仮定する。このとき，列車速度 v のときに，列車がその速度における最大牽引力 $T_{max}(v)$ を発揮している（これを「フルノッチで加速している」と呼ぶ）ときの電流を $I_{max}(v)$ とすれば，牽引力 T を出しているときの電流 I は

$$I = \left(\frac{T}{T_{max}}\right) \cdot I_{max} \tag{F.13}$$

で与えられる。ここで，$T/T_{max} = r$ とすれば，

$$T = r \cdot T_{max} \tag{F.14}$$

$$I = r \cdot I_{max} \tag{F.15}$$

と与えられる。r は牽引力および電流のフルノッチ時に対する比率を表すことになるので，これをフルノッチ比と定義する。なお，フルノッチ比は力行時には 0 〜 1，回生時は 0 〜 −1 の範囲の値をとるものとし，回生側も上と同様に定義する。

従来の走行パターンにおいては，フルノッチ比は力行時には 1，惰行時には 0，最大減速度でのブレーキ時には −1 となる。

このような「フルノッチ比」を用いて本章の以下の説明が行われる。

⟨F.2.3⟩ フルノッチ比の概念の拡張？

特に，ブレーキ時の振舞いを記述するには，フルノッチ比というのを電気的な部分と機械的な部分とに分ける必要を感じている。これは，電気的なブレーキ力（これはある効率で架線に返るから無駄とはいえない。仮に逆相ブレーキのような利用法をしたとしても，ブレーキシュー摩耗はないし，制御性も機械的ブレーキよりははるかによい）と，機械的なブレーキ力（これは効率ゼロである上ブレーキシューの摩耗まで招くから損は2重になる）をそれぞれ最適に制御しようなどという目的のためには，ブレーキ力の総和だけを1つの「フルノッチ比」で表現するのでは得策ではないと考えられるからだ。

「フルノッチ比」の考え方自体は，関数の与え方を工夫することでいろいろな車種に適用可能だろう。ただ，単純なシミュレーションをやっている分にはよいものの,「フルノッチ比」を用いて何かいいことをしようといった場合，特に抵抗制御電気車の場合はどの範囲まで制御可能かをいつもチェックしていなければならず，困難が伴うことが予想される。抵抗制御車の場合はノッチ戻し[2)]が不可だとかいった問題があるため，いずれにしても精密なモデリングは困難だろうとは思う。もっとも，いまさら抵抗制御電車を

[2)] ノッチ戻しとは，列車が加速中にいったんあげた「ノッチ」を低い値に戻すことをいう。こうするというまでもなく加速力が低下する。これができない電車では，通常連続走行可能なノッチを設けておき，加速を弱めたい場合にはそのノッチで「止める」こと

仮定したシミュレーションはないだろうという希望的な観測ができる時代になったのは，シミュレーションが楽だというようなつまらぬ観点からだけでなく，大変喜ばしいことだといってよいだろう。

⟨F.3⟩　回生絞り込みモデルと系の最高電圧

回生絞り込み特性は，シミュレーション結果としての回生率や回生電力量に大きく影響するパラメータとなる。これをプログラムがどうモデル化しているかについて，やや詳細に述べる。

⟨F.3.1⟩　2種の特性モデルと絞り込み対象の電流

回生絞り込み特性は通常図7.22（47ページ）のように与えられる。同図の(1)および(2)のいずれのケースも実際に存在しているそうだ。

(1)のケースでは，回生電流の絞り込みは次のように行われる。当該混雑率における最大回生電流をI_max（ただし$I_{empty} \leq I_{max} \leq I_{full}$，パンタ点電圧を$V_P$，絞り込みを行う前の回生電流を$I_R$とするとき

$$I_R \geq I_S \equiv I_{max} - I_{full} \times \frac{V_P - V_s}{V_e - V_s} \tag{F.16}$$

であるとき，I_RをI_Sまで（$I_S < 0$ならばゼロまで）絞り込む，というものだ。

一方(2)のケースでは，

$$I_R \geq I_S \equiv I_{full} \times \frac{V_e - V_P}{V_e - V_s} \tag{F.17}$$

であるときに，同様にI_RをI_Sまで（$I_S < 0$ならばゼロまで）絞り込む。

回生絞り込みを行う理由というのは架線電圧の異常な上昇を防ぐためである。従って，同一の電圧で満車時より空車時の電流が小さくなければならない合理的な理由はない。ゆえに図7.22(2)の回生絞り込み特性が実現できないが，同図(1)が実現可能ということはないだろう。そうなれば，(1)より(2)としたほうが空車時の回生能力をより有効に使うことができるといえる。(1)を採用しているケースでは，車両側の特性を変えることによる回生特性改善の余地が少なからず残っている，といういい方もできる。

RTSSは当初(2)のみをサポートしていた。これは曽根教授の考え方が反映されているもので，こちらがいいという過去の検討における結論を踏まえた割り切りである。いかにも曽根研のシミュレーションプログラムらしい特徴だったと思う。しかし，文献[73, 74]での検討に用いるさい実際の条件に合わせるべく，ver.2.1.3より(1)もサポートするように変更した。

ところで，この絞り込み特性の縦軸の電流はどの電流のことをいうのだろう。これにも考え方が2つあった。図7.23（48ページ）を見ていただきたい。考えられるのは「モータ電流」と「主回路回生電流」とである。

モータ電流はモータの消費している電力とはあまり関係がなく，むしろ電車の引張力（すなわちモータのトルク）との関係が強い。大きなトルクを出している低速では，モータ電流は大きいが，モータの消費する電力は小さいため，インバータの直流側に返る主回路回生電流は小さくなる。したがって，式(**F**.16)または(**F**.17)でI_Rほかが主回路回生電流であるとするならば，V_Pが絞り込み開始電圧V_sを越えていてもI_Rが小さいため絞り込みが行なわれない可能性が高くなる。このように，主回路回生電流が小さくなる中低速域では，主回路回生電流を見て絞り込む方式では絞り込みなしとなる場合でも，モータ電流を見て絞り込む方式では絞り込みを行うようなケースが出てくる。

混雑率の場合と同様に，同一の電圧で高速時より低速時の電流が小さくなければならない合理的な理由はない。したがって，回生絞り込みはモータ電流ではなく主回路回生電流に対して行った方が電気車の電力回生能力がより生かされる。実際の電車は制御の容易さからモータ電流を見る方式が主流のようだが，図7.22の特性(1)(2)の比較と同様，この方式では回生特性改善の余地を残していることになる。

をする。自動車のアクセルのように自在に力を加減することは，運転手の操作ではできないことになっているものが多い。直流の抵抗制御車でこれができない理由は，主としてカム軸で制御しているスイッチが接触器であり，電流を入れることはできても遮断できないことによる。最近のインバータ電車でも，ノッチ戻しができる領域を何らかの形で限っているものが多い。

RTSS は当初主回路回生電流を見る方式のみをサポートしていた。このあたりは曽根教授の考え方が反映されているが，絞り込み特性(2)だけをサポートして(1)は切り捨てるくらいだからこちらをこのようにするのはむしろ当然といえる。しかし，文献[74]での検討に用いるさい実際の条件に合わせるべく，ver.2.1.4 よりモータ電流を見る方式もサポートするように変更した。

⟨F.3.2⟩ 回生絞り込みと失効との区別

パンタ点電圧が高くなると絞り込みを始め，ある電圧より上では回生電流はまったく流れない，というような仕組みになっている。実際の電車では，あまり回生電流が小さくなった場合には主回路をオフにして，回生を切ってしまう（これが一般にいう回生失効の状態だ)。

ちなみに，ブレーキ時は回生ブレーキ（電気的ブレーキ）力と空気ブレーキ（機械的ブレーキ）力の和は等しくなるように制御されるため，列車としての減速度がマクロに見て変化するようなことはないが，実際には空気ブレーキ系の立ち上がりの遅さなどがあるため，とくに回生失効の状態になった際の切り変わり時にはショックがあったり，ATO 運転線区では定点停止に支障を来たしたりすることがあるようだ。

そして，いったんこうして主回路を切ってしまうと，その後空気ブレーキのみで電車は止まることになり，止まるまでの間にパンタ点電圧が下がっても回生ブレーキが再びかかることはないのが普通だ。もっとも，なかには，回生失効後20秒経つと再び回生にトライするという「飛びつき機能」を持っている電車がある。それに，インバータ制御電気車ならば，主回路を完全に開くのではなく，再び回生可能な状態になるまで待機するようなことも可能であるという。将来はそのような制御方法が一般化することになるだろう。

このシミュレータでは当初この回生ブレーキが「再びかからない」という条件のシミュレーションはできなかった。RTSS ver.2.1.3 以降，この条件のシミュレーションも可能となっている。この条件のシミュレーションにおいては，ある列車の主回路回生電流がある電流値より小さくなった場合に，それ以後の回生を禁止するフラグを立てる。禁止フラグが解除されるのは，フルノッチ比が非負となったばあい，もしくは回生オフ速度を下回った場合のいずれかである。

⟨F.4⟩ 補機負荷のモデル

列車補機負荷のモデリングは，大きく分けてパワー一定モデリングと電流一定モデリングとがある。従来はこの補機モデルが入っていなかったが，これは明らかに単にサボっていただけであった。媒介変数表示だから関数を追加するのはそう難しいことではないのである。

RTSS ver.2.1.1 から電流一定モデリングでサポートした。しかし，これでは評価量にかなりまずい影響があるため（補機パワーが電圧増加に伴って増大してしまう)，パワー一定モデルを ver.2.1.3 からサポートした。両方をデータファイルで選択できるようにしてある。

プログラミング自体は簡単である。‘train.hh’ というファイルで定義される train クラスには，teta_vi() 関数があり，‘train.cc’ にてコードが記述されている。ここで a （電流）および di（電流の媒介変数による微分値）を計算しているが，これを計算し終ったところに ttvi_auxcurr() なる関数を挿入しただけだ。この関数のコードは ‘trteta.cc’ にあるが，補機電流の値，およびその媒介変数による微分値を，a および di に加える操作をしている程度の簡単な関数だ。

特に注意すべきは，列車が消費する電流はマイナスの値をとる約束になっていることだ。また，回生失効率の計算に影響がないように，主回路分と補機込みとで電力計算を分ける必要が生じたため，この関数が主回路の電流の演算の後におかれていることを利用し，補機分を加える前の a の値を shibo という変数にコピーして保存している。

G

列車の運動モデル

この章では，列車の運動モデルを取り扱う。当然のことながら電気的モデルと運動モデルとは密接な関係があるので，なかなかこのように「きれい」に分割することは難しい。本章の読者は折りにふれて**付録 (III) F**を参照することになるだろう。

⟨G.1⟩　運動方程式，単位系

列車運動シミュレーション部は，列車を質点と見なし，列車の引張力・ブレーキ力・走行抵抗・勾配抵抗などをもとに列車の運動方程式をたて，それを数値的に解くものである。列車の運動方程式は次式で表される[19]。

$$W_k \cdot (1+\gamma) \cdot \frac{dV_k}{dt} = F_k - B_k - R_k - G_k \qquad (\mathbf{G}.1)$$

ただし，W_k : k 番目の列車の総質量 [t]
γ : 回転部分による補正係数
V_k : k 番目の列車の速度 [m/s]
F_k : k 番目の列車の引張力 [kN]
B_k : k 番目の列車のブレーキ力 [kN]
R_k : k 番目の列車の走行抵抗 [kN]
G_k : k 番目の列車の勾配抵抗 [kN]

現在の列車の位置・速度が与えられると，Δt 時間後の列車の位置・速度は式 (**G**.1) を解くことによって簡単に求められる。そこで，現在の列車群の配置およびすべての列車の速度が与えられているとき，この方程式を列車ごとに解いて行けば，Δt時間後の列車群の配置および各列車の速度は簡単に求められる。

