SSI-DT79175026

博士論文

電力系統の安定化に貢献する需要家リソース群の運用

中村勇太

2019 年 3 月

北海道大学 大学院情報科学研究科
システム情報科学専攻

本論文は北海道大学 大学院情報科学研究科に
博士（情報科学）授与の要件として提出した博士論文である。

中村勇太

審査委員：　主査　北　教授
副査　五十嵐　教授
小笠原　教授
原　准教授

電力系統の安定化に貢献する需要家リソース群の運用*

中村勇太

概要

低炭素化社会への関心の高まりから，近年風力発電や太陽光発電等の再生可能エネルギー電源の導入が進んでいる。しかしながらこれらの電源の発電電力は気象条件等によって変動するという性質から，電力系統へ大量導入された際に,電力系統内の電圧や電力需給バランスに悪影響を与え，系統運用者が適切に運用することが困難になる可能性がある。その従来の対策としては，電圧や電力需給を調整するための機器の設置が挙げられるが，その機器設置には多くのコストを必要とするという問題がある。この問題点を解決するために，需要家が所有する可制御機器（リソース）を活用することが検討されつつある。

そのリソースの中で，電気自動車用急速充電器（以下，充電器）は，需要家リソースの中でも比較的大容量機器であり，原理的には充電の有無に関わらず無効電力補償（発生・吸収）が可能であることから，現在電圧制御機器として使用されている静止型無効電力補償装置の代替となることが期待される。しかしながら，充電器は，通信機能を持たない，需要家の都合(充電)により補償可能な無効電力が制限される等，電圧制御機器にはない特有の性質を有する。そこで本論文では，これらの特徴に配慮した充電器の無効電力補償を用いた電圧制御手法を提案する。

一方，電力需給面に関しては，これまで行われてきた火力発電機等の出力調整能力（調整力）の代替として，需要家リソースの消費（あるいは発電）電力を調整することが検討されている。しかしながら，有効電力の調整がリソースを所有している需要家への経済性に与える影響は大きく，リソース活用に伴うコストの増分と対価（報奨金）を比較しながら経済的に運用するような手法が必要である。また，需要家のリソースは比較的小規模かつ多数存在しているという性質から,多数のリソース群を効果的かつ経済的に活用するためのアグリゲータの存在が不可欠であり，このアグリゲータの役割を含めた，リソース活用を目的とした運用手法の検討が必要である。そこで本論文ではコージェネレーションシステム（CGS）に着目し，調整力提供を目的とした CGS 群を統括・運用するアグリゲータの運用手法，および CGS 所有の需要家の経済性が最適となるような，需要家の運用手法を提案する。

また提案手法を適用した数値試算により，これらの需要家リソース群の活用能力を評価する。

キーワード：電圧制御，電気自動車用急速充電器，需給調整，需要家リソース

*北海道大学 大学院情報科学研究科システム情報科学専攻，博士論文，SSI-DT79175026，2019 年 2 月 13 日

Operation of customer's resources contributing stability of power system†

Yuta NAKAMURA

Abstract

For low carbon society, naturally fluctuating power supplies such as photovoltaic generation system and wind turbine generator are being installed in power systems, however, these power supplies with natural variability decrease stability of voltage and/or power balancing (supply and demand regulation) in the system. As conventional measures, the installation of regulation equipment is used, however, it is remained costly to install additionally. To resolve this problem, utilization of device (resources) with controllability, which is owned by customer, is considered.

Rapid charger (RC) for electric vehicles (EVs) which is one of customer's resources, would be suitable to regulate voltage in distribution system from volume of rated capacity and component of a typical RC. A novel method is required since a typical RC has two different characteristics from voltage regulation equipment: One is that reactive power compensation range is pushed away by current capacity of inverter when RC is utilized as EV charging. Another one is that a typical RC has no communication between other RC and voltage regulation equipment. Therefore, this paper proposes a new voltage regulation method that utilizes the reactive power compensation by RCs, which be employed autonomous and decentralized approach.

On the other hand, as alternative measures for power balancing control by generator, regulating consumption/generation power of customer's resource is considered, however, the customer impacts profitability by the regulation of active power because of decrease income or increase payment in terms of energy trade. Therefore, a new method to operate economically by means of comparing additional cost and benefit is required. Moreover, since, the size of a lot of resources for single consumer is small and the amount and the time with respect to available regulation capacity depend on the demand of each customer, aggregation of multiple resources is advantageous in realizing more efficient and flexible contributions. Therefore, the authors focus on co-generation system (CGS) as one of power balancing resources, this paper proposes a novel optimal operation strategy of CGS coordinated by the aggregator considering the energy balance and operation cost of individual CGS owner.

This paper also demonstrates the availability of these customer's resources for regulation ability through numerical case studies is employed.

Key words: voltage regulation, rapid charger, supply and demand regulation, customer's resources

†Doctoral Thesis, Division of Systems Science and Informatics, Graduate School of Information Science and Technology, Hokkaido University, SSI-DT79175026, February 13, 2019

目 次

図目次

表目次

第1章　序論

1.1　本論文の背景

世界における近年の電力エネルギー事情は，人類が生活する上でますます欠かすことのできないものとなっており，新興国は生活水準の向上，先進国は中心に IT 等を用いた機器間の通信機器の増加を背景に，電力エネルギーに関する需要の増加は今後も続くであろう。我が国でも，年間電力消費量は，1970 年では約 3500 億 kWh，2016 年では約 9500 億 kWh と高度経済成長期以降大幅に増加している[1]。また，図 1.1 は主要国における電力消費量を賄うための一次エネルギー消費の内訳を示している。同図より，我が国では大多数が石油や石炭，ガスなどの化石燃料であり，これらの自給率は約 6.2%程度と他の主要国に比べて低いことがわかる。この主な要因としては，これらの化石燃料の資源が乏しいことと，2011 年に発生した東日本大震災の影響から原子力発電の多くが安全性確保のために停止していることが挙げられる。近年では原子力発電の再稼動が進められてきているが，生活に欠かすことのできないエネルギー資源の大部分を海外から確保しなければならない。それゆえ，化石燃料の輸入国との関係や社会情勢によって燃料価格が高騰したり，供給そのものが遮断されたりなど，いわゆるエネルギーセキュリティに関する問題が我が国の重要な課題となっている現状がある。

また，エネルギー事情に関連して世界中で取り上げられている問題の一つとして，地球温暖化をはじめとする地球環境問題がある。二酸化炭素に代表される温室効果ガスはこれまで，生態系等の地球環境に適した量に保たれていたが，化石燃料の大量消費に伴い二酸化炭素等が多く排出され，地球全体の温室効果ガスが増加することで，世界全体の平均気温が増加したり，気候が変動したりするなどの地球環境のバランスを崩しているという報告も数多くなされている。これを踏まえ，近年世界中で二酸化炭素の排出量低減を目標とした対策がされつつある。

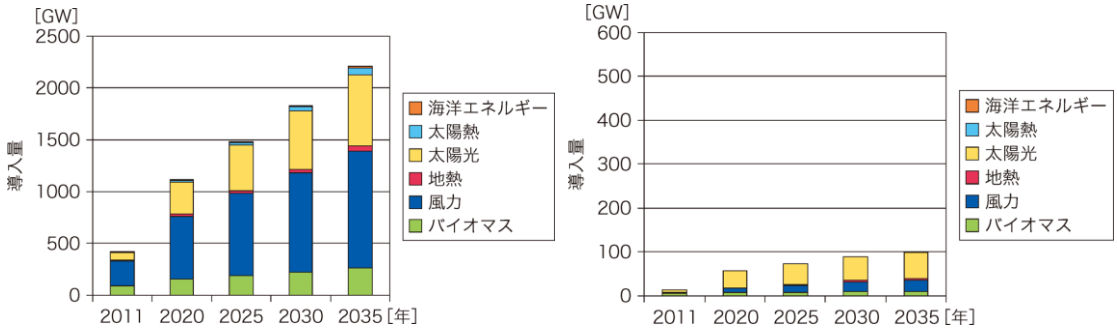
図 1.1 主要国の一次エネルギー消費構成と自給率（2015 年）[2]

このような背景から，我が国をはじめ世界中で，近年化石燃料に替わるクリーンなエネルギーが注目されており，その中でも特に再生可能エネルギーが脚光を浴びている。再生可能エネルギーとは我が国の法律で「エネルギー源として永続的に利用することができると認められるもの」と定義されている。再生可能エネルギーには，以前から発電に使用されている水に加えて太陽光や風力，地熱や太陽熱に代表される自然界に存在する熱，バイオマス等が含まれ，これらは一般的に資源の枯渇がなく，エネルギー利用時に二酸化炭素を排出しない[3]。そのため，再生可能エネルギーは地球環境に悪影響を与えずに利用することができるエネルギー源とされており，世界中で再生可能エネルギーを利用した発電システムの導入が進んでいる。

しかし一方で，再生可能エネルギーを利用した発電のコストは化石燃料をはじめとする従来電源よりも高いという問題がある。我が国では，再生可能エネルギーを利用した発電設備を設置しても十分採算が取れるように発電電力買取価格を一定期間固定にする「固定価格買取制度」が制定されている。これにより再生可能エネルギーの買取電力量は，固定買取価格制度が開始された 2012 年度から 2016 年度の 4 年間で年間約 56 億 kWh から約 430 億 kWh まで増加[4]し，図 1.2 のように，我が国だけでなく世界全体で将来に向けた導入拡大が見込まれている。

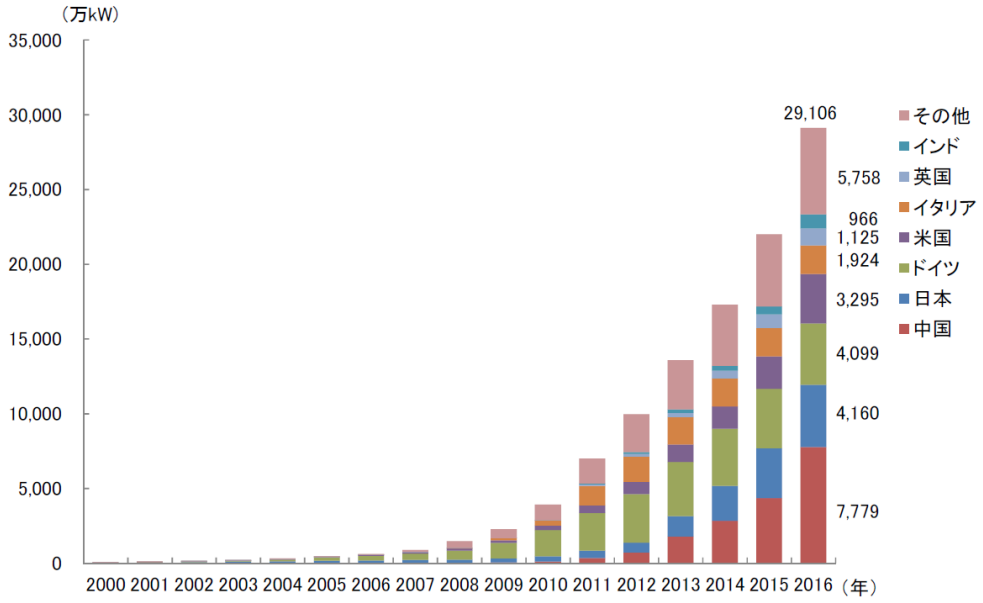
しかしながら，これらの再生可能エネルギーを利用した発電の多くは，天候や気温などの気象条件によって発電量が変化するという特性がある。特に太陽光発電や風力発電は「自然変動電源」とも呼ばれており，自然変動電源は図 1.3 に示すように更なる導入が予測されており，発電出力の変動増大によって電力系統の安定性が低下することが懸念されている。化石燃料を用いるような従来電源を主とした電力系統の運用では安定的な電力供給が比較的容易であるため，各電力会社の管轄エリア等，比較的広い範囲を対象にしているが，自然変動電源が大量導入されている場合では，より細分化した範囲を対象として管理・運用しなければならない。

したがって，化石燃料の枯渇と地球環境をはじめとするエネルギー問題を解決するために，ますます増加することが期待される，自然変動電源に代表される再生可能エネルギーを活用することを前提とした電力系統の運用方法や技術が必要不可欠であろう。

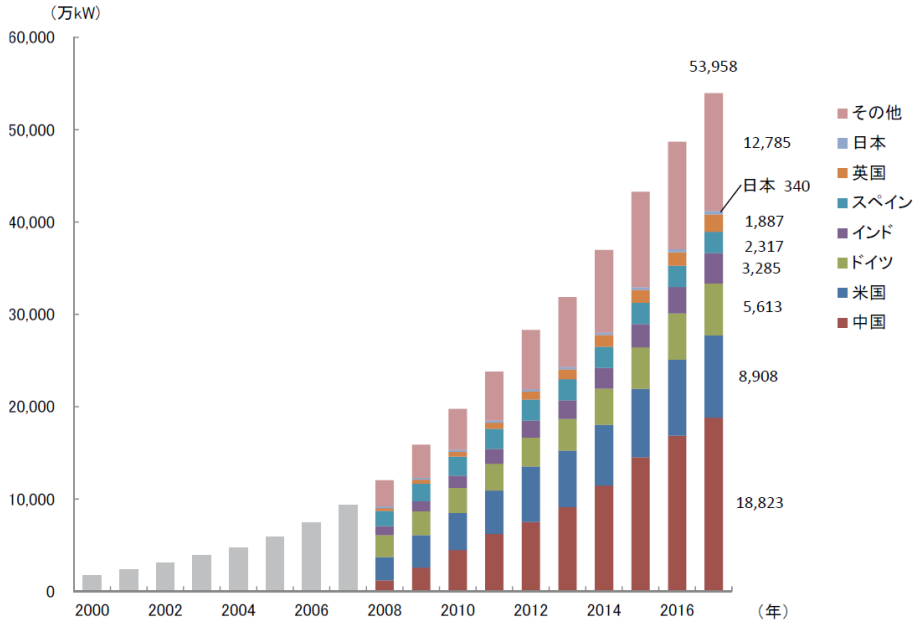


(a)世界　　(b) 日本

図 1.2 再生可能エネルギーの累積導入量予測[5]



(a) 太陽光発電



(b) 風力発電

図 1.3 世界の自然変動電源の導入状況[6]

1.2　自然変動電源の出力不安定性が電力系統に与える影響

前節でも述べたように，自然変動電源は気象条件により発電電力が大きく変動する電源のことであり，太陽光発電(Photovoltaic power generation: PV)や風力発電(Wind turbine generator: WT)がこれに含まれる。

PV は太陽電池パネルにより太陽光エネルギーを電気エネルギーに変換する装置である。太陽エネルギーの総量は膨大であり，発電する際にも環境への排出物は全くないことから最もクリーンなエネルギーともされている。PV は半導体で構成される太陽電池を複数合わせることで構成され，その太陽電池の構成を設置場所や関連機器の容量等に応じて容易に変化させることができるのも他の電源にはないメリットである。このことから，PV システムは一般住宅の屋根等のあまり利用されないスペースに導入できる。また PV システムは太陽電池やパワーコンディショナ，各種連系器等から構成されるが，可動部がなく基本的にメンテナンスフリーであることから設置・保守運用が比較的手軽である。

一方，WT は風力タービンにより風力エネルギーを回転エネルギーに変換，更にそのエネルギーを発電器により電気エネルギーに変換する。WT の特徴としては，基本的に機器で設定されている最大風速以下であれば風速の 3 乗に比例した発電を行うことができる。風力エネルギーの約 40%を電気エネルギーに変換でき，PV よりも高効率であることや設置コストが比較的低く，PV システムの約 5 分の 1 程度であることなどが挙げられる。

図 1.4 は我が国における再生可能エネルギーの設備容量を示している。WT は 2000 年から 2004 年にかけて設備容量が大きく増加しているが，2010 年以降も伸びは減少しているが依然として増加傾向にある。

一方 PV は，導入成長率は 2000 年後半に一度多少落ち込んだものの，一定の成長率を維持している。また図 1.5 に PV の国内導入量と PV システム価格の推移を示すが，PV システムの発電コストは従来の電源に比較して高いが年々減少傾向である。これは前節でも述べたように再生可能エネルギーの固定買取制度(Feed-in Tarif :FIT)により買取価格が比較的高値で設定されているため，一定期間発電(売電)すれば多くのケースで十分な採算が取れる。これらの背景から我が国の PV システムの導入台数は年々増加しており，今後もその傾向が続くことが予想される。

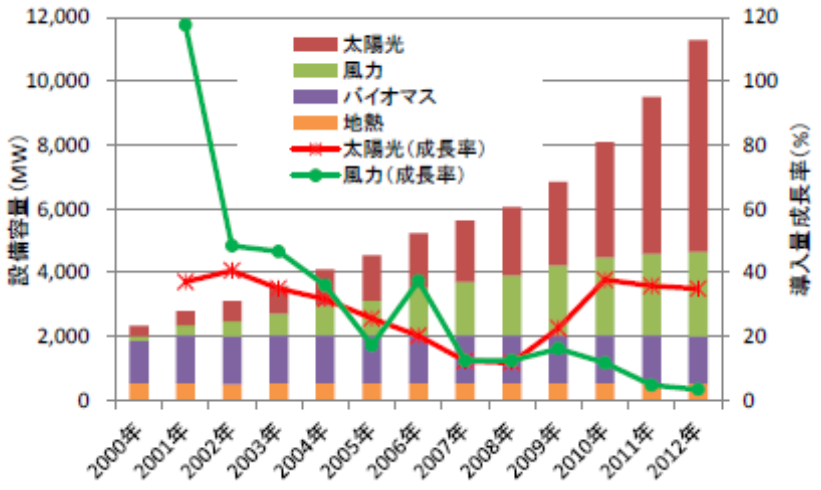


図 1.4 日本の再生可能エネルギーの設備容量[7]

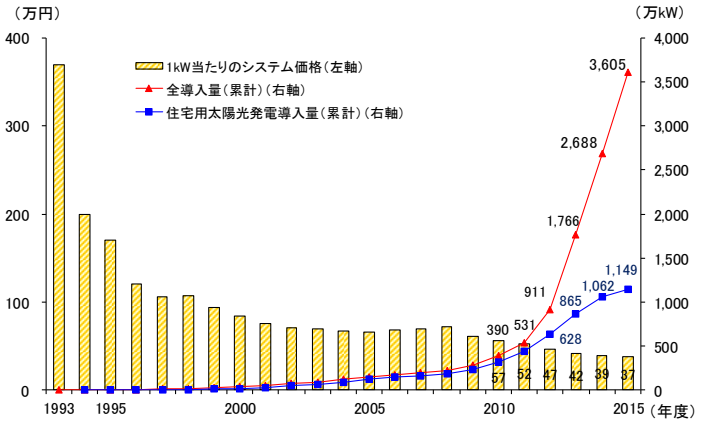


図 1.5 太陽光発電の国内導入量とシステム価格の推移[8]

以上の理由から，PV や WT に代表される自然変動電源は，将来的に多く導入されることが予想されるが，問題も存在する。固定価格買取制度による電気料金の増加などの金銭面の問題もあるが，特に電力系統運用者の運用面での問題が深刻である。PV システムの出力は日射量に依存するが，雲や太陽の移動等によって影が発生(または消滅)した場合，瞬間的な日射量の変化に伴って大きな出力変動を引き起こす。この出力変動が需要と供給のバランスに影響を与え，これまでの系統運用者による高品質・高信頼度の電力供給が困難になる可能性がある。

これらの自然変動電源の出力不安定性が電力系統に与える影響を，電圧面は 1.2.1 項，電力需給面は 1.2.2 項にそれぞれ分けて述べる。

1.2.1　電力系統の電圧に与える影響

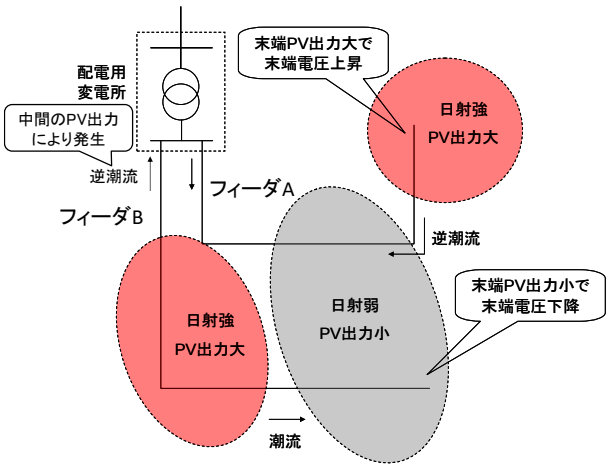
図 1.5 に示したように，PV は自然変動電源の中で一般住宅に導入容量および導入数が多い。これは，電力系統における電圧階級の中で一番低い，一般的に住宅が連系されている 100V または 200V の低圧配電系統に PV が集中的に接続されていることを意味している。

一般的に電力系統では，発電所から複数の変電所を経由して電圧階級を下げて，配電用変電所で低圧配電系統に小分けにして各需要家へ分配している。電力の潮流方向は系統の上位側から下位側に向かう(順潮流)であるが，PV の発電量が負荷需要よりも大きくなるとその潮流方向が逆になり，下位側から上位側に向かう向きとなる(逆潮流)。逆潮流が発生すると，電力潮流の方向から上位側よりも下位側の方が電圧は上昇する。さらに PV が大量に配電系統に導入されると系統内の電圧上昇がより顕著になるが，その導入が局所的になると系統内の電圧プロファイルも複雑化し，系統運用が困難になる。電気事業法では，「電圧を 100V の場合は 101±6V，200V の場合は 202±20V の電圧適正範囲内に収めなければならない」という規定となっているが，PV が大量導入されている系統ではその電圧適正範囲の上限値を逸脱する可能性がある。もし全ての低圧配電系統に PV が均一に導入され，また一定出力であれば，系統全体の電圧を下げるような対策を施すことで解決できる。しかしながら，前節で述べたように PV が発電ポテンシャルの高い地域を局所的に大量導入されることが予想されることから，系統全体の制御や対策が好ましくない可能性が高い。この解決策として，必要に応じて PV の出力を抑制する等の対策を講じることが現在

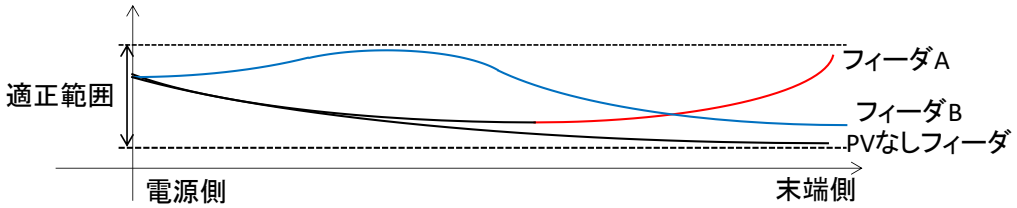
義務付けられている[9]が，出力抑制は売電機会や売電収入の減少に繋がる等の PV の所有者側への影響があるため最小限に抑えなければならない。

その例を図 1.6 を用いて説明する。多くの配電系統では図 1.6(a)に示すように配電用変電所から数 km の恒長のフィーダを複数有している。その中のあるフィーダ(フィーダ A)では，配電系統の末端側の PV 出力が大きいため，逆潮流が発生しているが中間付近では PV 出力が小さく，負荷の方が多いという負荷分布となっている。一方，別なフィーダ(フィーダ B)では電源側では PV 出力が大きく，末端側では出力が小さいことから中間付近から変電所に向かって逆潮流が発生している。このような現象は太陽や雲との位置関係の変化から数 km オーダ内でも発生が確認されている。このときの電圧分布は図 1.6(b)に示すように PV が導入されていない順潮流フィーダと PV が導入されており逆潮流が発生しているフィーダで大きく異なり，PV の導入により電圧分布が複雑化していることがわかる。したがって，低圧配電系統への局所的な PV 大量導入によって生じる，系統内の電圧分布の複雑化に対応した電圧管理・対策に関する検討は不可欠である。

従来行われている対策としては，配電用変電所に設置されている負荷時タップ切替変圧器(Load ratio Transformer: LRT)を用いた電圧管理が挙げられる。この LRT に着目して，自然変動電源導入による電圧分布が複雑化される電圧プロファイルを推定し，適切な LRT の送出電圧（タップ位置）を決定する手法が提案されている[10][11]。これらの手法では，機器の長寿命化および電圧適正範囲逸脱までの余裕の最大化を目的とした送出電圧決定がされるが，自然変動電源の出力が完全に予測可能であることを前提としている。また，LRT の送出電圧の変更だけでは局所的な電圧分布の変化に対応できない可能性が高い。



(a) PV 出力による潮流変化の例



(b) PV 出力による電圧分布変化の例

図 1.6 PV 出力による配電線潮流変化と電圧分布例

その LRT で賄うことができないような局所的な電圧分布変化に対応するために，近年，静止型無効電力補償装置(Static VAR Compensator：SVC)に代表される無効電力補償装置を配電系統内に設置することが検討されている。SVC は無効電力を補償することにより電圧制御できる機器であり，その補償量は電力用コンデンサや分路インダクタの投入･開放などの離散的な制御ではなく，投入する容量(kvar)を連続的にすることができる。そのため，SVC のこの特徴を活かした配電系統の電圧制御手法についても研究がなされている。文献[12]では電圧不感帯を設定し，電圧を電圧不感帯に収まるように連続的に制御する手法，文献[13]では，電圧の時間移動平均値を目標電圧値として制御する手法がそれぞれ提案されている。しかしながら，SVC は依然として設置コストが高いという経済面の問題がある。

また PV が導入されている場合には PV 自身の PV 用パワーコンディショナ(Power Conditioning Subsystem: PCS)等の，自然変動電源を含めた分散型電源の力率制御を用いた電圧制御手法が提案されている。文献[14]では PV 出力変化に応じて無効電力補償を行う電圧制御手法が，文献[15]では複数設置された分散型電源が電圧変化割合(インピーダンス)を推定し，その値に基づき無効電力補償を行う電圧制御手法がそれぞれ提案されている。また，文献 [15][16] では複数 PV 間の協調，文献[17][18]では複数 PV と複数 WT 間の協調と，各機器の特徴を活かした制御分担による電圧制御手法が提案されている。しかしながら，分散型電源は需要家や特定電気事業者等が所有している場合が多く，これらの制御手法の実施に対して何らかの補填(対価)が必要であると考えられる。そこで文献[19][20]では，分散型電源の力率制御（無効電力制御）を前提とした経済制度を提案しており，発電電力抑制および無効電力補償に伴う収入減少を金銭面で補償することを想定する。なお，[16]～[20]で挙げた文献は，いずれも情報通信技術を用いており，機器間の通信が可能であることを前提としている。

これまでに述べた電圧制御手法は単独あるいは複数の分散型電源に着目して研究・提案されてきたが，配電系統では LRT や SVC，PV を含む分散電源等，分散型電源と制御機器を組み合わせて活用することを前提とした，複数機器間の協調制御手法も検討されている。例えば，LRT と複数の自動電圧調整器(Step Voltage Regulator: SVR)の協調制御手法[21]，LRT との協調を前提とした SVC の電圧制御手法[22]，LRT と PV の力率制御の協調制御手法[23]，SVR と SVC の協調制御手法[24]，LRT，SVR および自励式 SVC(STATCOM)の協調制御手法[25]等である。上記のような多種・多数の機器間の協調制御を前提とした電圧制御・管理手法が数多く提案されており，これらの制御手法では，時間次元もしくは制御範囲等で役割分担を行う等の配慮がなされている。

1.2.2　電力需給バランスに与える影響

一方，自然変動電源に伴う発電電力(出力)の不安定性は，電力系統内の電圧以外に電力需給バランスにも大きな影響を与える。電力系統における需給は同時同量制約を満足させる，すなわち需給のバランスを常時取る必要がある。この制約が満たされない場合にはバランスの取り具合を示す周波数に影響を与え，周波数が既定されている範囲を逸脱する。さらに需給バランスが悪化すると，機器損傷等の影響を回避するために発電機を停止し，大規模な停電を招く可能性がある。したがって，電力系統内の電力需給の変化を常時モニタリングし，適切に制御しなければならない。

電力需要の変動は図 1.7 に示すように，一般に変動周期によって 3 つの部分に大別される。周期が数分程度までの微小変動成分（サイクリック成分），数分から数十分程度までの短周期変動成分（フリンジ成分），それ以上の長周期変動成分（サステンド成分）の 3 つである。これら 3 つの部分に対して，それぞれ異なる手法を用いることによって変動に対応することが一般的である。

微小変動成分については主に負荷の自己制御特性によって変動を解消し，それ以外の数十秒から数分程度までの成分についてはガバナフリー運転（Governor Free: GF）によって解消する。数分~十数分程度の短周期変動分については主に負荷周波数制御（Load Frequency Control: LFC）によって制御が行われる。LFC は中央給電指令所の自動周波数制御装置により周波数偏差を検出し，各発電所に偏差を解消するための制御信号を伝送することによって発電所出力を自動制御するものである。それ以上の変動成分である長周期変動成分に対しては，経済負荷配分制御（Economic Load Dispatching Control: ELD）が用いられる。これは発電効率の異なる各発電所（発電機）において最も経済的な出力配分を計算し，発電機出力を制御するといったものである。

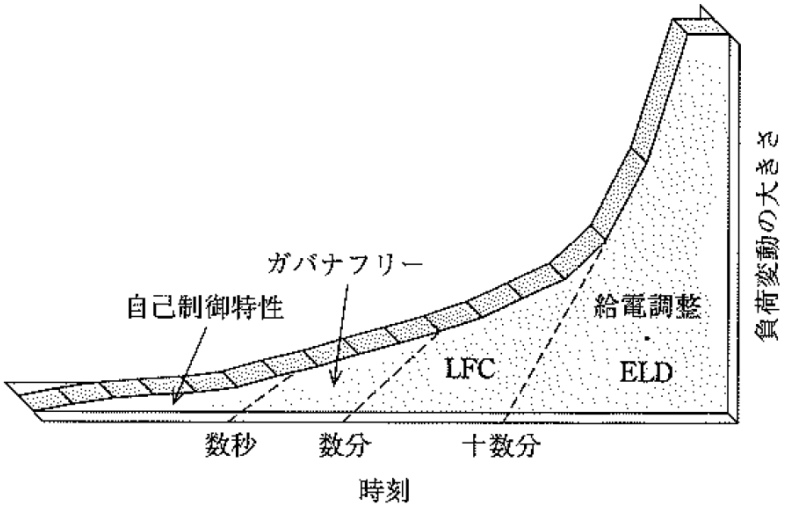


図 1.7 需要変動の制御分担[26]

現状では，火力発電機等の従来型電源が電力需給を調整することで安定化させる役割を果たしてきた。しかしながら，電力需給を不安定させる自然変動電源の大量導入に伴い，これらの既存の電源についてはすでに最大限度に近い形で利用されており，系統全体の発電量が自然変動電源へシフトしていく中で更なる需給調整する能力（以後，調整力）を捻出することは難しい。これは，また電源構成のうち自然変動電源の割合が増加していくことで，そもそも需給調整に利用可能な電源起動台数が減少すること，ならびに電源の出力レベルが最低出力付近となり発電出力減少方向の調整力が利用しづらくなる，いわゆる下げ代不足の両面から，更なる調整力を捻出することは困難であるといえる。また，自然変動電源の大量導入への対応として従来型電源の新規設置が考えられるが，設置に伴うコストや土地等の制約を考えると，自然変動電源の導入に追従して設置を繰り返すことは非常に困難であると考えられる。

これらの背景から我が国でも近年，調整力の取引を目的とした需給調整市場の創設に向けた議論がなされており，系統運用者が効率的に調整力を確保する手法や制度が検討されつつある[27]。その中で電力系統の運用に必要な調整力そのものを抑えるために，自然変動電源の発電出力を抑制することも検討されている[28]が，前項で述べた電圧面と同様，出力抑制の多頻度化・長期化に伴う発電機会の逸失や，それに伴う発電事業者の経済性への影響が懸念される。また出力応答に優れ，エネルギー貯蔵が可能な大型の蓄電池を，GF や LFC 制御として活用することも検討されている[29][30][31][32]。しかしながら，蓄電池のコストは依然として高いという問題点があり，蓄電池の設置コストを下げる構成機器の技術開発とともに，最小限の蓄電池容量で目的を満足するような制御方式・手法について検討されている。

したがって，環境負荷の少ない，クリーンな自然変動電源の更なる導入を促進するために，各種制御機器の設置費用をかけずに電力系統内の電圧や電力需給の安定性を担保するような，電力系統の運用に関する技術開発が不可欠である。

以上の背景から，系統運用側が所有している機器に加えて，需要家が所有している可制御機器（リソース）を活用することが検討されており，次節ではリソースの概要およびその活用例について述べる。

1.3　電力系統の安定化を目的とした需要家リソース活用研究の動向

前節では，PV 等の自然変動電源の大量導入による電圧管理および電力需給への影響と現状について述べた。本節では，需要家側として利用される機器やその機器に付随する機能により，電圧制御や調整力提供などの系統運用に貢献可能な需要家の可制御機器（リソース）の活用事例について述べる。

これらの需要家リソースには図 1.8 に示すようなシステムに付随する機器が該当し，代表的な機器について後述する。

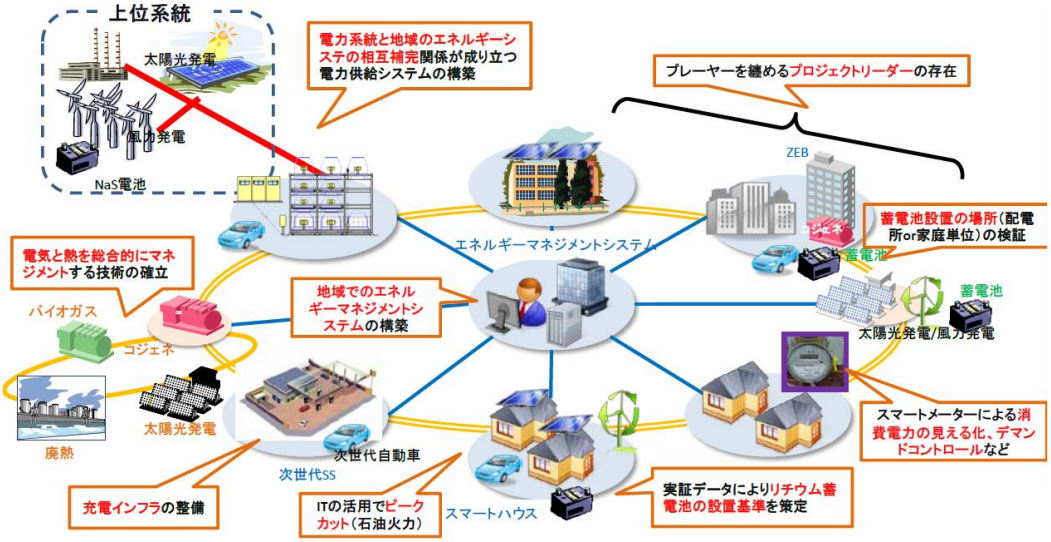


図 1.8「次世代エネルギー・社会システム実証事業」のイメージ図[33]

1.3.1　代表的な需要家リソース

(1)分散型電源 (Distributed Generators : DG)

DG とは，上位系統に接続され下位系統に電力を供給する従来(一方向)型の電源とは異なり，配系系統等の受電側の系統に直接接続される電源のことである。DG の種類としては，PV や WT 等の自然変動電源をはじめ，ガスエンジン発電機やディーゼル発電機，バイオガス発電機の可制御電源が挙げられる(コージェネレーションシステムも DG に含まれるがここでは別途記載)。しかしながら前節で述べたように，系統内の電力需要よりも供給量が多くなり逆潮流が発生すると，系統内の電圧が上昇し電圧管理が複雑になるという問題がある。そのため，各 DG は対象エリア内の需要と供給のバランスを監視しつつ，発電電力を適切に管理する必要がある。

DG を活用した電圧管理方法として，DG の力率制御(無効電力制御)がある。特に PV 等の交直変換用のインバータを有している DG は，このインバータのスイッチングのタイミングを適切に制御することで無効電力を制御することが可能であるとされている。系統の R/X 比 (抵抗とリアクタンスの比率)によっても異なるが，一般的に電力系統では無効電力の方が有効電力よりも電圧に与える影響(電圧感度)が高い。したがって無効電力を利用するのが効率的であるが，無効電力を大きくすると DG のインバータの電力容量を超える可能性がある。その場合はあらかじめ決められた力率以上(85%程度)を維持しつつ有効電力を抑制させ，空き容量を無効電力として利用する。この場合，連系点の電圧は図 1.9 のように有効電力の発電量(逆潮流量)の減少に加えて，遅れの無効電力（DG からみると消費）の増大により低下する。このように DG の力率制御は制御効果が高いことから注目されており，力率制御を用いた電圧制御手法およびその効果については，文献[16][17][18]で報告されている。

一方，自然変動電源の出力変動を低減する手法として，電力需給バランスを保つように，可制御電源の出力を制御することが行われている。電圧変動対策は自然変動電源が接続されている連系点付近で発生する局所的な問題である一方，電力需給面は電力系統全体の広い範囲の問題である。そのため，電力需給の安定化に貢献する可制御電源は，電力系統全体の需給バランスを一致するように，系統運用者からの指令等に基づき発電出力を調整する。

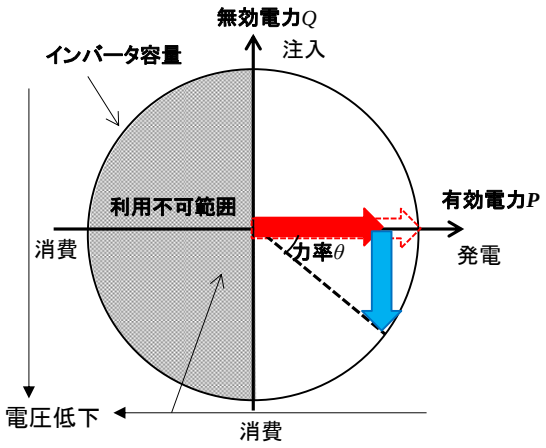


図 1.9 DG の力率制御のイメージ

(2) コージェネレーションシステム (Co-generation system : CGS)

前述したように， CGS も分散型電源(DG)として扱われることが多いが，本論文では CGS を取り扱っているため，別途分けて記載する。CGS は発電装置により発電を行いつつ，その発電によって生成される熱（排熱）を回収し，その熱をエネルギーとして活用するシステムのことを指す。ガスエンジン CGS やガスタービン CGS では，発電に伴って生じた高圧ガス（以下，排ガス）や低温熱を，冷房・暖房・給湯需要などに利用する。燃料電池では化学反応の際に発生する熱を回収する。

また近年は，電気を水素に変換する水素製造設備の普及に伴い，特に燃料電池 CGS の導入が急速に増えている。燃料電池は表 1.1 に示すように，個体高分子型(PEFC)，リン酸型(PAFC)，溶融炭酸塩型(MCFC)，固体酸化物型(SOFC)の 4 種類に大別される。その中で，2009 年から日本で販売されている家庭用燃料電池エネファームは，図 1.10 に示すように導入が加速しており，販売価格の低下がさらに進めば今後更なる導入が見込まれるだろう。

CGS の持つ利点として，発電と同時に利用するため高効率，省エネルギー・省コストであることが挙げられる。発電と熱を含めた総合効率は最高で 80%程度とされており，従来の火力発電効率よりも非常に高い，また，需要消費地に近い箇所で発電・熱供給できるため，エネルギーの伝送に関するロスやコストが少ないことが高効率である要因である。また，CGS の多くは比較的 CO2 排出量が少なく，石炭などの化石燃料を使用した火力発電機よりも環境負荷が少ない。さらに，CGS は上位系統が故障等により遮断された際も電力を供給することができ，病院等の人命に大きく関わる施設の非常用電源にも使用される。このような要因から，CGS は今後ますます導入が期待されており，日本ガス協会では 2030 年までに約 3000 万 kW の CGS の導入(電源構成の約 15%)を目標としている。

自然変動電源の出力変動対策として CGS を活用した研究は，文献[36][37][38]等が挙げられる。文献[36]では，CGS が導入されているマイクログリッドにおいて，WF の出力や需要の変動に対応することを目的とした最適運用計画手法を提案している，CGS からの排熱を貯めておく蓄熱槽の状態により起動停止させる，といったような CGS の実情に近い状態でのモデル化した検討がなされている。一方，文献[37][38]では酪農の廃棄物であるバイオガスを用いたバイオガス発電機による自然変動電源の出力変動対策について検討しており，電気と熱に関する制約に加えて，バイオガス生成(燃料)のタイミングによる影響を考慮した運用計画手法ならびに制御アルゴリズムの開発が行われている。

表 1.1 燃料電池の種類[34]

	固体高分子形（PEFC）	リン酸形（PAFC）	溶融炭酸塩形（MCFC）	固体酸化物形（SOFC）
電解質	高分子電解質膜	リン酸	炭酸塩	セラミックス
作動気体	水素			
			一酸化炭素	一酸化炭素
作動温度	常温～90℃	150～200℃	650～700℃	750～1,000℃
発電効率	30～40%	35～42%	45～60%	45～65%
主な用途	家庭用	工業用	工業用	工業用
	燃料電池車用	産業用	分散電源用	分散電源用

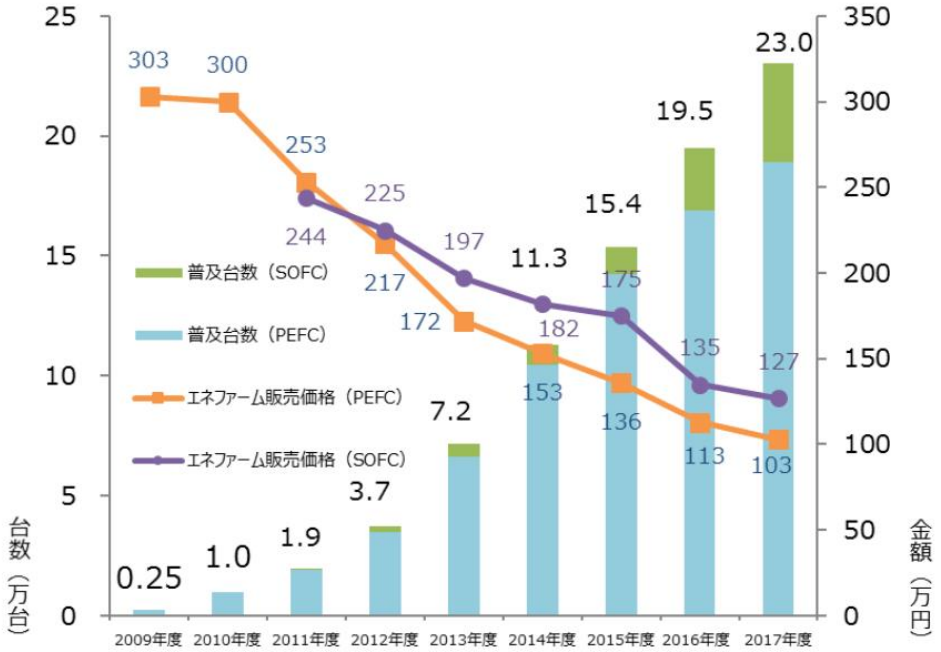


図 1.10 家庭用燃料電池の普及台数の推移[35]

(3) エネルギー貯蔵装置(蓄電池等)

蓄電池に代表されるエネルギー貯蔵装置は電気エネルギーを蓄えたり放出したりすることで，電力の需給のタイミングを変えることができる装置である。貯蔵するエネルギーとしては，現状蓄電池のように化学エネルギーが広く利用されているが，熱電変換装置での電気変換に利用される熱エネルギー，燃料電池に利用される水素エネルギー等の別なエネルギーに変換する装置も利用されつつある。

蓄電池の種類としては，リチウムイオン電池，鉛蓄電池，NAS 電池，レドックスフロー電池等があるがそれぞれ特徴があり，現状では用途や貯蔵する電力量，設置個所等に応じて利用する種類を決定することが多い。また，近年では電気を水素に変えて貯蔵し，燃料電池車などの水素需要として利用する等の検討もされている。

このようなエネルギー貯蔵装置の主な用途としてはピークシフトによる負荷平準化や電気自動車等の遠隔地や非常用の電源利用がある。電力自由化が進めば電気料金の価格は変動することが予想され，また昼夜料金や需要，時間帯等に応じて価格設定も行われることが予想される。そのため，「需要が少なく電気料金の安い時間に，需要以上電力を購入し蓄電池に充電しておき，電気料金の高い時間には電気を購入せずに蓄電池から放電される電力を利用する」といった使い方も今後進むことが予想されることから，蓄電池利用は今後ますます広がっていくだろう。

これらのエネルギー貯蔵装置の制御は，一般的に電力需要をある目的に達成するための有効電力（充放電）の制御であるが，蓄電池の充放電システム等のように交直変換器(インバータ)を有している場合には，有効電力だけでなく無効電力も制御できることが可能である。その場合には有効電力・無効電力の両者がインバータ容量以内であればどの向き・量にも制御することができることから，図 1.9 に示した DG の力率制御よりも柔軟な制御が可能となる。これに着目し，蓄電池の充放電システムの有効・無効電力に用いた電圧制御手法について検討されている[39][40]。

(4) 電気自動車 (Electric Vehicle : EV)

多くの場合で電気自動車(EV)は蓄電池の一つと考えられるが，本論文では EV について取り扱っているため，(3)の蓄電池とは別途記述する。まず EV は，動力源のモータを有する自動車，エンジンとモータを両方使用したハイブリット車，ハイブリット車に電源から充電する機能を有するプラグインハイブリット車に分類することができる。2013 年に閣議決定された「日本再興戦略」の中の「戦略市場創造プラン」では，2030 年までに全自動車の約 30%を EV とする普及数値目標が設定されている。また，政府や自治体による EV の導入に対する補助金助成などもあり，EV の普及は今後さらに進むことが予想される。

EV の充電器に利用される充電器は表 1.2 に示すように，「普通充電器」と「急速充電器」がある。普通充電器は一般的に EV での移動前後あるいは利用しない時間帯に充電を行うことを目的としており，家庭やオフィス等に設置される。その多くが単相 200V を入力とし，定格出力が 3kW 程度の仕様であり，現在販売されている EV の蓄電池容量は 15～24kWh 程度であることから，充電に最大 5～8 時間程度要する[42]。このときの有効(充電)電力制御を電力系統の運用に貢献することが検討されているが，その制御は機器構成上，EV が系統に接続(プラグイン)されているのみ実施することができ，尚且つ制御により次の出発までに充電が完了しなければ EV の利用ができない，あるいは移動の途中で継ぎ足し充電する可能性が高くなるなど従来利用および需要家の都合による制約・制限が発生する。そのため，この利便性への影響の考慮を前提とした負荷平準化[43]-[46]や周波数制御(LFC)[47]-[52]を目的とした EV の有効電力制御手法が複数提案されている。また，EV が配電系統の同一フィーダ大量に導入された場合，多数の EV が夜間に一斉に充電されることによる電圧降下問題の発生が懸念されている。この対策としては EV の普通充電器のインバータを活用した有効・無効電力制御による電圧制御手法[53][54]が提案されている。また，近年は，EV に充電した電力を放電し一般住宅などの電力需要に使用する「V2H(Vehicle to Home)」に対応した，充電・放電とも可能な充電器の導入が進んでいる。現在，普通充電器はコンセント・ケーブルタイプ合わせて約 80 万台出荷・利用[55]されており，今後更なる増加が予想される。

一方急速充電器は，近年 EV での目的地までの移動中や一時的な駐車中に利用する用途として市街地を中心に設置にされつつある。急速充電器は普通充電器よりも短時間で充電を完了させることを目的としているため，定格出力は最大 50kW と大きく，現在はさらに大容量化に向けた開発がされている。そのため，急速充電器の電力系統との連系は，普通充電器と異なり三相 200V であり，配電系統の高圧側(6.6kV)に変圧器を介して接続されることが多い。急速充電器は設置するための工事・手続きが多くことから，コンビニエンストア等の商用施設，充電スタンド(従来のガソリンスタンドに相当)等の交通量が多い施設に導入されている。現在日本では，約 7400 台の急速充電器が設置[56]されており，EV の導入促進に伴い更に増加していくだろう。

表 1.2 普通充電器および急速充電器の主な仕様[41]

	普通充電器	急速充電器
充電時間	5～7時間程度	30分程度
入力電圧	単相200V 交流	三相200V
定格出力	3kW	最大50kW

(5) 可制御性を有する負荷

これまで述べたリソースは発電あるいは蓄電を目的としたものであるが，上記以外に可制御な電気的な負荷を活用することが検討されている。その主な機器としては，ヒートポンプ給湯機[50],[57]-[63]，空調設備[64][65]などが挙げられ，各機器の特性や需要家の本来の目的（給湯や施設内の温度等）を考慮した周波数制御(LFC)の手法について検討されている。

1.3.2 需要家リソースの活用方法とアグリゲータの役割

ここでは，前項で述べたリソースをどのように制御・活用しているかについて述べる。比較的容量の大きい可制御性を有する DG 等の発電機については，多くが系統運用者からの出力指令に従って発電電力や無効電力を調整するが，需要家が所有しているエネルギー貯蔵装置や可制御負荷等については，系統運用者が直接指令することは接続されているリソース数やセキュリティ等の観点から非現実的であると考えられる。このような需要家リソース特有の性質から検討されている活用方法例について述べていく。

まず，需要家リソースの活用を促す方法として検討されているのが，電気料金や電力負荷のシフトに対するインセンティブ等の恩恵によって機器の発電・消費パターンを変更させる，デマンドレスポンス(Demand Response : DR)と呼ばれる考え方である。DR は主に(a)電力料金型 DR，(b)インセンティブ型 DR の 2 つに分類されている。(a) 電力料金型 DR は，ピーク時に電気料金を値上げするなど多様な電気料金を設定することで，需要家に DR を促すものである。一方，インセンティブ型 DR は，事前の契約に基づき系統運用者側からの指令により需要家に DR を促し，対価としてインセンティブ(報奨金)を支払うものである。電力需要を抑制(増加)させるものを下げ(上げ)DR とも呼ばれ，インセンティブ型 DR の下げ DR をネガワット取引，とも言われている。この DR の考え方は，蓄電池のようなエネルギー貯蔵装置を活用する際に採用されている。

しかしながら，この DR の考え方だけでは各時間帯(電気料金は基本的に 30 分間)の需要の上げ・下げが実現できても，時々刻々と変化する自然変動電源の出力に対して効率的かつ柔軟に制御することは困難である。そこで考えられている実体として，アグリゲータである。アグリゲータとは，需要家リソースや関連するエネルギーリソースを統括・制御することで，あたかも一つの発電所のように振る舞う。このことを仮想発電所(Virtual Power Plant:VPP)といい，VPP のイメージと活用例を図 1.11 に示す。同図に示すように，アグリゲータは需要家のリソースを統括することで電力系統に関する運用を実現するための役割を果たす。近年では VPP や DR を用いて，系統運用者や大型の需要家を取引先として，ピークカットや電気料金の削減などのサービスを提供する事業（アグリゲーションビジネス）に向けた検討[67]がされつつあり，小容量のリソースをアグリゲートし多様なサービスを提供するような検討もされている。

このようなアグリゲーションを前提としたアグリゲータおよび需要家のリソース活用に関する検討は文献[50]-[52]が挙げられる。文献[50]ではヒートポンプと EV を需要家リソースの対象としており，「DR による配電系統への影響を考慮しながらどのようにアグリゲーションするか」について検討されている。また，文献[51][52]では多数の EV を対象に調整力市場(Reserve Markets)への意思決定を行う方法について検討されている。

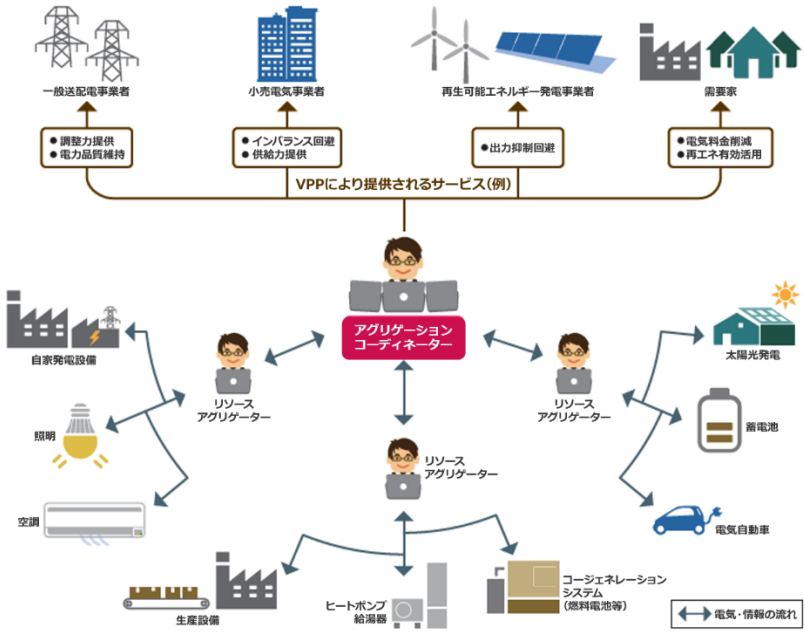


図 1.11 VPP のイメージ図と活用例[66]

1.4 本研究の目的と構成

前節までに述べたように，大量の自然変動電源が配電系統に局所的に導入されることが予想されるが，これによる電圧分布の複雑化への対策方策については確立されていないのが現状である。更に EV の普及により，市街地へ急速充電器(以下，充電器)が設置されつつある。これにより，PV の発電出力が大きく充電器の充電がない場合は電圧上昇，逆に PV の発電出力が小さく充電器の充電が多い場合は電圧低下と，さらに大きな電圧変動が発生し，電圧管理が複雑化になる可能性が高い。

一方，充電器は商用電力(交流)を直流に変換する交直変換器と各種保護回路等で構成される。交直変換器として自励式インバータが用いられている場合，原理的には EV の接続の有無に関わらず，無効電力の発生・吸収が可能である。この充電器による無効電力の調整により電圧変動の一部を解消することができれば，対策として必要な電圧制御機器(SVC)の容量を削減できるといったメリットが期待される。しかしながら，現在使用されている充電器の多くが通信機能を使用できる環境とは言えないため，各充電器が自律分散的に動作する必要がある。また充電器が本来の用途である EV の充電に利用されている間は，電流容量の制約により無効電力補償が制限されるため，充電器による電圧制御効果は，充電器の利用状況に依存することになる。そこで本論文では，まず電圧制御に貢献する充電器の運用(無効電力制御)手法を提案し，数値試算により提案手法適用時の制御能力を推定する。

また自然変動電源の大量導入による需給調整する能力（以下，調整力）の不足問題を解決するために，需要家リソースの活用することが検討されているが，検討されている手法は実運用断面における統括・制御する手法であり，前日などの事前段階でリソースの活用能力は保証されるものでない。またリソースを所有する需要家の観点からも，リソース活用に伴い生じる追加コスト

と対価（料金の値下げや報奨金）とを直接比較することで，需要家の経済性向上が期待される。しかしながら，需要家がリソースを系統に関する運用に活用するかどうかの意思決定を行うにあたり，リソースの運用に経済的な妥当性は必要不可欠であり，対価とコストの比較検討は重要な論点であろう。また，需要家側のリソースを調整力提供にどの程度利用できるかは，当該リソースの本来の用途や，定格容量や部分負荷運転の可否などの機器の運用上の制約などに大きく依存する。さらに需給運用上も，調整力がどの程度必要かは日時・気象条件・既存電源の起動停止状態などの様々な要因で変化する。したがって，アグリゲータは系統運用者からの要求に合わせるように，需要家側の都合を勘定しながら効率的かつ柔軟に調整力を事前に調達しておく機能が求められるだろう。

このような背景から本論文では，需要家側のリソースとして，コージェネレーションシステム（Co-generation system：CGS）に焦点を当て，アグリゲータが多数の CGS 群から調整力を調達する状況を想定し，具体的な調達手順を提案する。また，CGS を所有する個別需要家の視点から，調整力提供に関わる経済性への影響を考慮した CGS の最適運用手法を提案する。また，実需要データを用いた数値試算により，CGS 群の調整力を評価する。

さらに，需要家が所有するリソースの容量設計を調整力提供に貢献することによる対価を含めて行うことで，需要家の経済性がさらに向上することが期待される。そこで本論文では，そのリソースの容量設計を行うことを想定し，CGS の容量や利用形態（需要家が住戸単位で CGS を利用するのが，それとも複数住戸で CGS を共用するのか）の違いが需要家の経済性や調整力の大きさに与える影響を数値計算により明らかにする。

以下，本論文の構成および概要について記載する。

第 2 章では，本論文で提案する，電圧制御に貢献する充電器の無効電力補償を用いた運用手法の前提条件，各制御機能，および各制御に基づく無効電力補償の制御量の修正方法について記述する。また，提案手法適用時の電圧制御能力を推定する手法と数値試算結果についても記述する。

第 3 章では，本論文で想定する状況，調整力の考え方および需要家モデル，提案するアグリゲータが統括する CGS 群から調整力を調達する手順および需要家観点の運用手法に述べる。また，実測データを用いた数値試算により，提案手法適用時の CGS 群の調整力の評価結果についても記述する。

第 4 章では，本論文で比較検討を行う CGS の利用形態について述べる。また，第 3 章で提案した CGS の運用手法適用時の CGS の容量および利用形態の違いが需要家の経済性や調整力提供に影響を試算した結果を明らかにする。

第 5 章では，全体を通して得られた所見を含めた結論および今後の展望を述べる。

第2章　電圧制御に貢献する電気自動車用急速充電器の運用

前章で述べたように，PV 等の自然変動電源の大量導入により，今後の配電系統の電圧管理が複雑化されることが予想される。その対策として SVC 等の電圧制御機器の設置が挙げられるが，これらの機器の設置コストが依然として高い。また，DG の力率制御を実施することで電圧上昇を抑制することができるが，売電機会の逸失や収入の減少等の DG 所有者の経済性に影響を与える可能性がある。また，EV の充電をはじめとする需要負荷を増加させることで逆潮流を抑制することも可能であるが，特に導入が期待される PV においては昼間に発電電力が多く，その時間帯には多くの EV が移動または目的地へ移動しており，活用できる EV 数は限られているだろう。

このような背景がある中で電気自動車用急速充電器（本章では以後単に「充電器」とする）は，図 3.1 に示すように主に交直変換部および保護回路で構成されている[68][69]。交直変換部には AC-DC コンバータ部分と DC-DC コンバータ部分に 2 つ分けられるが，AC-DC コンバータ部分には三相の力率改善(Power Factor Correction:PFC)機能を有するコンバータ回路がある。また，DC-DC コンバータ部分には，安全上電源側と負荷側(自動車側)を電気的に絶縁するために絶縁トランスを設け，車載蓄電池の電圧値に対応した DC 電圧値に変換する。普通充電器は一般的に交直変換部のみで構成され, 絶縁トランスを用いずに電源側と負荷側を電気的に接続されていることから，基本的には有効・無効電力制御は負荷(EV)が接続されている間しか行うことができない。その一方で充電器は交直変換回路に加えて絶縁トランス等の保護回路があるため，原理的には EV の接続(プラグイン)の有無に関わらず有効・無効電力制御することが可能となる。また，充電器の容量は 10～50kVA 程度と普通充電器のそれよりも 10 倍程度大きく，より高い制御効果を発揮される可能性を秘めている。

そこで本論文ではこの点に着目し，充電器の無効電力補償(制御)を用いた電圧制御手法を提案する。本章の構成は以下の通りである。2.1 節では想定する前提条件，2.2 節では提案手法について述べ，2.3 節で提案手法における制御パラメータの設計方法，2.4 節で数値試算による妥当性検証結果について述べる。また，2.5 節で提案手法の制御能力を推定する手法を提案し，数値試算により能力評価を実施した結果について説明する。

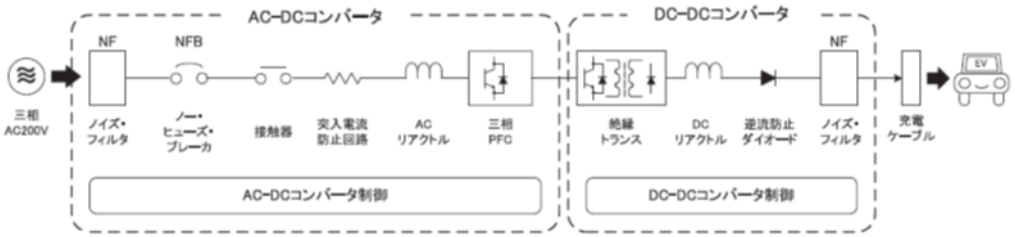


図 2.1 急速充電器の主な回路例[68]

2.1 想定する前提条件

本節では，電圧制御手法を提案するにあたり前提とする条件について記述する。図 2.2 は想定する前提条件を示す系統モデルの概略図である。同図の#S が配電用変電所を示し，電源側ノード(接続点)から#A，#B，…#E とノードを特定するため記号を振っている。前節でも述べたように充電器は主に三相 200V に接続される。配電系統の低圧側は単相 210V/105V 系統であるため，高圧(三相 6600V)に変圧器を介し三相 200V に変換して連系されるが，ここでは充電器がそれぞれ高圧ノード(接続点)に直接されているとみなしており，変圧器による影響は考慮していない。

次に，充電器の機能について記述する。充電器も有効電力，すなわち充電電力の制御を行うことができるが，充電電力を抑制する場合には充電器の一般的な用途(移動中や一時駐車中の利用)を考慮すると，普通充電器の充電利用よりも EV 所有者の利便性への影響が大きい。同様に充電器の容量の範囲内で充電電力を増加させることも可能であるが，実際に利用されている充電器は，EV の蓄電池の特性を考慮し蓄電量(kWh)に合わせて充電電力を制御している[69]。このとき充電電力を増加させてしまうと EV の蓄電池の寿命が短くなる等の問題があり，この問題は充電電力が大きい充電器特有のものである。これらの理由から，本研究では充電器の有効電力制御を実施せずに，充電電力は本来の用途(充電利用)を優先させ，空き容量の範囲を無効電力補償として用いることとする。

また，現在利用されている充電器には通信機能が備わっているものがある[69]。この通信機能は，充電器の利用状況や充電の予約等に利用されているが，標準機能ではなく充電器の規格にも必須事項でないことから，全ての充電器に搭載されている訳ではない。また，電圧制御のために通信機能を利用すると，充電器の利用状況(充電電力)を共有することによるプライバシーに関する問題や，機器の通信負荷の増大による本来用途の通信品質の悪化を招く可能性がある。これらを踏まえると，系統に接続された充電器をできるだけ多く活用するために，通信機能は使えない前提とすることが望ましく，本研究では「全ての充電器が通信機能を持たずに自律分散的に動作する」ことを前提とする。

このとき各充電器が効果的に制御を実施するためには，電力の潮流量や電圧状態等の系統状態を把握する必要がある。しかしながら，提案手法では通信機能を持たないことを前提としているため，各充電器は系統全体の電圧分布を把握することができない。そのため，各充電器はそれぞれの連系点における電圧(受電電圧)を特性上測定できることを前提として，その電圧値(入力)に基づいて無効電力補償量(出力)を決定する。各充電器が自身の補償量を決定するために，次節以降で記述する短周期制御，長周期制御の 2 つの制御機能，および各制御によって決まる補償量の修正のプロセスを実施する。

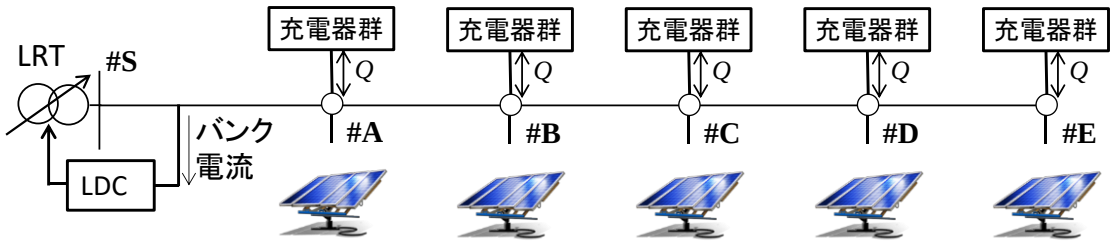


図 2.2 想定する前提条件の概略図

充電器の制御する周期を決定するあたり，似た回路構成を有する DG の力率制御，蓄電池の有効・無効電力制御は 0.1 秒以下の非常に短い時間で制御することができ，また充電器も原理的にはこれらと同様の同じ時間オーダで制御することができる。しかしながら，充電器の場合は本来の充電利用に影響を与えることを最小限にすることや，充電器の性能等により電圧計測・補償量決定のプロセスの時間が大きく異なってしまう可能性があることが挙げられることから，この時間オーダでの制御は難しい。特に後者の問題が深刻であり，例えば，ある充電器が電圧計測・補償量決定のプロセス実行中に他の充電器が同プロセスを 2 回行ったとすると，電圧計測した時点における電圧値と無効電力補償時の電圧値が大きく異なってしまい，場合によっては制御が悪化し，電圧変動を増大させてしまう可能性がある。また，測定した電圧値から無効電力量を決定する方法として，DG の力率制御として採用されている手法である，制御時間毎に無効電力量を単位量ずつ変化することで調整する方法が考えられるが，充電電力によって変化する空き容量が制御間隔毎に変化してしまい，十分に補償できない等の制御追従性の問題が考えられる。

そこで本論文では電圧測定時刻が同期されている(同時である)ことを前提とし，測定した電圧値から無効電力補償量(の目標値)を決定し，出力する一連のプロセスを制御間隔時間内(本論文では 1 分以内)に行うこととする。測定した電圧値(入力)から無効電力補償量(出力または目標値)を決定するにあたり，無効電力補償量の変化量に対する電圧値の変化に相当する電圧感度(V/Q 感度 : 比例制御の制御定数に相当)を算出する必要がある。この感度を算出するにあたり，基準となる配電用変電所から高圧配電系統のインピーダンスが既知である必要がある。実際の配電系統では様々な種類の機器や配電線を用いており，それらが複雑に構成されているため正確に把握することは困難であるが，インピーダンス推定装置等を用いることで精度よく推定でき，またこの推定に関する検討もなされている。よって配電系統の高圧系統内(図 2.2 のノード毎の)インピーダンスが既知であることを前提とする。

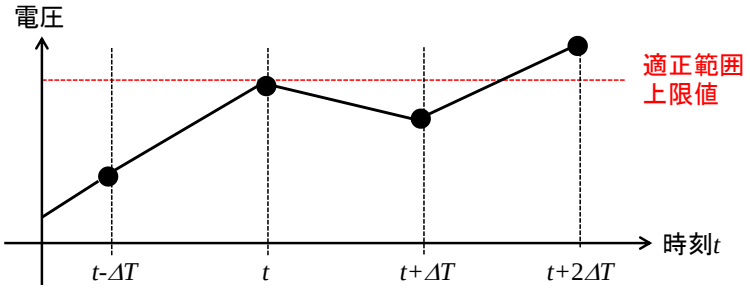
また，充電器を用いた電圧制御を実施するにあたり，既設されている電圧制御対策(LRT のタップ切替による電圧管理)にも配慮する必要がある。しかしながら，充電器は通信機能を持たないことから，当然 LRT と充電器の通信もできない。また，無効電力補償量を予測して LRT のタップ動作を行うことも可能であるが，充電器の利用状況によって無効電力補償量，すなわち電圧制御能力が制限されるため，期待される制御効果が発揮されない可能性もある。そこで本論文では LRT は従来から利用されている線路電圧降下補償器(LDC)方式に基づき動作することとする。なお，電圧制御機器は LRT の他に SVR，SC や ShR 等があるが，同様に取り扱うことが可能である(従来の動作・仕様で協調可能)。また，提案手法は電圧測定および無効電力補償が可能であればどの機器にも適用することが可能であり，SVC や DG，蓄電池等の無効電力補償に実施する際も 1 つの充電器(無効電力補償器)として適用することも可能である。

2.2　提案する電圧制御に貢献する運用手法

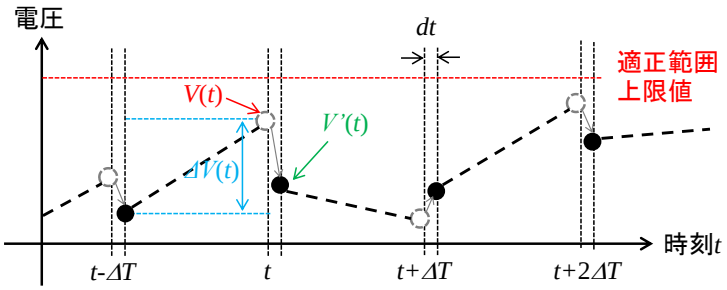
本論文では，本論文で提案する充電器を用いた電圧制御手法について説明する。具体的には，短周期制御および長周期制御に分け，また，これらの各制御機能に基づく無効電力補償量の修正方法について記載する。

2.2.1　短周期制御

短周期制御とは，制御時間間隔中の電圧変動を抑制することを目的とした制御である。その短周期制御のイメージ図を図 2.3 に示す。例として，同図(a)のように制御しない場合は電圧が変動しており，その結果として適正範囲上限値の逸脱が発生している時間帯を想定する。同図(b)のように前制御時間帯の制御後の電圧値$V'(t-\Delta T)$ [V]と測定した電圧$V(t)$ [V]に基づく無効電力制御量を決定する。この一連の動作を制御サンプリング時間(ΔT)毎に実施することにより，同図(b)の黒丸のように電圧変動が抑制される。なお，前節で述べたように各時刻における電圧測定から制御までの間(同図(b)の dt)，すなわち制御遅れ時間はインバータの特性を考慮すると比較的に短い時間(1 秒以下)であることが期待されるが，充電器の機器特性（充電能力に影響しないように配慮）やそれによって生じる制御の不安定性を考慮して制御サンプリング時間を比較的長い時間(ΔT =1 分)としている。電圧測定時刻 t から制御後時刻 $t+dt$ までは，他の充電器も含めた電圧制御実施時間であることが予想されるため電圧変動の算出には含めず，制御後時刻 $t+dt$ から次の電圧測定時刻 $t+\Delta T$ が電圧変動の算出時間としている。この制御遅れ時間が長すぎると電圧変動量を適切に算出できない可能性がある。



(a)制御しない場合



(b)短周期制御

図 2.3 短周期制御のイメージ図

具体的には，時刻tにおける無効電力制御量$dQ_i^s(t)$ [kvar]は，(2.1)式のように表される(上付き文字sは「短周期制御」を示す)。

$$dQ_i^s(t) = \frac{\Delta V_i(t)}{K_i^s} = \frac{V_i(t) - V_i'(t-\Delta T)}{K_i^s} \tag{2.1}$$

ここで，iは充電器番号，$V_i(t)$はiの充電器が接続されているノードのt時刻における電圧(測定電圧値)[V]，$V_i'(t-\Delta T)$は前の制御時間($t-\Delta T$)の制御後の電圧([V])を示している。また，$dQ_i^s(t)$は正が無効電力を消費(吸収)する向き，負が無効電力を発生(注入)する向きとなっている。すなわち，充電電力(正)と無効電力の正が同じ潮流方向になる。K_i^sは制御定数(V/Q 感度)[V/kvar]を示しているが，この値の算出方法は後述する。

しかしながら系統の電圧が上昇または低下傾向にある場合に，前述した無効電力補償を行うと充電器の空き容量値に張り付く可能性がある。これは，$dQ_i^{s1}(t)$のみに基づいて無効電力補償量を決定すると，たとえば常に電圧が上昇し続ける状況に対しては短周期制御における無効電力補償量が定常的に正の値をとることになる。提案手法ではこのような定常的な無効電力補償は後述する長周期制御，あるいはLRTやSVRなどの電圧制御機器により補償することを考えているため，提案手法では次式で定義されるフィードバック項$dQ_i^{s2}(t)$を重畳する。

$$dQ_i^{s2}(t) = -G_f \cdot Q_i^s(t-\Delta T) \tag{2.2}$$

G_fは短周期制御パラメータであるフィードバックゲイン(0～1)である。G_fが小さい(0 に近い)場合は，充電器が定常的に無効電力制御(補償)を行い，充電器の電流容量により短周期(制御サンプリング毎)の電圧変動を抑制できなくなるなどの不具合が懸念される。一方，G_fを大きくすると，電圧変動抑制効果が十分に得られず，電圧変動ならびに電圧逸脱を低減できなくなることが懸念される。したがって，適切なG_fを設定し適用する必要があるが，適切な値は系統の状態(電圧変動の大きさや系統構成)や充電器の利用状況(補償可能量)に依存する。また前述の通り充電器は通信機能を持っていないことから，受電点の電圧以外の系統の状態をリアルタイムで把握することは困難であり，さらには，外部からパラメータを設定（算出）することは困難であろう。適切なG_fの算出方法についても後述する。

短周期制御における最終的な無効電力補償量$Q_i^s(t)$は次のように表される。

$$Q_i^s(t) = Q_i^s(t-\Delta T) + dQ_i^s(t) \tag{2.3}$$

$$dQ_i^s(t) = dQ_i^{s1}(t) + dQ_i^{s2}(t) \tag{2.4}$$

2.2.2 長周期制御

長周期制御とは，短周期制御でも電圧制御しきれないような，長周期の(定常的な)電圧変動を抑制することを目的としている。一般的に配電系統の電圧分布は，配電用変電所から遠い末端側のノードほど，送出電圧からの変化量が大きい傾向にある。すなわち，電圧の適正範囲逸脱が生じやすいのは末端側のノードであるといえる。また，近隣のノード間ほど電圧の相関が強く，遠方のノード間の電圧の相関は弱い。そのため，配電系統の送出し点に近いノードで自端情報に基づいて長周期制御を実施すると，電圧逸脱が生じやすい末端側のノードにとっては返って悪影響を与える可能性も考えられる。そこで提案手法では，長周期制御は高圧配電系統の末端側のノードに接続された充電器でのみ実施するものとする。

例として，図 2.3(a)に示す制御なしの場合の電圧分布に対して長周期制御を実施したときのイメージ図を図 2.4 に示す。長周期制御を実施するにあたり，あらかじめ電圧不感帯(以後，単に「不感帯」とする)を設定する。この不感帯と測定電圧値$V(t)$に基づき無効電力補償の制御量を決定する。時刻tのように測定電圧値が電圧不感帯を逸脱している場合は無効電力制御により不感帯の範囲に収める。一方，時刻$t+\Delta T$のように測定電圧値が不感帯の範囲内に収まっている場合は前の無効電力補償を続ける。また次の時刻$t+2\Delta T$で測定電圧値が不感帯を超えているため無効電力補償量を増加させる…，このような判定・動作を制御サンプリング(ΔT)毎に実施するのが長周期制御である。すなわち，長周期制御は短周期制御と異なり，条件に分けて動作をする。

具体的には，図 2.5 に示すように 2 つの不感帯を設定する。2 つの不感帯のうち，電圧値が高い不感帯(図 2.5 の赤線)を上限側不感帯とし，低い不感帯(図 2.5 の青線)を下限側不感帯と呼ぶこととする。上限側不感帯は電圧上昇対策，下限側不感帯は電圧低下対策を想定して設定する。