⟨G.2⟩　列車の状態と状態遷移則

列車は,「力行」「定速走行」「惰行」「ブレーキ」などの状態を状況に応じて切替えながら走行する。また，同じ「力行状態」でも弱い力行，強い力行などの区別（ノッチの選択）が運転手の動作を完全に模擬することは難しいため，適当な仮定を置いてプログラムが組まれるのが普通だ。この部分は結果に少なからぬ影響を与える部分であり，またこのプログラムのもっとも特徴的な部分でもあるので，ここでやや詳細に説明を加えようと思う。

⟨G.2.1⟩　駅間走行時分一定化技術

この種のプログラムとしては，従来からいろいろな種類のものが書かれてきているし，現にいろいろな場所でいろいろなプログラムが用いられている。だが，従来のプログラムは，列車が加速をやめる位置や速度をデータとして与えて列車の駅間走行パターンを指定している。この方法でも列車の性能が一定であれば問題はないが，現実の列車はパンタ点の電圧によって性能が変化する。これをモデルに取り込めば必然的に列車の駅間走行時分が条件通りにならない現象が起きる。

饋電特性の中でももっとも重要な各種エネルギーの評価にとって，駅間走行時分は重要なパラメータである。駅間走行時分の狂いはシミュレーション結果の信頼度を下げてしまうことにつながる。これに対処するため，RTSS では駅間走行時分を高精度に一定化する列車運動シミュレーション技術を開発しインプリメントしてある。この点が，RTSS が他の多くの直流饋電システムシミュレータと異なる著しい特徴となっている。⟨A.1.1⟩に述べたように，このインプリメンテーション部分が RTSS の「いちばん古い部分」でもある。

パンタ点電圧が変動し，それに伴って列車性能も変化する条件のもとで，駅間走行時分を高精度に一定化するためには，加速をやめる位置・速度をそのときの条件に応じて求めればよい。パンタ点電圧は前もって予測することが難しいが，列車が加速していないとき（惰行・ブレーキ）には列車の運動はパンタ点電圧と無関係である。

惰行状態における運動がパンタ点電圧と無関係というのは理解しやすいと思われるが，ブレーキ時の列車運動がパンタ点電圧と無関係というのはなぜいえるのだろうか。これは，普通の電気車は電空併用演算をしており，電気ブレーキ力の不足時には空気ブレーキによる補足を自動的に行い，電気ブレーキと空気ブレーキのブレーキ力の和が一定となるように制御しているからである。この演算および制御が正確に行われている限りにおいて，マクロにみた場合の列車の減速度は一定に保たれるのである。

このことを利用すれば，加速中のある瞬間に列車が加速をやめ，あとは目標地点まで加速せずにゆくと仮定した場合の，目標地点までの到着時刻は計算できる。平坦な路線で速度制限などがなく，列車の駅間走行パターンが《力行→惰行→ブレーキ→停止》となる場合，列車は力行中この時間を毎回推定し，推定値が所定の時間より短くなったところで力行をやめればよい。このような，比較的簡単なアルゴリズムで，駅間走行時分一定化シミュレーションが実現できる。

もっとも，このように書くと簡単そうではあるが，実際には列車の走行時間を推定するのには列車の運動方程式を解く必要がある。これをするのにいちいち数値積分をしていたのでは時間がかかりすぎるので，近似計算ルーチンを作っておき，ある精度まではこの近似で追い込むようにしている。

駅間走行時分合わせに失敗した場合，列車の遅れ分は次の駅間にそのまま持ち越されるようにプログラムしている。一方，列車の進み分は停車時分に加算される。シミュレーション条件によっては，⟨G.2.2⟩に述べるように駅間走行時分合わせに失敗することもある。最小時間刻み幅程度の誤差はつねに起きる可能性がある。

⟨G.2.2⟩　一般的な列車状態と状態遷移則

⟨G.2.1⟩で述べた駅間走行時分一定化技術は，列車の駅間走行パターンとして「駅停車」「力行」「惰行」「ブレーキ」の 4 つのみを考え，しかも《力行→惰行→ブレーキ→停止》のような順に現れる単純なパターンのみを仮定している。しかし，局所的な速度制限や勾配が存在するなどの実際の路線の条件を考慮すると，このような単純なパターンだけで議論ができるわけではない。そこで，より現実的なモデルへの適用のためには，走行パターンになんらかの仮定をおく必要が出てくる。

このプログラムでは通常の列車の状態として前記の 4 つのほか「定速走行」の 5 つを考えた。「定速走行」状態は弱い力行または弱いブレーキのどちらにもなり得る状態である。ここでは列車の速度を一定に保とうとするモードであると簡単に理解していただこう。また，駅間の前半は力行領域，後半は惰行領域とし，力行状態，または弱い力行で速度を維持しなければならない定速走行状態は，力行領域以外では現

表 G.1: 一般的なシミュレーションモデルにおける5列車状態とその遷移条件

状態		条件
遷移前	遷移後	
停車	力行	停車時間終了
力行	惰行	ノッチオフ後次駅到着までの推定時間が定時以内(力行領域終了)
	定速走行	制限速度を超過
	ブレーキ	オフブレーキの条件（ブレーキオフ地点およびその地点での列車指定速度（駅停車の場合は停止目標位置およびゼロ速度）が与えられたとき，そこから所定の列車減速度を用いて逆算してちょうどよい地点に到達）
定速走行	惰行	力行領域にあり，かつノッチオフ後次駅到着までの推定時間が定時以内(力行領域終了)
	惰行	惰行領域にあり，かつ勾配を含んだ走行抵抗が正
	力行	力行領域にあり，かつ制限速度を下回った
	ブレーキ	オフブレーキの条件
ブレーキ	定速走行	制限速度を超過し，勾配を含んだ走行抵抗が負
	惰行	ブレーキオフ速度に到達
	停車	速度が0に到達
惰行	ブレーキ	オフブレーキの条件
	力行	ノッチアップ（速度制限解除）地点に到達

れないようにした。力行領域は，発駅側から，上記の方法によって判定したノッチオフ地点までとしている。ノッチオフ位置決定の計算の際，列車は必要な場所では定速走行を必ず維持できるものとしている。この5状態についての状態遷移則は表 G.1に示した。

なお，状態遷移は遅延なしに行われるほか，ジャークコントロールは考慮していない。実際の車両では数秒オーダの遅延が存在するほか，ジャークコントロールを行うことになっているものが多い（特にインバータ制御のような新しい電車はなおさらだ)。これらの条件を無視することは駅間走行時分で2〜3秒早く走ったことと等価になる可能性が強いため，列車消費エネルギーの低下となって現れるものと考えられる。

このルールは《力行→惰行→ブレーキ→停止》のパターンを基本とし，それを拡張したものになっている。状態遷移のようすを，図 G.1・G.2のふたつのケースについてこれに当てはめて考えてみよう。

1. 一定の比較的低い速度制限がかかっている場合（図 G.1)。駅を出た(A)列車は，B点まで力行し，速度制限にかかって定速走行状態に移行する。駅間走行時分一定化の条件によってCで惰行に移り，速度がだんだんおちる。D点で「オフブレーキの条件」を満たすのでブレーキ状態に移りE点で停車。
2. 発駅および着駅に近いほど低くなる形状の速度制限がかかっている場合（図 G.2)。駅を出た(A)列車は，B点まで力行し，速度制限にかかって定速走行状態に移行する。C点で速度制限が解除にな

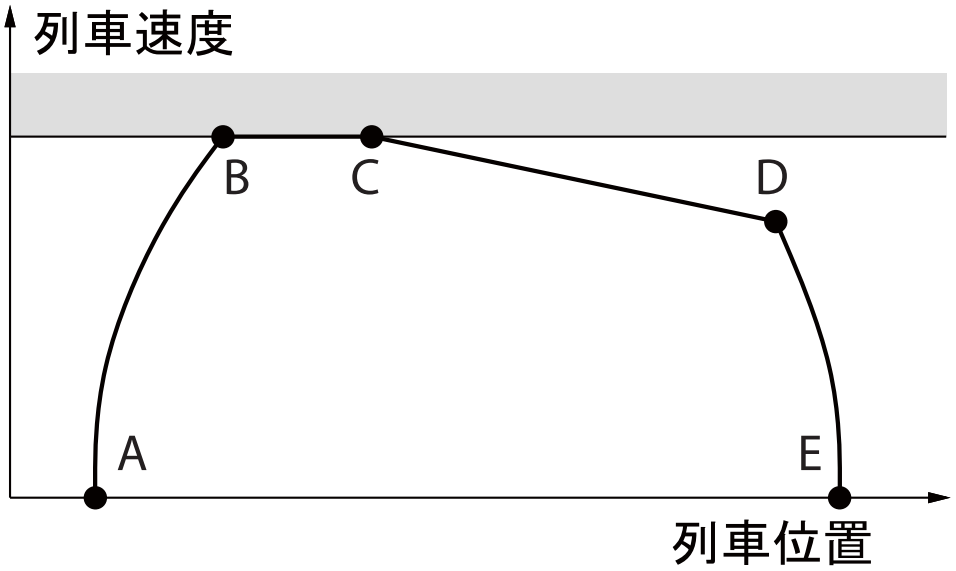


図 **G**.1: 走行パターン例 (1)

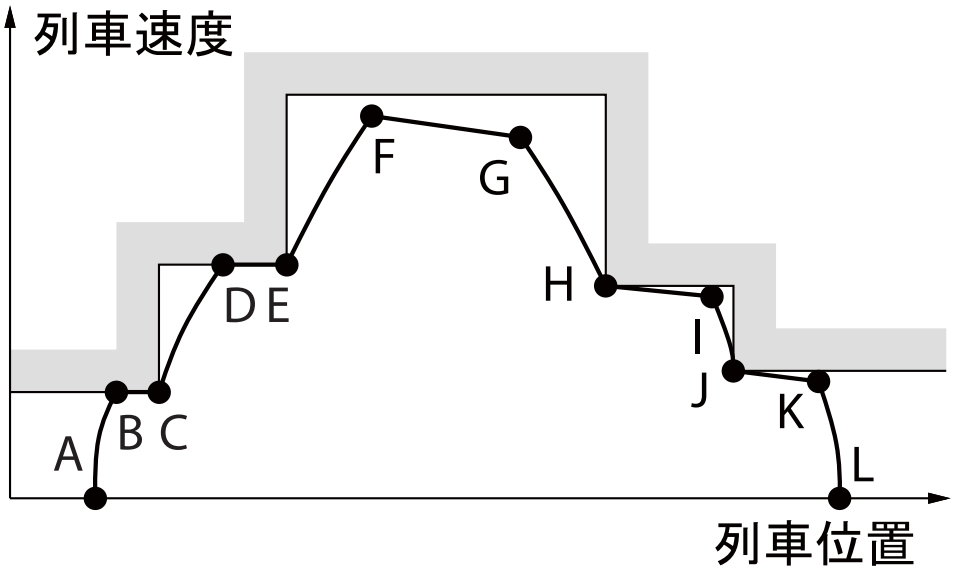


図 **G**.2: 走行パターン例 (2)

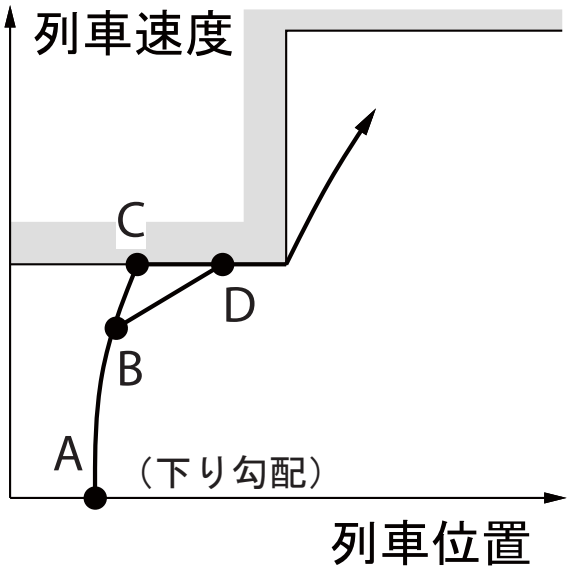


図 G.3: よくない走行パターンになる例 (1)