このように不感帯を 2 つ設定した理由としては，不感帯を 1 つだけ設定してその範囲内に収めようとする場合，LRT のタップ動作後も無効電力補償が定常的になるからである。これは，ある目標電圧範囲を 1 つの不感帯として設定した場合を考えると，不感帯を超えたときには無効電力補償の調整に加えて LRT のタップ切替が行われる。両者の制御により目標電圧範囲に収まった場合には前の無効電力補償を継続してしまう。さらに，電圧上昇または電圧低下が継続する場合にはこれを繰り返す。その結果として，充電器は必要以上の無効電力補償をしてしまい，LRT は充電器による電圧制御に対して電圧制御をする，というように充電器と LRT の間の協調が図れない可能性がある。協調が図れないという状態は例えば，充電器による電圧制御によって電圧が下がったすれば，LRT がその電圧を上げようと送出電圧を上げる。その結果電圧は不感帯よりも高くなり，充電器はさらに電圧を下げようとするが，LRT はまた送出電圧を上げようとする…とこのような一連の動作を繰り返すいったことを行ってしまうことである。

このような協調の問題を回避するために図 2.5 のように 2 つの不感帯を設定し，2 つの不感帯の間(同図の斜線の領域)は前の補償量を低減させる領域を持たせ，必要以上の無効電力補償が行われないようにする。

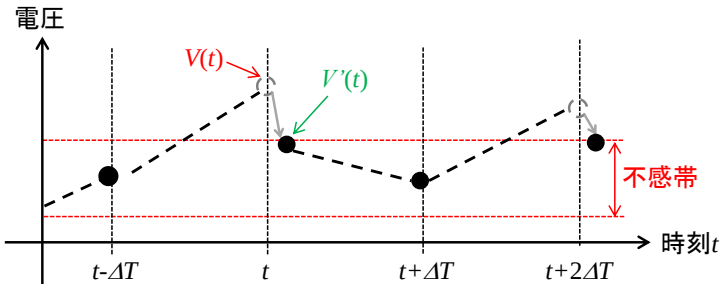


図 2.4 長周期制御のイメージ図

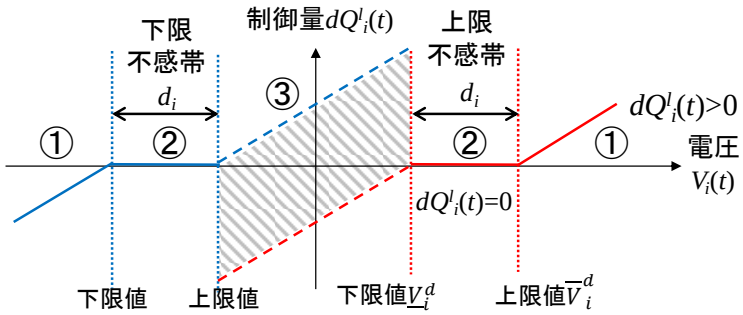


図 2.5 長周期制御における不感帯と無効電力制御量の関係

長周期制御においてはこの 2 つの電圧不感帯が制御パラメータとなる。本論文では，この不感帯設定を G_f と同様に過去の電圧情報を基に(オフラインで)設計することとする。具体的には，フィーダ内で電圧適正範囲上限値逸脱が発生しない各高圧ノードの最大電圧値を上限側不感帯の不感帯上限(図 2.5 の $\overline{V}_i^d$)に設定する。同様に下限側電圧不感帯の不感帯下限も，フィーダ内で電圧適正範囲下限値逸脱が発生しない各高圧ノードの最小電圧値となるように設定する。また 2 つの不感帯幅 d_i は，系統構成や電圧変動の大きさ等から決定する。

長周期制御における無効電力補償の制御量 $dQ_i^l(t)$ [kvar]は図 2.5 に示すように測定電圧 $V_i(t)$ [V]と不感帯の大小関係に基づき次の 3 通りに分けて決定する。なお，ここでは，電圧上昇対策として上限側不感帯(同図の赤色)に着目して説明するが，電圧降下時の対策にも同様の手法（ただし，制御の向きは反対）によって補償量を決定する。以下に示す①～③は図 2.5 の①～③に対応している。

①測定電圧 $V_i(t)$ [V]が不感帯上限値 $\overline{V}_i^d$ [V]よりも高い場合

この場合は，不感帯に収まるように無効電力補償(消費)を(2.5)式のように $dQ_i^l(t)$ [kvar]だけ増加させる(上付き文字 l は「短周期制御」を示す)。

$$dQ_i^l(t) = \frac{V_i(t) - \overline{V}_i^d}{K_i^l} \tag{2.5}$$

K_i^l は長周期制御に関する制御定数(V/Q 感度)[V/kvar]である。

②測定電圧$V_i(t)$ [V]が不感帯に収まっている場合

この場合は，電圧値を不感帯内に収まるように無効電力補償を継続する($dQ_i^l(t)=0$)

③測定電圧$V_i(t)$ [V]が不感帯下限値$\underline{V}_i^d$ [V]よりも低い場合

1 時点前の無効電力補償量が零であれば，そのまま維持する($dQ_i^l(t)=0$)。1 時点前の無効電力補償量が正の場合，充電器の無効電力補償量を低減させても不感帯の上限値の超過は発生しないと想定されることから，(2.6)式のように無効電力補償量を低減させるように制御する。

$$dQ_i^l(t) = -\frac{\underline{V}_i^d - V_i(t)}{K_i^l} \tag{2.6}$$

①～③の計算によって決定される無効電力の制御量$dQ_i^l(t)$から，(2.7)式のように長周期制御の無効電力補償量$Q_i^l(t)$を決定する。

$$Q_i^l(t) = Q_i^l(t-\Delta T) + dQ_i^l(t) \tag{2.7}$$

ただし，③の場合で調整前後の制御量($Q_i^l(t-\Delta T)$と$Q_i^l(t)$)の符号が異なる場合は長周期制御の制御量$Q_i^l(t)$を零とする。

一方，電圧下限対策時(下限不感帯)も同様の手順で向き(符号)を反転させて取り扱う。例えば，測定電圧値が下限値不感帯の下限値以下であれば，無効電力補償量を増加させる($dQ_i^l(t)<0$)。長周期制御は上記のように測定電圧値と不感帯の大小関係に基づき，制御サンプリング時間毎に場合分けおよび制御量計算を行う制御である。

2.2.3　各制御機能に基づく無効電力補償量の修正

短周期制御と長周期制御のそれぞれで決定した無効電力補償の制御量$dQ_i^s(t)$, $dQ_i^l(t)$が同一符号の場合，単純に両者に相当する分だけ無効電力補償量を制御すると，個々の制御で必要とされているよりも多く制御することとなり，過補償となる可能性が高い。逆に，各制御機能の制御量が異符号の場合には実際に補償する量は打ち消されるため，個々の制御で必要な制御が実施されない可能性もある。したがって，各制御機能に基づき適切に補償量(出力)を修正する必要がある。

そこで提案手法では，図 2.6 に示すフローチャートに沿って$dQ_i^s(t)$, $dQ_i^l(t)$を修正する。この修正の基本方針は，両制御量が同一方向の場合には必要最低限な制御量に留めること，また逆方向の場合には,電圧を適正範囲に収めることを目的としている長周期制御を優先させることにある。図 2.6 の①～③の各手順詳細は次の通りである。

手順①）前述の通り提案手法は充電器の空き容量を利用した無効電力補償を念頭においている。そのため，無効電力補償に利用可能な容量($Q_{i,\max}(t)$ [kvar])は，次式に示すように同時間帯の充電電力$P_i(t)$ [kW]の影響を受ける。

$$Q_{i,\max}(t) = \sqrt{S_i^{\,2} - P_i(t)^2} \tag{2.8}$$

ここで，S_iは充電器iの容量である。手順①ではまず，優先的な制御量である$dQ_i^l(t)$だけを仮に考慮した場合に，(2.8)式で求められる$Q_{i,\max}(t)$の範囲内で実施可能かどうかを判定し，容量を逸脱する場合には，ちょうど$|Q_i(t)| = Q_{i,\max}(t)$が成立するように$dQ_i^l(t)$を修正する(修

正後の制御量を $dQ_i^l{}'(t)$ とする)。この手順①により，長周期制御の制御量は確定することになる。

手順②）$dQ_i^s(t)$ と $dQ_i^l{}'(t)$ が同符号の場合，過補償を回避するために，最終的な制御量($dQ_i^s{}'(t)$ と $dQ_i^l{}'(t)$ の合計値)が $dQ_i^s(t)$ と $dQ_i^l{}'(t)$ の絶対値が大きい方と同値になるように， $dQ_i^s(t)$ を修正する(右辺 1，2 行目に相当)。このように修正することで，短周期制御・長周期制御の双方に対して必要制御量を確保することが可能となる。一方，$dQ_i^s(t)$ と $dQ_i^l{}'(t)$ が異符号の場合は，$dQ_i^l{}'(t)$ を優先させることから $dQ_i^s(t)$ を零とする。

手順③）手順②によって修正された短周期制御の制御量 $dQ_i^{s-}(t)$ が充電器容量制約を満たすように，再度修正を施す(修正された制御量を $dQ_i^s{}'(t)$ とする)。具体的には，(2.9)式が成立しない場合は $|Q_i(t)| = Q_{i,\max}(t)$ となるように $dQ_i^s{}'(t)$ を決定する。

$$|Q_i(t)| = \left|Q_i(t-\Delta T) + dQ_i^s{}'(t) + dQ_i^l{}'(t)\right| \le Q_{i,\max}(t) \tag{2.9}$$

なお，最終的な制御の仕上がりだけを考えた場合， $dQ_i^s(t)$ と $dQ_i^l(t)$ の合計値にのみ着目すれば十分であるように思われるが，前述の通り提案手法では $dQ_i^s(t)$ の決定過程においてフィードバック項($dQ_i^{s2}(t)$)を用いていることから，各制御の内訳($Q_i^s(t)$, $Q_i^l(t)$)を明らかにする必要がある。そのため提案手法では各制御機能における制御量($dQ_i^s(t)$, $dQ_i^l(t)$)を個別に修正している。

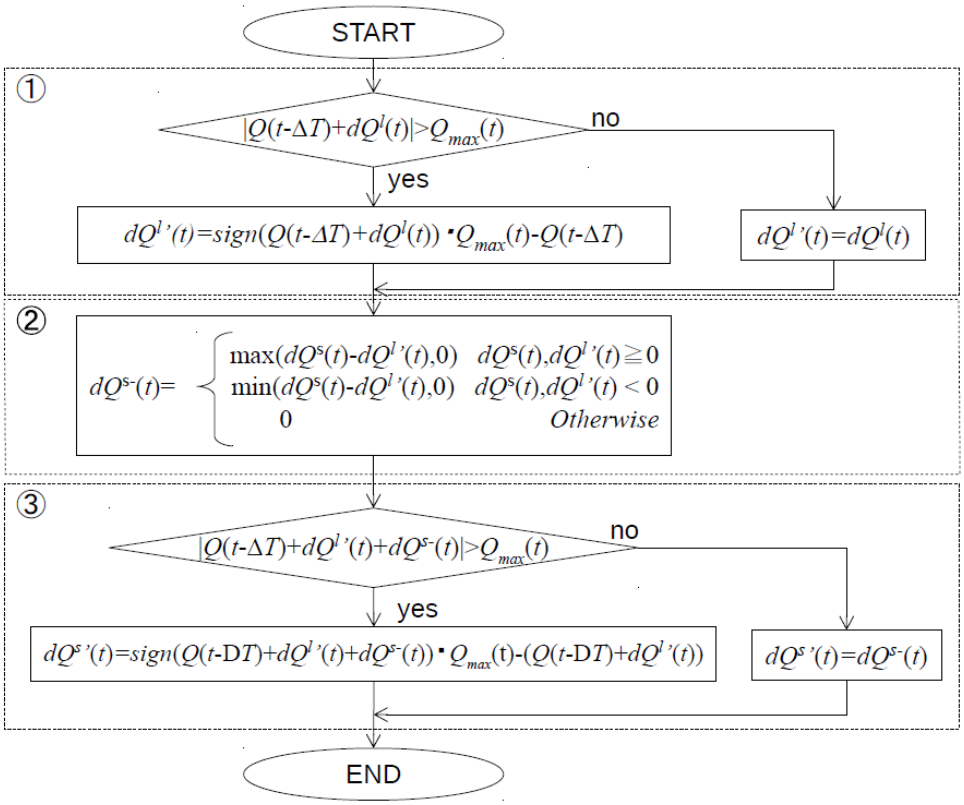


図 2.6 無効電力補償量の修正のフローチャート

2.3 制御パラメータの設計

ここでは，提案手法における制御パラメータである V/Q 感度，フィードバックゲインおよび不感帯の設計方法について述べる。

2.3.1 V/Q 感度

ここでは，提案手法における制御パラメータである V/Q 感度，フィードバックゲインおよび不感帯の設計方法について述べる。

提案手法で用いている V/Q 感度は，実現したい電圧調整量に対して無効電力補償量をどの程度調整すべきかを決定する際に用いる定数である。ある充電器の無効電力補償は，その充電器が接続されているノードの電圧のみならず，他の隣接ノードの電圧にも影響を与える。本論文では同一配電系統内に複数の充電器が設置されており，それらが一斉に電圧制御に参加する状態を想定しているため，V/Q 感度の設定においては他の充電器の挙動を考慮することが，過補償回避の観点からは重要である。なお，本論文では充電器は外部との通信機能を持っていない状況を想定しているため，提案手法では他の充電器の挙動をリアルタイムに正確に把握することはできない。

そこで本論文では，「系統に接続されている全ての充電器が，同じ量だけ無効電力補償量を調整する」ものと仮定して V/Q 感度を設計することとする。具体的な手順を図 2.7 に示す配電系統モデルを使用するが，このモデルは高圧ノード数および各高圧ノードに接続されている充電器台数は任意性・一般性を持たせており，分岐がある場合も含め，系統によらず適用可能である。

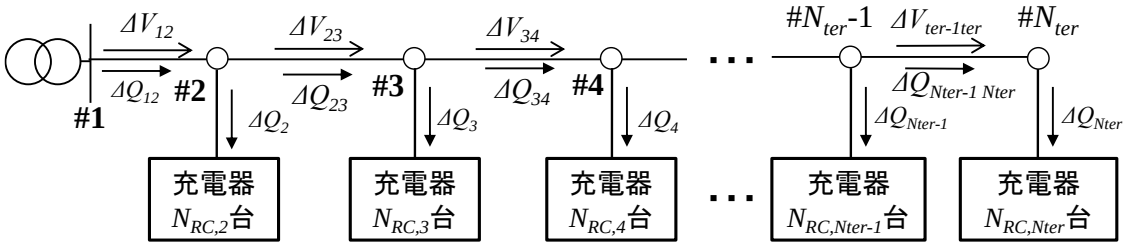


図 2.7 一般化された複数ノードの配電系統モデル図

まず，隣接する二つのノード $j-1, j$ をつなぐ配電線での電圧降下 $\dot{V}_{j-1}-\dot{V}_j$ は(2.10)式で表現できる。

$$\dot{V}_{j-1}-\dot{V}_j=\frac{R_{j-1,j}P_{j-1,j}+X_{j-1,j}Q_{j-1,j}}{\dot{V}_j}+j\frac{X_{j-1,j}P_{j-1,j}-R_{j-1,j}Q_{j-1,j}}{\dot{V}_j} \tag{2.10}$$

ここで，$R_{j-1,j}, X_{j-1,j}$：ノード $j-1, j$ 間の抵抗，リアクタンス[Ω]，$P_{j-1,j}, Q_{j-1,j}$：ノード $j-1, j$ 間を流れる有効・無効電力潮流(ノード j において観測された値)[W]，[var]である。無効電力制御量が $\Delta Q_{j-1,j}$ だけ変化したときに，(2.10)式の値の変化量 $\Delta V_{j-1,j}$ は，近似的に(2.11)式で求められる。

$$\Delta V_{j-1,j}\approx\left|\frac{1}{V_j}(X_{j-1,j}-jR_{j-1,j})\Delta Q_{j-1,j}\right|=\frac{1}{V_j}\sqrt{{X_{j-1,j}}^2+{R_{j-1,j}}^2}\,\Delta Q_{j-1,j} \tag{2.11}$$

j よりも末端側の全ての充電器が無効電力量を一律 dQ [var]だけ調整する場合，$\Delta Q_{j-1,j}$ は(2.12)式で近似できる。

$$\Delta Q_{j-1,j} = \left(\sum_{k=j}^{N_{ter}} N_{RC,k} \right) \times dQ \tag{2.12}$$

ここで，$N_{RC,k}$ はノード k に接続され，なおかつ電圧制御を実施する充電器台数，N_{ter} は高圧配電系統の最末端のノードである。したがって，無効電力補償量の調整 dQ によって生じるノード i の電圧変化量 ΔV_i は，(2.11)式を配電用変電所(ノード 1)からノード i まで積算することで，次式によって表される。

$$\Delta V_i \approx \sum_{j=2}^{i} \left(\frac{\sqrt{X_{j-1,j}^2 + R_{j-1,j}^2}}{V_j} \sum_{k=j}^{N_{ter}} N_{RC,k} \right) dQ \tag{2.13}$$

ここで，厳密には上式中の V_j は時間とともに変化する値であるが，その値を各充電器がオンラインで推定することは困難であるため，提案手法では V_j としてある一定値を用いることとする。その際，V_j として実際の電圧値よりも大きい値を利用すると，(2.13)式で算出される無効電力制御量は本来必要な制御量よりも大きくなり，結果的に過補償を引き起こしたり，またさらにはハンチングを引き起こしたりする可能性がある。そこで，本設計では V_j の代表値として電圧適正範囲の下限値 V_{low} [V]を用いて，次式により V/Q 感度を設定する。

$$K_i \approx \frac{1}{V_{low}} \sum_{j=2}^{i} \left(\sqrt{X_{j-1,j}^2 + R_{j-1,j}^2} \sum_{k=j}^{N_{ter}} N_{RC,k} \right) \tag{2.14}$$

この K_i は無効電力制御を実施する充電器台数・接続箇所，配電線のインピーダンスなどの事前に把握可能な情報に基づいてオフラインで算出することができる。なお，短周期制御と長周期制御では制御を実施する充電器台数($N_{RC,k}$)が異なることから，K_i^s と K_i^l は異なる値となる。

2.3.2　フィードバックゲイン

フィードバックゲイン G_f が小さい(0 に近い)場合は，充電器が定常的に無効電力補償を行ってしまい，充電器容量制約により所望の短周期変動抑制が実施できないなどの不具合が懸念される。一方，G_f を大きくすると，ある時間断面で決定した補償量が次の時間断面以降の早い時間断面で無効になるため，短周期制御による電圧変動抑制効果が小さくなることが懸念される。したがって，適切な G_f を用いる必要があるが，その値は系統の状態(変動の大きさ)や充電器の利用状況に依存する。しかしながら，前述したように充電器は通信機能を持っていないことから，これらの外部状況をリアルタイムで把握する，あるいは外部から逐次パラメータを設定することは困難である。そこで本研究では，G_f をオフラインで試行錯誤的に設計するものとする。試行錯誤の過程においては，過去の電圧分布や需要パターン，PV 等の分散電源の出力パターンを想定したシミュレーションを活用することになる。

2.3.3 不感帯

長周期制御で使用する不感帯は，過去の電圧情報を用いて設計する。具体的には，まず，配電系統内のノード電圧の時系列データの中から，各時間断面における系統内の電圧の最大値・最小値を求める。次に，その最大・最小電圧が許容範囲(法令で決まっている適正範囲をある一定のマージンだけ狭めた範囲)にぎりぎり収まる時間断面を抽出する。こうして抽出された時間断面における各ノード電圧を電圧上昇側の上限値 $\overline{V}_i^{d,over}$ ならびに電圧降下側の下限値 $\underline{V}_i^{d,under}$ とする。このように設計された両不感帯の外側の境界値よりも内側に各ノード電圧を収めることができれば，系統内の全ノード電圧が許容範囲内に収められることが期待される。

なお，両不感帯の内側の境界($\underline{V}_i^{d,over}$，$\overline{V}_i^{d,under}$)については，外部境界から一定量だけ内側に設定するものとする。この際に用いる不感帯の幅は制御結果(電圧逸脱が起こるかどうか)に直接影響を与えないが，狭く設定すると長周期制御の補償量を低減させられる反面，電圧が定常的に変化しやすくなり，反対に不感帯を広く設定すると必要以上の補償を継続的に行ってしまう。本論文では不感帯幅は G_f と同様，試行錯誤的に設計することとする。

以上の不感帯の設計においては各高圧ノードで時系列の電圧値データが利用可能であることを前提としている。このようなデータの入手が難しい場合には，G_f の設計と同様に，想定需要データに基づくシミュレーション結果などを活用することになる。

2.4 数値試算による提案手法の有効性評価

本節では，数値試算(シミュレーション)により提案手法(各制御および各制御の制御量の修正方法)の有効性について検証・評価する。

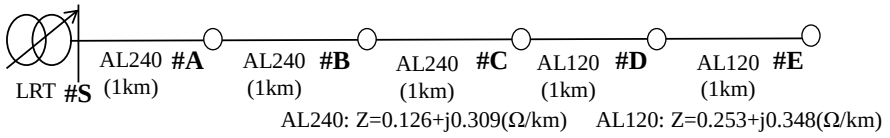
2.4.1 試算条件

本論文の試算に使用する配電系統モデルを図 2.8 に示す。図 2.8(a)に示すように，配電用変電所の 1 バンクに 1 本のフィーダ(恒長は 5km)が接続され，配電用変電所(LRT)をノード#S(以後，単に#S と表記)とし，系統の末端側に向かうにつれて，ノード#A，#B，…，#E と 5 つの高圧ノード(ノードの間隔は 1km)が存在する。図 2.8(b)は各高圧ノードのモデルを示しており，各高圧ノードに 30 台の柱上変圧器が接続されている。柱上変圧器の 2 次側(低圧側)は図 2.8(c)のようなモデルと想定しており，1 台の柱上変圧器に 12 軒の住宅が接続されている。これより，各高圧ノードでは 360 軒，系統全体では 1800 軒の住宅が接続されていることになる。また図 2.8 には，文献[70][71]を基に想定した配電線の種類および配電線や柱上変圧器のインピーダンスを表記している。定格 4kW 級の PV が全住宅に接続されているものとする。また，単機容量 30kVA とする充電器は各高圧ノード 6 台，系統全体で 30 台設置するものとし，これらの充電器は次項で述べるモデルに従うものとする。ただし，上記で述べた配電系統モデルは，試算の都合上，一部簡略化している。省略化した内容は以下の通りである。

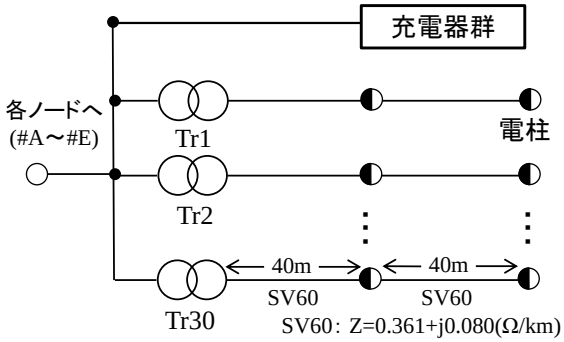
まず，各高圧ノードの 30 台の柱上変圧器のうち 1 台については低圧側も計算するが，残りの 29 台の柱上変圧器については低圧側の計算をせずに，各高圧ノード以下に残りの住宅軒数(348 軒)分の負荷接続されているものとする。さらに，低圧系統内にも同様の手法を適用させ，同図(c)の

下側の住宅を柱上変圧器もしくは電柱に集約させる。この処理により，各高圧ノードで計算する住宅軒数は 6 軒，系統全体では 30 軒と集約され，全ノード数は柱上変圧器および電柱を含め 51 ノードとなる(ノードマップについては付録に記載)。また，PV は全住宅のうち実際に計算する 1 台の柱上変圧器以下の住宅(全ノードで 60 軒)には全て PV を接続し，それ以外の住宅に対して残りの PV 導入台数(1380 台)分をランダムに割り振って各高圧ノードに集約している。

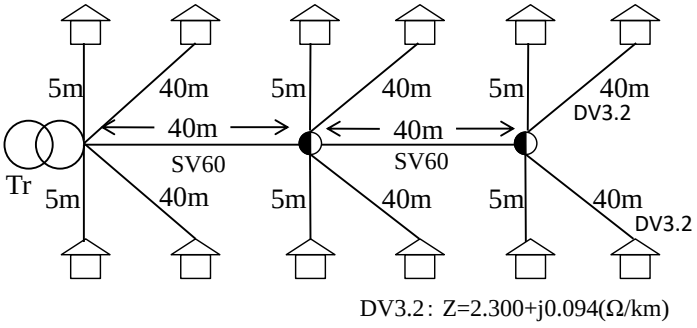
また，本試算で使用する住宅負荷および PV 出力は全ての住宅で同じパターンを使用することとしている。図 2.9 は本節の試算に使用する住宅負荷および PV 出力のパターンを示しているが，住宅の負荷は文献[72][73]を基に，PV の出力パターンは新エネルギー・産業技術総合開発機構(NEDO)が計測した群馬県太田市で測定された実測データを基に，出力ピークおよび出力変動が大きいデータを生成した。なお，PV 出力がない時(または夜間時)の電圧降下も考慮し，柱上変圧器の 1 次側のタップを#A～#C では 6600V，#D～#E では 6450V に選定する。ただし，シミュレーション刻み幅は 1 分，試算期間は 6:00～18:00 の 12 時間とする。



(a) 高圧フィーダモデル



(b) 各高圧ノードモデル



(c) 低圧フィーダモデル

図 2.8 数値試算に用いる配電系統モデル

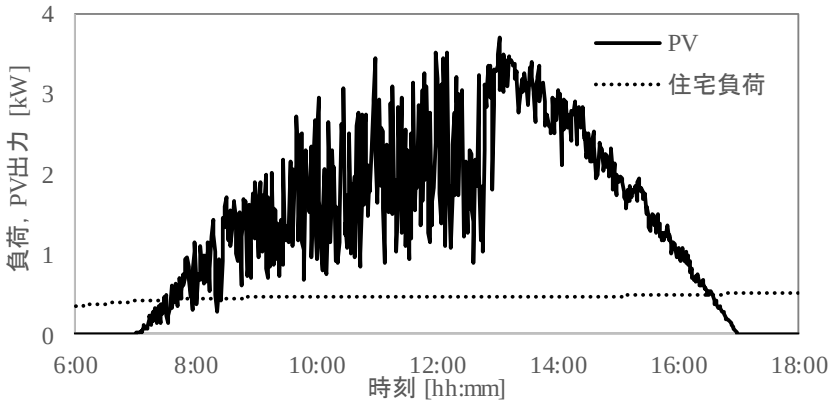


図 2.9 住宅負荷および PV 出力のパターン

数値試算に用いる LRT のパラメータを表 2.1 に示す。#S の LRT の動作として，従来から LRT の制御として用いられている LDC 方式を採用することとする。具体的には，LRT の送出点に流れる電流を計測し(バンク電流 $\dot{I}_B(t)$)，(2.15)式により最適な送出電圧 $V_s(t)$ を推定する。

$$V_s(t) = \left| \dot{Z}_{REF} \dot{I}_B(t) + V_{REF} \right| \tag{2.15}$$

ここで，$\dot{Z}_{REF}$ は送出点から負荷中心点までのインピーダンス，V_{REF} は目標電圧値である。推定電圧値 $V_s(t)$ が目標電圧値 V_{REF} を中心とする不感帯幅を一定量継続して逸脱した際に LRT のタップを切替する方式である。一般的に負荷需要は各ノード・時間で異なるため，負荷中心点および整定値($\dot{Z}_{REF}$，V_{REF})を系統条件から直接算出することができず，過去データ等を用いて整定値をあらかじめ決定しておく必要がある。本試算では，ノード間で異なる潮流量と配電線のインピーダンスの違いによる電圧降下の大きさ，LRT のタップ数等を考慮して，近似的に#C を負荷中心点とし，$\dot{Z}_{REF}$ は#S から#C のインピーダンス値，V_{REF} は適正範囲中央値の 101V(低圧換算)とした。この LDC の整定値を適正値にすることでさらなる制御効果も期待できるが，充電器利用による無効電力補償可能量の不確実性を考慮する必要もあるため，これついては今後の展望とする。

LRT のタップ幅，すなわちタップ間隔が 0.0125[p.u.](基準が 6600V)であり，タップ不感帯をその間隔の半分とする。また，LRT のタップ動作を行う不感帯逸脱積算量を 0.03p.u.min としており，不感帯からの連続逸脱量がこの値を超えるとタップを切り替える。LRT のタップ切替動作は機械的な動作を伴うため一定回数の寿命があり，本論文では LRT のタップ切替動作に 10 分間の動作制限時間を設けている。ただし不感帯逸脱積算量の計算はこの動作制限時間中も行う。

表 2.1 LRT 設定パラメータ

タップ幅(間隔)	±0.0125p.u.
タップ不感帯	±0.00625p.u.
LRTタップ動作不感帯逸脱積算量	0.03p.u.min
LRT動作制限時間	10min

本論文では，電圧制御効果を測る指標(電圧制御指標)として，LRT のタップ切替回数，電圧変動量，電圧逸脱量を用いる。LRT のタップ切替回数は 1 日の試算期間内における LRT のタップ切替回数である。電圧変動量V_f [V]は(2.16)式に示すように，1 日の試算期間内における高圧ノード毎の前の制御サンプリング時間との電圧変化の平均偏差として算出する。

$$V_f = \sqrt{\frac{1}{T}\frac{1}{N_{ter}}\sum_{n=1}^{N_{ter}}\sum_{t}^{T}\left[V_n(t)-V_n(t-\Delta T)\right]^2} \tag{2.16}$$

一方，電圧逸脱量V_v [Vmin]は(2.17)式に示すように，制御サンプリング時間(1 分)毎の系統内の最大（または最小）電圧の適正範囲からの逸脱量を積算したものである。

$$V_v = \sum_{t}^{T}\left\{\max\left[V_{\max}(t)-V_{high}, V_{low}-V_{\min}(t), 0\right]\right\} \tag{2.17}$$

ここで，T は試算期間，$V_{\max}(t)$，$V_{\min}(t)$ は時刻 t における最大・最小の電圧(低圧換算値)，V_{high} は適正範囲上限値(107V)を示している。

本試算のフローチャートを図 2.10 に示す。まず，潮流計算により各充電器の測定電圧値を計算する。次に各充電器はその電圧値に基づき無効電力補償量を決定し，再び潮流計算により補償後の電圧を計算，LRT の動作を行う。この一連の動作を全試算期間で繰り返し実施する。

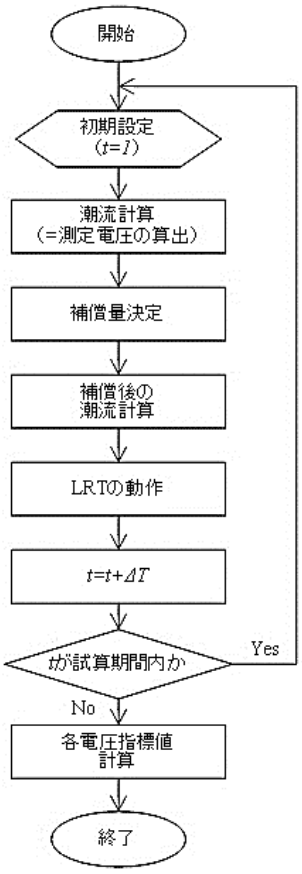


図 2.10 数値試算のフローチャート

2.4.2 充電器の利用状況のモデル化

充電器の利用状況は，顧客(EV の所有者)の到達時間(プラグイン時間)および到達時の充電量(State of charge: SOC)，充電目標 SOC，EV の最大充電電力などによって決定される依存するため，不確実性を有している。本論文では，充電器を利用する EV 台数をあらかじめ決定し，プラグイン時間，到達時の SOC を確率分布によってモデル化する。

まず，充電器を利用する EV 台数に着目する。第 1 章で述べたように，EV の充電器の充電時間は約 30 分である。よって，1 台の充電器あたり最大約 48 台の EV が充電することができるが，夜間・早朝は住宅やオフィス等の普通充電器で充電され，また，朝方は満充電になった EV が走行し始めるので十分に SOC があることが予想される。このことから，昼間～夜にかけての時間帯に充電が集中されるであろう。さらに現在は，充電器が大型の商用施設等に設置されつつあるが，日本に設置されている充電器のうち約 4 割が夜間は充電サービスを提供していないというデータ[74]もある。本論文では比較的大きな負荷である充電器の充電電力による電圧変動も考慮することから，充電器 1 台あたり 5～10 台/1 日の EV の充電が妥当ではないかと考えられる。そこで本論文では充電器 1 台あたり 5 台/1 日，系統全体で 150 台の EV が充電器を利用することとする。

充電器の到着時間は，充電器が設置されていると想定する施設の到着時間を確率的に表現した，確率密度分布に基づく乱数により決定する。想定する施設は表 2.2 に示すように，コンビニエンスストア(コンビニ)，スーパーマーケット，ガソリンスタンドおよびその他(役所や道の駅，商業以外の施設等)の 4 種類である。この 4 種類の施設毎にプグライン時間の確率密度関数のパターンを設定する(コンビニから順にパターン 1 とする)。施設毎のプグライン時間の確率密度関数を図 2.11 に示す。これら 4 つのパターンは，それぞれが想定されている商業施設の利用実態調査等(文献[75]はその例)を基に表 2.2 のようなピーク回数およびそれぞれのピークの時間を割り出し，ばらつき具合(分布)は正規分布になると仮定している。正規分布の標準偏差については利用実態調査データに沿った分布になるように設定した。

充電器の利用状況をモデル化するにあたり，以下の手順を実施する。

(1) 充電器毎に想定する商業施設のパターンを割り当てる。本試算では，高圧ノード毎にパターンを割り当てており，#A に接続された充電器 6 台はパターン 1，#B に接続された充電器 6 台はパターン 2，…等のように各高圧ノードを 1 つのグループとみなしている。なお，全パターンがいずれかの充電器に割り振られているようにランダムに決定する。

(2) EV も同様に 4 パターンの中から 1 つをランダムに決定する。

(3) 各 EV がプラグインする対象の充電器はその EV と同パターンである充電器の中からランダムに決定され，各 EV の到着時間はそのパターンの確率密度関数に従う乱数により決定される。

表 2.2 想定する充電器設置場所の到着時間の確率密度関数

	想定する商業施設	ピーク回数	ピーク時間
パターン1	コンビニ	2回	13,18時
パターン2	スーパー		11,16時
パターン3	ガソリンスタンド	1回	18時
パターン4	その他		16時

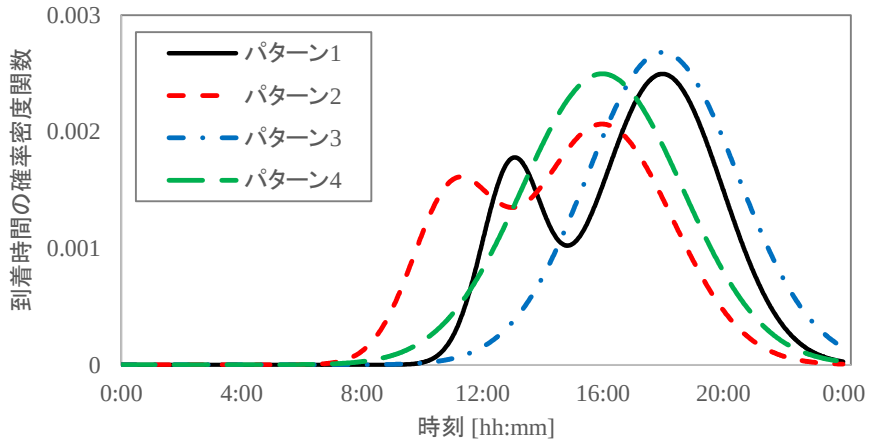


図 2.11 充電器到着時間の確率密度関数

充電器到達時の充電量(SOC)と充電器の利用状況の関係を考慮するにあたり，SOC が満充電(100%)近くであれば充電器を利用する可能性は低いと考えられる。また，1 日の平均走行距離[76](実働の自家用車(軽自動車含む)で平均約 32km)および EV の蓄電池 1kWh あたり走行距離(文献[42]の仕様からの試算により 5～8[km/kWh]程度，電費とも呼ぶ)を考慮すると 1 日の平均的な走行は SOC 換算で 20～25%程度である。この通常の走行だけでは充電しない可能性の方が高いとすると，充電器を利用する EV の充電器到達時の充電量(SOC)は，0～80%の範囲と考えることができ，本論文では EV 毎にこの範囲内でランダムに決定し，SOC が 100[%]になるまで充電するものとする。

EV への充電電力については，次のように決定する。一般的に EV に搭載されている蓄電池の特性を考慮して充電電力は SOC に応じて調整される[77]ことに鑑み，本論文でも同様に，図 2.12 の破線のように SOC の変化に応じて，充電電力を変更するものとする。その充電電力により無効電力補償可能範囲も同図の青の斜線のように変化する。なお，青線は(2.8)式の $Q_{\max}(t)$ に相当する。

各充電器の充電電力決定アルゴリズムを図 2.13 に示す。まず，既に到着している EV の有無を判断する。到着している EV がない場合は充電電力($P_i(t)$)を 0[kW]とし，EV がある場合はプラグインされているかを判断する。ここでプラグインされていなければプグラインを行い，その時間における SOC から充電電力と，充電電力量に応じた分だけ SOC を加算する。加算結果の SOC が 100%を超えた場合はプラグアウトを行う。以上のような判定・充電電力計算を各時間・充電器毎に行い，試算期間の充電電力プロファイルを生成する。

また，充電器の各制御の制御定数と制御パラメータを表 2.3 に示す。表 2.3 の第 1 列は充電器が接続されているノードを示している。各制御の制御定数(K^s, K^l)は(2.14)式から算出した値である。また，長周期制御は#D,#E に接続された充電器だけ実施することから，長周期制御の制御定数 K^l は短周期制御よりも値が小さい(相対的に 1 台あたりの補償量が多い)。

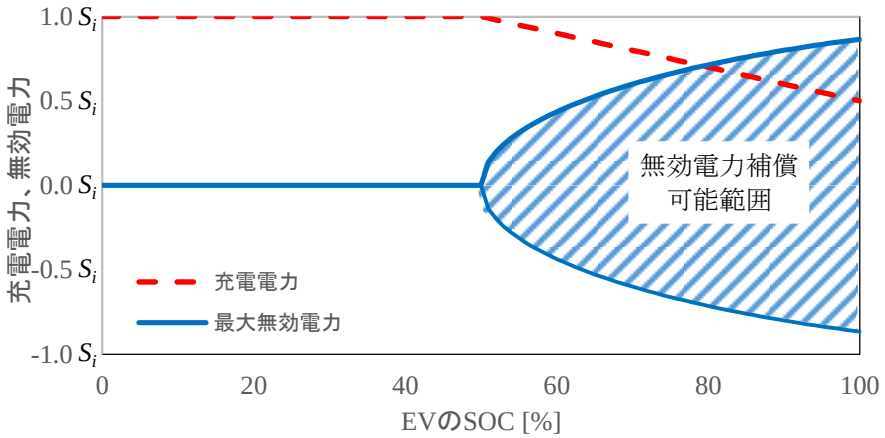


図 2.12 EV の SOC に基づく充電器の充電電力および無効電力補償可能範囲

表 2.3 充電器の各制御の制御定数および制御パラメータ

充電器	短周期制御		長周期制御			
	K^s [V/kvar]	G_f	K^l [V/kvar]	上限不感帯上限値[V]	下限不感帯下限値[V]	不感帯幅[V]
#A	0.0254	0.1				
#B	0.0458					
#C	0.0610					
#D	0.0738		0.0434	102.5	96	0.3
#E	0.0803		0.0498	103	95	0.5

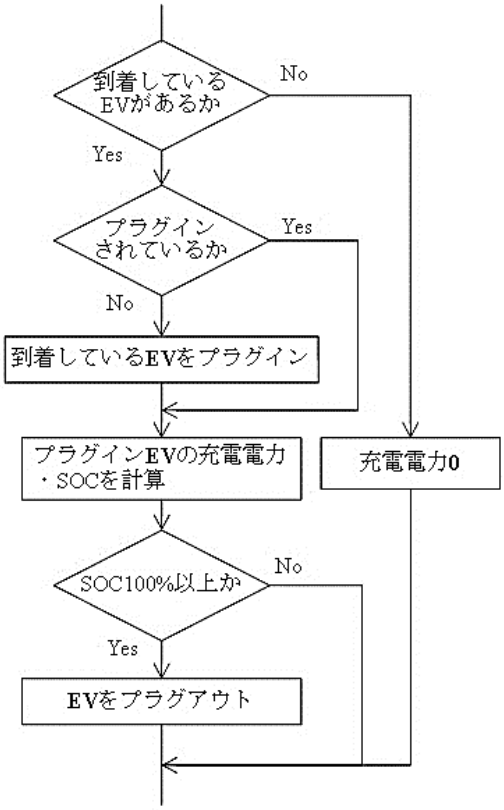


図 2.13 充電器の充電電力決定アルゴリズム

2.4.3　試算結果

はじめに，制御しない場合の各高圧ノードの電圧と系統内の最大電圧($V_{max}(t)$ ：グラフの凡例は(t)を省略)を図 2.14 に示す。PV 出力がほとんどない，朝時間帯や夕方以降では，電圧は比較的一定を保っている。しかしながら，午前中(10:00～12:00)を中心に図 2.9 に示した PV の出力による，大きな電圧変動が発生している。結果的に，適正範囲の上限値(同図の赤破線:107V)の逸脱が 12:00 付近を中心に発生している。

これに対して各制御(短周期制御または長周期制御)のみを実施した場合と，提案した各制御および制御量の修正を行った場合（以後「提案手法」）の結果を図 2.15 に示す。

図 2.15(a)の短周期制御では電圧変動が抑制され，PV の出力変動が大きい時間帯でも各高圧ノードの分布がこのグラフから判断できる程度になっている。その結果，系統内の電圧逸脱も低減されている。また 12:45～13:00 の定常的な電圧上昇も制御なしに比べて抑制されているが，タップ位置が最下限(97.125V)に切替される 13:00～13:18 は定常的に高い電圧値となっており，この時間内で複数時間の電圧逸脱が発生している。したがって，短周期制御により制御サンプリング時間間隔(1 分)の電圧変動を抑制することができるが，定常的な電圧変動（偏差）を解消することができず，電圧逸脱が発生する可能性がある。

一方，図 2.15(b)のように長周期制御のみでは，制御なしと同等の電圧変動の大きさであるが，特に 13:00 付近での定常的な電圧の偏差が制御なしの場合よりも小さい。その結果として，電圧逸脱回数(時間)が制御なしよりも少なく，電圧逸脱は短周期(制御サンプリング間隔)の変動によって発生している。

図 2.15(c)が提案手法の結果を示しているが，短周期制御と同様，電圧変動が共に抑制されている。また，長周期制御により，短周期制御のみの場合で電圧逸脱が発生していた 13:00～13:18 の電圧変動(偏差)が低減され，その結果として電圧逸脱量・タップ切替数が低減されている。

これらの 4 つの場合における電圧分布から電圧制御指標を算出した結果を表 2.4 示す。制御なしの場合に比べ，短周期制御は電圧変動・電圧逸脱量，長周期制御は LRT タップ切替回数および電圧逸脱量が低減され，提案手法を実施することにより全ての指標が試算の中で最小の値となっている。

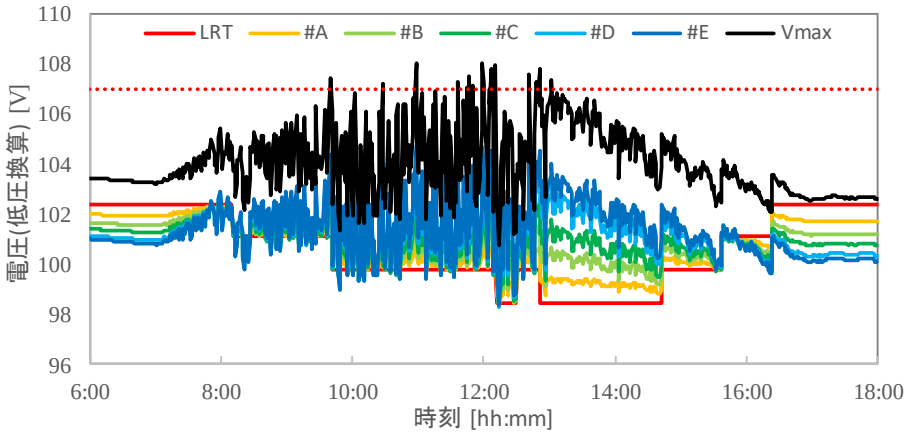
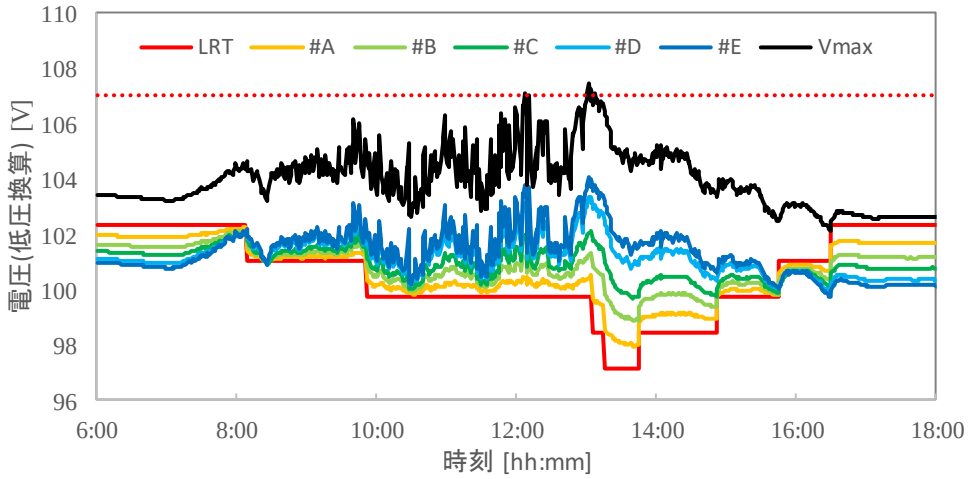
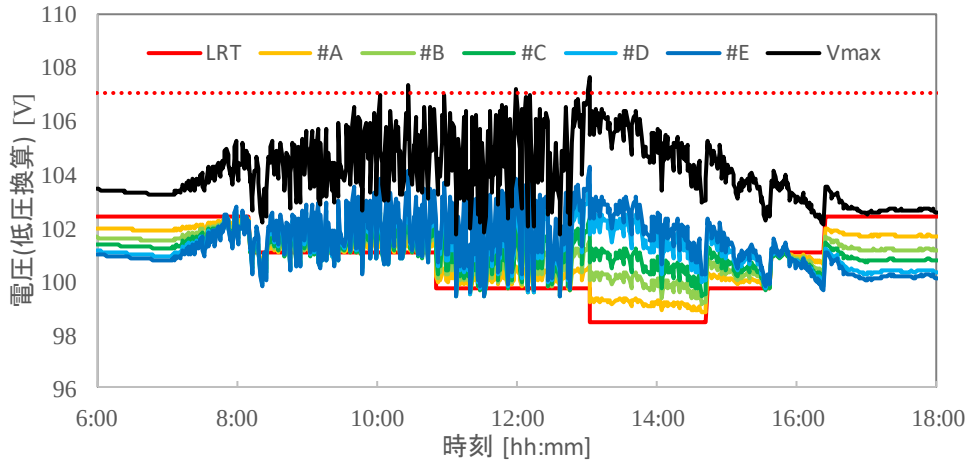


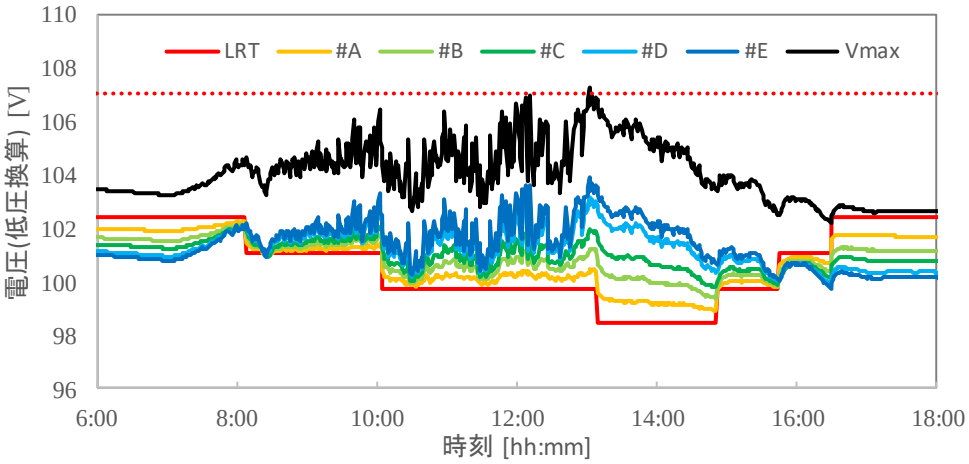
図 2.14 制御なしの場合の電圧分布



(a) 短周期制御



(b)長周期制御



(c) 提案手法

図 2.15 各制御および提案手法を実施した場合の電圧分布

次に各制御における合算した無効電力補償量について比較する。図 2.16 が各制御適用時と提案手法適用の全充電器の無効電力補償量の合算値を示しており，図 2.17 は提案手法における全充電器の無効電力補償量の短周期制御と長周期制御の内訳を示している。なお，両図の正値は充電器が消費（吸収）する向き，負値は充電器が発生（系統に注入）する向きを示している。本試算では定常的な電圧変動は電圧上昇傾向であるため, 長周期制御による補償は正方向のみである。これらの図を比較すると，提案手法は短周期制御と長周期制御の両方の機能を活用した結果，各制御よりも良い電圧制御指標となっていることが考えられる。

表 2.4 電圧制御指標の算出結果

	LRTタップ切替回数	電圧変動量[V]	電圧逸脱量 [Vmin]
制御なし	8	1.590	7.402
短周期制御	8	0.459	1.030
長周期制御	6	1.320	1.209
提案手法	6	0.456	0.230

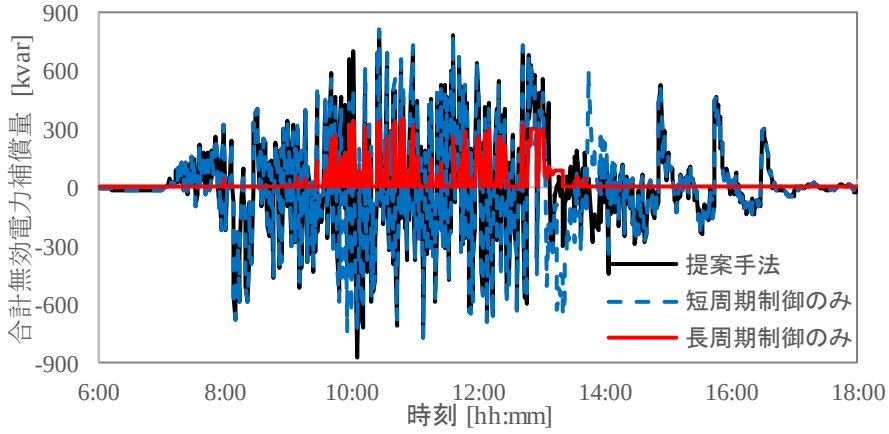


図 2.16 各制御適用時と提案手法適用の全充電器の無効電力補償量

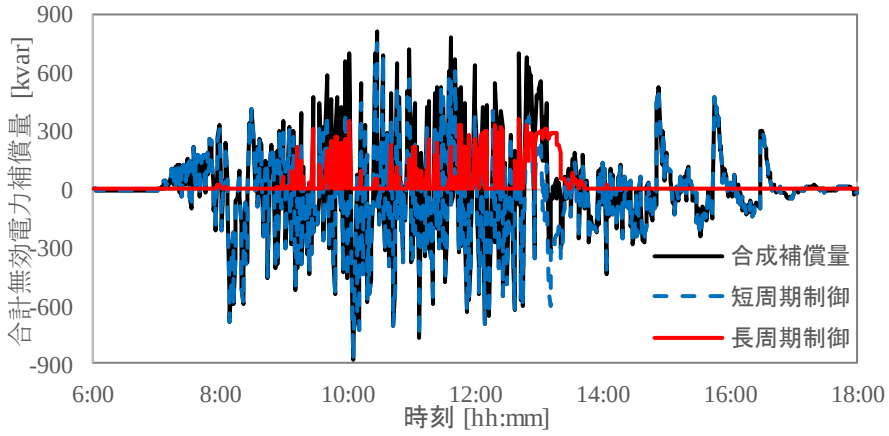


図 2.17 提案手法適用時における全充電器の合算無効電力補償量とその内訳

次に，制御なしの場合と提案手法の制御による LRT タップ切替動作を影響について比較する。図 2.18 は LDC 方式に基づく LRT の推定電圧，図 2.19 は LRT の推定電圧の不感帯逸脱積算量を

示している。制御なしの場合では，大きな電圧変動に伴い推定電圧値も大きく変化している。特に 9:30～13:00 の電圧変動が大きく，頻繁に推定電圧値の不感帯(同図の破線)を逸脱しており，図 2.19 の不感帯逸脱積算量も正負に大きく変動している。12:30～12:44 では PV 出力が比較的小さい時間が継続したことによりタップの上げ動作がされているが，このタップの上げ動作後に PV 出力が大きくなり，図 2.15 に示すように電圧逸脱が発生している。

一方，提案手法を適用すると，電圧変動と同様に LRT の推定電圧の変動も抑制され，その結果として不感帯の逸脱回数，量も減少している。また，不感帯逸脱発生するが LRT タップ切替不感帯逸脱積算量(図 2.19 の破線)に達しないケースが少なく，多くのケースで破線に達していることから，短周期の電圧変動低減によって LRT のタップ動作(適正なタップ位置)の判断が容易となったといえる。その結果として，提案手法では 12:31 および 12:51 のタップの上げ・下げ動作を回避しており，LRT のタップ動作と協調を図ることができていると考えられる。

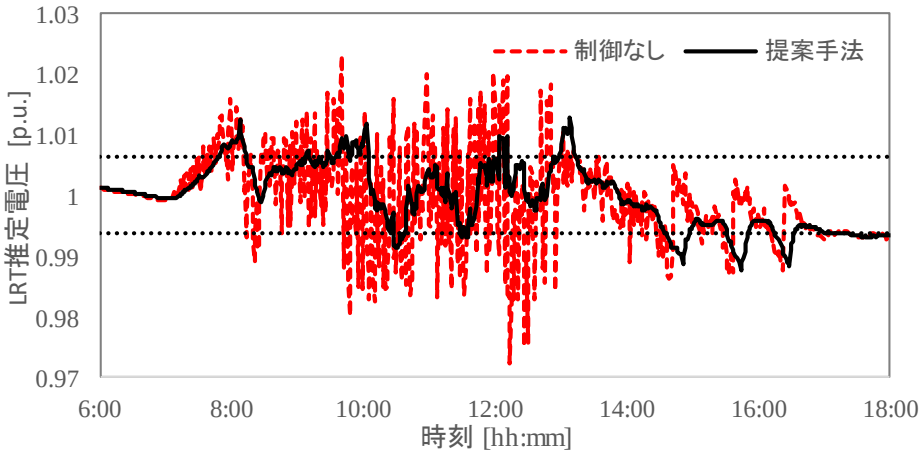


図 2.18 各条件における LRT の推定電圧

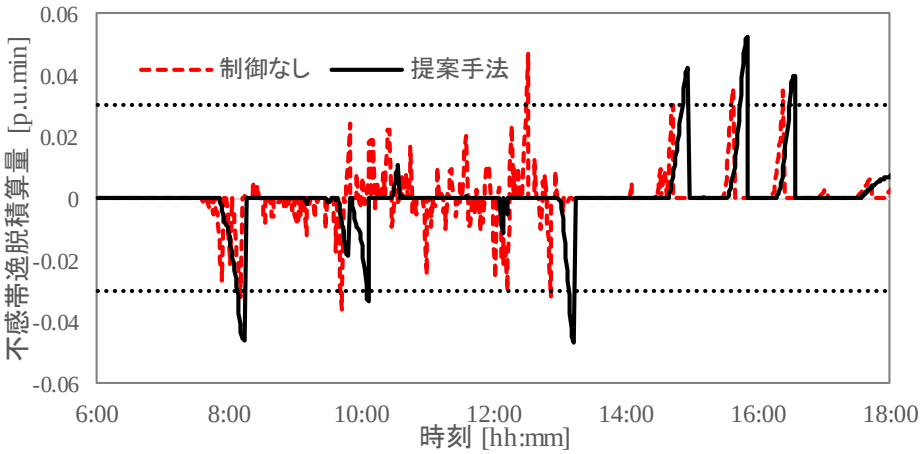
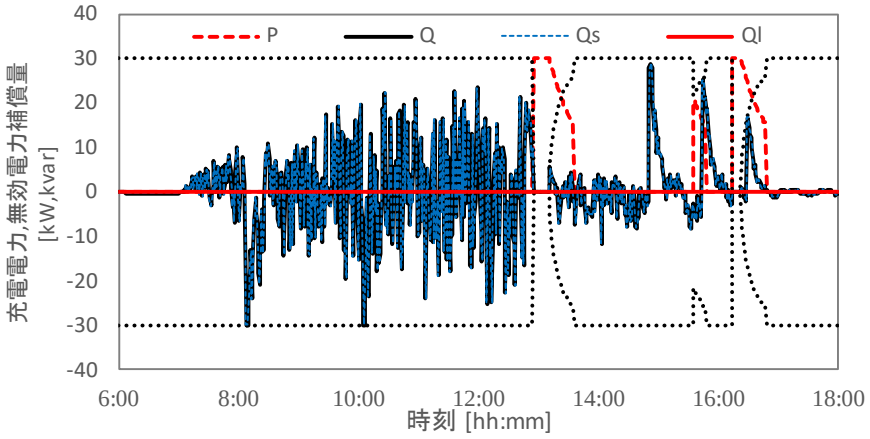
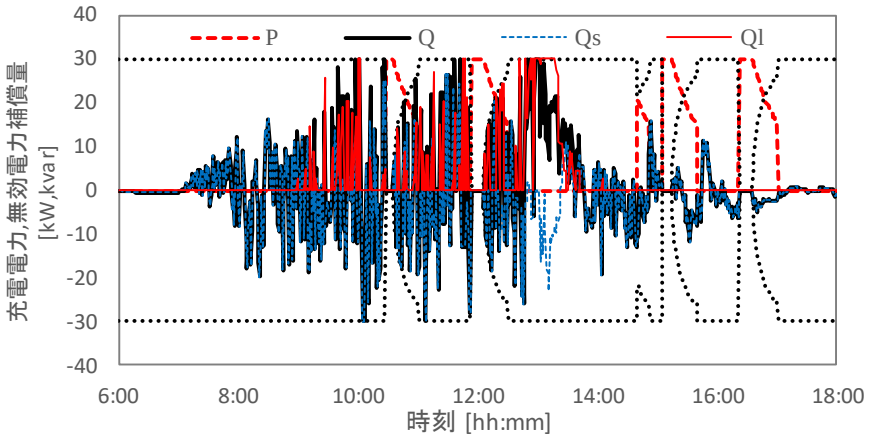


図 2.19 各条件における LRT の不感帯逸脱積算量

また，図 2.20 は提案手法において，代表的な 2 つ高圧ノード(#A:電源側，#E:末端側)に接続された充電器 1 台の充電電力および各制御内訳を含む無効電力補償量を示している。同図の黒破線は充電器の無効電力補償可能範囲を示しており，充電電力(図中の P)によって制限される。充電器の利用状況が異なるため，無効電力補償量は同じノードに接続された充電器であっても一致しない。図中の短周期制御(Q_s)は接続個所によらず同程度の大きさの補償を行っており，制御定数は妥当であると考えられる。



(a) 電源側(#A)ノード

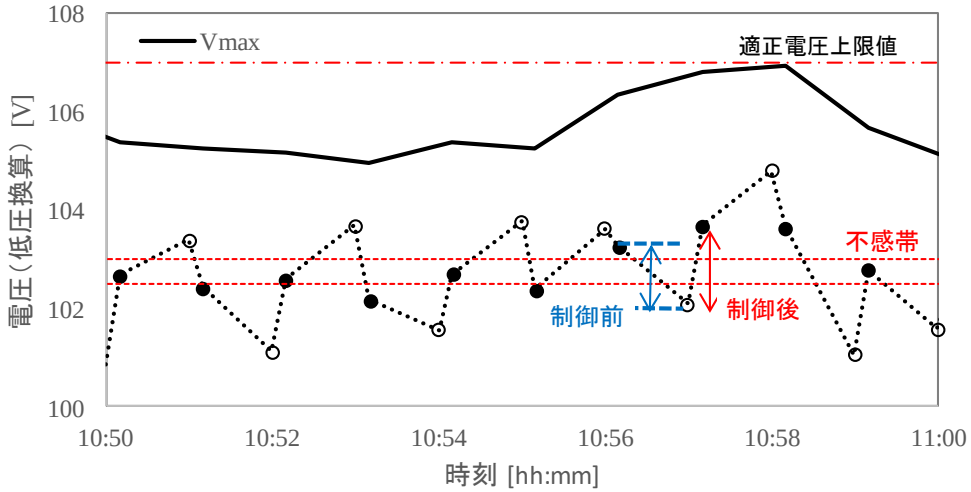


(b) 末端側(#E)ノード

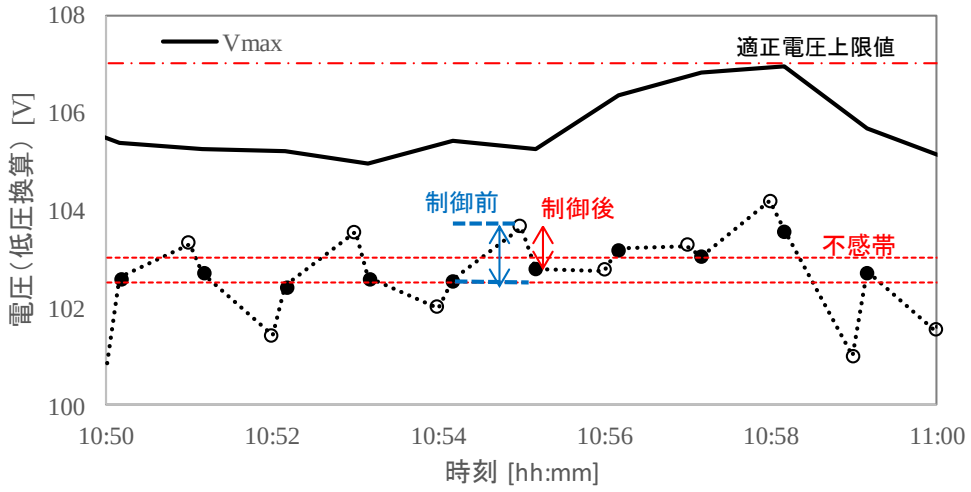
図 2.20 各ノードに接続されている代表的な充電器 1 台の無効電力補償量

次に，各機能に基づく補償量の修正方法の有効性について検証する。図 2.21 は，補償量の修正の有無による電圧分布の較結果を示している。図 2.21 の赤の破線はこのノードに接続されている充電器の長周期制御の不感帯，赤の一点破線は適正電圧の上限値，黒の破線がこのノードで測定される電圧の推移，黒の実線が系統全体の電圧値をそれぞれ示している。同図のプロット点が各時間帯における制御前後の推移であり，白丸のプロットが制御前の電圧$V_i(t)$，黒丸のプロットが制御後の電圧$V_i'(t)$，青矢印が制御前の電圧変動量$(V_i(t)-V_i'(t-\Delta T))$，赤矢印が制御後の電圧変化量$(V_i'(t)-V_i(t))$を示している。図 2.21(a)の結果を見ると，各制御機能で算出した制御量を足し合わせて補償しているため過補償が生じており，10:57 の断面で本来補償したい電圧変動（青矢印）よりも，最終的な電圧変化量(赤矢印)が大きくなっている。一方，図 2.21 (b)は制御量修正を実施した場合の結果であるが，同時間帯に必要以上の補償とならずに済んでいる様子が見て取れる。

これらの結果から，提案手法は自律分散動作を前提とした充電器において，必要最小限の補償で電圧制御を行う手法であり，充電器の利用状況によって変化する補償可能範囲内で有効活用できているのではないかと考えられる。同様の効果が，10:51，10:53，10:55 等でも観察できる。



(a) 修正を行わない場合



(b) 修正を行った場合(提案手法)

図 2.21 各時間帯における制御前後の電圧と系統内の最大電圧

2.5　提案手法の制御能力推定

本節では，充電器の利用状況によって無効電力補償可能量が変化する，という不確実性を考慮し，提案手法の電圧制御能力を推定する。本節では，その電圧制御能力を推定する手法と数値試算に基づく推定結果について議論する。

2.5.1　推定手法

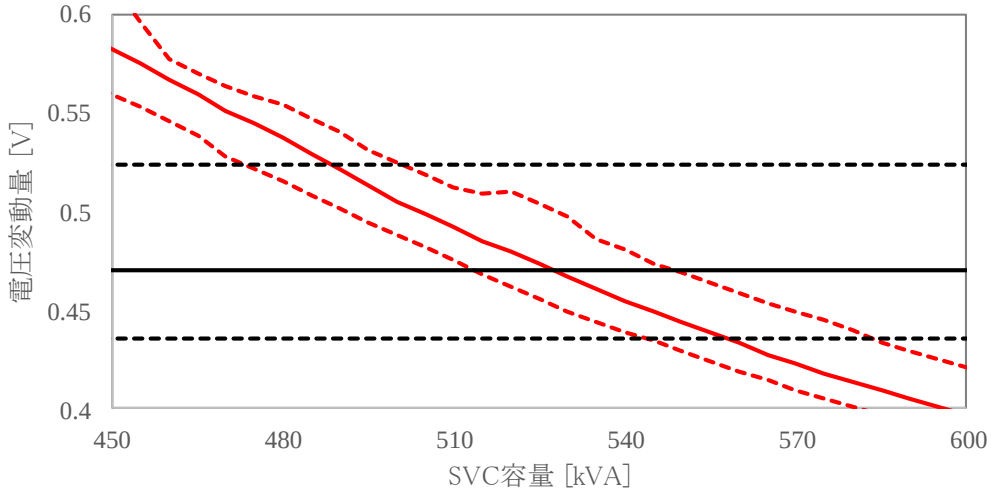
これまでに述べたように，充電器は本来の利用である充電利用（充電電力）により補償できる無効電力が制限される，という電圧制御機器にはない特徴がある。そのため，この能力を正確に把握することができれば，本来必要な電圧制御機器の容量の削減量が明らかになり，その対策のコスト低減に繋がるなどの利点が挙げられる。そこで本論文では，同じ無効電力補償機器である SVC に着目し，充電器を用いた提案手法を実施することにより，どれだけの SVC の容量が削減されるかという「等価 SVC 容量」を用いることとする。

この等価 SVC 容量を推定するにあたり，同条件において充電器を使用した場合の各電圧制御指標値と充電器の代わりに SVC を使用した場合の電圧制御指標を比較する。このとき，SVC 容量を段階的に変更して試算を繰り返し行い，充電器の各電圧制御指標値に最も近い値を等価 SVC 容量とする。なお，ここでは，充電器の利用状況の変化による等価 SVC 容量値の変化について検証するために， 図 2.9 で示した 1 日の PV および負荷パターンに対して，100 日分の充電器の利用シナリオ(充電電力)を用いる。一般的に SVC は系統の末端側に接続されることから，同様に系統の末端ノード(#E)に接続する。制御の違いによる影響をなくすために，SVC の制御方式は本論文で提案した測定電圧に基づく制御とする。なお，ここでは各制御の制御パラメータ(G_f および不感帯)は表 2.3 の#E の充電器と同じ値を採用するが，制御実施台数が異なるため制御定数の値は異なる($K^s = K^I = 0.0047$ [V/kvar])。

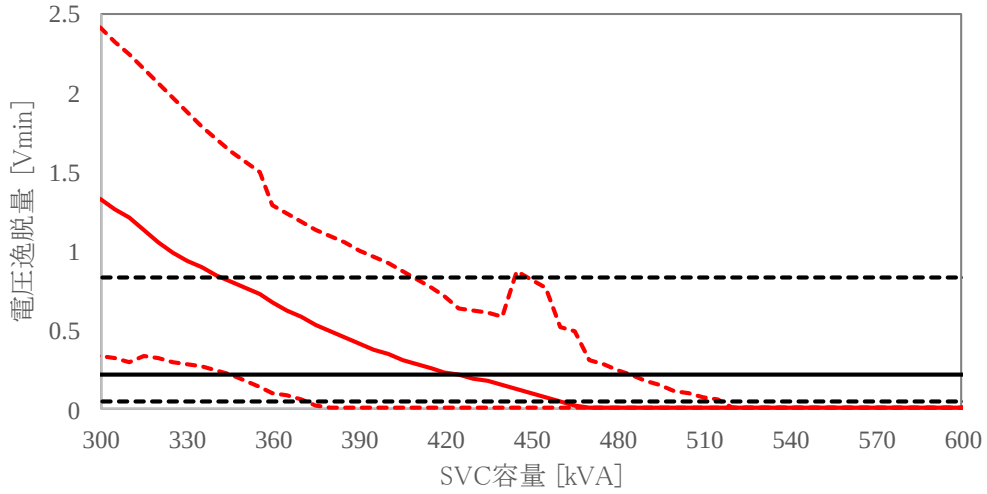
2.5.2　推定結果

図 2.22 は充電器を使用した場合および SVC を使用した場合の各電圧制御指標を分けて示しており，図 2.22 (a)は電圧変動量，図 2.22 (b)は電圧逸脱量である。同図の黒色で示したのが充電器を使用した制御，赤色で示したのが SVC を使用した制御における指標値を示している。また，破線は最大・最小を示しており，実線が平均値(期待値)を示している。

これらの図の SVC を使用した制御の値から，SVC 容量の増加に伴ってどちらの指標値も減少する傾向にある。電圧逸脱量の最大は 460kVA 付近で増加しているが，これは LRT のタップ切替のタイミングによって突発的な電圧逸脱によるものである。



(a) 電圧変動量



(b) 電圧逸脱量

図 2.22 充電器または SVC を用いた制御時の各電圧制御指標値

次に 1 日試算(1 充電パターン)毎に等価 SVC 容量を算出したヒストグラムを図 2.23 に示す。電圧変動量は，500kVA から 570kVA に分布しているが，電圧逸脱量は，335kVA から 475kVA と等価 SVC 容量値に差がある。また，等価 SVC 容量の平均値(期待値)は，電圧変動量に関しては 535kVA，電圧逸脱量に関しては 425kVA となる。

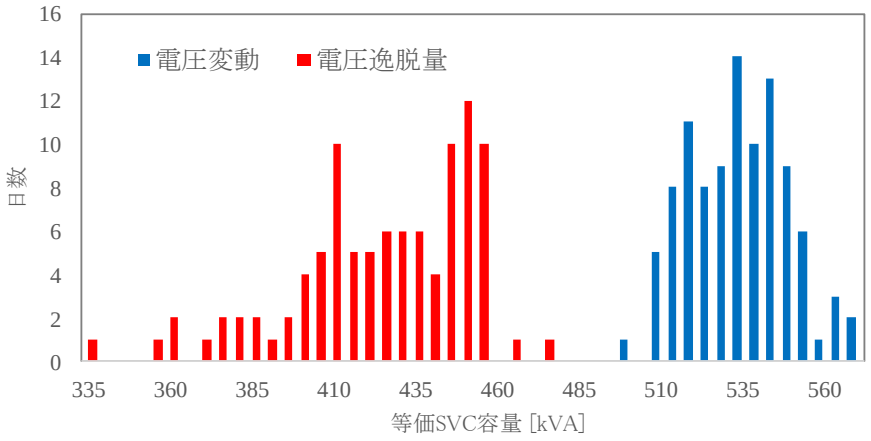


図 2.23 等価 SVC 容量の推定結果

SVC を設置した際の電圧分布の傾向について示すために，容量が 300，400，500, 750kVA の 4 種類の SVC を設置し，制御した場合のそれぞれの系統内の最大電圧を図 2.24 に示す。300kVA では電圧変動量・電圧逸脱量は大きいが，400kVA にすると 10:58 の電圧逸脱が解消され，500kVA 以上の容量では全ての時間で電圧逸脱は解消されており，電圧変動も抑えられている。

この結果から充電器の電圧制御能力は充電器の利用状況(充電電力)によって左右されるが，電圧変動量に関しては全充電器容量(30 台×30kVA=900kVA)の約 6 割，電圧逸脱量に関しては約 4～5 割程度の等価的な SVC 容量を持つことが確認できた。

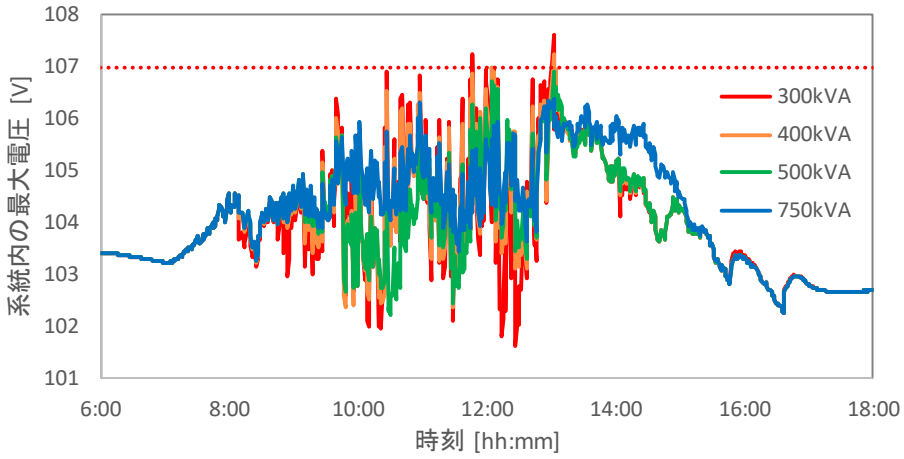


図 2.24 SVC を設置した場合の系統内の最大電圧

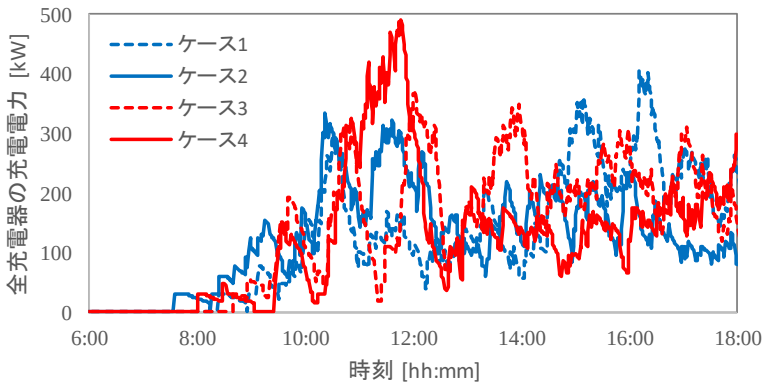
次に，前述したように各電圧制御指標値で差が生じた理由について考察するために，表 2.5 に示す各指標値による等価 SVC 容量の最大値・最小値にとなる日の 4 ケースについて検証する。