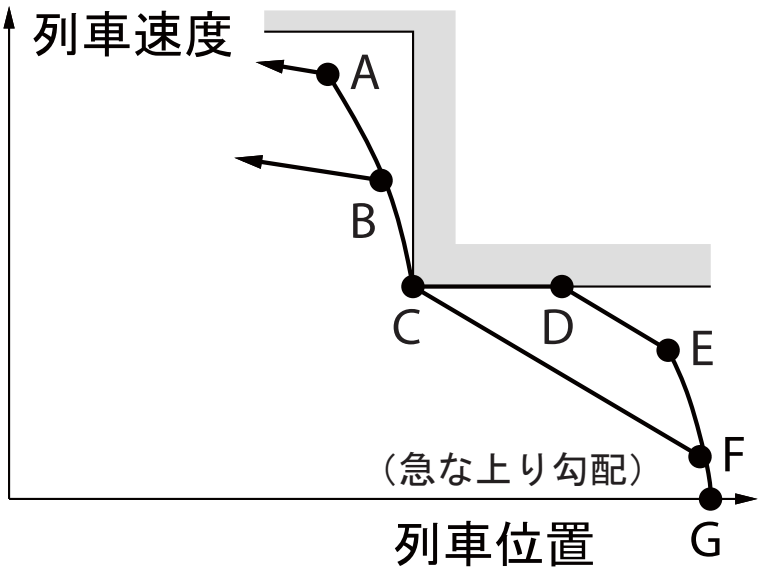


図 G.4: よくない走行パターンになる例 (2)

り，まだ力行領域にあるため再力行。D 点でふたたび速度制限にかかるが，E 点で同様に再力行する。駅間走行時分一定化の条件によって F 点で力行から惰行に移ると同時に，ここで力行領域が終了する。その後，G 点で「オフブレーキの条件」を満たすためブレーキに入り，H 点までに速度制限より低い速度まで減速する。力行領域は F 点で終了しているので，H 点から惰行に入る（力行領域ならば定速走行となる）。I 点，J 点はそれぞれ G 点，H 点と同様。J 点から同様に K 点まで惰行で走って再びブレーキをかけ，L 点で停車する。

これらのケースのように「行儀のよい」線路条件のもとであれば，このルールで定性的に見てノーマルで，比較的最適にも近いパターンが生成できることがわかる。しかし，次のようなケースには問題が出て

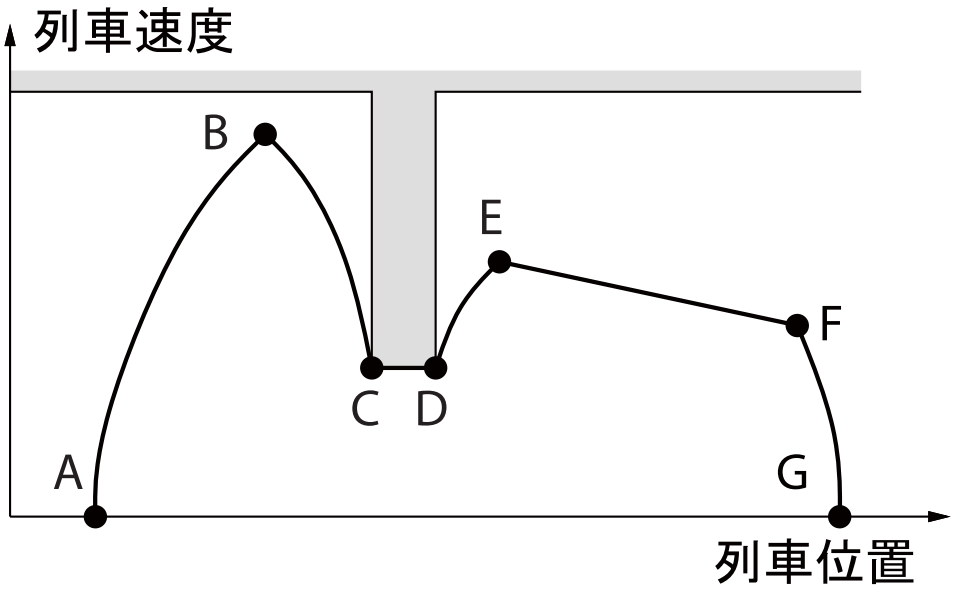


図 **G**.5: よくない走行パターンになる例 (3)

くることが考えられる。

- 駅間起点付近に制限速度が低く下り勾配の急な区間がある場合（図 **G**.3）。このルールでは，区間前半（力行領域）は速度制限にかかるまでは力行を続けることになるため，この例ではA点を発車した列車はC点まで力行して定速走行状態に移る。しかし，下り勾配のため定速走行状態は弱いブレーキであるから，このように走ると力行の直後にブレーキをかけることになり，好ましくない。このパターンよりは，B点で惰行に入りD点まで勾配によって加速するパターンとし，速度制限が解除された地点の先でこの分の遅れを取り戻すほうがエネルギー的にも得になるはずである。
- 駅間終点付近に制限速度が低く登り勾配の急な区間がある場合（図 **G**.4）。このルールでは区間後半（惰行領域）に力行または速度維持のための定速走行ができないため，速度制限のためC点まで減速した列車はF点までに大きく速度を下げざるを得ない。この結果速度制限の前の区間でのブレーキ初速をA点のように高くせざるを得ない。このような場合は，前の区間のブレーキ初速をB点のように下げる代わりにC点からD点まで定速走行モードを入れて速度を維持したほうがエネルギー的に得になるケースがある。さらに，このような場合，低い速度での近似計算がうまくゆかないため駅間走行時分が精密に一定化できない場合も出てくる。
- 駅中間に制限速度が低い場所がある場合（図 **G**.5）。基本的に力行領域は区間前半に，惰行領域は区間後半になるわけだから，図のように前半ではオフブレーキ運転となる。E点でノッチオフとなり力行領域が終了するまでは速度制限より低ければとにかく力行するルールのためにこうなってしまう。極端な場合，速度制限箇所の先ではまったく力行しないこともある。

これらは，主に列車力行エネルギーの増大として結果に現れるはずである。また，力行を行う場所が変化することから，回生ブレーキの効き具合にも影響がある。

これらの「悪い」条件に対応するには，駅間をいくつかの小区間にさらに細分すればよい。例えば図 **G**.5はA点（発駅）からC点までおよびC点からG点（着駅）までの2区間に分割し，それぞれの区間を走行する時間をランカーブより与えるようにする。それぞれの区間についてはこのルールを適用することが可能であり，駅間走行時分一定化の基本的な考え方も変更する必要がない。ただし，この議論では，ラ

ンカーブにおいて各区間ごとの時間の配分はある程度最適化されたものを与えなければならない。すなわちランカーブがあらかじめ最適化されていることを前提にしている。

RTSS は，ver.2.2 より駅間分割をサポートした。また，ランカーブの最適化機能も，どのような形にするかおよびどのような最適化手法によるかは別として導入する考えだが，これは ver.3 以降でということになるだろう。

⟨G.3⟩ 回転部等価質量係数

列車の運動方程式を考える場合，質量だけでなく回転部等価質量も与えなければならない。通常回転部等価質量は車両の質量が $(1 + w_r)$ 倍されたものとして考慮している。

RTSS ver.2.1.0 までは荷重分も含めて $(1 + w_r)$ 倍するように考えていたが，これでは正確ではない。このため，RTSS ver.2.1.1 以降では空車質量のみを $(1 + w_r)$ 倍するように改めた。

これを実現するために，`trainmotion` クラスに新たに `eqrot()` 関数を設けた。この関数の中では空車質量 W_{empty} および荷重を含めた現在の列車質量 W_{curr} を求め，

$$w_R = \frac{w_r \times W_{empty}}{W_{curr}} \tag{G.2}$$

で与えられる w_R を返すようにしてある。従来直接 w_r（変数名は `wrotate`）を参照していた場所はすべてこの `eqrot()` 関数の返り値を利用するように改めてある。

H

列車ダイヤのモデル

列車ダイヤのモデルは，列車の路線全体にわたる走らせ方を規定するものであり，diapattern クラスが一つのダイヤパターンを管理する。diapattern クラスは実は nextsta クラスの配列である。さらに nextsta クラスは gradcrv クラスの派生クラスとなっている。

列車ダイヤ，ないしは列車の「走り方」を表現する最小のデータ単位が次駅データで，それを表現するためのクラスが nextsta クラスである。全体を駅間ごとに切りとって管理し，列車がある駅からその次の駅（場合によっては駅間に特定の地点を設け，その地点までと限定することもある）までの走行に必要な情報を格納する。

勾配・曲線情報などのデータもこの次駅データクラス nextsta によって管理される。ファイルリード時には独立させ，別に gradcrv クラスの配列をつくることとし，データを記述するファイルも別にしてある。しかし，ファイルリードが終われば，nextsta クラスは必要な勾配・曲線情報をも自分で保持する。

次駅までの走行に関するデータがすべて nextsta クラスによって管理されるのだから，nextsta オブジェクトを必要数だけ並べれば列車の行路表が描かれたことになる。これが diapattern クラスである。場合によっては（というか高木が今までにシミュレーションしたほぼすべてのケース）このダイヤパターンの上に等時隔で複数列車を張り付けることもできるようになっている。

なお，ダイヤパターンデータとは別に，信号系のシミュレーションのためのモデルを構築中である。

⟨H.1⟩　勾配・曲線・速度制限データクラス gradcrv

勾配・曲線・制限速度は，路線を適当な区間に分けたとき，その区間内では一定である。となりの区間と連続である必要はない。すなわち，緩和曲線や縦曲線は考慮しない。そこで，路線をいくつもの区間に分けて,「区間始点」「区間終点」「勾配」「曲線」「制限速度」を一組としたデータを与えてやればよい。この一組のデータを gcvel データ と呼ぶ。このことは⟨J.2⟩にて述べた通りだが，このデータ1つを格納するのが gcvel 構造体である。その定義は ‘nextsta.hh’ にあるが，変数5つだけの簡単なものだ。

次駅データとの関連から，すなわち次駅データはある駅から次の駅までの走行に関するデータをまとめて保持するものだから，この gcvel 構造体データも駅間ごとに与える。こうすると，gcvel データは駅間ごとにいくつかが1組となる。この1組を1つの gradcrv データと称することも ⟨J.2⟩ にて述べたが，この gradcrv データを格納するのが gradcrv クラスである。この定義も ‘nextsta.hh’ にあるが，zeroreftable テンプレートクラスの機能を使ったため定義自体は簡単である。

⟨H.2⟩ 次駅データクラス nextsta

列車の次駅までの走行パターンを決めるためのデータをセットにしたのが次駅データクラス nextsta である。このクラスはgradcrv からの派生として定義されている。

次駅データとは，次のデータの集まりである:

- 現駅（または，データの対象となる区間の開始点）の位置
- 次駅（または，データの対象となる区間の終了点）の位置
- 次駅（または，データの対象となる区間の終了点）での列車速度
- 次駅（または，データの対象となる区間の終了点）までの所要時分
- 次駅での停車時分
- ノッチオフ速度（現在のプログラムでは意味をなさないが，0.0 を書き込む。将来必要が出てきた場合は有効にする計画である）
- 列車走行方向（通常キロ程が増加する方向が (+)1，反対は -1）
- 饋電線番号
- 勾配・曲線・制限速度データ（gradcrv データ）
- 次駅データの種類を表す記号
- 次駅データ切替地点（駅以外の地点で切替を行う場合に必要）
- 次の「次駅データ」へのポインタ

特に強調すべきこととして，次駅データは勾配・曲線・速度制限データを持っていることだ。つまり，ファイルリードが終われば，nextsta クラスは必要な勾配・曲線情報をも自分で保持する（該当するgradcrv クラスへのポインタとしてではなく，自分で持っている）。この方法はメモリ容量をいささか無駄使いするが，メモリ容量制限の多かったパソコンでの使用は考えていないので，現在までのところ目だった問題にはなっていない。

⟨H.3⟩ ダイヤパターンデータクラス diapattern

⟨H.3.1⟩ 基本

次駅データは，ある駅から次の駅までの走行に要するデータを集めたものである。次駅までの走行に関するデータがすべて nextsta クラスによって管理されるのだから，すでに ⟨C.6⟩（128ページ）にて述べたように，nextsta オブジェクトを必要数だけ並べれば列車の行路表が描かれたことになる。これがダイヤパターンデータと呼ばれるものの基本であり，diapattern クラスが管理する。次駅データは，ダイヤパターンデータの中で配列として与えられている。なお，RTSS がシミュレート可能な列車ダイヤは，完全に周期的なダイヤであり，ダイヤパターンデータクラスの設計にもそのことが前提として考えられている。

ダイヤパターンデータとは，詳しくは

- 次駅データの配列
- このダイヤパターン 1 周期の所要時間
- このダイヤパターンの中で走る列車の数
- パターン倍数
- 列車群の位相（最初の列車が最初の駅を出発する時刻）
- 列車の初期化用変数群

などからなるデータである。

列車が次駅データの有効な範囲を越えた場合は，次の次駅データに移る。この移行のときに，列車位置・列車が電力の供給を受ける饋電線の番号・列車の走行方向などのデータを，ある程度自由に変えることができる。また，次駅データ配列の最後のデータの次は，周期性の条件から，最初に戻る約束になっている。

ダイヤパターンデータにおいて与えられるダイヤは，必ずしも運用上「実現可能」である必要はない。例えば直線的な路線の起点から終点まで走る列車が5分に1本あるが，その他の列車は存在しないようなもの（この場合，起点および終点に無限大容量の車庫がないと実現しない）も，シミュレート可能である。

⟨H.3.2⟩ 複数のダイヤパターンデータが存在する場合

ダイヤパターンデータは通常複数存在する。次駅データの配列 $\boldsymbol{A}$, $\boldsymbol{B}$ があり，それぞれ $A_1, A_2, \cdots, A_n$ および $B_1, B_2, \cdots, B_m$ という要素を持たせる。このとき，次駅データ $A_1, A_2, \cdots, A_n$ の順に走る列車群と，$B_1, B_2, \cdots, B_m$ の順に走る列車群とがそれぞれ存在しうる。例えば，JR山手線のような完全な環状線にあっては，このような2つの次駅データが存在してはじめて，内・外回りの列車運動をデータとして与えることができる。また，昼間時の営団銀座線のように，普通の複線で普通列車1種類のみの平行ダイヤ，運転区間も1種類のみである場合は，ダイヤパターンデータは1つであるが，急行・普通の2種の列車があれば2つのダイヤパターンを用意しなければならないし，区間運転などが存在すればダイヤパターンはより多く必要になる。

このような場合，それぞれのダイヤパターンデータにおいて周期 T_{circ} を求めることを考える。このダイヤパターン1周期の所要時間（t_{whole} とする）およびこのダイヤパターンの中で走る列車の数（N_{cars} とする）のふたつのデータから，

$$T_{\mathrm{circ}} = \frac{t_{\mathrm{whole}}}{N_{\mathrm{cars}}} \tag{H.1}$$

となるはずだが，これがすべてのダイヤパターンに対して一定であるとは限らない。そこで，パターン倍数（N_{dptm} とする）という変数（整数）を用意しておき，すべてのパターンについて

$$T_{\mathrm{gcs}} \equiv \frac{t_{\mathrm{whole}}}{N_{\mathrm{cars}}} \cdot N_{\mathrm{dptm}} \tag{H.2}$$

で定義されるシミュレーションサイクル T_{gcs} が一定となるようにしてある。

⟨H.3.3⟩ 列車位相

ダイヤパターンがひとつしかない場合，等間隔に列車の配置を行うルールだけを持っていれば十分だが，複数ダイヤパターンがある場合にはこれだけでは済まず，ダイヤパターン相互の時間的な関係を規定する必要がある。これが列車位相である。その定義は図**H**.1に示す通りである。

ダイヤパターンごとに,「ダイヤパターン起点駅」が決まっている。ダイヤパターンクラスの持っている `nextsta` オブジェクトの配列の先頭がこれに当たるのが自然だろうから，RTSS でもそれに従ったインプリメンテーションとしてある。

そして，図**H**.1に示した通り，その起点駅を列車が出発する時刻が列車位相となる。例えば，図で起点駅Aはダイヤパターンの，起点駅Bはダイヤパターンの，起点駅Cはダイヤパターンの，それぞれ起点駅と定義されているものとしよう。このとき，ダイヤパターンの位相 $t_P(A)$ が 0 の場合，ダイヤパターンに属する列車は位相原点になる時刻（$t_{\mathrm{sim}} = 0$）に起点駅を発車する。ダイヤパターンには列車位相 $t_P(B)$ が与えられていたとするなら，列車は $t_{\mathrm{sim}} = t_P(B)$ なる時刻に発車する。についても同様だ。そして，複数列車が1つのダイヤパターン内に存在するなら，起点駅 D および位相 $t_P(D)$ を持つダイヤパターン D においては，T_{circ}（式(**H**.1)，161ページ）を用いれば $t_P(D)$, $t_P(D) + T_{\mathrm{circ}}$, $t_P(D) + 2T_{\mathrm{circ}}$, ... がD駅の列車出発時刻となる。

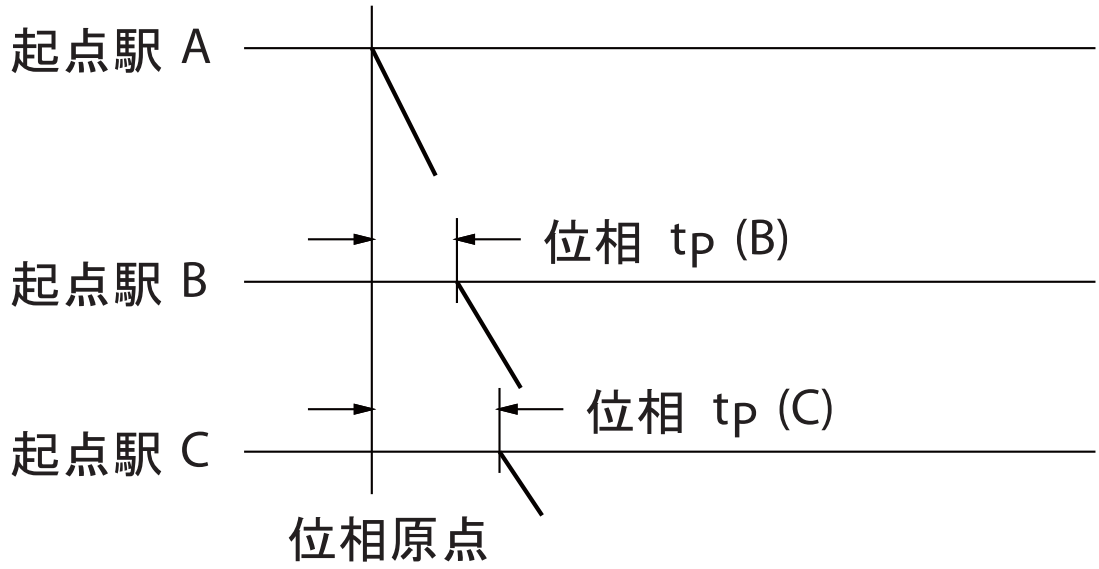


図 **H**.1: 列車位相の定義

⟨H.3.4⟩ `trainvar` コマンドでデータを「だます」

`trainvar` コマンドを用いて，データをいわば「だます」ような操作を行うことが可能だ。これは，周期的でないダイヤでのシミュレーションに使われる。

列車位相の定義は図 **H**.1で示した通りだ。ここで，周期的ダイヤを前提にして考えよう。同図で C 駅を位相 $t_P(C)$ にて出発する列車があるとしよう。ダイヤは周期的だというのが前提になっているから，C駅での停車時分 t_{stop} が $t_P(C)$ より短い時間で設定されていたならば，列車は $t_P(C) - t_{\mathrm{stop}}$ なる時刻にC駅に到着するように走行する初期設定となるはずである。

通常はこれで何ら困らないわけだが，周期的でないダイヤを取り扱おうとする場合に問題が生じる。周期的でないダイヤの場合，1ダイヤパターンあたり1列車を張り付けるようにしてデータを記述するが，C駅での停車時分 t_{stop} が $t_P(C)$ より短い時間で設定されていると困ることが起きる。すなわち，列車は時刻 0 から $t_P(C)$ まではC駅で動かずにいて欲しいのに，このままでは時刻 0 から $t_P(C) - t_{\mathrm{stop}}$ までの間はC駅に到着すべく動いてしまうからである。

本来は，ダイヤパターンデータ全体をきちんと書き直し，C駅での停車時分を長くとるのが本当だろうが，いわばデータを「だます」やり方として，`initialdeparturetime` フラグを利用する方法がある。このフラグを利用すると，`trainvar` コマンドに駅出発時刻を記入することができる。こうすることにより，データを「だまし」，列車が時刻 0 から $t_P(C) - t_{\mathrm{stop}}$ までの間に動くことだけを防止することができる。

なお，`trainvar` コマンドで駅出発時刻を指定する場合，`phase` コマンドで指定する位相のほうはゼロと与えるべきである。

付録 IV

RTSS マニュアル・使い方

I

コマンド行の引数

rtss プログラムの立ち上げは，次のようにすればよい。

```
rtss [options]
```

コマンドオプションは次のいずれか（複数指定可）である。

- -g勾配・曲線・制限速度データファイル名
- -n次駅データファイル名
- -s変電所特性データファイル名
- -p列車性能データファイル名
- -f饋電線データファイル名
- -t列車初期位置データファイル名
- -r出力データファイル名
- -dディレクトリ名

-d オプションは，それまでに指定したファイル名すべてに，同じディレクトリ名を付加するオプションである。これらのオプションは一つ一つ空白で区切る。しかし，一つのオプションは空白を空けずに続けて書く必要がある。つまり，-g, -n, ... などのあとには空白を入れてはいけない。

なお，これらのデータファイル名は，デフォルトではプレゼントワーキングディレクトリの次のような名称のファイルである:

勾配・曲線・制限速度データファイル‘gradcrv.dat’（G ファイルと略称）

次駅データファイル‘nextsta.dat’（N ファイルと略称）

変電所特性データファイル‘ssdata.dat’（S ファイルと略称）

列車性能データファイル‘pbdata.dat’（P ファイルと略称）

饋電線データファイル‘feeder.dat’（F ファイルと略称）

列車初期位置データファイル‘trdist.dat’（T ファイルと略称）

出力データファイル‘result.dat’（R ファイルと略称）

なお，-t オプションで指定する列車初期位置データファイルは，現在無効となっている。

これらのオプションはかなり複雑で，与えるのも面倒だろう。そこで，適切なシェルスクリプトのようなものを書いてやることをお勧めしたい。

J

データファイルの仕様と書き方

饋電システムのシミュレーションに必要なすべてのデータは，データファイルによって与えられなければならない。このファイルは5つにわかれている。それぞれについて説明しよう。

⟨J.1⟩ 基本

データファイルのひとつの**データ行**は，すべて**コマンド**とデータ（コマンドの種類によっては省略可能だったり個数可変だったりする）からなり，次のような構成となっている:

```
commandname [data...]
```