表 2.5 各電圧制御指標値における等価 SVC 容量の最大・最小値

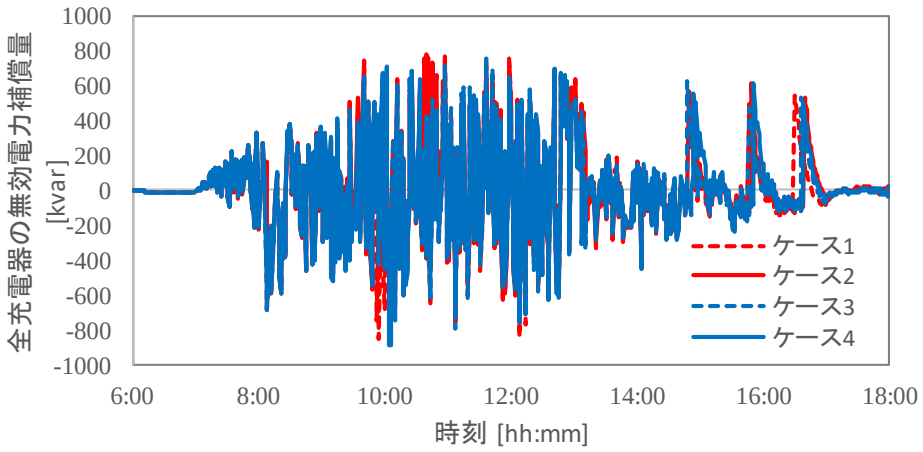
	各電圧指標値		等価SVC容量[kVA]
ケース1	電圧変動量	最大	570
ケース2		最小	500
ケース3	電圧逸脱量	最大	475
ケース4		最小	335

図 2.25(a)～(d)は 4 ケースの全充電器の充電電力，無効電力補償量，LRT のタップ位置，系統内の最大電圧を示している。図 2.25 の線の色はケース分けした指標(青：電圧変動，赤：電圧逸脱量)，線種は最小(破線)・最大(実線)でそれぞれ示している。推定値が最小である 2 ケースはどちらも 10:00～12:00 の充電電力が比較的大きく，ケース 2 では 10:42 で LRT のタップが上限方向に切替されているが，その後の充電電力の減少により 10:58 で電圧逸脱が発生している。一方，ケース 4 では 11:00～12:00 で 500kW 近い充電電力があったがその時間帯の PV 出力も大きく，結果的にタップ切替はされなかった。なお、電圧逸脱量は 10:58 の大きな電圧逸脱が発生するケース 2 が最も多いが，SVC による制御時も同様に多い傾向にあり，ケース 4 の方が電圧逸脱量における等価 SVC 容量の推定値は小さい。

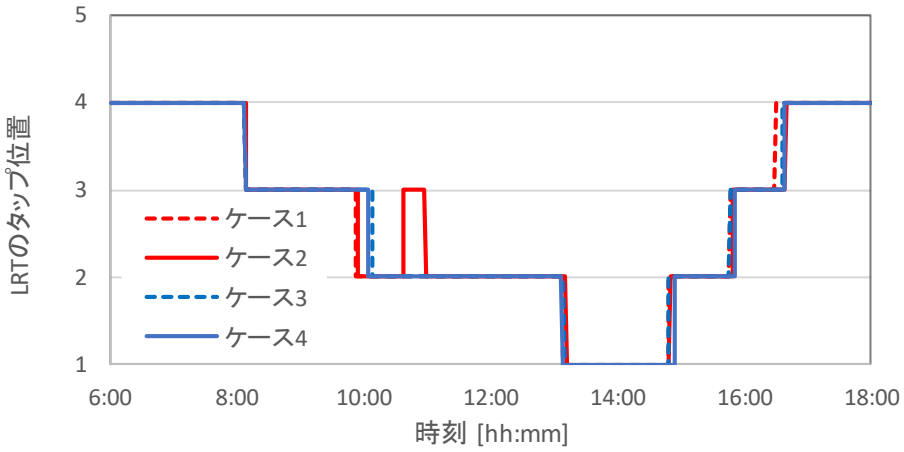
これより，充電器の利用シナリオ(充電電力)だけでは規則性を見つけるのは困難であると考えられる。特に電圧逸脱量に関しては，全てのケースで電圧逸脱が発生している時間帯(13:02)の無効電力補償量を見てもほとんど同じである。提案手法では前時間の制御に基づいて次の制御量を決定することから，等価 SVC 容量の推定には時系列の数値試算を実施することが不可欠であると考えられる。



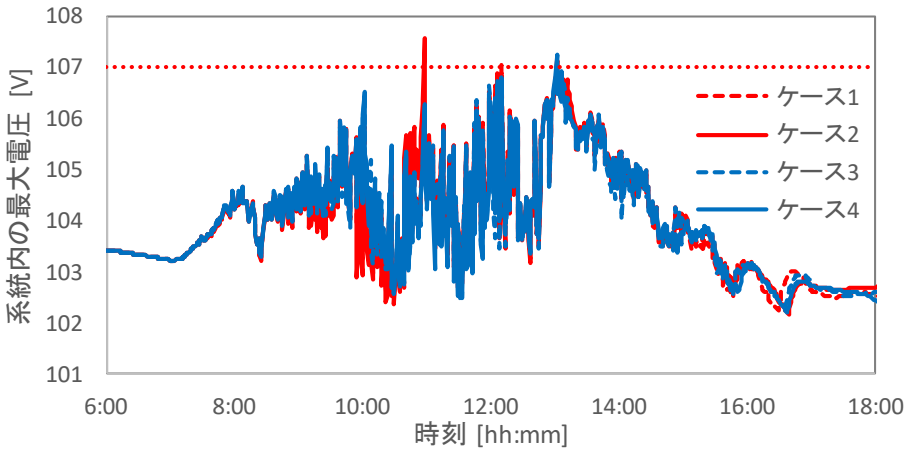
(a) 全充電器の充電電力



(b) 全充電器の無効電力補償量



(c) LRT のタップ位置



(d) 系統内の最大電圧

図 2.25 等価 SVC 容量値が最大・最小となるケースの検証結果

第3章　需給調整に貢献するコージェネレーションシステムの運用

第 1 章で述べたように，PV 等の自然変動電源の大量導入により，電力の需給バランスを調整する能力(調整力)が不足することが予想される。しかし，これまで調整力の主力を担ってきた火力発電機等の既存電源は既に最大限度に近い形で利用されている状況にある。また電源構成のうち自然変動電源の割合が増加していくことで，そもそも需給調整に利用可能な電源ユニット台数が減少すること，ならびに起動ユニットの出力レベルが最低出力付近となり発電出力減少方向の調整力が利用しづらくなる，いわゆる下げ代不足の両面から，更なる調整力を捻出することは困難であるといえる。将来の調整力不足の解決策の一つとして，需要家側の可制御機器(リソース)を系統運用に活用することが検討されているが，需要家側リソース活用に伴い生じる追加コストと対価とを直接比較しながら調整力を評価した事例は見当たらない。リソース活用を提供する需要家の視点からは, 調整力として提供するかどうかの判断に経済的な妥当性は必要不可欠であり，対価とコストの比較検討は重要な論点であろう。

また, 多数の需要家に需給運用に貢献してもらうためには, 需要家と系統運用者との間に立ち，多数のリソース群を束ねて統合的な運用を実現するアグリゲータの存在が必要不可欠であると著者らは考える。当然，需要家側のリソースを需給調整にどの程度利用できるかは，当該リソースの本来の用途や，定格容量や部分負荷運転の可否などの機器の運用上の制約などに大きく依存する。また，需給運用上も，調整力がどの程度必要かは日時・気象条件・既存電源の起動停止状態などの様々な要因で変化する。アグリゲータは，系統運用者からの要求に対して，需要家側の都合を勘案しながら効率的かつ柔軟に調整力を調達する機能が求められるであろう。

このような背景から本章では, 需要家側のリソースとしてコージェネレーションシステム（CGS）に焦点を当て，アグリゲータが多数の CGS 群から調整力を調達する状況を想定し，具体的な調達手順を提案する。また，CGS を所有する個別需要家の視点から，調整力提供に関わる経済性への影響を考慮した最適運用手法を提案し，北海道地域における実需要データを用いた数値試算により，CGS 群の調整力を評価する。

本論文の構成は次の通りである。まず 3.1 節では，想定する状況および調整力の考え方について説明する。3.2 節では本論文で提案する，アグリゲータが統括する CGS 群から調整力を調達する手順および需要家観点の運用の決定手法を説明し，3.3 節では実測データを用いた数値試算により，提案手法の妥当性の評価を行う。最後に 3.4 節では，CGS 群の調整力の評価を行う。

3.1 想定する前提条件

3.1.1 想定する調整力提供の枠組

本項では本論文で想定する調整力提供の枠組について述べるが，本論文では現在創設に向けた検討がなされている調整力市場に参加することを想定しているため，現状の調整力市場の検討と調整力提供の枠組を併せて説明していく。

まず系統運用者は，調整力市場における調整力の公募調達ルール[78]に従い，調整力を事前に調達する。そのルールの中には「一般送配電事業者による電源等の形態」として，電源Ⅰと電源Ⅱと呼ばれる 2 つの電源に大別され，それぞれ以下のように定義されている。

a)電源Ⅰ…送配電事業者の専用電源として，決められた時間帯の調整代を常時確保する電源

b)電源Ⅱ…小売電気事業者の供給力等と送配電事業者の調整力の相乗りとなる電源

図 3.1 は各電源の市場への入札方法および清算の概略図を示している。調整力市場の落札は，電源の種別に問わず，入札価格の安い順(メリットオーダ)に基づき行われる。一方，清算時には，電源Ⅰには常時の調整代確保(ΔkW)の対価と実際に使用された電力量(kWh)の対価の両者で支払われるが，電源Ⅱには実際に使用された電力量(kWh)の対価で支払われる，という違いがある。

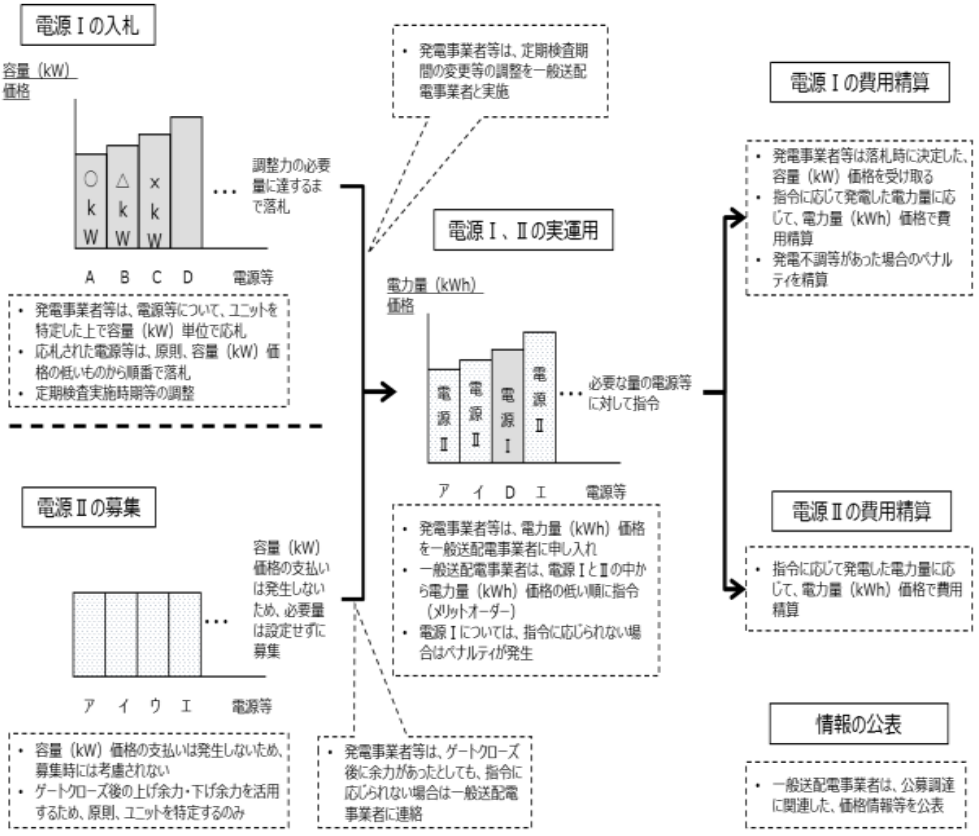


図 3.1 調整力市場における各電源の公募と清算[78]

次に，現状の料金の精算方法について述べる。文献[78]では，「調整力の募集に対して応募した者は，送配電事業者に対して，出力上げ調整単価、下げ調整単価、起動費等の単価表およびその算定基準となる火力発電機の熱消費量特性曲線より求めた定数等を定期的に提出」とされている。具体的には，「出力上げ(下げ)調整単価」とは，対象の発電機の出力を上げる(下げる)ことに調整に対して設定される単価であり，実際の上げ(下げ)に生じる燃料費の増分(減分)を補填するように価格設定される。これらの清算は，実際に需給バランスを逸脱(インバランスを発生)させた事業者が系統運用者に支払われるペナルティの一部から捻出される。

調整力市場に参加する発電機を所有する事業者は出力調整の上げ，下げに伴う費用の変化が大きいが一方で，CGS 等の需要家リソースからの調整力提供は，出力調整による変化は電源の比較的に安いとされており(3.3 節，3.4 節の試算結果でも述べるが単価 0 円でも一部提供可能)，需要家リソースが市場へ参入することよる効果は大きいことが予想されている。

ただし，現在検討されている調整力市場への参加要件は数千～数万 kW オーダであり，市販されている需要家リソースの容量は最大でも数百 kW 程度であることから，これらのリソースを群として活用することを要求され，リソース群を統括するアグリゲータの存在が不可欠である。また，需要家リソースは需要家都合(本来の利用)等により，常時確保は困難であることから，常に調整代(ΔkW)容量を常時確保する電源 I への参入は難しいだろう。このことから，本論文では，CGS 群を統括するアグリゲータは調整力市場へは電源Ⅱとして参入することを想定する。

これらを踏まえて本論文では，系統運用者，アグリゲータ，需要家の間に図 3.2 に示すような枠組みを想定する。系統運用者は，系統運用上必要な調整力(以後，必要調整力と呼ぶ)を，従来から用いられてきた出力調整用電源に加えて，需要家側リソース群を統括するアグリゲータからも調達する。CGS 群から調達した調整力が必要調整力に達しない場合は，その不足分をアグリゲータが管轄する他のリソースや調整用電源から補填する。また，アグリゲータは，系統運用者からの需給調整の指令に対して，自身が管轄する CGS 等のリソース機器の出力を調整してもらうことになる。その際アグリゲータは，系統運用者から需給調整に得られる報奨金の一部を，需給調整に貢献した需要家に対して報奨金を支払う。

このような想定する枠組みの中で，本章の検討対象は図 3.2 の破線で示すように，必要調整力に対して，できるだけ多くの調整力を CGS 群から調達することである。なお，図 3.2 の破線の部分は「事前に CGS 群から調整力を電源 I のように調整代を確保し，その確保量に対して報奨金を支払う」ということを想定している。これは「CGS 群は調整力市場に対して電源Ⅱとして機能するが，アグリゲータに対しては電源 I のように機能する」ことを意味している。前述した調整力市場の取り決めは系統運用者とアグリゲータ間であり，アグリゲータと需要家の取り決めについては調整力市場の取り決めに必ずしも沿う必要はない。むしろ，アグリゲータとしては事前に調整力調達を行わなければ，アグリゲータの他の調整用電源や需要家リソースの運用を決定し，エネルギー市場や調整力市場への意思決定ができず，アグリゲートすることの効果が十分に発揮されないだろう。

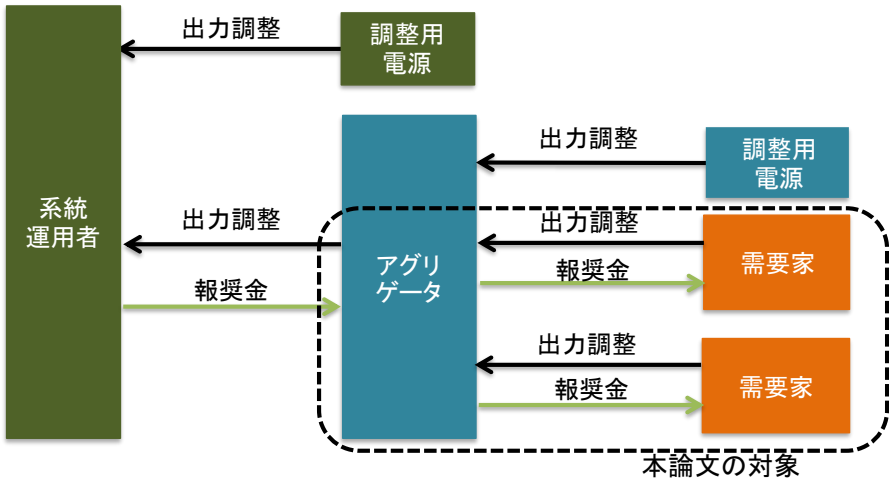


図 3.2 想定する調整力提供の枠組み

3.1.2 想定する需要家モデル

本章で想定する需要家は，燃料電池(PEFC)型の CGS を所有している一般住宅である。エネルギー供給モデルは図 3.3 に示すような，実際に市販されている燃料電池 CGS の一般的なものを想定する。需要家は電力系統およびガス系統に連系され，電力・給湯・暖房需要をそれぞれ賄う。暖房需要については補助熱源としてのガスボイラ，給湯需要は CGS およびガスボイラから供給するが，CGS はその排熱を一時的に貯めることができる蓄熱槽を有する。なお，エネルギー効率および環境性の観点から CGS の運転に関しては，熱出力を熱需要で吸収できる範囲に限定することとする(熱廃棄禁止)。ただし，CGS 発電出力が電力需要を上回る場合には，電力系統側への逆潮流(売電)を認めることとする。

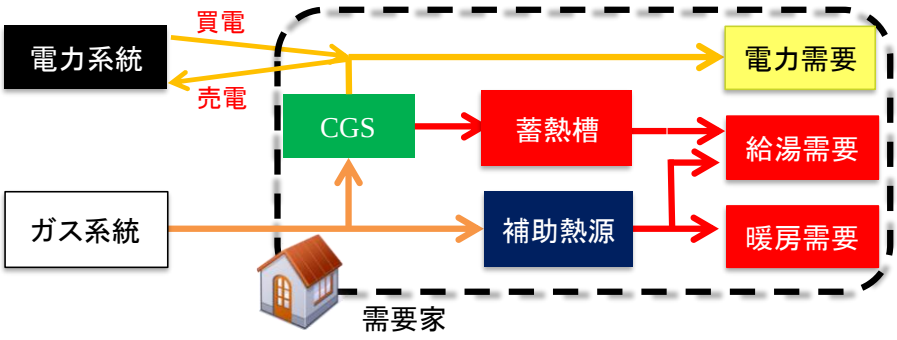


図 3.3 想定する CGS 所有需要家のエネルギーフロー

3.1.3　調整力提供の考え方

図 3.4 は本論文で想定する調整力提供の考え方を示している。CGS がアグリゲータからの要請に従って対応周期(Δt_{reg} [h]，事前の契約により指定されているものとする)の半分の期間($\Delta t^{kW} = \Delta t_{reg} / 2$)，その出力を計画値(当初の運転状態 $E_{CGS}(t)$)[kW]から $\Delta E_{CGS}(t)$ [kW]だけ持続的に増加(減少)させられる状態にあるとき，「出力増加(減少)方向の調整力を提供可能」と定義する。これに対し，系統運用者からの要請に応じて実際に CGS の出力を調整することを「調整力を発動する」と定義する。

また，調整力の大きさと対応周期の半周期の積($\Delta E_{CGS}(t)\Delta t^{kW}$,図 3.4 の各色で塗られた面積に相当)は，CGS が提供(あるいは発動)する調整力の量(調整量)を表しており，その意味は「1kW の調整力を 1 時間提供(あるいは発動)した」ことを表しているため，エネルギーの単位([kWh])と区別する目的で本論文では[kW・h]と表記することとする。

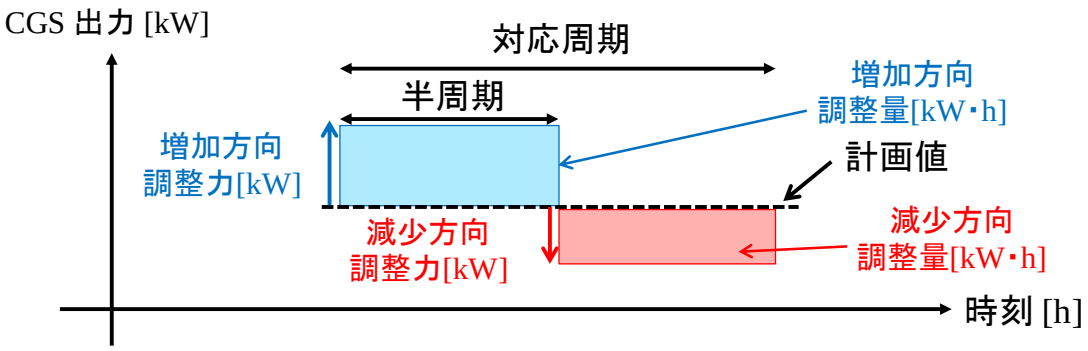


図 3.4 想定する調整力提供の考え方

3.2　提案する需給調整に貢献する CGS 群の運用

3.2.1　アグリゲータが調整力を調達する手順

前節で述べたように，アグリゲータの役目は複数の需要家を統括することで，必要調整力を CGS 群から柔軟に調達し，またその見返りとして需給調整に貢献した需要家に対して報奨金を支払うことである。しかしながら，アグリゲータが対象需要家の CGS の特性や電力・熱需要を把握した上で全ての CGS を直接的・統括的に運用することは，需要家のプライバシー保護や計算負荷の観点から現実的ではない。また，CGS からの調整力提供は需要家の各種需要プロファイルに依存するため，各 CGS が提供可能な調整力は需要家毎・時間帯毎で偏りがあることが自然であり，全 CGS に対して同量の調整力提供を依頼しても必要調整力を調達できなくなる可能性がある。

そこで本論文では，アグリゲータが各需要家に優先順位を付け，その優先順位が上位の需要家に対して順に調達すべき必要調整力の残存分と報奨金単価を提示し，また各需要家はアグリゲータから提示された情報に基づいて CGS の運用を自ら決定する手順を提案する。図 3.5 は提案す

る需要家から調整力を調達する手順とそのイメージ図を示している(図中の①～⑤は以下の手順の番号と対応)。

①全需要家に優先順位(1,2,…)を割当する。

アグリゲータが各需要家に対して優先順位を決定する。ただし，優先順位によって需要家間で不公平性が生じてしまう可能性があるため，できるだけ公平になるように，運用計画(1日)毎に過去の報奨金の累積額を昇順で並べ替え，優先順位を変更する。

②優先順位が高い需要家 1 軒に各種情報(必要調整力・報奨金単価)を提示する。

アグリゲータは必要調整力の残存分(必要調整力に満たすための残り量)とあらかじめ決定した報奨金単価を i 番目の需要家(以後，需要家 i)に提示する。

③対象需要家は運用計画を実施し，アグリゲータへ調整力提供・報奨金受取を行う。

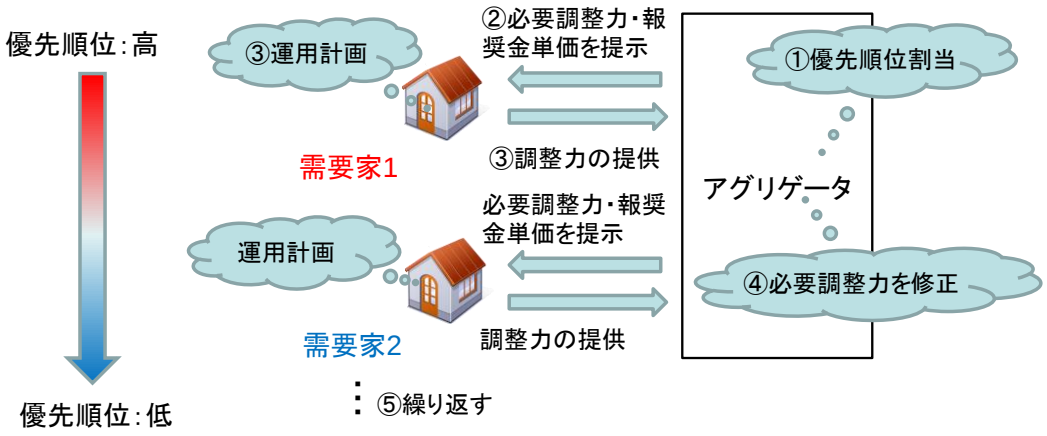
アグリゲータから各種情報が提示された需要家は，提示された必要調整力の残存分と報奨金単価に基づき，自身の運用を次項に示す手法に従い計画する。運用計画後，需要家は運用計画によって確保した調整力をアグリゲータに提供し，一方，アグリゲータは需要家に対して報奨金を付与する。なお，本論文で取り扱う報奨金は運用計画時における調整力提供への対価であり，運用時の調整力発動の状況によらず支払われるものである。

④提供された調整力に応じて次の需要家に提示する必要調整力の残存分を修正する

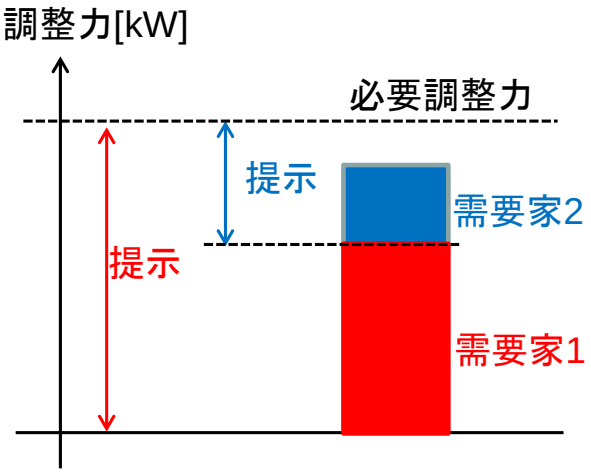
次の需要家 $i+1$ に提示する必要調整力の残存分を修正する。これは，もともとの必要調整力から，1～(i -1)番目の需要家からすでに調達した調整力を差し引いたものである。

⑤上記の②～④を需要家軒数だけ繰り返す

上記の手順を全ての需要家に対して優先順位順に実施する。



(a) 提案する需要家から調整力を調達する手順の概要



(b) 需要家に提示する必要調整力の残存分と調達した調整力の関係

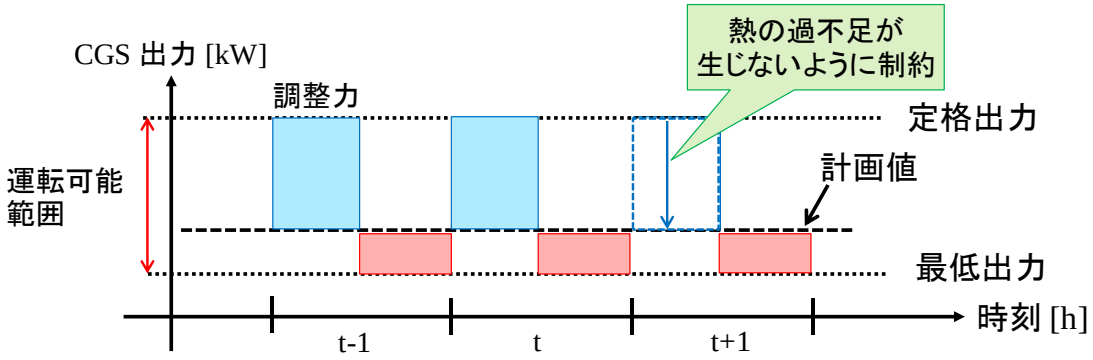
図 3.5 提案する需要家から調整力を調達する手順とそのイメージ

3.2.2　各需要家の運用計画

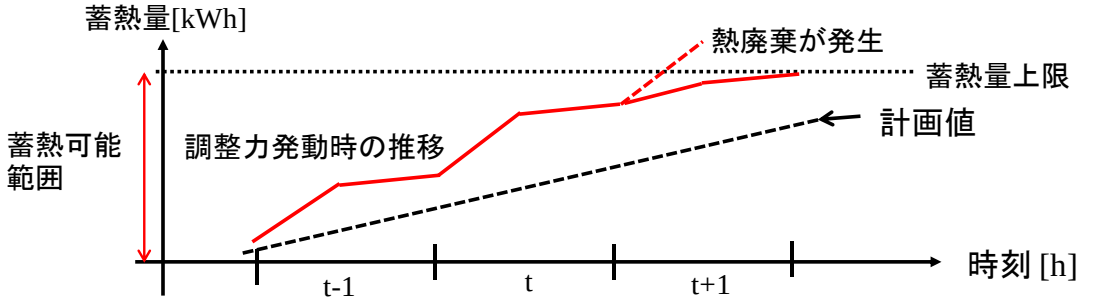
各需要家は，アグリゲータから提示される報奨金単価および必要調整力を元に，自身のエネルギー供給手段の計画を立てる。具体的には(3.1)式に示すように，各需要家は，エネルギー供給コストから報奨金として得られる収入を差し引いた，トータルコストTC_i（i ：需要家の優先順位）の最小化を目的関数とする，混合整数線形計画問題（Mixed Integer Linear Programming :MILP）を解くことで運用計画を行う。

なお，確保しておいた調整力がどのように発動されるかによって，蓄熱槽内の蓄熱量[kWh]は変化することになる。ただし，調整力発動のタイミングや方向，大きさは運用時における系統全体の需給バランスに依存する。そのため，運用計画段階では，調整力がどのように発動されるかの不確実性のシナリオを考慮して蓄熱槽の運用のマージンを確保する必要がある。そこで本論文では調整力発動のシナリオとして「提供する出力増加、出力減少方向の調整力の全てが対応周期の半分ずつ発動される状況」を想定し，計画時にはこのシナリオのもと，各種運用制約を違反しないように，運用のマージンを設けることとする。図 3.6 はそのイメージ図を示しているが，この図において，図 3.6(a)の CGS 出力で見ると，t+1 で計画値から定格出力まで出力増加方向，また計画値から出力減少方向の両方向で調整代(ΔkW)提供可能である。しかしながら，t-1～t+1 の間で提供した調整力に対して全量の出力調整指令が来ると，図 3.6(b)の蓄熱槽内の蓄熱量(kWh)が蓄熱可能範囲を逸脱し，熱廃棄が生じてしまう。

そのため本論文では，上記のシナリオにおいて，CGS 出力(ΔkW)，蓄熱量(kWh)の両者のマージンが十分にある範囲で調整力提供を行う。



(a) CGS 出力の調整力発動に対するマージン



(b) 蓄熱量の調整力発動に対するマージン

図 3.6 運用計画段階における調整力発動に対するマージンのイメージ

本最適化問題における変数は以下のようにまとめる。これらの変数のうち，決定変数については添え字以外を大文字で表している。また，図 3.7 は図 3.3 の需要家のエネルギーフローにエネルギーバランスに関する変数を追加した変数の関係図である。

【各種変数】

Δt ：計画時間間隔[h],

$\Delta t / \Delta t_{reg}$ ：計画時間間隔(Δt)中の調整力の対応周期の数，$G_{pur,i}(t)$ ：ガス消費[kW],

p_{gas} ：ガス購入単価[¥/kWh],

$E_{pur,i}(t)$, $E_{sell,i}(t)$ ：買電・売電電力[kW],

p_{pur} , p_{sell} ：買電，売電単価[¥/kWh],

$inc^{up}(t)$, $inc^{dn}(t)$ ：調整力提供に対する報奨金単価[¥/kW・h],

$\Delta E^{up}_{CGS,i}(t)$, $\Delta E^{dn}_{CGS,i}(t)$ ：方向別の調整力の大きさ[kW],

$G_{CGS,i}(t)$, $G_{B,i}(t)$ ：CGS およびボイラーのガス消費[kW],

$B_{CGS,i}(t)$ ：CGS の起動停止バイナリ変数（0：停止，1：起動），

α_{CGS} , β_{CGS} ： CGS 発電電力に対するガス消費量を線形近似した 1 次係数・定数項(β_{CGS} の単位は[kW]),

$HW_{B,i}(t)$, $HD_{B,i}(t)$ ：ボイラーからの熱供給（給湯，暖房）[kW]，η_B ：ボイラー効率,

$e_{dem,i}(t)$ ：電力需要[kW],

$B_{e,pur,i}(t)$ ：買電を表すバイナリ変数（0：売電，1：買電),

$e^{\max}_{pur}$, $e^{\max}_{sell}$ ：買電・売電の最大値[kW],

$hw_{dem,i}(t)$, $hd_{dem,i}(t)$ ：給湯・暖房需要[kW],

$HW^{out}_{TS,i}(t)$ ：蓄熱槽からの排熱[kW],

$e^{\max}_{CGS}$, $e^{\min}_{CGS}$ ：CGS の定格・最低発電出力[kW],

$H_{CGS,i}(t)$ ：CGS から排出される熱[kW],

ehr_{CGS} ：CGS 出力の熱電比，$H^{kWh}_{TS,i}(t)$ ：蓄熱槽の蓄熱量[kWh],

$h^{kWh,\max}_{TS}$ ：蓄熱槽の蓄熱量上限[kWh],

$\Delta H^{kWh}_{TS,i}(t)$ ：調整力発動による蓄熱量の変化分[kWh],

$\Delta e^{up}_{req}(t)$, $\Delta e^{dn}_{req}(t)$ ：方向別・時間帯別の必要調整力 [kW]

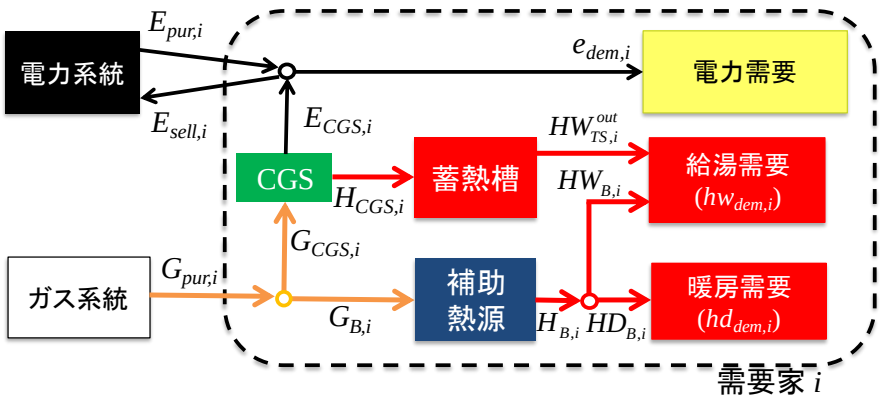


図 3.7 需要家のエネルギーバランスに関する変数

【目的関数】

$$\min TC_i$$

$$\begin{aligned} TC_i = & \sum_t \left\{ G_{pur,i}(t) \cdot p_{gas} + E_{pur,i}(t) \cdot p_{e,pur} - E_{sell,i}(t) \cdot p_{e,sell} \right\} \Delta t \\ & - \frac{\Delta t}{\Delta t_{reg}} \sum_t \left\{ inc^{up}(t) \Delta E^{up}_{CGS,i}(t) + inc^{dn}(t) \Delta E^{dn}_{CGS,i}(t) \right\} \Delta t^{kW} \end{aligned} \tag{3.1}$$

$$G_{pur,i}(t) = G_{CGS,i}(t) + G_{B,i}(t) \tag{3.2}$$

$$G_{CGS,i}(t) = \alpha_{CGS} E_{CGS,i}(t) + \beta_{CGS} B_{CGS,i}(t) \tag{3.3}$$

$$G_B(t) = \frac{1}{\eta_B} \left\{ HW_{B,i}(t) + HD_{B,i}(t) \right\} \tag{3.4}$$

(3.1)式はトータルコスト TC_i を最小化する目的関数である。(3.1)式の TC_i に関する式の右辺第 1 項がエネルギー供給に伴うコスト(供給コスト)であり，第 2 項は調整力提供によって得られる報奨金を示している。供給コストは電力系統およびガス系統からの取引量に元に算出され，報奨金はそれぞれの方向別の調整量に，方向・時間帯別の報奨金単価 $inc^{up}(t)$, $inc^{dn}(t)$ を乗ずることで算出される。(3.2)～(3.4)式は時刻 t における(3.1)式の供給コストの内訳を示しており，(3.2)式はガス購入，(3.3)式は CGS 発電に伴うガス購入，(3.4)式は補助熱源の出力をそれぞれ示している。

【電力の需給平衡制約】

$$e_{dem,i}(t) = E_{CGS,i}(t) + E_{pur,i}(t) - E_{sell,i}(t) \tag{3.5}$$

$$E_{pur,i}(t) \le B_{e,pur,i}(t) \cdot e^{\max}_{pur} \tag{3.6}$$

$$E_{sell,i}(t) \le \left(1 - B_{e,pur,i}(t)\right) \cdot e^{\max}_{sell} \tag{3.7}$$