例えば，勾配・曲線・制限速度データファイルに出てくる gcvel コマンドは次のような形をしている。

```
gcvel  9
 16.382  16.458  -5.00    0.0  80.0
 16.458  16.480  -5.00  622.0  80.0
 16.480  16.578 -35.00  622.0  80.0
 16.578  16.728 -35.00    0.0  80.0
 16.728  17.100 -33.00  450.0  80.0
 17.100  17.160  -2.00  450.0  80.0
 17.160  17.220  -2.00  355.0  80.0
 17.220  17.316  -2.00  355.0  40.0
 17.316  17.382   5.00  355.0  40.0
```

このうち，最初のデータとなる 9 は整数で，その後のデータ（5つの倍精度実数で1組）の組数を表す。データは先頭から読まれ，最初の整数データによって全体のデータの個数を可変としている。この場合は，最初のデータが9であったから，これに45個の実数データが続くことになる。

データの読み込みは，C 言語の標準的な入出力関数のひとつ printf() を利用して記述してある。このため，ひとつながりの数字の間でなければ，どこに改行コードやスペース文字，タブ文字などを挿入しても大丈夫だ。従って，上のように適当な場所でデータを改行し，読みやすくすることができる。

また，ファイルの先頭に**フラグ**が書き込まれることがある。フラグはコマンドと同一の書式を持つが，データファイルにおいて最初のコマンドが出現するまでの間に書かれなければならない。もちろん，コマンドだからといってどこに書いてもよいわけではなく，順序などがある程度決まっている。また，多くの場合フラグは「書かなければならない」ものではなく，コマンドは「書かなければならない」という違い

がある。しかし，コマンドの中にも省略可能なものがあるし，書かれたフラグの種類によってコマンドが省略可能になったり，データの個数や種類が変更になったりすることもある。

また，コメント行はデータファイルの任意の場所に挿入することが可能である。コメントは UNIX の C シェルスクリプトなどと同じく，# 文字から行末までをコメントとみなす約束になっている。例えば，上の gcvel コマンドの場合，

```
gcvel  9
 16.382  16.458  -5.00    0.0  80.0
 16.458  16.480  -5.00  622.0  80.0
 16.480  16.578 -35.00  622.0  80.0 #This is a comment
 16.578  16.728 -35.00    0.0  80.0
 16.728  17.100 -33.00  450.0  80.0
 17.100  17.160  -2.00  450.0  80.0
 17.160  17.220  -2.00  355.0  80.0
 17.220  17.316  -2.00  355.0  40.0
 17.316  17.382   5.00  355.0  40.0
```

のように挿入することも可能だ。なお，漢字コードへの対応についてはきちんと行っていない。EUC 漢字コードならば大丈夫であると思われるが，確認はしていないので，at your own risk にて行うこと。

なお，例えばG ファイル（勾配・曲線・制限速度データファイル）にあるコマンドであれば，G コマンドと略称することがある。N コマンド, P コマンド, S コマンド, F コマンド なども同様。G フラグ, N フラグ, P フラグ, S フラグ, F フラグ なども同様である。

⟨J.2⟩　勾配・曲線・制限速度データファイル（G ファイル）

勾配・曲線・制限速度は，路線を適当な区間に分けたとき，その区間内では一定である。となりの区間と連続である必要はない。すなわち，緩和曲線や縦曲線は考慮しない。そこで，路線をいくつもの区間に分けて,「区間始点」「区間終点」「勾配」「曲線」「制限速度」を一組としたデータを与えてやればよい。この一組のデータを gcvel データ と呼ぶことにする。

次駅データとの関連から，gcvel データは駅間ごとに与える。こうすると，gcvel データは駅間ごとにいくつかが1組となる。この1組を1つの GRADCRVでーた称する。場合によって，上り下りで別な gradcrv データを与えることも，同一の gradcrv データを共用することも，どちらも可能である。

⟨J.2.1⟩　フラグ

このデータファイルに書き込めるフラグは，現在のところ次の1種類だけが認められる。

- vlimitmikomi フラグ: 速度制限に見込む余裕。データは倍精度実数の1つだけで，単位は km/h とする。このフラグを与えると，速度制限がこの分だけ低くなったのと同じ振舞いをする。データファイルそのものを修正しても同じ効果は得られる。

⟨J.2.2⟩　コマンド

このデータファイルには次のようなコマンドを，**ここで出てきた順に**書き込む必要がある。

1. number コマンド: gradcrv データの数。

   ```
   number 29
   ```

   のように書く。データファイルにはこれ以降 gcvel コマンドがここで指定した回数だけ現れなければならない。

2. gcvel コマンド: 可変長データを持つコマンドであり，このコマンド1つで gradcrv データ1つを表現する。

   最初のデータは整数で，この gradcrv データの中に gcvel データがいくつ含まれるかを示す。次いで，(この整数×5) 個の倍精度実数データが与えられ，必要な数の gcvel データを表現する。

   1つの gradcrv データにつきこのコマンドが1つずつ対応するはずであるから，最初の number コマンドで指定された数だけの gcvel コマンドが現われなければならない。

   gcvel データを表す5つの倍精度実数は順に「区間始点 (km)」「区間終点 (km)」「勾配 (‰)」「曲線 (m, ゼロは直線)」「制限速度 (km/h, ゼロは制限なし)」を意味する。「区間始点」は必ず「区間終点」より小さな数字でなければならない。また，勾配は「区間始点」から「区間終点」に向かう列車に対する上り勾配で表示する。また，ひとつの gcvel コマンド中において，となりあうふたつの gcvel データのうち，データファイルの先頭側にあるものの「区間終点」と末尾側にあるものの「区間始点」は同一の数字でなければならない。

   ```
   gcvel 2
   0.000 0.400 0.00 0.0 0.0 0.400 0.800 -3.00 0.0 0.0
   ```

   のように書く。

   データファイルでは通常複数の gcvel コマンドを書くことになるが，これは RTSS に対して複数の gradcrv データを入力したことと同じである。このとき，gradcrv データには内部で自動的に番号が付けられる。この番号は，最初の gcvel コマンドに対応するものがゼロ，次が1，次いで2，……のようになる。最初がゼロである理由は C 言語を利用している人ならばわかると思う。なお，この番号はあとで次駅データを作成する際に必要となるものなのだが，データファイルを見ていちいち数えるのではわかりにくいので，対応する番号をデータファイル上にコメント行として記入しておくとわかりやすいだろう。

⟨J.2.3⟩ サンプル

データファイルの例を次に示す。本来 gcvel コマンドが29個なければならないが，長いので省略している。

```
#########
# Flags #
#########

vlimitmikomi    5.0

#########################
# Beginning of commands #
#########################

number 29

# (0) Tokyo - Yurakucho
gcvel 3
    0.000  0.400  0.00  0.0  0.0
    0.400  0.580  0.00  400.0  0.0
    0.580  0.780  0.00  0.0  0.0

# (1) Yurakucho - Shimbashi
gcvel  8
    0.780  1.000  0.00  0.0  0.0
    1.000  1.190  0.00  600.0  0.0
```

```
    1.190  1.490  0.00  0.0  0.0
    1.490  1.550  5.00  0.0  0.0
    1.550  1.660  5.00  400.0  0.0
    1.660  1.750  0.00  400.0  0.0
    1.750  1.870  0.00  520.0  0.0
    1.870  1.890  0.00  0.0  0.0

# (2) Shimbashi - Hamamatsu-cho
gcvel  12
    1.890  1.960  0.00  0.0  0.0
    1.960  2.000  0.00  0.0  0.0
    2.000  2.060  5.00  460.0  0.0
    2.060  2.070  0.00  460.0  0.0
    2.070  2.120  -5.00  0.0  0.0
    2.120  2.160  0.00  0.0  0.0
    2.160  2.210  -11.70  0.0  0.0
    2.210  2.310  -10.00  0.0  0.0
    2.310  2.610  -9.10  0.0  0.0
    2.610  2.680  0.00  0.0  0.0
    2.680  2.760  -5.00  0.0  0.0
    2.760  3.050  0.30  0.0  0.0

#                          ……以下略
```

⟨J.3⟩ 次駅データファイル（N ファイル）

次駅データファイルは，列車の次駅までの走行に関するデータ，およびダイヤパターンデータを書き込むファイルである。

⟨H.2⟩（160ページ）にて述べたことの繰り返しとなるが，次駅データとは，次のデータの集まりである:

- 現駅（または，データの対象となる区間の開始点）の位置
- 次駅（または，データの対象となる区間の終了点）の位置
- 次駅（または，データの対象となる区間の終了点）での列車速度
- 次駅（または，データの対象となる区間の終了点）までの所要時分
- 次駅での停車時分
- ノッチオフ速度（現在のプログラムでは意味をなさないが，0.0 を書き込む。将来必要が出てきた場合は有効にする計画である）
- 列車走行方向（通常キロ程が増加する方向が (+)1，反対は −1）
- 饋電線番号
- 勾配・曲線・制限速度データ（gradcrv データ）
- 次駅データの種類を表す記号
- 次駅データ切替地点（駅以外の地点で切替を行う場合に必要）
- 次の「次駅データ」へのポインタ（データとして与える必要はない）

ダイヤパターンは常に周期をなしており，次駅データはダイヤパターンデータの中で配列として与えられている。ダイヤパターンデータとは，

- 次駅データの配列
- このダイヤパターン 1 周期の所要時間（twhole）
- このダイヤパターンの中で走る列車の数（cars）
- パターン倍数（dptm）

- 列車群の位相（最初の列車が最初の駅を出発する時刻）
- 列車の初期化用変数群

などからなるデータである。

列車が次駅データの有効な範囲を越えた場合は，次の次駅データに移る。この移行のときに，列車位置・列車が電力の供給を受ける饋電線の番号・列車の走行方向などのデータを，ある程度自由に変えることができる。また，次駅データ配列の最後のデータの次は，最初に戻る約束になっている。

シミュレート可能な列車ダイヤは，完全に周期的なダイヤでなければならない。

ダイヤパターンデータにおいて与えられるダイヤは，必ずしも運用上「実現可能」である必要はない。例えば直線的な路線の起点から終点まで走る列車が5分に1本あるが，その他の列車は存在しないようなもの（この場合，起点および終点に無限大容量の車庫がないと実現しない）も，シミュレート可能である。

また，ダイヤパターンデータは通常複数存在する。次駅データの配列 $\boldsymbol{A}$, $\boldsymbol{B}$ があり，それぞれ $A_1, A_2, \cdots, A_n$ および $B_1, B_2, \cdots, B_m$ という要素を持たせる。このとき，次駅データ $A_1, A_2, \cdots, A_n$ の順に走る列車群と，$B_1, B_2, \cdots, B_m$ の順に走る列車群とがそれぞれ存在しうる。例えば，JR山手線のような完全な環状線にあっては，このような2つの次駅データが存在してはじめて，内・外回りの列車運動をデータとして与えることができる。また，昼間時の営団銀座線のように，普通の複線で普通列車1種類のみの平行ダイヤ，運転区間も1種類のみである場合は，ダイヤパターンデータは1つであるが，急行・普通の2種の列車があれば2つのダイヤパターンを用意しなければならないし，区間運転などが存在すればダイヤパターンはより多く必要になる。

このような場合，それぞれのダイヤパターンデータにおいて周期を求めると，このダイヤパターン1周期の所要時間（twhole）およびこのダイヤパターンの中で走る列車の数（cars）のふたつのデータから，

```
twhole / cars
```

となるはずだが，これがすべてのダイヤパターンに対して一定であるとは限らない。そこで，パターン倍数（dptm）という変数を用意しておき，

```
twhole / cars * dptm
```