(3.5)式は電力需給平衡制約を示しており，(3.6),(3.7)の制約式中のバイナリ $B_{e,pur,i}(t)$ により，同時に買電・売電を行わないようにしている。ただし，買電・売電の最大値 $e^{\max}_{pur}$, $e^{\max}_{sell}$ は一般的に契約電力を示すが，ここでは(3.5)式が必ず成立するよう(電力需要・CGS 定格以上)に十分大きい値とする。

【熱の需給平衡制約】

$$hw_{dem,i}(t) = HW_{TS,i}^{out}(t) + HW_{B,i}(t) \tag{3.8}$$

$$hd_{dem,i}(t) = HD_{B,i}(t) \tag{3.9}$$

熱についても電力と同じように需給平衡制約が成立するようにする。

【CGS に関する制約】

$$E_{CGS,i}(t) - \Delta E_{CGS,i}^{dn}(t) \geq B_{CGS,i}(t) \cdot e_{CGS}^{\min} \tag{3.10}$$

$$E_{CGS,i}(t) + \Delta E_{CGS,i}^{up}(t) \leq B_{CGS,i}(t) \cdot e_{CGS}^{\max} \tag{3.11}$$

$$H_{CGS,i}(t) = ehr_{CGS} E_{CGS,i}(t) \tag{3.12}$$

$$H_{TS,i}^{kWh}(t) = H_{TS,i}^{kWh}(t-1) + \left\{ H_{CGS,i}(t) - HW_{TS,i}^{out}(t) \right\} \Delta t \tag{3.13}$$

$$0 \leq H_{TS,i}^{kWh}(t) \leq h_{TS}^{kWh,\max} \tag{3.14}$$

$$H_{TS,i}^{kWh}(t) + \Delta H_{TS,i}^{kWh}(t-1) - ehr_{CGS} \Delta E_{CGS,i}^{dn}(t) \Delta t \geq 0 \tag{3.15}$$

$$H_{TS,i}^{kWh}(t) + \Delta H_{TS,i}^{kWh}(t-1) + ehr_{CGS} \Delta E_{CGS,i}^{up}(t) \Delta t \leq h_{TS}^{kWh,\max} \tag{3.16}$$

$$\Delta H_{TS,i}^{kWh}(t) = \frac{\Delta t}{\Delta t_{reg}} ehr_{CGS} \sum_{\tau=1}^{t} \left\{ \Delta E_{CGS,i}^{up}(\tau) - \Delta E_{CGS,i}^{dn}(\tau) \right\} \Delta t^{kW} \tag{3.17}$$

(3.10),(3.11)式は CGS 出力の上下限制約式であり，CGS が調整力を発動した場合にも CGS 本来の出力可能範囲に収まるように制約をかけている。(3.12)式は CGS の発電出力と排熱の関係性，(3.13),(3.14)式は調整力を発動しないときの蓄熱槽の熱量(計画値)の上下限制約を示している。一方，(3.15)～(3.17)式は運用断面において確保しておいた調整力が発動された場合の蓄熱槽の熱量の上下限制約である。ここで，(3.17)式左辺の $\Delta H_{TS}^{kWh}(t)$ は調整力発動に伴う蓄熱量の変化分(図 3.6(b)の黒破線と赤実線の差分)であり，調整力の方向別に調整量(kW・h)の値が異なる場合の影響を，計画開始時刻から時刻 t まで積算することで求めている。ただし，蓄熱槽内の熱の損失(自然放熱)についてはいずれも考慮していない。

【調整力提供に関する制約】

$$\Delta E^{up}_{CGS,i}(t) \le \Delta e^{up}_{req}(t) - \sum_{j=1}^{i-1} \Delta E^{up}_{CGS,j}(t) \tag{3.18}$$

$$\Delta E^{dn}_{CGS,i}(t) \le \Delta e^{dn}_{req}(t) - \sum_{j=1}^{i-1} \Delta E^{dn}_{CGS,j}(t) \tag{3.19}$$

(3.18)式が出力増加方向，(3.19)式は出力減少方向に関する調整力提供に関する制約式である。それぞれの右辺は，必要調整力 $\Delta e^{up}_{req}(t)$, $\Delta e^{dn}_{req}(t)$ から対象需要家 i より高い優先順位の需要家 $(1 \sim (i-1))$ が提供した調整力を差し引いた，アグリゲータから提示される必要調整力の残存分[kW]を示しており，需要家 i が提供する調整力は，この残存分以下に制約される。

3.3 数値試算による提案手法の妥当性検証

3.3.1 試算条件

本論文では，2008 年に北海道で実測された一般住宅 20 軒分のデータ(データ間隔 10 分，1 軒あたりの最大電力需要 3.6kW，最大給湯需要 56.0kW)に対して，CGS 群が提供する調整力を数値試算により評価する。表 3.1 は試算に用いたパラメータをまとめたものであり，各需要家には表 3.1 (a)に示す仕様の各種機器が設置されているものとする。各種料金単価や時間に関するパラメータは表 3.1 b)にまとめるが，ガス購入単価については北海道ガス(株)が定めている家庭用コージェネレーション契約料金[79](使用量区分 D・平成 27 年 9 月 1 日～)を，電力購入単価については北海道電力(株)が定めている家庭用電力の一般電気料金[80](従量電灯 B(40A)・平成 26 年 11 月 1 日～)を採用した。

運用計画は翌日 1 日分を対象期間として，前日の最終時間(24:00)に策定する。各計画の開始時刻(0:00)の蓄熱槽の蓄熱量 $H^{kWh}_{TS}(0)$ は，前日の最終時間(24:00)時点における運用時の(調整力発動の変化を含めた)蓄熱量とする。ただし，調整力提供によるコストの変化を明らかにするため，試算期間の開始・終了の蓄熱量は全ての条件および全需要家で同一になるように制約する。なお，前述の通り本試算では CGS の排熱は給湯需要にのみ供給される状況を想定しているため，以下の評価では暖房需要への熱供給やそれに付随するエネルギー供給コストを除いている。

調整力に関する評価指標としては，調整量[kW・h]および調整力提供時間を用いる。ここで調整力提供時間とは，「アグリゲータが CGS 群から調達した調整力が出力増加・出力減少方向共に必要調整力を満たしている時間」と定義する。なお，前述したように，調整力発動のタイミングや方向，大きさは，実運用断面の系統全体の需給バランスに依存するが，ここでは，提供した調整力を全て発動する，調整力発動のシナリオを想定する。また，需要家の各種需要も計画段階と実運用断面で変化しないものとしているため，策定した計画に従って運用，調整力発動を実施しても各種運用制約には抵触しない。そのため，本試算では「提供した調整量」と「発動した調整量」は同一の値であり，本節ではこれらを区別せずに「調整量[kW・h]」と表記する。

以後の試算では，アグリゲータが提示する報奨金単価($inc^{up}(t)$, $inc^{dn}(t)$)および必要調整力($\Delta e_{req}^{up}(t)$, $\Delta e_{req}^{dn}(t)$)をパラメータとして変化させながら調整量・調整力提供時間を評価するが，本論文ではこれらのパラメータが結果に対して与える影響を原理的に理解することが目的であるため，方向や時間帯に依存せず，一律の値を用いることとする。

表 3.1 数値試算に用いた各種パラメータ

(a) 各種機器

CGS の定格容量	0.7 kW
CGS の最低出力	0.2 kW
CGS 出力の熱電比	1.65
CGS の発電に対するガス消費係数	2.66(α_{CGS}), 0.11(β_{CGS})
蓄熱槽容量	3.25 kWh
補助熱源の熱効率	0.9

(b) 各種料金単価および運用に関する時間

買電電力単価	29.72 ¥/kWh
売電電力単価	13.0 ¥/kWh
ガス購入単価	5.4 ¥/kWh
計画時間間隔	10 分
調整力対応周期 (調整力発動継続時間)	10 分(5 分)

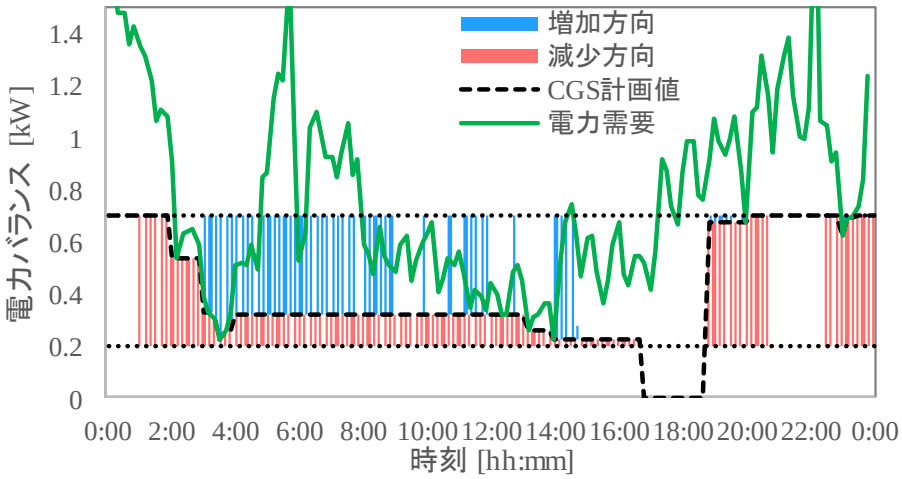
3.3.2 報奨金単価が与える影響

本項では冬季・夏季の平均的な日負荷曲線に近い需要が存在する日をそれぞれの季節の代表日とし，報奨金単価を変化した場合のある需要家 1 軒の運用について検証する。なお，ここでは調整量とコスト関係に着目しているため，調整力提供に関する制約((3.18) ,(3.19)式)は削除し，需要家は優先順位に依存せず，自身の経済性の観点から望むだけの調整力を提供できる状態を想定した。

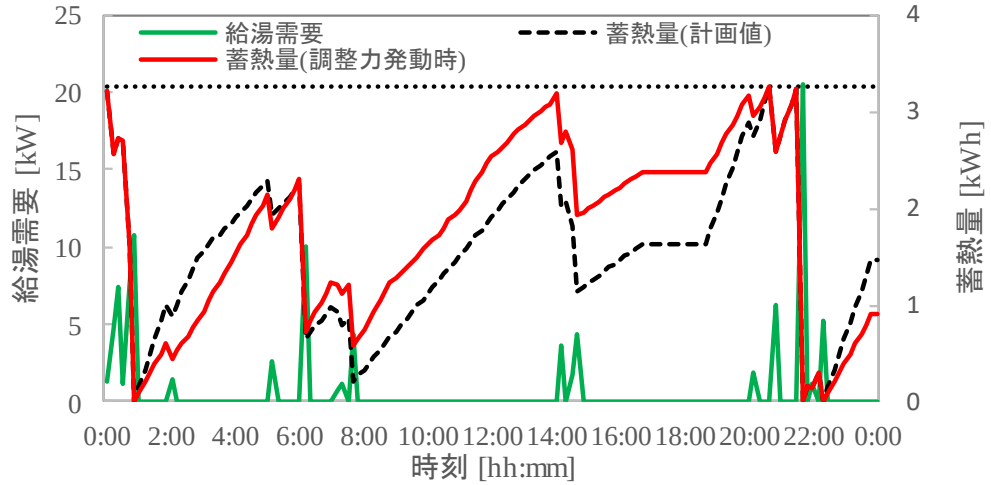
図 3.8 は各種需要が多い冬季の平均的な日負荷曲線に近い日(3 月 12 日)における，報奨金単価 0 ¥/kW・h の場合の需要家 1 軒のエネルギーバランスを示している。図 3.8(a)の緑線と黒破線はそれぞれ電力需要と CGS 発電出力を示しており，両者の差分は電力系統からの買電量(CGS 出力の方が多い場合は売電量)に相当する。また，図 3.8(a)の 0.2kW と 0.7kW に引いた細い破線の間が CGS 出力可能範囲であり，CGS 出力がこの下限を下回っている時間帯では CGS が停止していることを示している。一方，図 3.8(b)は熱(給湯)のバランスを示しており，緑線は給湯需要(第 1 軸)を，黒破線と細い黒破線はそれぞれと蓄熱槽内の蓄熱量の計画値(第 2 軸)とその上限(蓄熱槽容量)を示している。図 3.8(a),(b)から蓄熱量が上限値を超過しない範囲で CGS を最大限出力してお

り，特に 0:00～1:00 と 20:00～22:00 の大きな給湯需要がある時間帯で蓄熱槽内の蓄熱量が利用されていることがわかる。

図 3.8(a)の青と赤の実線は，提供(発動)した方向別の調整力の大きさを，図 3.8(b)の赤実線は，調整力を発動した際の蓄熱量の推移をそれぞれ示している。これより報奨金単価 0 ¥/kW・h であっても比較的多くの時間帯で調整力を提供しているが，0:00～1:00 や 20:00～22:00 では CGS が定格運転(0.7kW)にあるのにも関わらず，出力減少方向の調整力を提供できていない。これは，同時間帯の給湯需要への供給に蓄熱量を利用し，なおかつ CGS の排熱を最大限に利用せざるを得ず，したがって出力減少側の調整力提供するためには補助熱源を焚き増しする必要があり，コスト増加に繋がってしまうためである。



(a) 電力バランス

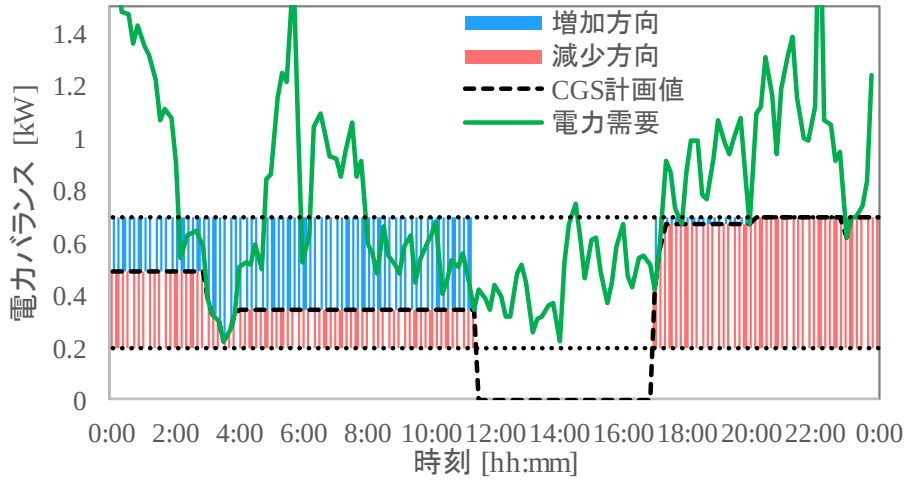


(b) 熱バランス

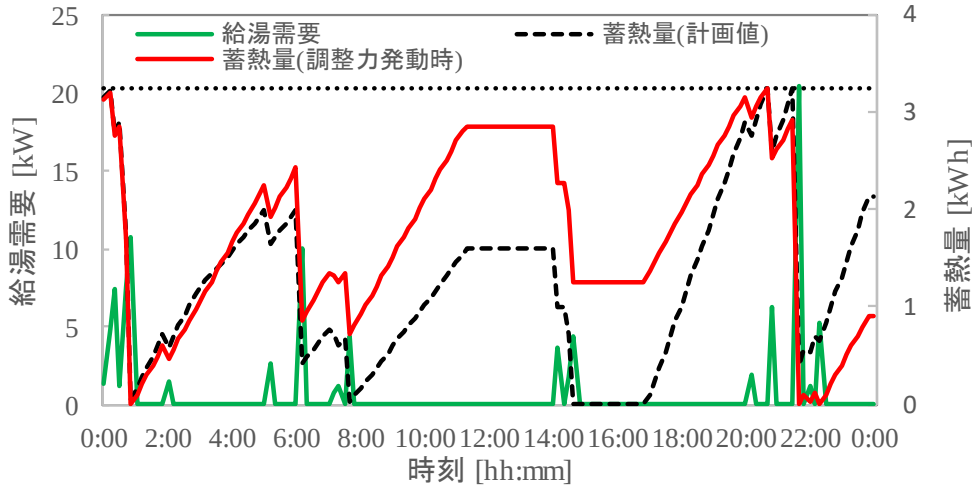
図 3.8 冬季の代表日の運用(0 ¥/kW・h)

これに対し，報奨金単価 30¥/kW・h とした場合の同じ需要家のエネルギーバランスを図 3.9 に示す。この場合，報奨金単価 0¥/kW・h の運用に比べて CGS の起動時間を短く，なおかつ出力を絞り運転している。その結果，CGS が起動している全時間帯で調整力を提供・発動しており，出力増加・出力両方向ともに調整量(各図(a)の赤・青線の面積に相当)が増加している様子が見て取れる

。



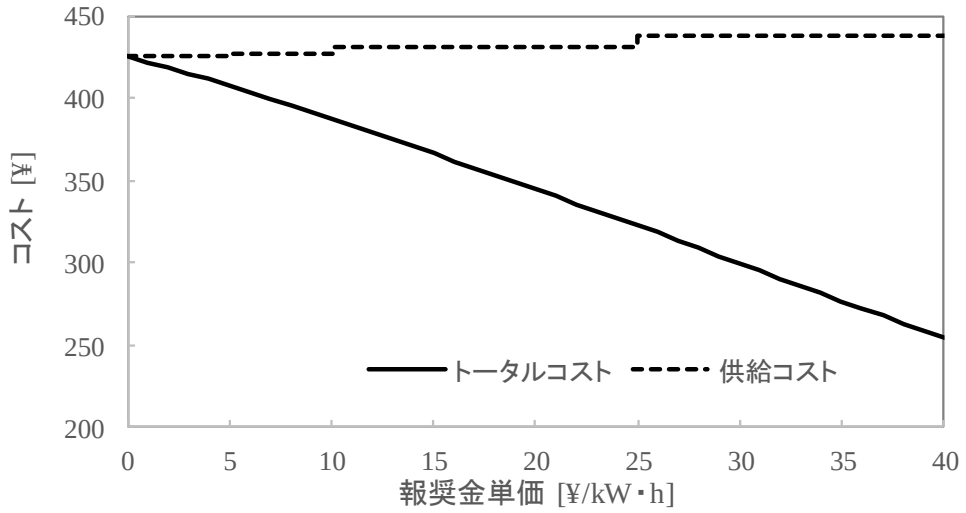
(a) 電力バランス



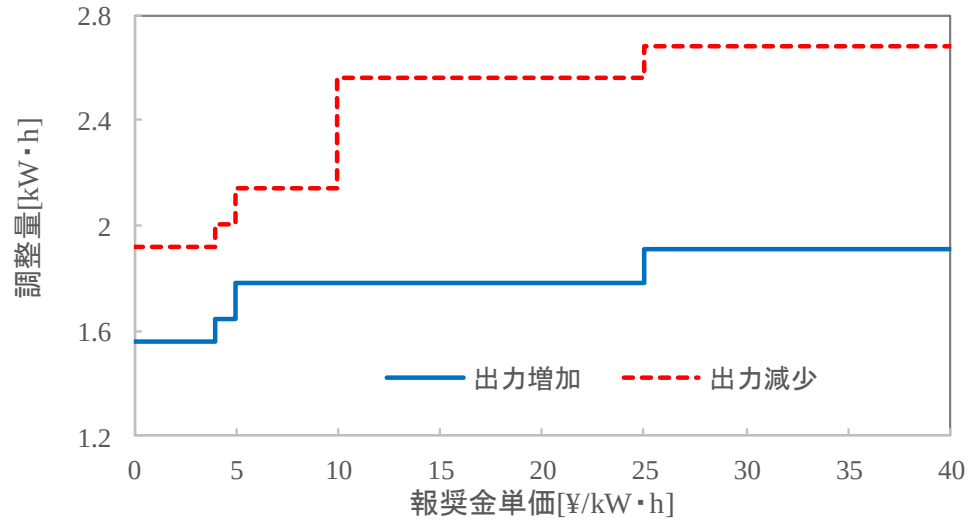
(b) 熱バランス

図 3.9 冬季の代表日の運用(30 ¥/kW・h)

図 3.10 は冬季の代表日のある需要家において，報奨金単価を変化させた際の各種コストの変化を示しており，図 3.10(a)は供給コストおよびトータルコスト，図 3.10(b)は方向別の調整量を示している。トータルコストは報奨金単価に応じて単調減少しているが，供給コストおよび調整量は報奨金単価の増加に伴い段階的に増加している。これは，需要家の運用は，需給調整への貢献(調整量増加)に伴う供給コストと報奨金の増分の比較によって決定される，ことを意味している。



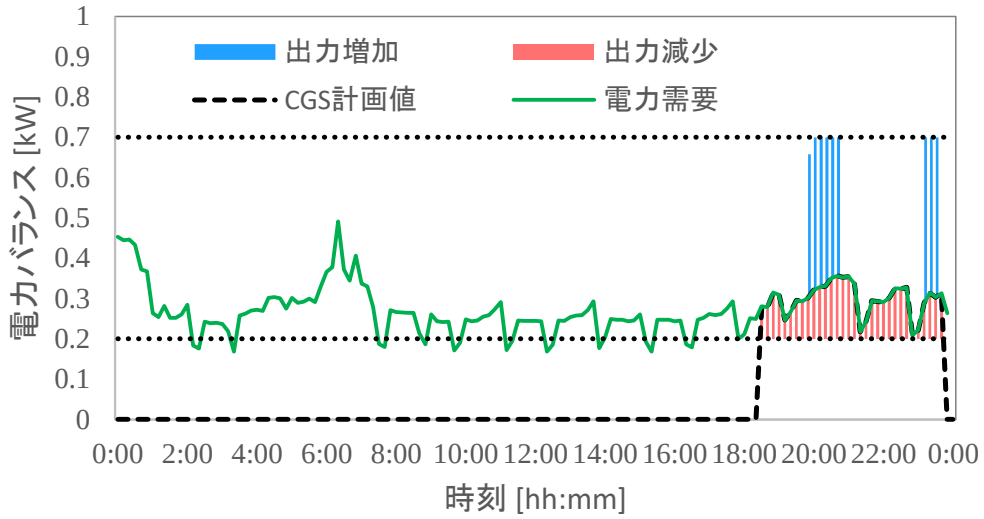
(a) 供給コストとトータルコスト



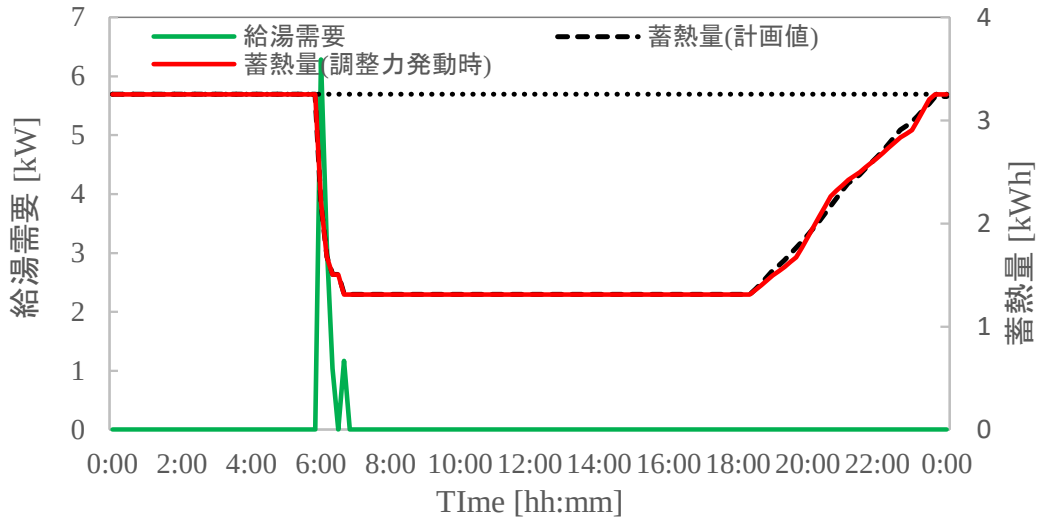
(b) 方向別の調整量

図 3.10 報奨金単価の変化によるコストと調整量(冬季の代表日)

一方，各種需要が少ない，夏季の平均的な日負荷曲線に近い日(8 月 12 日)のある需要家 1 軒において，報奨金単価 15,30 ¥/kW・h の試算結果を図 3.11，図 3.12 にそれぞれ示す。冬季の代表日と同様，報奨金単価の増加に伴い，CGS 出力を絞ることで出力増加方向の調整量が増加する。その一方で，給湯需要が少ないため CGS 出力は最低出力付近で推移しており，出力減少方向の調整代を確保できず，出力減少方向の調整量は減少する。



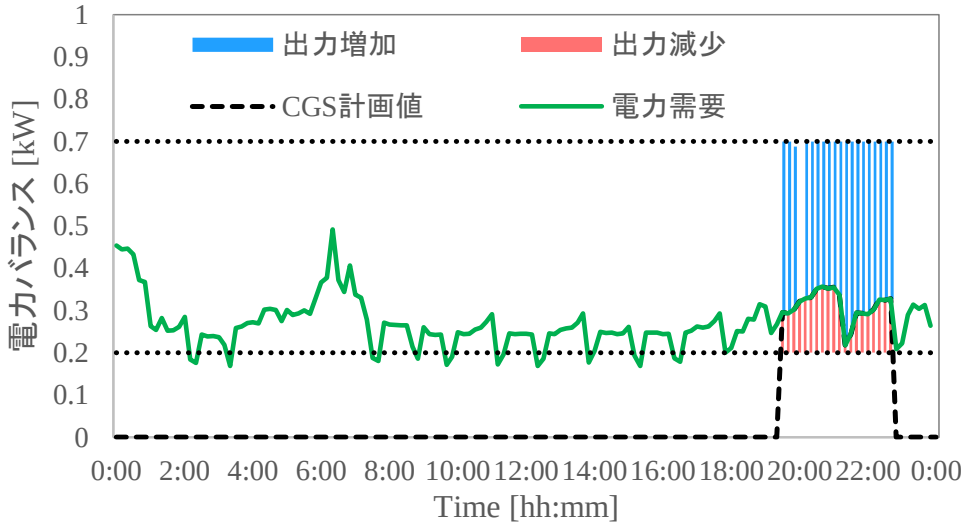
(a) 電力バランス



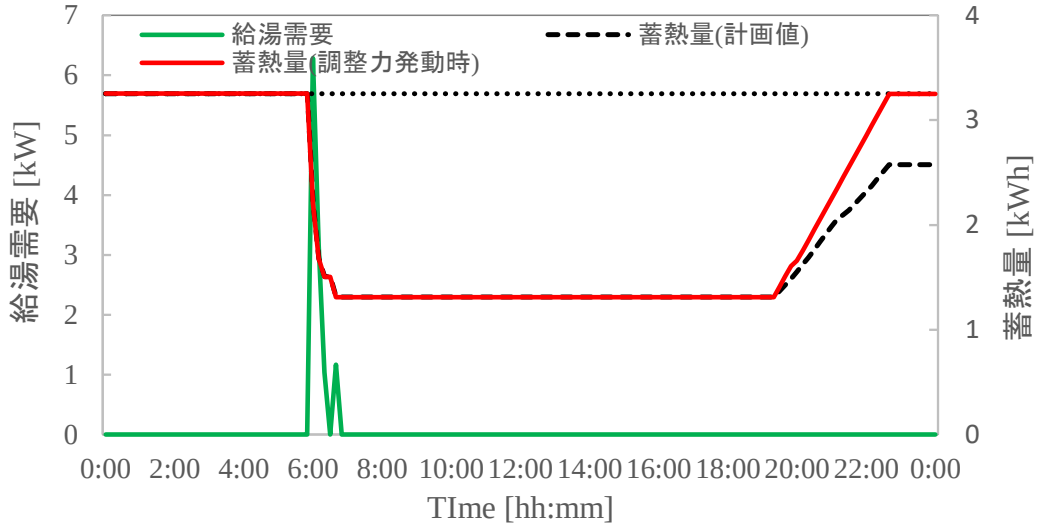
(b) 熱バランス

図 3.11 夏季の代表日の運用(15 ¥/kW・h)

図 3.13 は夏季の代表日のある需要家において，報奨金単価を変化させた際の各種コストの内訳を示している。減少方向の調整力は減少しているものの，両方向合わせた調整量の合計値は増加しており，それに伴う報奨金の増分はコストの増分よりも大きいため，報奨金単価の増加に伴い，図 3.11 から図 3.12 のように運用が変化していることが読み取れる。



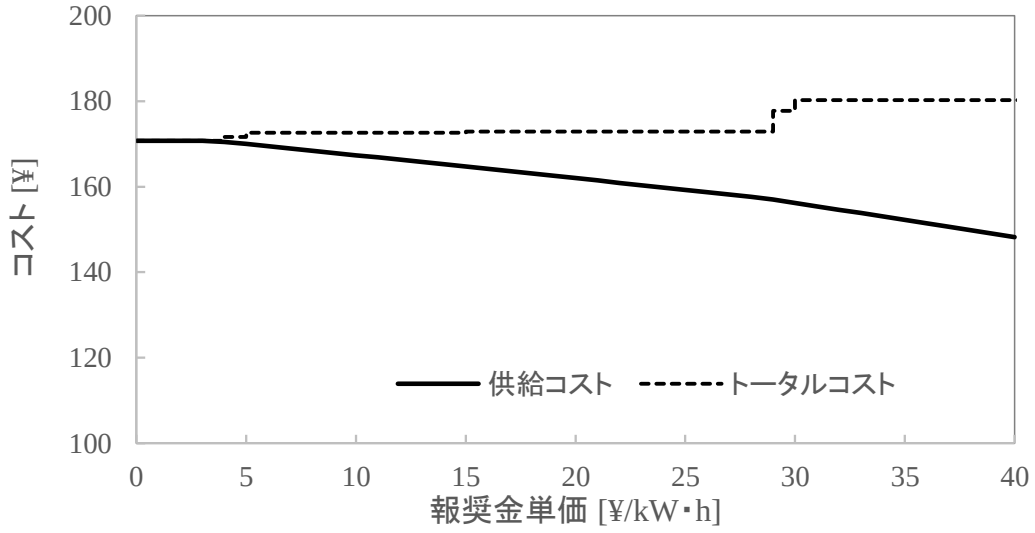
(a) 電力バランス



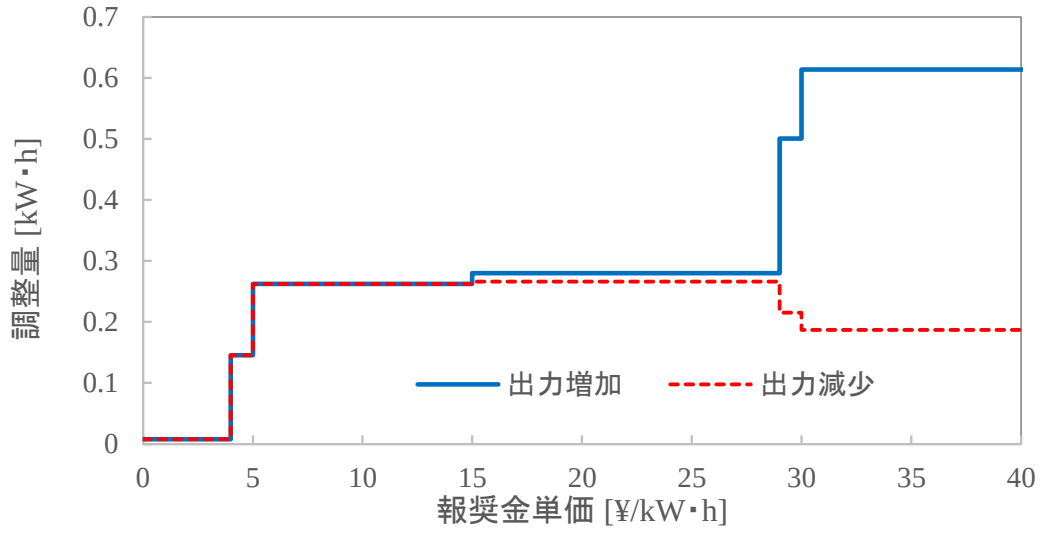
(b) 熱バランス

図 3.12 夏季の代表日の運用(30 ¥/kW・h)

これらの結果から CGS の運転が変化する報奨金単価は，調整力提供による対価と追加コストの比較によって決まることがいえる。報奨金単価が高くなると，エネルギー供給単価が安い CGS の出力を絞った運転となるため需要家のエネルギー供給コストは増加する一方で，報奨金も増加するためトータルコストは低減させられることがわかる。したがって，提案した運用手法を実施することで，需要家の経済性が向上することが明らかとなった。



(a) 供給コストとトータルコスト



(b) 方向別の調整量

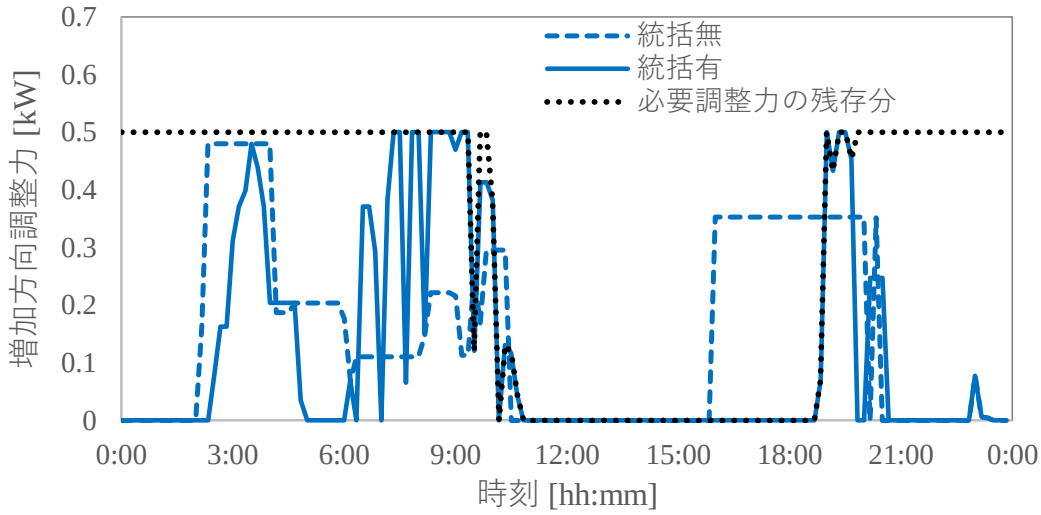
図 3.13 報奨金単価の変化によるコストと調整量(冬季の代表日)

3.3.3 調整力提供に関する制約が与える影響

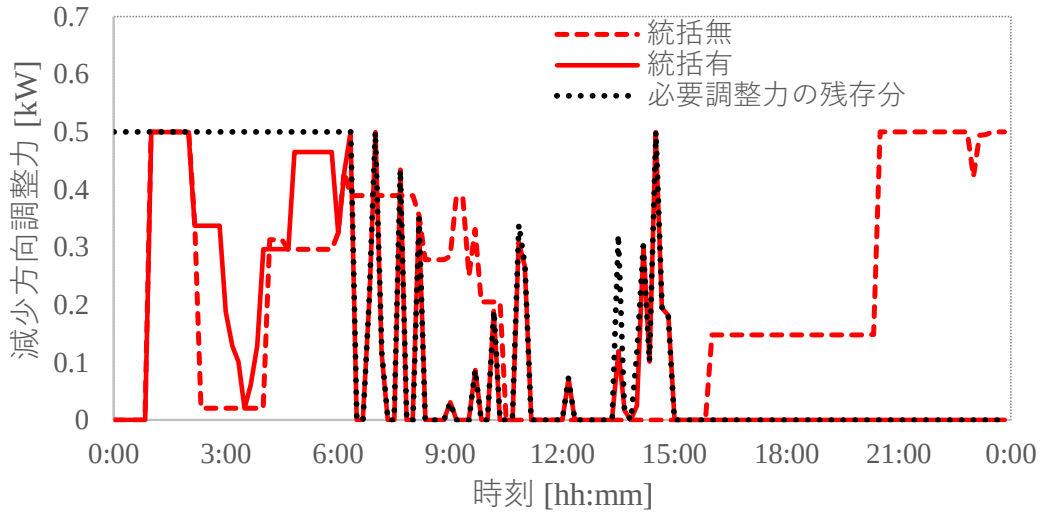
本項では，前項で扱ったように調整力提供に関する制約((3.18) ,(3.19)式)を削除し，経済性の観点から需要家が望むだけの調整力を提供できる状態を想定した「統括無ケース」と，本論文で提案した，アグリゲータが優先順位に従って CGS 群から調整力を調達する手順(((3.18) ,(3.19)式によ

る制約)を適用した「統括有ケース」の 2 ケースで調整量と調整提供時間を比較する。この 2 ケースを比較することで，提案した手順が必要調整力に対する調整力調達の効率性にもたらす効果を検証することができる。比較検証においては前節で取り扱った冬季の代表日において，必要調整力が両方向ともに 2.8kW，報奨金単価（$inc^{up}(t), inc^{dn}(t)$）も両方向共に 10¥/kW・h である状況を対象とする。

まず，アグリゲータが提示される必要調整力の残存分の大きさが需要家の CGS の運用に与える影響を明らかにするために，前項で取り扱った 3 月 12 日において，比較的運用の変化が大きい，優先順位が 17 である需要家 1 軒について着目する。図 3.14 は，アグリゲータがこの需要家に提示する必要調整力の残存分，および各ケースで需要家が提供した調整力をそれぞれ示している。なお表記の都合上，必要調整力の残存分は，上限が 1 台の CGS の出力可能範囲(0.5kW)以下になるようにキャップをかけた値を示している。