が一定となるようにしてある。dptm は整数だ。

⟨J.3.1⟩ フラグ

このデータファイルに書き込めるフラグは，現在のところ次のものが認められる。

- conges_station フラグ: 混雑率を駅間毎に変更できるようにするフラグで，データはない。このフラグを与えると，congestion コマンドを各次駅データ毎に指定でき，混雑率が駅間毎に変更できるようになる。
- ontprec_delay_set フラグ: 定時ノッチオフ位置決定ルーチンを使用しているにもかかわらず列車があまりに進んだり遅れたりするばあい，フリーランの回数を増やした上で定時ノッチオフ位置決定ルーチンのうち近似計算のルーチンの使用をその次駅データについて抑制する。このフラグで，この操作を行う列車の進み遅れの閾値を設定することができる。データはこの閾値が倍精度実数型でひとつだけ指定可能（単位・秒）である。このフラグがない場合，デフォルト値の 0.6 が採用される。
- ontprec フラグ: 定時ノッチオフ位置決定ルーチンのうち近似計算のルーチンの使用をすべてのケースで抑制するフラグである。このフラグを指定すると，計算時間がかかるようになるが，フリーランの回数がゼロの場合に有効だろう。データはない。

- initialdeparturetime フラグ: trainvar コマンドで tdept，すなわち列車の出発時間を指定できるようにするフラグである。データはない。この機能は，いわば RTSS を「だます」ために利用することができる（⟨H.3.4⟩, 162ページ）。
- station_object_valid フラグ: station_obj オブジェクトを有効にするフラグである。このオブジェクトは現在設計中である。

⟨J.3.2⟩ コマンド

データファイルには次のようなコマンドを，**ここで出てきた順に**書き込む必要がある。ただし，文中に明示してあるコマンドについては，順序が問われないこともある。

1. patterns コマンド: ダイヤパターンデータの数。データの先頭に1つ書かれる nextsta コマンド，およびそれに続く cycletime, phase, cars, patterncirc, trainvar の各コマンド群，および nextsta コマンドで指定した数の次駅データ指定コマンド群を合わせたものが，1つのダイヤパターンデータとなる。データファイルには，ダイヤパターンデータがここで指定した回数だけ現れなければならない。

   ```
   patterns 2
   ```

   のように書く。

2. ダイヤパターンデータ指定コマンド群: nextsta, cycletime, phase, cars, patterncirc, trainvar の各コマンド群，および nextsta コマンドで指定した数の次駅データ指定コマンド群を合わせたものだ。これらのうち cycletime, phase, cars, patterncirc, trainvar の各コマンドは，ダイヤパターンデータの中の最初の次駅データが現われる前に，どの順で現われてもよい。1つのコマンドが複数回現われてはいけない。また，どのコマンドも省略できない。

   2A. nextsta コマンド: 1組のダイヤパターンデータの先頭には，必ずこのコマンドを置かなければならない。一つのダイヤパターンデータの中に，次駅データがいくつあるかを示す。

   ```
   nextsta 29
   ```

   のように書く。

   2B. cycletime コマンド: ダイヤパターンデータの1周当たりの時間を，秒単位で与える。

   ```
   cycletime 3600.0
   ```

   のように書く。

   2C. phase コマンド: パターンの位相ずれを，秒単位で与える。

   ```
   phase 0.0
   ```

   のように書く。

   例えば，山手線のダイヤを内・外回りとも東京駅発の次駅データ配列として与え，内回りのパターンに対して phase 0.0，外回りのパターンに対して phase 60.0 と与えると，内回り電車が東京駅を時刻 0.0 秒に出発すると外回り電車は東京駅を 60.0 秒に出発するパターンとなる。

   2D. cars コマンド: ダイヤパターンデータの中を走っている列車数を与える。

   ```
   cars 25
   ```

   のように書く。

   車両は等間隔にダイヤパターン内に配置される。従って，

   ```
   cycletime 3600.0
   cars 25
   ```

のように指定すると，列車の間隔は

$$\frac{3600[\mathrm{sec}]}{25\,\text{本}} = 144[\mathrm{sec}]$$

となる。これが1パターン当たりの最低周期である。

2E. `patterncirc` コマンド: パターン倍数。各ダイヤパターンが完全に周期的であるとしても，1パターンあたりの最低周期が一致するとはかぎらない。そこで，`cycletime`, `cars`, `patterncirc` の各コマンドにて指定した値をそれぞれ c, n, p とするとき

$$\frac{c}{n} \cdot p = \mathrm{const.} \tag{J.1}$$

となるような整数 p を与える。

```
patterncirc 1
```

のように書く。

2F. `trainvar` コマンド: 列車データの初期値を与えるコマンドである。

```
trainvar 0.0 0.0 0 0.0
```

のように書く。

`trainvar` の後にはかならず4つのデータを書き込む。順に

- 列車初期位置 [km]
- 列車初期速度 [km/h]
- 列車初期状態，整数で与える。この整数は，プログラム中の列挙データ型 `car_stat` において，該当する整数または文字列を与える。
  - `Power` または 0 (力行)
  - `Brake` または 1 (ブレーキ)
  - `Coast` または 2 (惰行)
  - `Stn_stop` または 3 (駅停車)
  - `Const_vel` または 4 (定速走行)
  - `Regenerate` または 5 (回生のみ)
- 列車駅出発時初期遅れ [s]

プログラムは列車の初期配置を自動的に決定する関数を持っている。これらのデータは，この初期配置を決定する際に列車に初期の状態変数として与えられるデータである。

なお，`initialdeparturetime` フラグがある場合，`trainvar` コマンドに与えるデータは5つとし，順に

- 列車初期位置 (上記と同じ)
- 列車初期速度 (上記と同じ)
- 列車初期状態 (上記と同じ)
- 列車出発時刻 [s]
- 列車駅出発時初期遅れ (上記と同じ)

を書き込む。当然ながら，列車出発時刻の指定は列車初期状態が「駅停車」でなければ意味を持たない。この時刻は当該駅での所定の停車時分と無関係に設定できるので，結果的にデータをだます機能を持つ（⟨**H**.3.4⟩, 162ページ）。

2G. 次駅データ群: `startpoint`, `endpoint`, `endvelocity`, `startstoptime`, `stationstop`, `notchoff`, `direc`, `nfline`, `nextsta_pattern`, `gradcrvnumber`, `gradcrv`, `congestion` および `nextnxx` のコマンド群（ただし `nextnxx` は省略可能，`congestion` は `conges_station` フラグを書かなかっ

た場合は記入禁止，またgradcrvは1つとは限らない）の1まとまりをもって，1つの次駅データを表現する。この次駅データを，nextstaコマンドで指定された数だけ配置する。

なお，これらのコマンドのうち，nextnxxだけが省略可能である。これらのコマンドはどの順で出てきてもよい。ただし，gradcrvnumberコマンドの直後には，コマンドで指示された数だけのgradcrvコマンドが出てくる必要があるほか，nextnxxコマンドの配置方法は注意を要する。

(i) startpointコマンド: 次駅データの開始地点。通常は，発駅のある地点を指す。

```
startpoint 0.0
```

のように書く。

(ii) endpointコマンド: 次駅データの終了地点。通常は，着駅のある地点を指す。

```
endpoint 0.78
```

のように書く。

(iii) endvelocityコマンド: 次駅データの終了地点での速度。通常は，終了地点は着駅のある地点であるから，0.0を与えればよい。

```
endvelocity 0.0
```

のように書く。

(iv) startstoptimeコマンド: 次駅データの開始地点から終了地点までの所要時分（単位秒)。

```
startstoptime 70.0
```

のように書く。

(v) stationstopコマンド: 次駅での停車時分。

```
startstoptime 20.0
```

のように書く。

(vi) notchoffコマンド: ノッチオフ速度を入れるが，現状のプログラムでは完全なダミーパラメータとなっている。

```
notchoff 0.0
```

と書く。

(vii) direcコマンド: 列車走行方向。キロ程の数字が増加する方向に走るとき (+)1，その逆は−1 と与える。

```
direc -1
```

のように書く。

(viii) nflineコマンド: 饋電線番号。列車に電力を供給する饋電線の番号を整数として与える。

```
nfline 0
```

のように書く。饋電線の数を n とすると，ここに書きうる数字は $0, 1, \cdots, n-1$ である。範囲外の数字を書き込んではいけない。なお，饋電線数は饋電線データファイル‘feeder.dat’のなかで，feedersコマンドによって指定する。

(ix) nextsta_patternコマンド: 次駅データの種類を与える。いまのところStationタイプ，つまり次駅データの終点が停車駅である場合にしかプログラムが対応していない。

```
nextsta_pattern Station
```

のように書く。

(x) gradcrvnumberコマンド: gradcrvコマンドが現われる回数を整数で記入する。このコマンドに続けてgradcrvコマンドをその回数だけ書き込まなければならない。

```
gradcrvnumber 3
```

のように書く。いうまでもなく，gradcrvnumber 0 などという指定はできない。

(xi) congestion コマンド: この次駅データが有効な範囲内での混雑率を指定する。混雑率は % ではなく，比で記入することに注意されたい。すなわち，いわゆる200%乗車の場合なら

```
congestion 2.0
```

のように書く。

なお，conges_station フラグを記入しなかった場合，このコマンドを記入するとエラーとなる。

(xii) gradcrv コマンド: 勾配・曲線・制限速度データ（gradcrv データ）を選択し，場合によっては適当に変換して，次駅データの中に取り込む。仮に複数の gradcrv コマンドが1つの次駅データに与えられているならば，選択された gradcrv データは連結されて1つの配列とされる。

```
gradcrv 3 0
```

または

```
gradcrv 3 1 2.5
```

のように書く。

gradcrv コマンドの後には2ないし3個の数字を記入する。まず，最初の数字は整数であって，gradcrv データの番号を記入する。この番号は，⟨J.2⟩で述べた勾配・曲線・制限速度データファイルにおいて，最初に現われる gradcrv データを 0 番として，後に出てくるものを $1, 2, \cdots,$ として定める。

次の数字も整数である。これが 0 の場合は3つめの数字は記入する必要はなく，gradcrv データはデータファイルのまま代入される。それ以外の場合は3つめの数字が必要である。

まず，2つめの整数が 0 より大である場合は，新たな距離原点を3つめの数字の位置にとり，距離を書き改める。例えば，ある gradcrv データに含まれる gcvel データにおいて，始点が 5.0, 終点が 6.5 であるとしよう。この gcvel データを含む gradcrv データを

```
gradcrv 3 1 2.5
```

なるコマンドで呼び出したとすれば，新しい距離原点を 2.5 にとるので，始点の 5.0 は 2.5 に，終点の 6.5 は 4.0 に，それぞれ修正される。

```
gradcrv 3 1 6.5
```

ならば，始点は -1.5 に，終点は 0.0 になる。このようなルールですべての gcvel データが修正され，新しい gradcrv データが生成される。

2つめの整数が 0 より小である場合は，gcvel 配列の順序を逆にする。同時に始点と終点の位置も入れ換えられる。さらに，距離原点は 3つめの数字にとり，距離の符号も反転させる。例えば，上と同じように gcvel データにおいて始点が 5.0, 終点が 6.5 であるときに，この gcvel データを含む gradcrv データを

```
gradcrv 3 -1 2.5
```

なるコマンドで呼び出したとすれば，始点・終点を入れ換えて，新しい距離原点を 2.5 にとったうえ符号を反転させるので，始点は -4.0 に，終点は -2.5 に，それぞれ修正される。

```
gradcrv 3 -1 6.5
```

ならば，同様にして始点は 0.0 に，終点は 1.5 になる。このようなルールですべての gcvel データが修正され，入れ換えられて，新しい gradcrv データが生成される。

複数の gradcrv コマンドがあるときは，これらの変換の結果出てくる gcvel データの配列が連続していなければならない。つまり，ある gradcrv コマンドで生成された gradcrv データの配列先頭にある gcvel データの区間始点は，直前の gradcrv コマンドで生成された gradcrv データの中の配列末尾にある gcvel データの区間終点と一致していなければならない。

(xiii) nextnxx コマンド: このコマンドのみ省略可能である。このコマンドは，次駅データに与えられている区間の中間で，饋電線番号を変更するときにのみ用いる。

```
nextnxx 53.32
```

のように書く。この位置を列車が過ぎたならば，次駅データを次のものに移行させる。次の次駅データは，nfline コマンド以外の必須コマンドとしてすべて同一の値を書き込まなければならない。

なお，startpoint, endpoint, endvelocity, startstoptime, stationstop, notchoff, direc, nfline, nextsta_pattern, congestion, gradcrvnumber の各コマンドはすべて省略不可能である（ただし，conges_station フラグがないならば congestion コマンドは記入禁止）。これらのコマンドはどの順で出てきてもよい（ただし，gradcrvnumber コマンドの直後には，コマンドで指示された数だけの gradcrv コマンドが出てくる必要がある）。これらが1つでも欠けていたり，複数あったりするとエラーとなる。gradcrvnumber 0 が許されていないから，実質的には gradcrv コマンドも必須である。

しかし，nextnxx コマンドは省略可能であるため，次のような問題を生ずる。すなわち，プログラムは必須コマンドがすべて出つくしてしまうと，次駅データの記述が終わったと考えて次に進んでしまうのである。そこで

```
startpoint 55.45
endpoint 48.44
nextnxx 53.32
endvelocity 0.00
startstoptime 265.0
stationstop 20.0
notchoff 0.0
direc -1
nfline 9
gradcrvnumber 1
gradcrv 19 0
nextsta_pattern Station
```

のように，必須コマンドの中間にいれないと正しく認識されない。

```
startpoint 55.45
endpoint 48.44
endvelocity 0.00
startstoptime 265.0
stationstop 20.0
notchoff 0.0
direc -1
nfline 9
```

```
gradcrvnumber 1
gradcrv 19 0
nextsta_pattern Station
nextnxx 53.32
```

とすれば，この nextnxx コマンドは，これらの一連の startpoint 〜 nextsta_pattern コマンド群で示される次駅データに対するコマンドではなく，この次に記述される次駅データに対するコマンドとして理解されてしまうので，注意を要する。

データファイルの例を次に示す。本来次駅データがダイヤパターンごとに29個なければならないが，長いので省略している。

```
patterns 2

nextsta 29
cycletime 3600.0
phase 0.0
cars 25
patterncirc 1
trainvar 0.0  0.0  0  0.0

startpoint 0.000
endpoint 0.780
endvelocity 0.00
startstoptime 70.0
stationstop 20.0
notchoff 0.0
direc 1
nfline 0
nextsta_pattern Station
gradcrvnumber 1
gradcrv 0 0

startpoint 0.780
endpoint 1.890
endvelocity 0.00
startstoptime 90.0
stationstop 30.0
notchoff 0.0
direc 1
nfline 0
nextsta_pattern Station
gradcrvnumber 1
gradcrv 1 0

#                          ……以下，次駅データ省略

nextsta 29

cycletime 3600.0
phase 60.0
cars 25
patterncirc 1
trainvar 34.475  0.0  0  0.0

startpoint 34.475
```

```
endpoint 33.225
endvelocity 0.00
startstoptime 100.0
stationstop 20.0
notchoff 0.0
direc -1
nfline 1
nextsta_pattern Station
gradcrvnumber 1
gradcrv 28 0

#                              ……以下略
```

⟨J.4⟩ 変電所特性データファイル（S ファイル）

変電所特性データファイルは，各変電所の特性を与えるファイルである。変電所の特性は，変電所ごとに設定することができる。変電所の特性はV–I平面上で折れ線で与えられるものに限る。

⟨J.4.1⟩ フラグ

このデータファイルには，現在のところフラグの書き込みは認められていない。

⟨J.4.2⟩ コマンド

データファイルには次の項目を，この順に記入する。順番を変えてはいけない。

1. substations コマンド: 変電所数を整数で示す。

   ```
   substations 11
   ```

   のように書く。データファイルには，subchar コマンドがここで指定した回数だけ現われなければならない。

2. subchar コマンド: 可変長データを持つコマンドであり，このコマンド1つで1変電所分の特性データを表現する。

   最初のデータは整数である。変電所の特性は折れ線で表示されるが，その折れ線の端点の数を整数で示す。このあとに，端点データがここで指定した数だけ続いて現われなければならない。当然ながら，ここで 1 以下の数を指定しても意味がない。

   端点データは1点あたり3つの倍精度実数値で表現され，折れ線の端点の電圧・電流と，媒介変数の値を示す。媒介変数0のときは例えば無負荷時送出電圧となるように考えるべきであろう。また，媒介変数の値のとり方を変にすると計算失敗の原因となることもある。データは，媒介変数，電圧，電流の順に，実数を3つずつ書き込む。例えば

   ```
   subchar 3
   -200.0 1800.0 0.0
   0.0 1600.0 0.0
   20000.0 1100.0 20000.0
   ```

   のように書く。

   なお，複数の端点データがあるとき，各行の媒介変数は，行が下に行くにしたがって単調に増加，ないしは減少しなければならない。また，媒介変数の変域は，最初の端点データにある値と最後の端点データにある値との間に制限される。したがって，はじめと終わりの端点データは実質的に最高電圧・最低電圧などを決定する点になる。この点は，シミュレーションの目的にかなうように，または計算がきちんと終了するように，適当に与えればよい。

3. ratedcurrent コマンド: 変電所の定格電流値（単位 A），I_{Sl}, I_{Sc} および I_{Sm} の値（単位 p.u.）を，実数で与える。

    ratedcurrent 4800.0 2.0 2.5 3.0

    のように書く。通常のシミュレーションではあまり深い意味は持たないが，指定はしなければならない。

⟨J.4.3⟩ サンプル

データファイルの例を次に示す。

```
substations 3

# (0) A SS
subchar 5
        -43300.0        1800.0          -3333.333334
        -23000.0        1551.0          -3333.333334
        -20000.0        1551.0          -1.0
        0.0             1521.0          0.0
        20000.0         1091.0          10000.0
ratedcurrent 2000.0 2.0 2.5 3.0

# (1) B SS
subchar 3
        -20000.0        1800.0          -1.0
        0.0             1521.0          0.0
        20000.0         1091.0          10000.0
ratedcurrent 2000.0 2.0 2.5 3.0

# (2) Dummy SS (con. 0)
subchar 2
        -200.0          1800.0          0.0
        1200.0          400.0           0.0
ratedcurrent 0.0 2.0 2.5 3.0
```

⟨J.5⟩ 列車性能データファイル（P ファイル）

列車の性能は，入出力インタフェースの問題から現状では全列車同一のものしか与えられない。また，インバータ制御車両が前提となっている。

⟨J.5.1⟩ フラグ

このデータファイルに書き込めるフラグは，現在のところ次の 6 種類が認められる。すべてデータのないフラグである。これらのフラグは 2 度以上書き込まないようにすること。2 度書き込むと，トグルになっていて元に戻ることになり，フラグの指定を無効としてしまう。

1. regendantei フラグ: 回生失効をシミュレートする。このフラグを与えると，回生絞り込みと回生失効を区別してシミュレーションを行う。
2. differentregend フラグ: 回生絞り込み終了電圧と系の最高電圧が異なるデータを取り扱う。このフラグを与えると，回生絞り込み終了電圧を最高電圧とは別にデータとして与えることができるようになる。
3. shiboriparallel フラグ: 回生絞り込み特性を (1) （⟨**F**.3⟩，図 7.22参照）とする。このフラグを与えると回生率は悪くなることだろう。
4. auxpower フラグ: 補機負荷をパワー (kW) で与えるようにする。デフォルトは電流値 (A) である。

5. torque_shibori フラグ: 主回路回生電流（⟨**F**.3⟩，図7.23参照）のかわりにモータ電流を回生絞り込みの基準に用いる。このフラグを与えると回生絞り込みが必要でない場合にも行われることになる。
6. anteimotor フラグ: 回生失効の判定基準にもモータ電流を用いる。このフラグを与えると，新たにモータ電流をデータとして与える必要が出てくる。

⟨J.5.2⟩ コマンド

データファイルには，この順に次のコマンドをすべて書き込まなければならない。

1. cs_command コマンド: 列車で行う制御の種類を示す。None（従来方式・制御なし），Ec_auto（フルノッチ比制御あり），Ea_cvauto0（定数調整制御あり）のなかから選んで，文字列で記入する。
2. powerdata コマンド: 列車の力行性能を記述するコマンド群の始まりを指定するコマンドである。このコマンド群は以下のようなものだ。
   - 2A. ec コマンド: 性能を表す数字は，電車線電圧がこのコマンドで指定される電圧値であるときのものを記入する。
   - 2B. zerocur コマンド: 0km/h フルノッチ力行時の電流値を記入する。
   - 2C. minimumvoltage コマンド: 最低電圧を記入する。400V など，かなり低くてよい。
   - 2D. bminvoltage コマンド: ダミーパラメータである。通常1200V程度を記入する。
   - 2E. maximumvoltage コマンド: 最高電圧を記入する。
   - 2F. empty コマンド，mid コマンド，full コマンド: 応荷重装置のシミュレーションに用いる特性指標を記入する。empty のあとのものが空車，full のあとのものが満車で，もうひとつmid はその中間のものを与える。mid は「3点指示方式」の採用によって必要となったもので，詳しいことは ⟨**F**.2.1⟩を参照されたい。これら3点とも指定することが求められる。
     それぞれのコマンドの下には以下のコマンド群をすべて記入する。
     - (i) congestion コマンド: 混雑率。empty の下は0.0，full の下は最大乗車時を記入。なお，full の混雑率以上では応荷重装置は働かない（最大性能で固定される）ように考える。
     - (ii) maximumcurrent コマンド: 編成あたりの最大パンタ点電流 (A) を記入。
     - (iii) lowvelocity コマンド: 定トルク領域終端速度 (km/h) を記入。
     - (iv) highvelocity コマンド: 定パワー領域終端速度 (km/h) を記入。
     - (v) tractionforce コマンド: 定トルク領域引張力 (tf) を記入する。
     - (vi) full.lowvel コマンド: ここでは力行満車時の定トルク領域終端速度を
3. brakedata コマンド: 列車のブレーキ時性能を記述するコマンド群の始まりを指定するコマンドである。このコマンド群は以下のようなものだ。
   - 3A. ec コマンド: ブレーキ性能を表す数字は，電車線電圧がこのコマンドで指定される電圧値であるときのものを記入する。
   - 3B. regenoff コマンド: 回生ブレーキ終端速度 (km/h) を記入する。
   - 3C. minimumvoltage, bminvoltage, maximumvoltage コマンド: 力行時データと同じ。
   - 3D. regenlimitstartvoltage コマンド: 回生絞り込み開始電圧 (V) を記入する。
   - 3E. regenlimitendvoltage コマンド: 回生絞り込み終了電圧 (V) を記入する。differentregend フラグがない場合はこのコマンドを記入するとエラーとなる。
   - 3F. regenoffanteika コマンド: パンタ点電流がこれを下回ったら回生失効とする電流値 (A) を記入する。regendantei フラグがない場合はこのコマンドを記入するとエラーとなる。また，anteimotor フラグが立っているとこれはモータ電流のこととなる。

3G. empty コマンド，mid コマンド，full コマンド: 力行時と同じく，応荷重装置のシミュレーションに用いる特性指標を記入する。empty のあとのものが空車，full のあとのものが満車で，もうひとつ mid はその中間のものを与える「3点指示方式」である。

それぞれのコマンドの下には力行時と同じコマンド群をすべて記入する。もちろん回生ブレーキ時の特性として記入するべきである。なお，anteimotor フラグが立っている場合に限り次のコマンドをも記入する。

- motorcurrent コマンド: そのときの最大モータ電流 (A) を記入する。

4. autodata コマンド: cs_command コマンドに Ec_auto を指定した場合に意味があるコマンド群である。意味がない場合も省略できないので，以下のコマンドすべてに適当な値を入れておく。