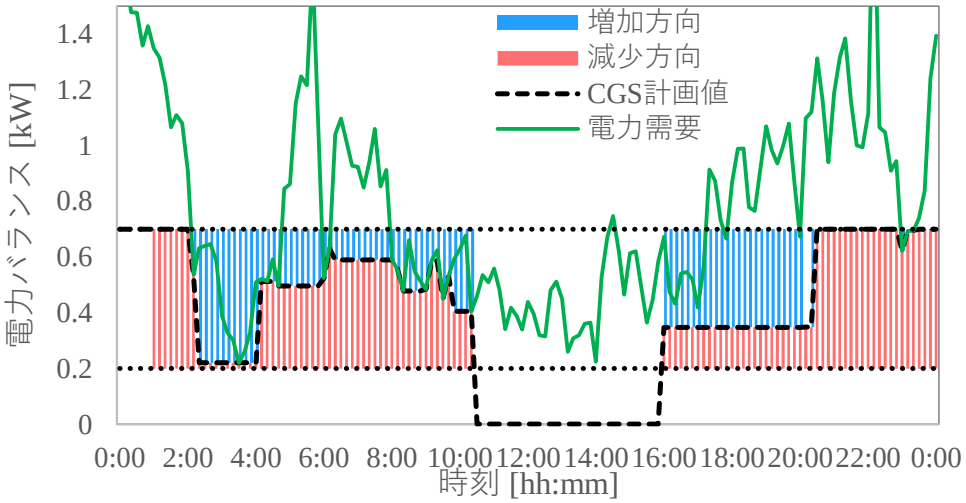
(a) 出力増加方向



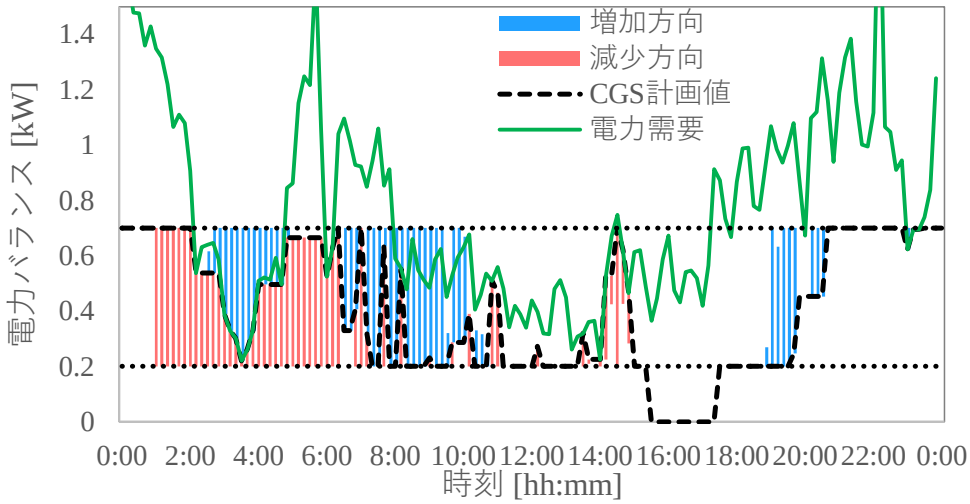
(b) 出力減少方向

図 3.14 需要家に提示される必要調整力の残存分と各ケースで提供した調整力

図 3.15 は両ケースにおける電力バランスを示しているが，図 3.15(b)の統括有ケースでは需要家は必要調整力の残存分に対して調整力を提供するように，図 3.15 (a)の統括無ケースとは異なる運用を行っていることが読み取れる。このような運用の変化は，必要調整力の残存分が少ない，すなわち優先順位が低い需要家ほど大きく，優先順位の高い需要家は，アグリゲータから提示される必要調整力の残存分が多いため，調整力提供に関する制約((3.18) ,(3.19)式)の影響が小さく，統括無ケースのように，比較的自由に望むだけの調整力を提供するような運用を行う。



(a) 統括無ケース

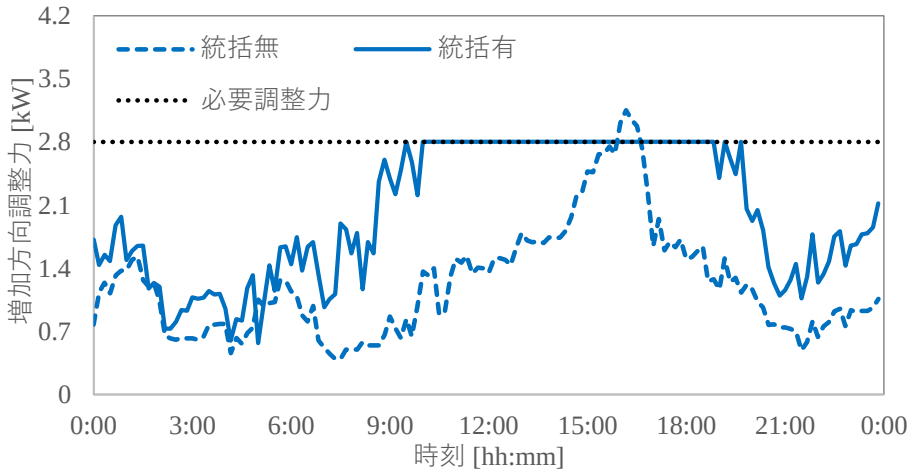


(b) 統括有ケース

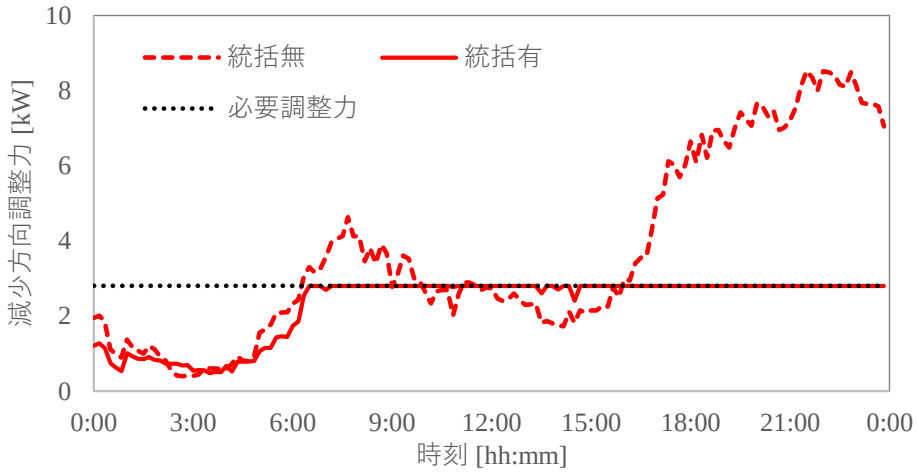
図 3.15 調整力に関する制約(統括)の有無による需要家の運用の変化

次に，需要家全 20 軒分の試算結果について述べる。図 3.16 は両ケースにおいて，アグリゲータが CGS 群から調達した各方向の調整力の大きさを示している。複数の需要家では図 3.14 に示す需要家単位よりも調整力提供における時間変化・間欠性は少なく，比較的安定して調整力を調達している様子が見て取れる。これは，需要家特有の給湯需要によって生じる，調整力の時間と量のばらつき具合が平準化され，相対的に小さくなっていることを意味している。

また図 3.16(a),(b)中の必要調整力(2.8kW)を黒破線でそれぞれ記載しているが，どちらの方向においても，統括有ケースの方が調達した調整力が必要調整力に達する時間が長いことが読み取れる。これは，統括有ケースでは図 3.15(b)に示すように，優先順位の低い需要家は必要調整力に達していない，残存分の多い時間や方向の調整力を提供していることが要因であり，その結果として調達した調整力が平滑化されていることが言える。



(a) 出力増加方向



(b) 出力減少方向

図 3.16 各ケースにおける全需要家から調達した調整力

表 3.2 は需要家から調達した調整量(各方向の合計値)と調整力提供時間，需要家のトータルコストの内訳を示している。ただし，調整力提供時間以外は 1 需要家あたりの平均値を示している。統括無ケースは提供する調整量は多いものの，調整力提供時間は短い。一方，統括有ケースでは，需要家群が提供する調整力は必要調整力の残存分以下に限定されるため調整量は少なくなっているが，調整力提供時間は長くなる。

試算に要した時間は両ケース共に，全需要家 20 軒分で約 30 秒であった。また需要家 20 軒分の最適化をまとめて一括して解くことを試みたが，使用した計算環境(Core i7 3.6GHz，メモリ:16GB)ではメモリ不足により実行不可であった。そのため参考値となるが，住宅 15 軒をまとめた 1 回の最適化で運用計画を実施すると約 420 秒と大きく増加する一方で，解の精度を意味する調整量および調整力調達時間の増加は数%程度であった。したがって，計算量の観点から，運用計画の実施対象は，需要家を統括するアグリゲータ単位よりも需要家単位の方が実用的であるだろう。

以上の結果から，アグリゲータが提案した手順を適用することで，CGS 群の調整力提供に柔軟性を持たせることが可能になり，必要調整力に対してより多くの調整力が調達できることが明らかになった。次項では，本項の統括有ケースで年間試算を行った結果について報告する。

表 3.2 需要家から調達した調整力およびコストの内訳

	調整力		エネルギー供給コスト[円]	報奨金[円]	トータルコスト[円]
	量 [kW・h]	時間 [min]			
統括無	3.68	190	376.25	1.84	374.41
統括有	2.59	530	375.92	1.29	374.63

3.3.4　年間試算による CGS 群の調整力評価

本項では，報奨金単価と必要調整力をパラメータとした際の年間試算結果について検証する。まず，試算に用いた需要データについて，1 日積算の電力需要・給湯需要量[kWh]の年間推移(20 軒平均値)を図 3.17 に示す。図 3.17 により，電力需要よりも給湯需要の方が季節性による 1 日需要量(kWh)の変化が大きいものの，年間を通して一定の需要が存在していることがわかる。

図 3.18 は前項の統括無ケースにおける，CGS 群から調達した調整量の年間推移を示している(調整量を需要家 1 軒分に正規化している)。調達した調整量は図 3.17 に示す給湯需要と相関が高く，給湯需要が多い冬季では多く，夏季では少ない。このような季節性が認められるものの，年間を通して CGS 群から一定以上の調整力を提供できることが期待される。一方，調整量を方向間で比較すると，出力減少方向が出力増加方向よりも多い。これは，出力減少方向は運用上の制約下で最大限 CGS を運転している状態で調整力を提供することができるが，出力増加方向は運用計画時点で CGS 出力を絞らなければならず，買電量・補助熱源からの熱供給量の増加により，エネルギー供給コスト増に繋がってしまうことが主な要因である。

図 3.19 は，提案した手順を適用した統括有ケースにおいて，必要調整力を横軸，代表的な報奨金単価をパラメータとして変化させた年間試算結果を示しており，図 3.19(a)は 1 日あたりの調整量，図 3.19(b)は調整力提供時間，図 3.19(c)は需要家 1 軒当たりのトータルコストをそれぞ

れ示している。必要調整力の増加に伴って調整量は増加するが，その必要調整力そのものが大きくなるため調整力提供時間は短くなる。その一方で，必要調整力が小さくなると，調整力提供に関する制約による影響が顕著に表れるため，需要家群が提供する調整量は減少するものの，調整力提供時間が長くなる。報奨金単価 0¥/kW・h から 10,30¥/kW・h と高くすると，調整力提供による追加コストよりも対価が大きくなり，各需要家が調整力をできるだけ多く提供するように運用するようになる。その結果，調整量および調整力提供時間は増加し，またトータルコストは低減させられていることが読み取れる。

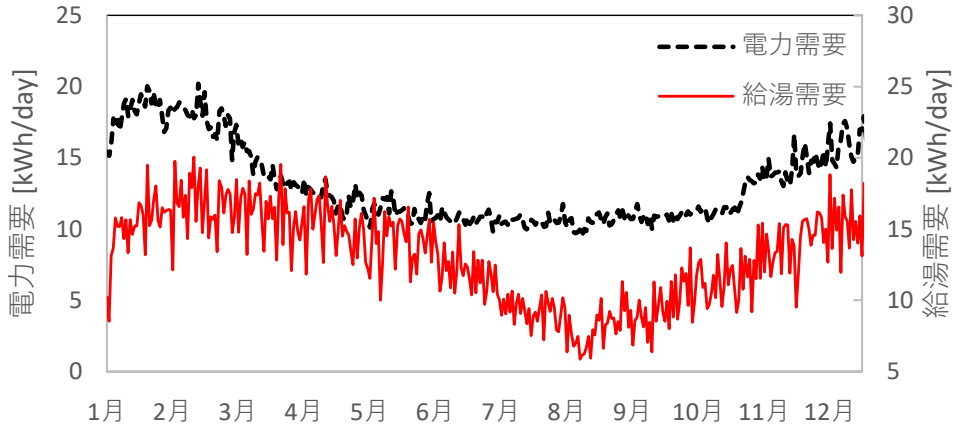


図 3.17 1 日積算の電力需要・給湯需要量[kWh]の年間推移

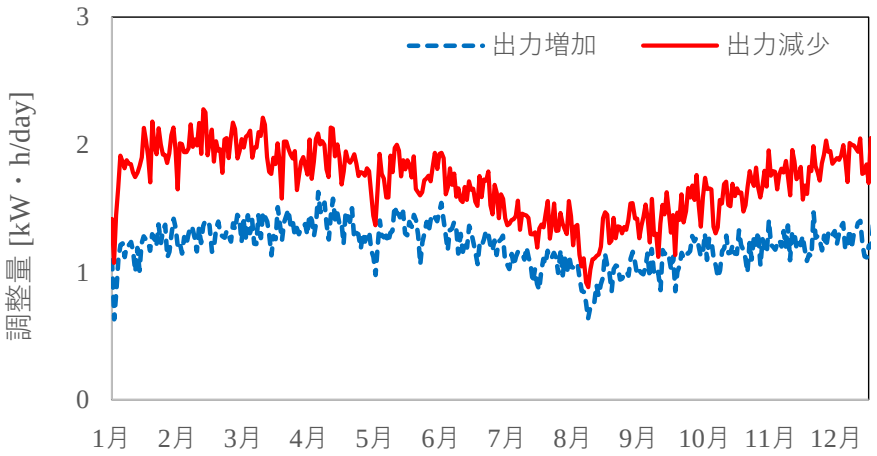
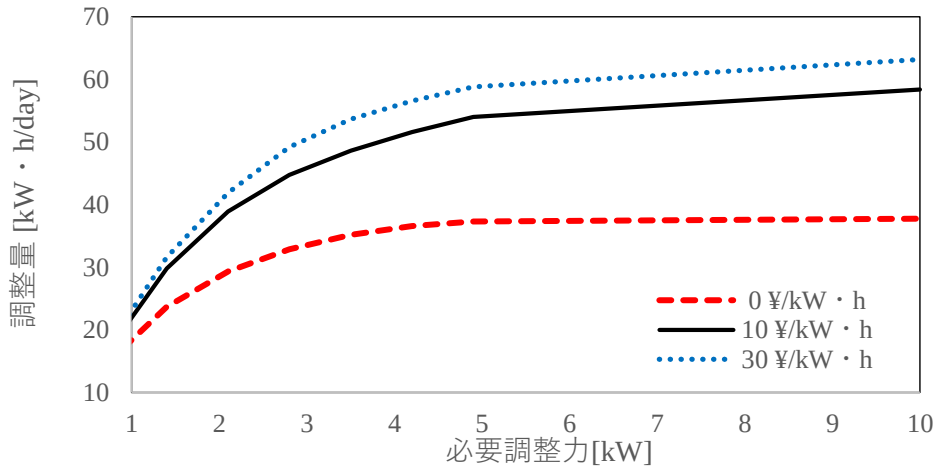


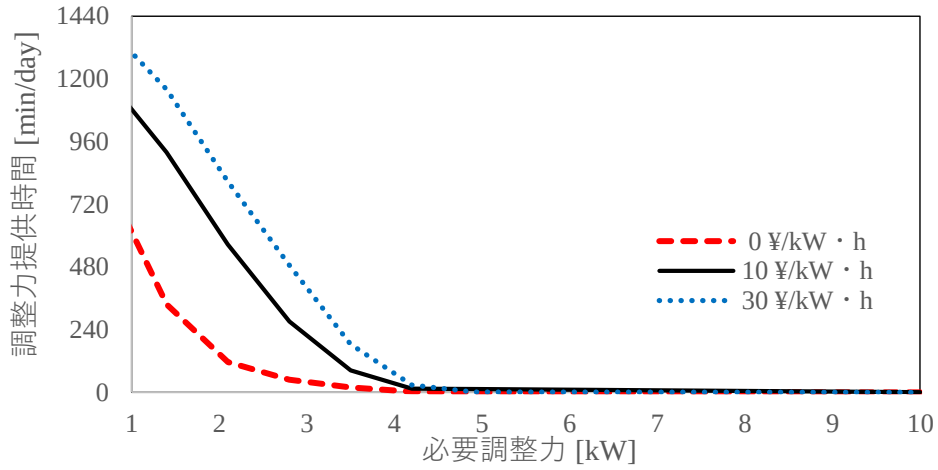
図 3.18 方向別の調達した調整量の年間推移

以上の結果から，必要調整力の大きさにより，調整量と調整力提供時間はトレードオフの関係で変化することがわかる。必要調整力を大きくすると調整量は増加し，需要家リソース群を所有している需要家の利益は増加傾向にあるが，必要調整力に満たないことによるリスクも大きくなる。このことから必要調整力の適切な大きさ(これはアグリゲータと系統運用者の契約，あるいは需給調整市場などによって決定される)は，需給運用状況，アグリゲータが統括する需要家数，需要家から提供される調整力の性質(量や時間，調整方向)，調達した調整力が必要調整力に満たなかった場合の補填(ペナルティ)などの様々な要因によって変化することが考えられる。また，報奨金単価を時間帯別や調整方向別に設定することで，特定の時間帯や調整方向に集中させ調整力を調達することも可能であるだろう。

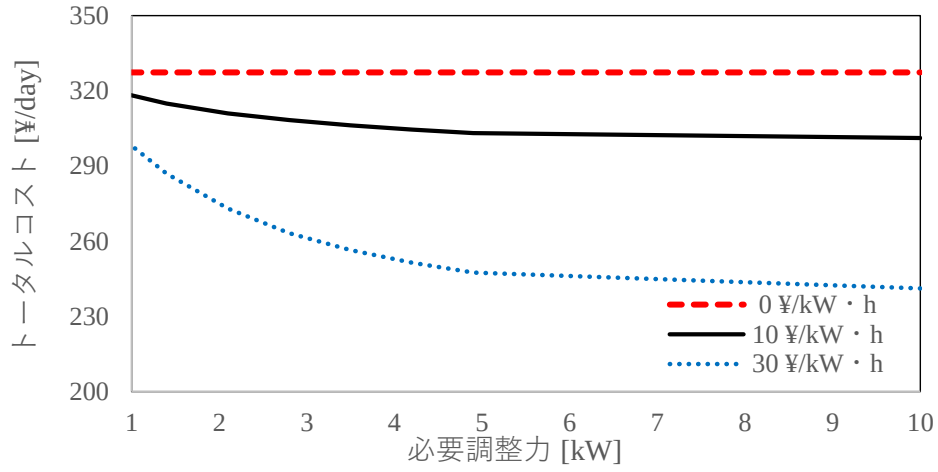
したがって，本論文で提案した手順のように CGS 群を統括することで，アグリゲータが必要調整力をより多く調達するようになるなど，調整力調達に柔軟性を持たせることが確認できた。また，リソースを所有する需要家も調整力提供に伴い生じる追加的コストと対価の比較を考慮することで，経済的に最適な運用が可能となることが確認できた。



(a)調整量



(b)調整力提供時間



(c)トータルコスト

図 3.19 統括有ケースの年間試算結果

第4章　需給調整に貢献するコージェネレーションシステムの最適構成

第 3 章では，調整力提供に貢献する CGS の運用手法の提案を行い，提案手法実施時における CGS の運用と数値試算による調整力評価を行った。その一方で，CGS のエネルギー効率改善効果や，調整力としての付加価値は，CGS の利用形態や容量，エネルギー供給の対象である需要家の電力・熱需要のプロファイル等に大きく依存する。

例えば，文献[81]では業務建物を，文献[82]では一般住宅を対象として，CGS の各種容量を変化させた際の省エネルギー性，経済性に関する評価を行っている。さらに，文献[83]では，集合一般住宅を想定し，戸別に CGS を 1 台ずつ設置するシステムと，集合住宅全体に 1 台の CGS を設置するシステムの 2 つの CGS の構成において，省エネルギー性の観点から最適な CGS 容量をそれぞれ算出，比較を行っている。

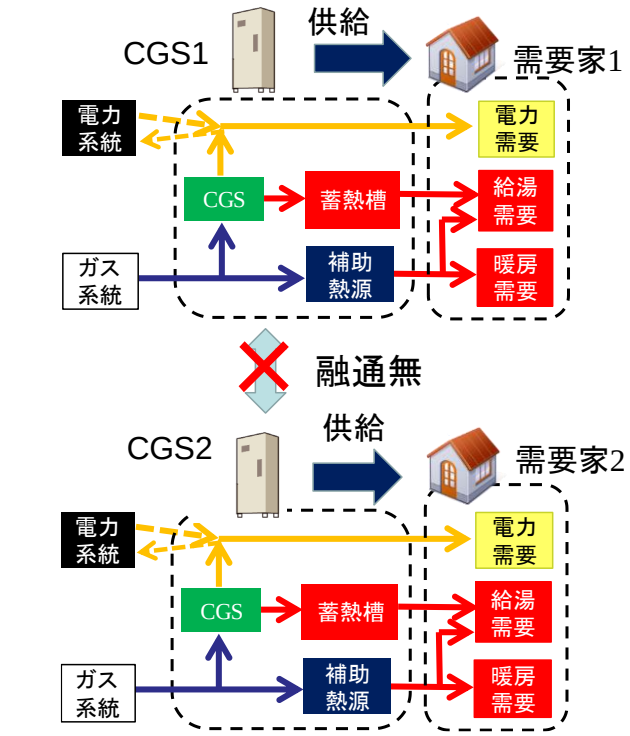
そこで本論文では，これらの関連研究[81] [82] [83]に考慮されていない CGS の調整力としてのリソース活用を念頭におき，CGS の容量や利用形態(住戸単位で CGS を利用するのか，それとも複数住戸で CGS 容量を共用するのか)の違いが，経済性(トータルコスト)やリソース活用能力(利用可能な調整力の大きさ)に与える影響を，数値試算により明らかにする。

4.1　本論文で取り扱う CGS 構成

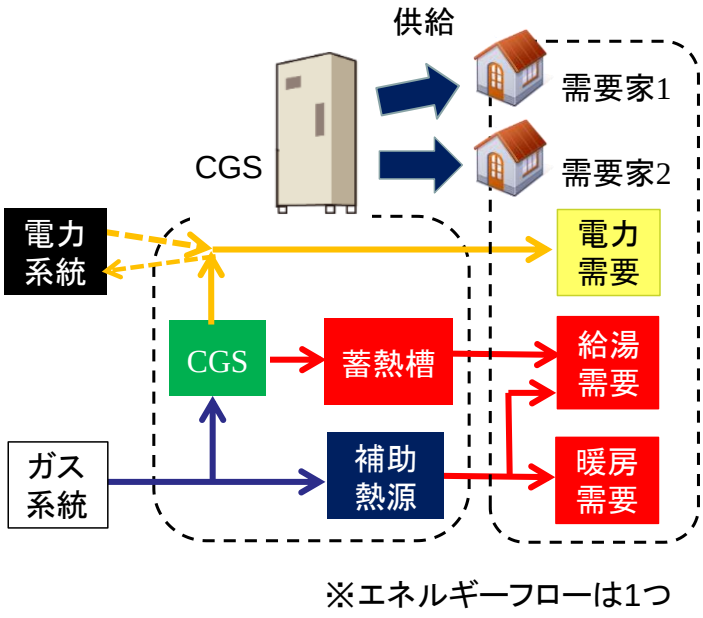
本論文では，複数の需要家(住戸)で構成される集合一般住宅に，CGS を設置されている状況を想定する。なお，基本的なエネルギー供給モデルについては，前章の需要家モデル(図 3.3)と同一である。

本論文では，各住戸に対して 1 台の CGS を個別に設置・利用する構成(以後，「個別 CGS」)と集合住宅全体に対して 1 台の共有の CGS を設置・利用する構成(以後，「共用 CGS」)の 2 つの構成を取り扱う。図 4.1 は住戸数が 2 つのときの本論文で取り扱う 2 つの構成例を示している。図 4.1(a)の個別 CGS では集合一般住宅内の住戸数(2 戸)だけ存在し，それぞれのエネルギーフローが独立して存在することになる。そのため，個別 CGS 間で電力や熱などのエネルギーの融通については考慮していない。

一方，共用 CGS では，図 4.1(b)に示すように集合一般住宅全体に対して 1 つのエネルギーフローだけが存在することになる。



(a) 個別 CGS



(b) 共用 CGS

図 4.1 各構成のエネルギーフロー(住宅 2 戸の例)

4.2 各 CGS の運用計画

本章では，前章で取り扱った調整力提供の枠組みのもと，最適な CGS 構成に関する検討を行う。具体的には，各 CGS はアグリゲータから提示される報奨金単価を元に，CGS 自身が対象とする需要家へのエネルギー供給手段の計画を立てる。具体的な方法としては，前章で提案した需要家の運用計画を決める最適化問題((3.1)～(3.19)式)における添え字 i が示す対象を，需要家から CGS に置き換えるだけであり，3 章の最適化問題と本質的な変化はないため省略する。この置き換えは．例えば $e_{dem,i}(t)$ は，前章では「i 番目の需要家の電力需要」を示していたが，ここでは「i 番目の CGS の供給対象である需要家の電力需要」に置き換わる，ということである。

ただし, 本章では調整力を最大限提供した際の CGS を設置している需要家の影響について評価するため，調整力提供に関する制約は設定しない(前章の「統括無」ケースに相当)。

4.3 実測データに基づく CGS 構成の比較

本論文では，住戸 10 戸で構成される 1 つの集合一般住宅を想定し，数値試算により各構成(個別 CGS と共用 CGS)の運用に関するコストと調整力を評価する。本節の構成は，以下の通りである。はじめに，本節における試算条件について述べる。次に 2 つの構成の CGS の運用について示す。最後に 2 つの構成において，CGS 容量および報奨金単価が変化した際のトータルコストおよび調整力に与える影響について述べる。

4.3.1 試算条件

使用するデータは，2008 年に北海道で実測された年間データ(データ間隔 10 分，1 戸あたりの最大電力需要 3.6kW，最大給湯需要 56.0kW)である。図 4.2 に 10 戸分の 1 日電力需要・給湯需要量[kWh]の年間推移を示す。

個別 CGS では住戸 1 戸に対して 1 台の CGS，すなわち合計 10 台の CGS が前節で述べた手法により運用する(i の範囲は 1～10)。一方，共用 CGS では，1 台の CGS が 10 戸分の住戸を対象として運用するが，このとき CGS は各住戸(需要消費地)までの熱配管の距離が長くなり，個別 CGS に比べて熱供給時の損失が大きくなると考えられる。そこで本試算では，「熱供給時には各時間帯の給湯負荷に対して一定割合の損失が発生する」と仮定し，全住戸の給湯需要を一定割合(0%あるいは 90%)増加させることで，損失を模擬して評価を実施した。

表 4.1 は本試算で使用する各種パラメータを示している。表 4.1(a)CGS の各種機器の仕様を示しているが，その数値は CGS の単機定格容量を 1.0kW に規格化した値であり，CGS 発電電力に対するガス消費量の定数項(β_{CGS})と蓄熱量の上限($h_{TS}^{kWh,\max}$)は，CGS 定格容量($e_{CGS}^{\max}$)を乗じた値を用いている。

その他の条件は表 2 にまとめるが，ガス購入単価については北海道ガス（株）が定めている家庭用コージェネレーション契約料金[79](使用量区分 D・平成 27 年 9 月 1 日～)を，電力購入単価については北海道電力(株)が定めている家庭用電力の一般電気料金[80](従量電灯 B(40A)・平成 26 年 11 月 1 日～)を採用した。

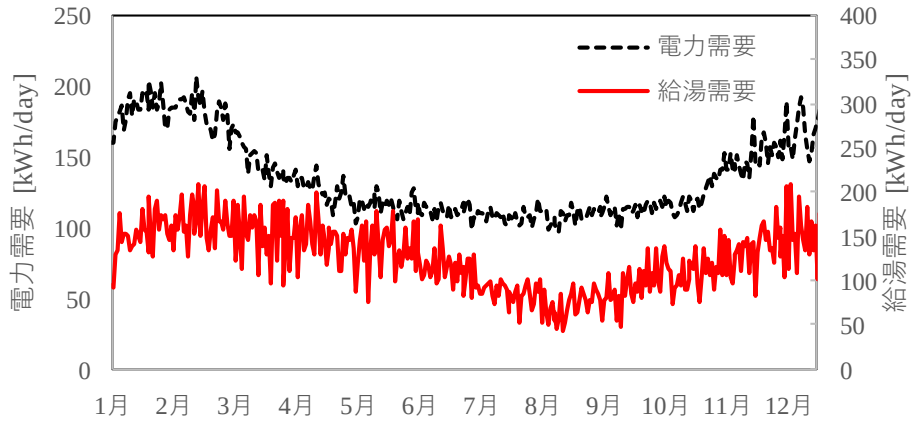


図 4.2 1 日電力需要量と給湯需要量の年間推移(10 戸合算値)

表 4.1 数値試算に用いた各種パラメータ

(a) 各種機器

CGS 運転範囲	28.6～100%(定格)
熱電比	1.58
発電出力におけるガス消費係数	$2.66(\alpha_{CGS}), 0.16(\beta_{CGS} / e_{CGS}^{\max})$
蓄熱槽容量	4.64　$(h_{TS}^{kWh,\max} / e_{CGS}^{\max})$
補助熱源の熱効率	0.9

(b) 各種料金単価および運用に関する時間

買電電力単価	29.72 ¥/kWh
売電電力単価	13.0 ¥/kWh
ガス購入単価	5.4 ¥/kWh
計画時間間隔	10 分
調整力対応周期(調整力発動継続時間)	10 分(5 分)

各 CGS が策定する運用計画は翌日 1 日分を対象期間として，前日の最終時間(24:00)に策定するものとした。また，各計画の開始時刻(0:00)の蓄熱槽内の蓄熱量($H_{TS}^{kWh}(0)$)は蓄熱可能量上限の 50%とし，最終時間(24:00)時点における蓄熱量は調整力発動に伴う蓄熱量の変化分を含め，50%以上になるようにする。

本試算において評価・比較する指標は，CGS が提供した調整量[kW・h]および供給コストから調整力提供に対する報奨金を差し引いたトータルコストである。

以後の試算では，CGS 容量および報奨金単価（$inc^{up}(t), inc^{dn}(t)$）をパラメータとして変化させながら評価をするが，本論文では報奨金単価が結果に対して与える影響を原理的に理解することが目的であるため，方向や時間帯に依存せず，一律の値を用いることとした。

4.3.2　代表日における各構成の CGS の運用

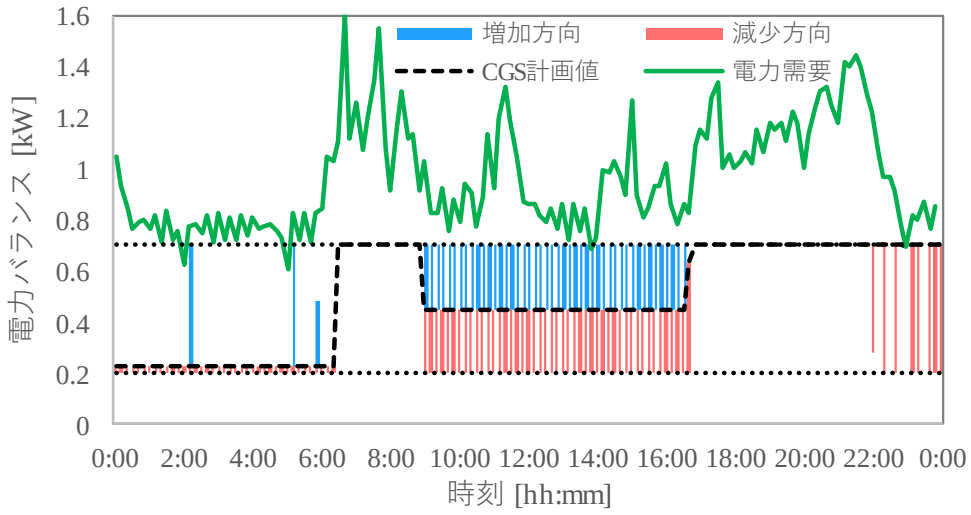
本節ではまず，報奨金無(単価 0¥/kW・h)の条件下において，冬季の代表日(2 月 5 日)における，個別 CGS と共用 CGS の運用を比較する。想定する CGS 容量は，個別 CGS では，一般住宅向けに市販されている定格容量 0.7kW（蓄熱量上限 3.25kWh）が各住戸に設置されているものとした。共用 CGS では，個別 CGS のケースと総定格容量が等しくなるように，定格容量 7.0kW の CGS を 1 台設置する状況を想定した。

図 4.3 は，各種需要が多い代表日(2 月 5 日)における，ある個別 CGS の 1 台の運用を示している。基本的な見方は 3 章で掲載した図と同一であるが，図 4.3(a)の緑実線と黒破線はそれぞれ電力需要と CGS 発電出力を示しており，両者の差分は電力系統からの買電量(CGS 出力の方が多い場合は売電量)に相当する。また，図 4.3(a)の縦軸値 0.2kW と 0.7kW の場所に引いた細い破線の間が CGS 出力可能範囲に相当する。一方，図 4.3(b)の 2 つの破線はそれぞれ給湯需要(第 1 軸)と蓄熱槽内の蓄熱量の計画値(第 2 軸)を示している。図 4.3(a),(b)から蓄熱量が上限値(細い黒破線)を超過しない範囲で最大限 CGS を出力しており，特に 6:00～7:00 と 20:00～21:00 の時間帯で，大きな給湯需要に熱供給するために蓄熱槽内の蓄熱量が利用されていることがわかる。

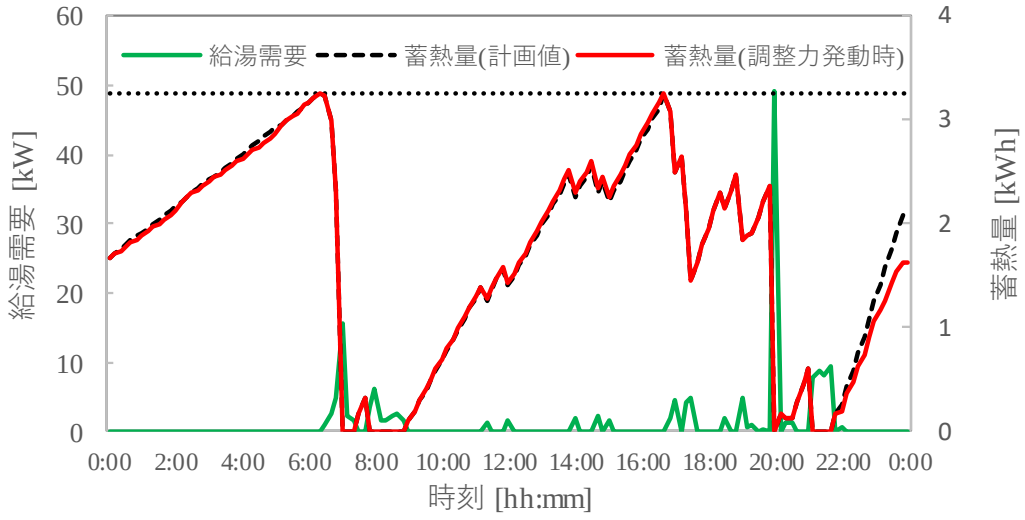
図 4.3(a)の実線は提供した方向別の調整力の大きさを，図 4.3(b)の実線は，調整力を発動した際の蓄熱量の推移をそれぞれ示している。第 3 章でも述べたように，報奨金単価 0 ¥/kW・h であっても，特に昼間の時間帯で調整力を提供しているが，朝方や夜間では CGS の出力調整代があるにも関わらず，調整力を提供できていない。これは，蓄熱量の過不足が要因である。たとえば朝方の時間帯で出力増加方向の調整力を発動すると，蓄熱量増加に伴い蓄熱量上限を超過してしまう。一方，夜方では，出力減少方向の調整力を発動すると，蓄熱量が不足し補助熱源を焚き増しする必要があり，コスト増加に繋がってしまう。

図 4.4 は同じ代表日における共有 CGS(損失 0%)の運用を示す。共有 CGS では全住戸を対象にしているため，個別 CGS よりも需要の大きさは増加しているが，各需要の時間変動は相対的に小さくなっており，これより蓄熱量は蓄熱可能量の上下限に抵触しにくくなっていることが読み取れる。その結果，大きな給湯需要が発生する 20:00 付近を除き多くの時間帯で調整力が提供できていることがわかる。

図 4.5 は損失 90%とした共有 CGS の運用を示しているが，見掛け上の給湯需要増加に伴い蓄熱量の排熱利用も増加している。その結果として， CGS 出力を全体的に増加させて CGS の排熱量を増加しようとしている様子がうかがえる。