4A. minimumvelocity コマンド: このコマンドは，Ec_auto および Ea_cvauto0 のモードにおいて，制御をかけはじめる最低の速度 (km/h) を示す。この速度より下では，列車はどのモードであっても None とまったく同じふるまいをする。

4B. fullvelocity コマンド: このコマンドは，制御をフルにかける最低の速度を示す。minimumvelocity コマンドで指定した速度とこの速度の間では，制御をフルにはかけない状態になる。この速度より上では制御はフルにかかる。

4C. powerlimitvoltage コマンド: 力行時，列車の電流を絞り込みはじめる電圧。これより電車線電圧が低いと「フルノッチ比」が下がる。惰行時には，列車がピーク救済のための回生ブレーキをかけはじめる電圧である。

4D. poweroffvoltage コマンド: 力行時，列車の電流を完全に絞り込み終わる電圧。これより電車線電圧が低いとフルノッチ比は一定となり，速度が fullvelocity コマンドで指定された速度より高ければフルノッチ比0，すなわち力行電流0となる。惰行時には，列車がピーク救済のための回生ブレーキをもっとも強くかける領域の上限電圧となり，これより電車線電圧が低いとフルノッチ比は最低となり，速度が fullvelocity より高ければフルノッチ比は -1 である。

4E. regenerationstartvoltage コマンド: 惰行時，回生失効防止のため惰行車が加速しはじめる電圧。

4F. regenerationfullvoltage コマンド: 惰行時，回生失効防止のため惰行車が最大加速する領域の下限電圧。

5. congestion コマンド: 列車の混雑率を与える。%で与えてはならない。

```
congestion 1.0
```

のように書く。

6. auxcurrent コマンド: 列車の補機電流 (A) を与える。auxpower フラグが立っていれば補機パワー (kW) を与える。

```
auxcurrent 30.0
```

のように書く。

7. setweight コマンド: 順に「電動車自重」「電動車定員乗車時荷重」「付随車自重」「付随車定員乗車時荷重」を，4つの実数によって与える。

```
setweight 190.0 50.0 150.0 50.0
```

のように書く。

8. setrest コマンド: 走行抵抗式を6つの定数で与える。6定数を順に a, b, c, d, e, f，列車速度 v[km/h]，電動車質量 M_d[t]，付随車質量 M_t[t]，編成両数 n[両] とするとき，走行抵抗 R[kgf] は

$$R = (a + bv)M_d + (c + dv)M_t + \{e + (n-1)f\}v^2 \qquad (\mathbf{J}.2)$$

と与える。通常は

```
setrest 1.65 .0247 .78 .028 .0358 .0078
```

などと書けばよい。

9. setmres コマンド: 曲線抵抗の係数，回転部分補正係数，平坦線減速度，編成両数を入れる。

```
setmres 800.0 0.0825 3.0 10
```

のように書く。

⟨J.5.3⟩ サンプル

データファイルの例を下に示す。

```
# train characteristics file for RTSS ver. 2.1.1

regendantei     # This flag enables regeneration stop simulation.
differentregend # This flag makes regendvol and maxvol different.
shiboriparallel # This flag makes overvoltage limit narrower.
auxpower        # This flag makes auxcurr point auxiliary power.
torque_shibori  # This flag makes motor current for regeneration limit.
anteimotor      # With this flag, RTSS will use motor current for
                # regeneration down decision.

cs_command None

powerdata
        ec 1350.0
        zerocur 100.0
        minimumvoltage 400.00
        bminvoltage 1200.0
        maximumvoltage 1800.0
        empty
                congestion 0.0
                maximumcurrent 1750.0
                lowvelocity 64.0
                highvelocity 64.0
                tractionforce 11.9995
        mid
                congestion 0.5
                maximumcurrent 1850.0
                lowvelocity 60.54
                highvelocity 60.54
                tractionforce 13.6405
        full
                congestion 2.5
                maximumcurrent 1850.0
                lowvelocity 42
                highvelocity 60.54
                tractionforce 20.2459

brakedata
        ec 1650.0
        regenoff 5.0
        minimumvoltage 400.00
        bminvoltage 1200.0
        maximumvoltage 1800.0
```

```
        regenlimitstartvoltage 1650.0
        regenlimitendvoltage 1700.0
        regenoffanteika 70.0
        empty
                congestion 0.0
                maximumcurrent 1340.0
                lowvelocity 87.0
                highvelocity 87.0
                tractionforce 10.2432
                motorcurrent 376.0
        mid
                congestion 2.5
                maximumcurrent 1670.0
                lowvelocity 69.81
                highvelocity 69.81
                tractionforce 17.2975
                motorcurrent 616.0
        full
                congestion 2.5
                maximumcurrent 1670.0
                lowvelocity 69.81
                highvelocity 69.81
                tractionforce 17.2975
                motorcurrent 616.0

autodata
        minimumvelocity 20.0
        fullvelocity 35.0
        powerlimitvoltage 1450.0
        poweroffvoltage 1200.0
        regenerationstartvoltage 1600.0
        regenerationfullvoltage 1750.0

congestion 0.20
auxcurrent 30   # [kW]
setweight 65.0  16.665  51.4  15.4
setrest 2  .11  1  .0132  .063  .0078
setmres 600.0  0.0825  3.0  4
```

⟨J.6⟩ 饋電線データファイル（F ファイル）

饋電線とその接続関係を示すデータファイルである。また，饋電システム全体に関するデータもここに書き込まれる。

⟨J.6.1⟩ フラグ

このデータファイルに書き込めるフラグは，現在のところ次の 1 種類のみが認められる。

- options フラグ: このフラグでいろいろなオプションを設定することができる。このフラグを複数書くことができるが，同じオプションを 2 度書くとトグルスイッチになっているため指定が無効化されて元に戻ってしまうから注意が必要だ。データはオプションを指示する次の文字列のいずれかである：
    1. ccalresult: これを与えると，ログに計算結果を等価回路演算毎に吐き出す。ログファイルが大量になるのであまりお勧めできない。

2. ssdirection: これを与えると，変電所での方面別電流を計算する。まだバグが残っているようだ。
3. kilowatt_subout: 出力ファイルにおいて，変電所別出力電力量を kW 単位で表示する。
4. per_car_out: 列車の力行・回生電力量を列車あたりで出力。列車がすべて同じような走り方をするのでないと意味がない。

⟨J.6.2⟩ コマンド

このデータファイルには次のようなコマンドを，**ここで出てきた順に**書き込む必要がある。

1. feeders コマンド: 饋電線の数，およびダイヤパターンに対する指示のふたつのデータを書き込む。
   饋電線の数は整数で記入する。ダイヤパターンに対する指示は，余裕時分の再配分を行なうときは Buffer_yes，早着防止を行なうときは Delay_yes，何もしないときは No_comm を，それぞれ文字列で記入する。

   ```
   feeders 2 No_comm
   ```

   のように書く。

2. freeruns コマンド: シミュレーション開始時までの空回しの回数を整数で記入する。多くの場合，大きいほど正確なシミュレーションができるが，時間がかかる。ただし，場合によっては空回しが行えないモデルもあろう。

   ```
   freeruns 3
   ```

   のように書く。

3. simulatings コマンド: シミュレーションを行なう周期を記入する。周期的に列車が動いてくれる分には，大きい必要はない。通常1ないし2で十分である。

   ```
   simulatings 2
   ```

   のように書く。

4. setfeedline コマンド: すべての饋電線について，その長さ，饋電定数，形状を記入する。このコマンドは，feeders コマンドで指定した饋電線数と同じ数だけ現われなければならない。

   ```
   setfeedline 34.475 0.0327 Long
   setfeedline 34.475 0.0327 Long
   ```

   のように書く。データは，それぞれ饋電線長 [km]，饋電定数 [Ω/km]，饋電線形状である。形状は，線分状のときは 0 または Long，環状のときは 1 または Circle を，それぞれ記入すればよい。また，最初の setfeedline コマンドが饋電線番号0，次が1，などのように対応する。この饋電線番号が次駅データファイル‘nextsta.dat’にて nfline コマンドで指定される饋電線番号に対応するので注意されたい。

5. ssfeederconnections コマンド: 可変長データを持つコマンドであり，このコマンド1つで変電所と饋電線との接続関係をすべて表現する。
   最初のデータは整数で，変電所と饋電線との接続点に関するデータがいくつあるかを記入する。次いで，この整数の数だけの饋電線・変電所接続点データが続く。
   ひとつの饋電線・変電所接続点データは2つの整数と3つの倍精度実数の都合5 つの数字を1組としたデータである。このうち，最初の整数が饋電線番号，次の整数が変電所番号，3つ目（倍精度実数）が接続点位置 (km)，4つ目・5つ目（どちらも倍精度実数値）はそれぞれ Busbar から起点側接続点までのインピーダンス，および Busbar から終点側接続点までのインピーダンス（どちらも単

位 Ω）を示す。なお，変電所が饋電線から極端に離れていない多くのシミュレーションケースでは，最後の2つのデータは 0 となるが，変電所が饋電線から離れており，Busbar と饋電線との間のインピーダンスが無視できない場合には0以外の値を入れる。

```
ssfeederconnections 6
0 1 0.0 0.0 0.0
0 0 5.0 0.0 0.0
0 2 10.0 0.0 0.0
1 1 0.0 0.0 0.0
1 0 5.0 0.0 0.0
1 2 10.0 0.0 0.0
```

のように書く。

データファイルの例を下に示す。

```
# "feeder" data file format for RTSS 2.1.1

# options command: you can write any of them before feeders command
#       options ccalresult      # toggle g_sw_ccalresult, default: FALSE
        options ssdirection     # toggle g_sw_ss_direction, default: FALSE
        options kilowatt_subout # toggle g_sw_kilowatt_subout
        options per_car_out     # toggle g_sw_per_car_out

feeders 2 No_comm
freeruns 3              # free run 3 cycles
simulatings 2           # simulation 2 cycles

setfeedline     7.0     0.033   Long    # No. 0 feedline, Long/Circle
setfeedline     7.0     0.033   Long    # No. 1 feedline, Long/Circle

ssfeederconnections 6
#       nfline  csno    pos     sttimp  endimp
        0       1       1.0     0       0
        0       0       6.0     0       0
        0       2       11.0    0       0
        1       1       1.0     0       0
        1       0       6.0     0       0
        1       2       11.0    0       0
```

⟨J.7⟩ 列車初期位置データファイル

このデータファイルは，なければプログラムが勝手に計算して作るので，ユーザは気にする必要はないものだった。列車初期位置の計算には非常に時間がかかるので，このようなファイルをつくって少しでも計算時間を短縮しようとしたものである。しかし，RTSS ver.2.1 開発の過程でバグが発見されたため，現在はデータのセーブはするものの利用していない。将来はプログラムを整備し，再び機能を復活させる予定であるが，いつになることやら……。

この機能が有効になった暁には次のことに注意されたい。すなわち，条件を変えれば当然初期位置データファイルの中身も変わるので，条件を変えた場合に別な条件における列車初期位置データファイルを誤って読み込むことのないよう，コマンドオプションの指定には注意すべきであろう。ただし，列車初期位置は電車線電圧などには依存しない（電車線電圧としてノミナル値を仮定するため）ので，ある程度の共用はできる。

RTSS マニュアル・一般概念索引

き

こ

さ

し

じ

だ

て

で

と

の

ば

ぱ

ひ

ふ

ぶ

へ

め

も

り

れ

RTSS マニュアル・関数・変数索引

A

C

D

E

F

G

M

N

O

R

S

T

V

Y

Z

RTSS マニュアル・クラス・構造体名索引

RTSS マニュアル・コマンド名索引

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

V

Z

RTSS マニュアル・記号索引

C

I

K

N

R

T

V