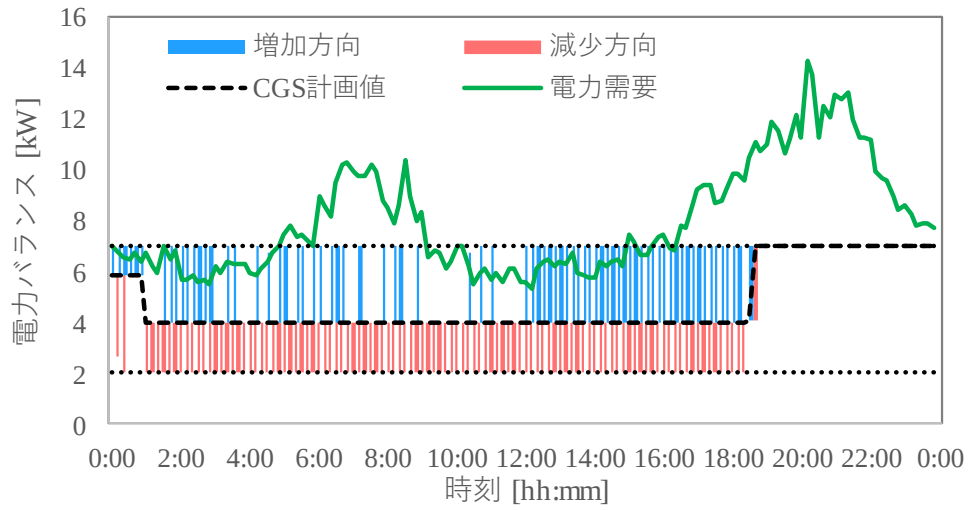


(a) 電力バランス

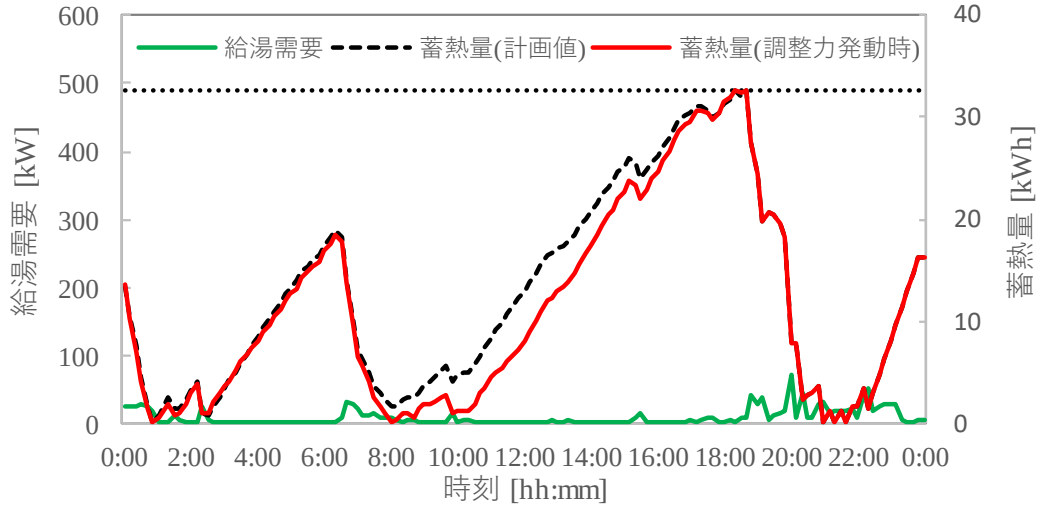


(b) 熱バランス

図 4.3 代表日における個別 CGS(1 台)の運用

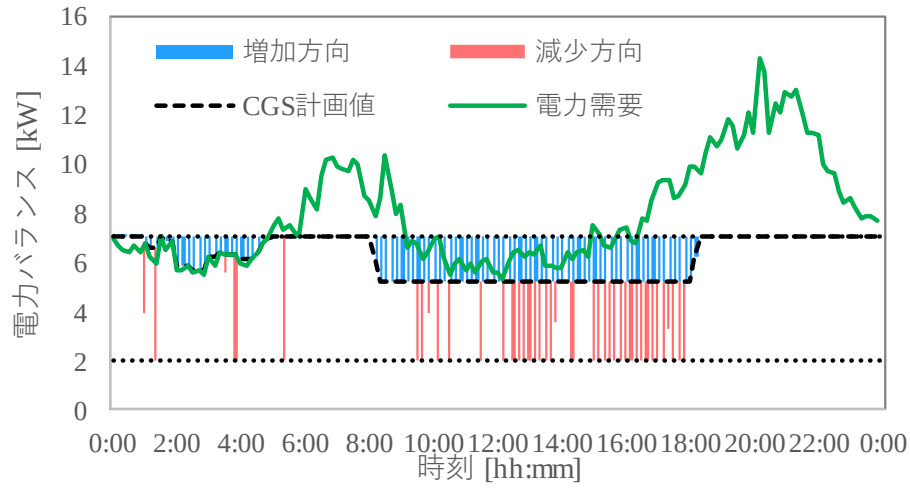


(a) 電力バランス

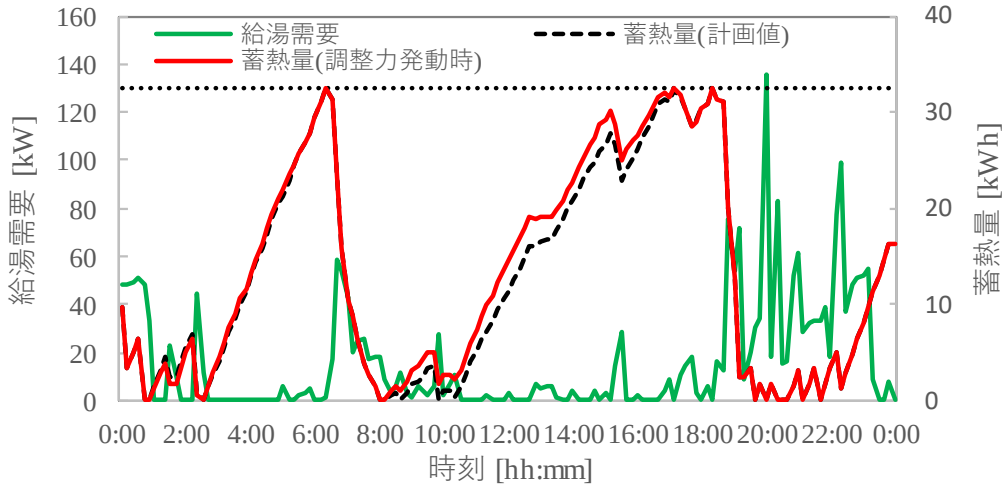


(b) 熱バランス

図 4.4 代表日における共有 CGS の運用（損失 0%）



(a) 電力バランス



(b) 熱バランス

図 4.5 代表日における共有 CGS の運用（損失 90%）

表 4.2 は図 4.3～図 4.5 で示した結果である，個別 CGS(CGS10 台の合計値)および共有 CGS(損失 0,90%)の場合のエネルギー供給の内訳と供給コスト，および提供した調整量[kW・h]を示す。まず損失無(0%)の構成間で比較すると，共有 CGS は個別 CGS よりも CGS 発電(排熱)量が大きくなっており，その他の供給源(買電量・補助熱源)からの供給量が減少している。その結果，共有 CGS の供給コストは個別 CGS よりも約 16%削減される。また，損失がある場合では，給湯需要増加により補助熱源の出力は大きく増加するが，CGS 発電量の増加により買電量は減少するため，供給コストそのものは大幅な増加には繋がっていない。その結果，熱供給時の損失を伴ったとしても共用 CGS の方が，調整力提供，コスト面共に個別 CGS よりも優位である結果となった。

表 4.2 代表日における供給コストと調整量の内訳

方式(損失)	エネルギー供給量 [kWh]				供給コスト	調整量
	買電量	売電量	補助熱源	CGS発電量	[¥]	[kW・h]
個別CGS (0%)	110.9	0.2	103.5	80.4	5213.2	13.7
共用CGS(0%)	77.9	0.0	46.3	113.2	4375.6	36.0
共用CGS(90%)	44.7	0.8	170.5	147.2	4622.3	20.8

4.3.3　年間試算結果

ここでは，両構成において CGS 容量および報奨金単価を変化させた場合に，年間のコストや提供した調整量にどのような影響を与えるかを比較，検証する。

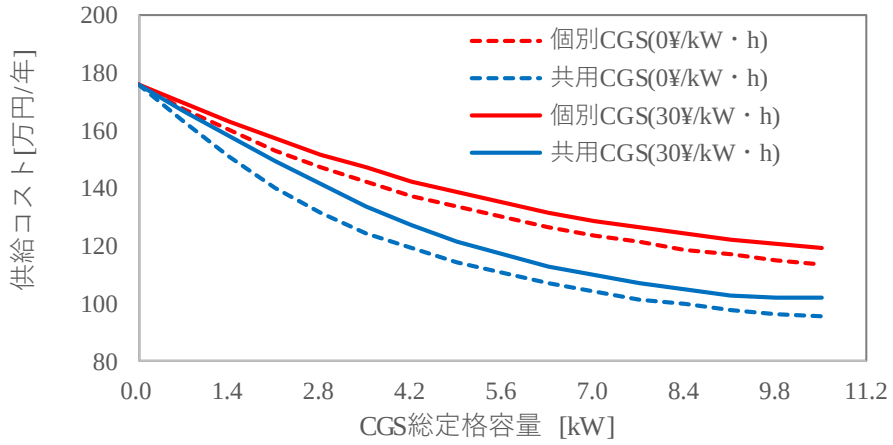
まず図 4.6 は，損失無(0%)において，報奨金単価 0，30 ¥/kW・h とした場合の両構成の比較結果を示しており，図 4.6(a)は年間供給コスト，図 4.6(b)は提供した 1 日平均の調整量[kW・h/day]を示している。図 4.6(a),(b)の横軸は CGS の総定格容量としているが，これは前項で行った比較のように，構成の違いによらず，CGS の機器的な容量が等しくなるようにしている。なお，前節の結果は同図の CGS 総定格容量 7.0 kW に相当する。図 4.6 からどの結果においても，CGS の容量増加に伴い供給コストは低減し，また調整量は増加していることが読み取れる。また構成間で比較すると，共有 CGS の方が供給コストは少なく，調整量は多い。しかしどちらの構成においても，報奨金単価を 0 から 30¥/kW･h に設定すると，調整量に加え供給コストも増加する。これは，報奨金無 0¥/kW・h では，供給コストを最小とするために CGS を最大限出力させるような運用を行うのに対して，高い報奨金単価 30¥/kW･h では，CGS の出力を絞り提供する調整量を増加させるような運用を行った方がトータルコストは最小となるからである。

図 4.7 は個別 CGS および共用 CGS(0,90%)の年間のトータルコスト削減額を示しており，図 4.7(a)は報奨金単価 0¥/kW・h，図 10(b)は 30¥/kW・h の結果を示している。ここで，年間トータルコスト削減額は，CGS を未設置の年間供給コストを基準としており，そのコストから調整力提供による報奨金を含めたトータルコストを差し引いた，CGS 運用によって得られる経済効果額に相当する。図 4.7 を見ると，CGS 総定格容量および報奨金単価に関わらず，共用 CGS の方が個別 CGS よりも年間トータルコスト削減効果が大きいことがわかる。

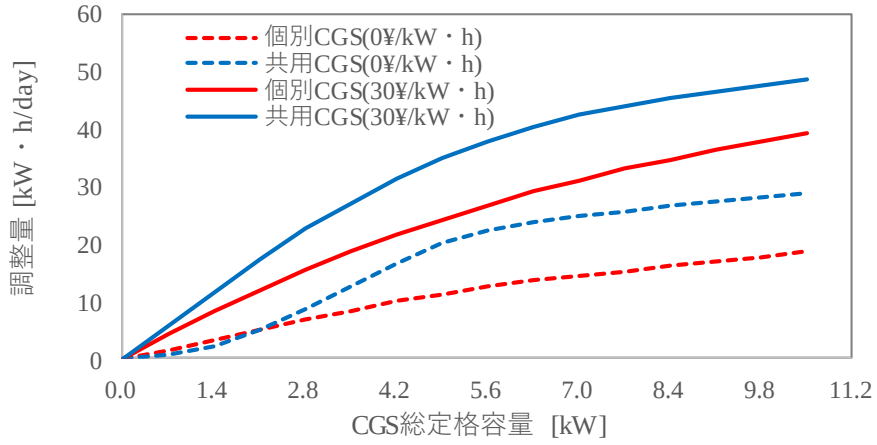
また，共用 CGS において損失を有する場合は，CGS 容量が小さいと見かけの給湯需要の増加に伴う補助熱源の出力増加によりトータルコスト削減効果は薄れ，コスト増を引き起こす(トータ

ルコスト削減額が負となる)。しかしながら，CGS 容量が増加するにつれて CGS 発電(排熱)量が増加し，損失無い場合とのコスト削減効果の差は小さくなっていく。CGS 容量がさらに増加して 7.0kW 以上に大きくなると，熱供給損失による給湯需要の増分よりも CGS 発電量の増加による買電量の減少分の影響が大きく，トータルコスト削減効果は損失無い場合のそれよりも高くなっている。

以上のことから，熱供給損失を考慮しても共用 CGS の方が，供給コストおよび提供可能な調整量の両者の観点で優位である。また，調整力提供による報奨金を含めると，共用 CGS のコスト削減効果は更に大きくなり，構成間の差は広がることが言える。

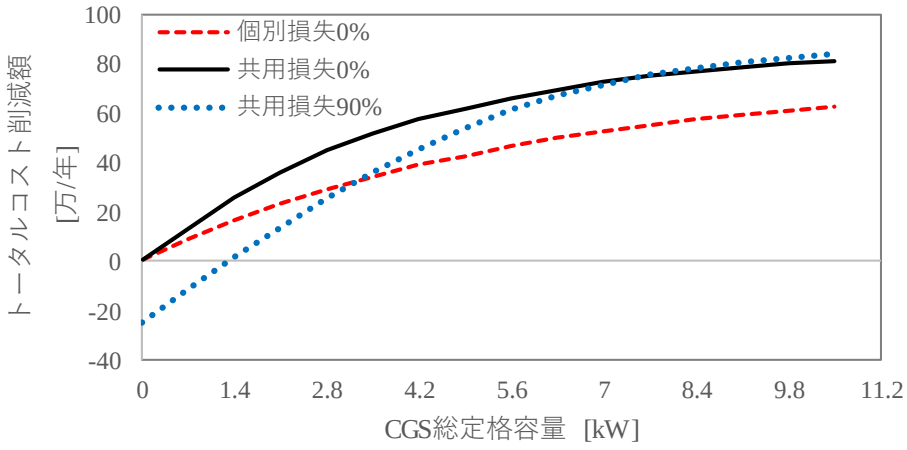


(a)　年間供給コスト

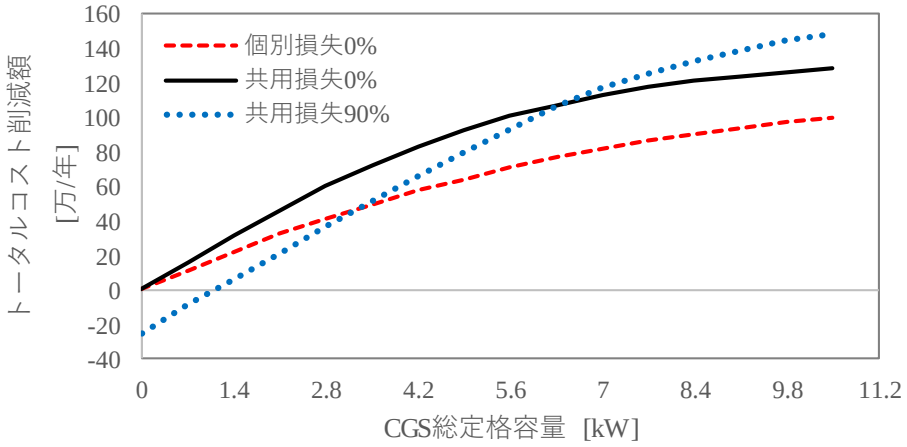


(b)　1 日平均調整量

図 4.6 両構成の比較結果（損失 0%）



(a)　報奨金単価 0¥/kW・h



(b)　報奨金単価 30¥/kW・h

図 4.7 トータルコスト削減額の比較結果

このことから，集合一般住宅などの複数需要家に対して一括して熱を供給する共用 CGS のようなシステムが，電力・熱需要プロファイルが CGS の運用に与える影響を緩和しやすいために，エネルギー効率効果が高く，提供可能な調整量も増加し，この両者の効果により経済的効果は大きく増加することが明らかになった。

ただし，実際に CGS の構成を検討・設計する際には，多数の需要家に熱供給を行う共用 CGS は，配管などの熱供給に関する設備構成にかかるコスト(イニシャルコスト)を含めて行うことが望ましいが，この調整力を提供することを前提とした運用を行うことで，運転の自由度(調整代の確保のしやすさ)が高い共用 CGS を設置する優位性が増すことが期待される。

第5章　結論

本章では，本論文で述べた内容のまとめおよび今後の展望について記載する。

5.1 本論文のまとめ

本論文では，太陽光発電や風力発電などの自然変動電源が電力系統に大量に導入されることを想定し，電力系統の電圧および電力需給バランスの安定化を図ることを目的とした需要家リソースの運用に関する検討を行った。

＜第1章＞

本論文の背景として，自然変動電源の現状や自然変動電源が電力系統の電圧および電力需給バランスに与える影響について述べ，その対策として検討されている需要家リソースを活用した研究動向について記述した。

＜第2章＞

自然変動電源の大量導入による電圧変動対策として，電気自動車用急速充電器(以下，充電器)に着目し，電圧制御に貢献する充電器の運用方法を提案し，その制御能力効果を評価した。提案した手法の特徴は以下である。

1．充電器の空き容量を用いているため，従来の利用(充電)の影響を与えない範囲で電圧制御できる。充電器による制御の能力は本来の利用によって左右されるが，その能力を推定することで本来必要な電圧制御機器の対策コストを削減することができる。
2．電圧制御対象が受電点の測定電圧に基づき自律分散的に動作することから，通信機能を必要としない。
3．充電器以外の需要家側負荷や電圧制御機器である無効電力補償装置にも適用できる。
4．過去の電圧情報等を用いて適切に制御パラメータを設計することで，従来の電圧制御機器との協調を図ることが可能である。

＜第3章＞

自然変動電源の大量導入による電力需給バランスの調整力不足対策として，多数の CGS 群を統括することで調整力を調達するアグリゲータを想定し，必要調整力に対してできるだけ多くの調整力を CGS 群から調達する手順を提案した。また，CGS を所有する需要家の視点から，調整力提供に関わる経済性を考慮した CGS の最適運用手法を提案した。さらに，実需要データを用いた数値試算により，CGS 群が提供する調整力を評価した。

提案した手法の特徴は以下である。

1．需要家は調整力提供(需要家リソース活用)に伴う増分コストとその対価(報奨金)を比較しながら運用を決定するため，需要家の経済性が向上する。
2．アグリゲータが各需要家の需要や機器特性などの諸条件の把握を必要とせず，調整力提供に関する情報(必要調整力・報奨金単価)に関して需要家とのやりとりを行う(プライバシーの問題に配慮されている)。
3．CGS 以外の他の需要家のリソースや出力調整用の電源，あるいはこれらの機器を組み合わせてアグリゲートした場合にも適用可能である。
4．アグリゲータは需要家から調整力を事前調達するため，需要家分の調整力を見込んだアグリゲータ自身の運用計画(市場への意思決定)などを戦略的に行うことができる。

＜第 4 章＞

集合一般住宅にエネルギー供給を行う CGS を想定し，個別 CGS と共用 CGS の 2 つの構成を対象として，これらの構成の違いが，提供可能な調整量および調整力提供による報奨金を含めたトータルコストに与える影響を数値試算により明らかにした。その結果を以下にまとめる。

1．共用 CGS の方が提供可能な調整量は多く，一方でトータルコストが少ないことから，CGS 導入による経済性の効果は大きいことが明らかになった。
2．調整力提供による報奨金単価が高くなると，経済的効果の構成間の差は広くなり，さらに共用 CGS の優位性が増すことがわかった。

5.2 今後の展望

今後の展望について，第 2 章「電圧制御に貢献する充電器の運用」と第 3,4 章「需給調整に貢献する CGS 群の運用」に分けて挙げる。

＜電圧制御に貢献する充電器の運用＞

1．本論文では，乱数により決定したある一通りの EV の到着時間・充電電力量の時系列シナリオを用いたが，より多数のシナリオを想定し，提案手法の効果をより統計的に評価することが今後の展望として考えられる。また，本論文では制御原理の開発を目的としたため，充電器の応答遅れや，充電器間の制御間隔・タイミングの相違の影響などは考慮していないが，実機にはこれらの影響を考慮して各種ゲインや不感帯等を調整することが挙げられる。

2．本論文では，夕方から夜間の時間帯における，住宅などに設置されている普通充電器の充電によって生じる電圧低下問題に対する有効性については対象としておらず，充電器の充電利用が多い場合には空き容量不足により電圧低下抑制に必要な無効電力を補償できなくなる，といった課題が挙げられる。このような充電器の充電利用による電圧変動(低下)の影響と，充電器のその充電利用に伴う空き容量による電圧制御能力の関係の解明について検討することが挙げられる。

3．今回の電圧制御手法では通信機能を有していない充電器を想定していたが，最近は通信機能を活用した充電器も導入されつつある。ただし，仮に需要家の全負荷(あるいは全需要家)が，スマートメータに代表される通信機能等を用いることで，データを完全に把握できれば最適化することは比較的容易であるが，コスト面や情報のデータ速度等を考慮すると，一部の機器または需要家だけが通信可能となることが予期される。よって，上記のような限られた通信間隔や情報量で最大限の電圧制御に貢献するような運用・制御手法も検討することが必要になる。

＜需給調整に貢献する CGS 群の運用＞

1．今回の試算では，必要調整力および報奨金単価は時間によらず一律としているが，これらの値は，現在創設に向けた検討がなされている調整力市場の価格やアグリゲータが管轄する他の需要家リソースや調整用電源の稼働状況を加味し，系統運用にかかる全体のコスト(あるいは利益)が最小になるように決定する必要がある。その適切な値の決定方法およびその影響について明らかにすることが挙げられる。

2．提案した運用手法における調整力提供は，系統内の需給運用の状況や調整力市場に参加する他の事業者の行動によって，計画時に提供した調整力が運用時に発動しない(不要となる)可能性がある。さらに需要家からの調整力提供は各種需要プロファイルに依存するため，運用計画と異なるような機器利用や需要の変化により，運用時に調整力を提供できなくなる可能性が大きい。したがって，調整力発動に関する不確実性を考慮した調整力提供の能力評価や需要家の運用手法を検討することが挙げられる。

3．第 4 章にて，CGS の運用に着目した比較を行ったが，CGS の設備・構成の設計を調整力提供およびその提供時の対価(報奨金)を含めて行うことで，CGS を所有する需要家は更なる経済的効果を得られることが可能になることが予想される。したがって，その調整力提供による対価を含めた際の CGS の最適設計について検討することが挙げられる。

4．また， CGS を含む需要家リソース群を統括するアグリゲータが調整力市場へ参入するにあたり，需要家の利用により各リソースが十分に機能しない場合にはペナルティが大きくなることから，図 3.2 に示した想定する調整力提供の枠組みのように，調整用電源を併せることが望ましいだろう。このとき，調整力市場に加えてエネルギー市場(卸市場)も同時に考慮する必要があり，アグリゲータが両市場に対してどのように意思決定を行っていくか，調整用電源およびリソース群からどのように調整力を事前に調達し，系統運用者からの調整力発動指令を配分するか，など，実際の市場への技術要件等を満足しつつ効率的かつ合理的な枠組み・運用方法に関する検討が必要である。

参考文献

[1] 資源エネルギー庁「総合エネルギー統計」(2016)

[2] 一般社団法人 海外電力調査会：「主要国の一次エネルギー消費区政と自給率」
(URL：https://www.jepic.or.jp/data/g01.html アクセス日：2018-11)

[3] 経済産業省 資源エネルギー庁：「なっとく!再生可能エネルギー」
(URL : http://www.enecho.meti.go.jp/category/saving_and_new/saiene/renewable/outline/ アクセス日：2018-11)

[4] 再生可能エネルギー発電設備　電子申請サイト　情報公開用ウェブサイト：「C 表 買取電力量および買取金額の推移」
(URL：https://www.fit-portal.go.jp/PublicInfoSummary , アクセス日:2018-11)

[5] 独立財政法人 新エネルギー・産業技術総合開発機構(NEDO)：「NEDO 再生可能エネルギー技術白書」第 2 版

[6] 経済産業省：「平成 29 年度エネルギーに関する年次報告(エネルギー白書 2018)」
(URL：http://www.enecho.meti.go.jp/about/whitepaper/2018pdf/　アクセス日：2018-11)

[7] 環境省：「平成 26 年度 2050 年再生可能エネルギー等分散型エネルギー普及可能性検証検討委託業務報告書」第 2 章 (2014)

[8] 経済産業省：「平成 28 年度エネルギーに関する年次報告(エネルギー白書 2017)」
(URL：http://www.enecho.meti.go.jp/about/whitepaper/2017pdf/　アクセス日：2018-11)

[9] 「太陽光・風力発電と系統連系技術」出版社：オーム社

[10] 林 泰弘，松木純也，鈴木良治，武藤英司：「分散型電源が連系された配電系統における最適送出し電圧の決定手法」，電学論 B，Vol.125，No.9，pp.846-854 (2005)

[11] 松田勝弘，和田勝，渡辺雅浩，高橋玲児：「分散型電源導入に対応した配電系統の柱上変圧器タップ最適設定手法の検討」，電学論 B，Vol.127，No.1，pp.137-144 (2007)

[12] 高橋尚之，林　泰弘：「不感帯制御 SVC による配電系統の動的電圧制御手法」，電学論 B，Vol.133，No.4，pp.396-403(2013)

[13] 米澤征司，高山聡志，石亀篤司，伊藤隆治，阿部勝也，南　雅弘：「移動平均制御を用いた SVC による配電系統の自動電圧制御」，電学論 B，Vol.136，No.3，pp.302-310 (2016)

[14] 八太啓行：「PV 出力に応じた無効電力制御による SVC 容量低減効果と配電線路損失への影響評価」，電学論 B，Vol.135，No.2，pp.106-110 (2015)

[15] 田中俊輔，鈴木宏和：「分散形電源の自律分散制御による電圧補償制御方式の検討」，電学論 B, Vol.129, No.7, pp. 869-879 (2009-7)

[16] 大城将人，千住智信，與那篤史，浦崎直光，舟橋俊久：「無効電力出力分担を考慮した配電系統の電圧制御法」，電学論 B，Vol.130, No.11, pp.792-980 (2010)

[17] Frédéric Olivier, Petros Aristidou, Damien Ernst: “Active Management of Low-Voltage Networks for

Mitigating Overvoltages Due to Photovoltaic Units", IEEE Trans, Smart Grid, Vol.7,No.2 (2016)

[18] Vito Calderaro, Vincenzo Galdi, Lamberti, Antonio Piccolo: "A Smart Strategy for Voltage Control Ancillary Service in Distribution Networks", IEEE Trams, Power Systems, Vol.30,No.1 (2015)

[19] 辻隆男, 佐藤典幸, 橋口卓平, 合田忠弘, 丹下誠視, 野村俊夫 : "将来型配電系統の電圧分布制御方式における経済制度の検討", 電学論 B, Vol.129, No.6, pp.745-754 (2009)

[20] 古林薫, 小出明, 辻隆男, 大山力 :「スマートメータの通信機能を考慮した配電系統の無効電力プライシング方式」, 電学論 B, Vol.136, No.4, pp.382-389 (2016)

[21] 渡辺雅浩, 高橋玲児, 松田勝弘, 瀬戸寿之 :「自端計測情報の相関関係を利用した複数台 SVR の協調制御手法の検討」, 電学論 B, Vol.135, No.10, pp.598-604 (2015)

[22] 川崎章司, 林　泰弘, 松木純也, 山口益弘 :「LRT との制御分担を考慮した SVC の協調型電圧制御法および SVC の定格容量と制御パラメータの決定手法」, 電学論 B, Vol.130, N0.11, pp.963-971 (2010)

[23] 川崎章司, 金本憲明, 田岡久雄, 松木純也, 林　泰弘 :「太陽光発電システム群の力率制御と LRT による協調型電圧制御法」, 電学論 B, Vol.132, N0.4, pp.309-316(2012)

[24] 餘利野直人, 三木崇裕, 大和右季, 造賀 芳文, 佐々木博司 :「SVC と SVR の協調のための時間スケール分割による電圧制御方式」, 電学論 B, Vol.124, N0.7, pp.913-919(2004)

[25] 川崎章司, 黒川尚大, 田岡久雄, 中嶋祐也 :「STATCOM の容量低減化を考慮した系統電圧制御機器の協調制御手法」, 電学論 B, Vol.134, N0.5, pp.378-385(2014)

[26] 大久保仁 :「電力システム工学」出版社 : オーム社

[27] 経済産業省 :「需給調整市場の創設に向けた論点」
(URL:http://www.meti.go.jp/committee/sougouenergy/denryoku_gas/denryoku_gas_kihon/seido_kento/pdf/007_03_00.pdf　アクセス日:2018-11)

[28] 西崎康, 入江寛, 横山明彦, 多田泰之:「風力発電連系系統の周波数制御のための風車ピッチ角制御とその蓄電池容量削減効果」, 電学論 B, Vol.129, No.1, pp.50-56 (2009)

[29] 株式会社三菱総合研究所「蓄電池を活用した新たなエネルギー産業に関する調査」
(URL: http://www.meti.go.jp/meti_lib/report/2016fy/000300.pdf)

[30] 小野貴, 荒井純一 :「電力系統に大量の風力発電が導入された場合の電力貯蔵装置の不感帯型周波数制御」, 電学論 B, Vol. 132, No. 8 pp.709-717 (2012)

[31] 名古屋洋之, 駒見慎太郎, 荻本和彦:「太陽光発電が大量導入された電力系統における蓄電池を用いた負荷周波数制御の一方式」, 電学論 B, Vol.132, No.4, pp.325-333 (2012)

[32] 福島敏 :「電力系統における蓄電池利用・制御技術の最新動向」, 電学論 B, Vol.137, No.10, pp644-647 (2017)

[33] 経済産業省資源エネルギー庁省エネルギー・新エネルギー部 :「『次世代エネルギー・社会システム実証事業』選定結果について―」(2010)

[34] 燃料電池.net
(URL: https://xn--qevu4mf0e768b.net/　アクセス日 : 2018-11)

[35] 経済産業省 :「水素エネルギーは何がどのようにすごいのか？」
(URL: http://www.enecho.meti.go.jp/about/special/johoteikyo/suiso.html アクセス : 2018-11)

[36] Manijeh Alipour, Behnam Mohammadi-Ivatloo, and Kazem Zare, "Stochastic Scheduling of Renewable andCHP-Based Microgrids", IEEE Transactions on Industr-ial Informatics, VOL. 11, NO.

5, October 2015.

[37] 濱本篤志，原亮一，北裕幸：「HP/BG 併用熱供給システムによる風力発電出力変動抑制の検討」，電気学会論文誌 B 分冊，Vol.137，No.6，pp. 446-452 (2017)

[38] 原亮一, 北裕幸, 石川志保, 平瀬貴之：「HP/BG 併用熱供給システムを用いた風力発電の計画発電手法」，電気学会論文誌 B 分冊，Vol.138 ，No.6，pp. 521-528 (2018)

[39] 吉永淳，赤木覚，伊藤雅一，林泰弘，石橋一成：「PV 導入量拡大を目的とした LRT および SVR と蓄電池による協調電圧制御手法」，電学論 B，Vol.136，N0.3，pp.291-301(2016)

[40] 関﨑真也，青木睦，鵜飼裕之，重藤貴也，佐々木俊介：「太陽光発電大量導入時における小容量蓄電池群を用いた配電系統電圧制御手法」, 電学論 B, Vol.133, N0.5, pp.439-448(2013)

[41] 一般社団法人 次世代自動車振興センター：「充電システムについて」，(URL: http://www.cev-pc.or.jp/kiso/juden.html アクセス:2018-11)

[42] パナソニック株式会社：「電気自動車の充電について学ぶ」，(URL:http://www2.panasonic.biz/es/densetsu/haikan/elseev/ev_kisotisiki.html アクセス:2018-11）

[43] 小柳文子、瓜生芳久：「負荷平準化を目的とする電気自動車の充放電時間制御方策」，電学論 B，Vol.118, No.5, pp505-510 (1998)

[44] 高木雅昭, 岩船由美子, 山本博巳, 山地憲治, 岡野邦彦, 日渡良爾, 池谷知彦：「プラグインハイブリッド車の負荷持続曲線に基づいたボトム充電アルゴリズム」、電学論 B, Vol.130, No.8, pp727-736 (2010)

[45] 池上貴志, 矢野仁之, 工藤耕治, 荻本和彦：「負荷平準化による発電燃料費低減を目的とした電気自動車の多数台充電制御効果の評価」，電学論 B，Vol133, No.6, pp.562-574(2013)

[46] 高木雅昭, 田頭直人, 浅野浩志：「電気自動車の使用者利便性を考慮した夜間充電負荷平準化方策」，電学論 B, Vol. 134, No.11, pp908-916 (2014)

[47] 太田豊：「電力システムと電気自動車の協調」、電学論 B, Vol. 133, No. 6, pp.497-500 (2013)

[48] 高木雅昭, 山本博巳, 山地憲治, 岡野邦彦, 日渡良爾, 池谷知彦：「LFC 信号を用いたプラグインハイブリッド車の充電制御による負荷周波数制御手法」,電学論 B,Vol.129, No.11, pp.1342-1348 (2009)

[49] 清水浩一郎，益田泰輔，太田豊，横山明彦：「系統周波数変動抑制のための電気自動車群の使用者利便性を考慮した SOC 同期制御手法制御手法」，電学論 B, Vol.132, No.1, pp.57-64 (2012)

[50] Iker Diaz de Cerio Mendaza, Jayakrishnan Radhakrishna Pillai, and Birgitte Bak-Jensen: “Demand Response Control in Low Voltage Grids for Technical and Commercial Aggregation Services”, IEEE Transactions on Smart Grid, Vol. 7, Issue. 6, pp2771-2780 (2016)

[51] Mushfiqur R. SarkerYury Dvorkin, and Miguel A. Ortega-Vazquez, ”Optimal Participation of an Electric Vehicle Aggregator in Day-Ahead Energy and Reserve Markets”, IEEE transactions on power systems, vol. 31, no. 5 (2016)

[52] Enxin Yao and Robert Schober “Optimization of Aggregate Capacity of PEVs for Frequency Regulation Service in Day-Ahead Market” IEEE Transactions on Smart Grid, Vol. 9, Issue:4 (2018)

[53] 野田琢，樺澤祐一郎，福島健太郎，根本孝七，上村　敏:「電気自動車普通充電器からの無効電力注入による夜間一斉充電時の需要家電圧低下補償手法」，電学論Ｂ, Vol. 132, No. 2, pp.163-170 (2012)

[54] 三栗祐己，原亮一，北裕幸，神谷英志，滝祥治，平岩直哉，小暮英二：「電気自動車の充電調整と無効電力注入による夜間一斉充電時の配電系統電圧低下補償に関する研究」，電学論 B，Vol.133，No.2, pp.157-166 (2013)

[55] 一般社団法人 電動車両用電力供給システム協議会(EVPOSSA) :「普通充電器出荷自主統計」(URL: http://evpossa.or.jp/pdf/shukka201809.pdf, アクセス日: 2018-11)

[56] CHAdeMO 協議会 HP (URL: https://www.chademo.com/ja/, アクセス: 2018-11)

[57] 益田泰輔, 郡司掛安俊, 横山明彦, 多田泰之：「大量の再生可能エネルギー電源が導入された電力系統における需要家の利便性と不確実性を考慮した多数台の可制御ヒートポンプ給湯機の統計的モデリングとその周波数制御への適用」, 電学論 B, Vol.131, No.1, pp.9-19 (2011)

[58] 奥谷和也, 馬場旬平, 太田豊：「家庭用ヒートポンプ給湯機の可制御負荷利用時における需要側・系統側影響の検討」，電学論 B, Vol.136, No.1, pp.72-78 (2016)

[59] 益田泰輔, 井上孝弘, 横山明彦：「負荷周波数制御と経済負荷配分制御のための多数台のヒートポンプ給湯機の運転計画作成手法」，電学論 B, Vol.133, No.4, pp.302-312 (2013)

[60] 池上貴志, 岩船由美子, 荻本和彦：「電力需給調整力確保に向けた家庭内機器最適運転計画モデルの開発」，電学論 B, Vol.133, No.10, pp.877-887 (2010)

[61] 大嶺英太郎, 八太啓行, 浅利真宏, 上野剛, 小林広武：「ヒートポンプ式給湯機と電力貯蔵装置を用いた太陽光発電余剰電力利用のための需要地系統運用手法」，電学論 B, Vol.133, No.7, pp.631-641 (2013)

[62] S. Ali Pourmousavi, Stasha N. Patrick, and M. Hashem Nehrir : Real-Time Demand Response Through Aggregate Electric Water Heaters for Load Shifting and Balancing Wind Generation, IEEE Transactions on Smart Grid, Vol. 5, No. 2(2014)

[63] Young-Jin Kim, Elena Fuentes, and Leslie K. Norford : Experimental Study of Grid Frequency Regulation Ancillary Service of a Variable Speed Heat Pump, IEEE Transactions on Power Systems, Vol. 31, No. 4(2016)

[64] 飯野穣, 藤川勉, 金子沙織, 下田学, 片岡和人, 荻本和彦:「デマンドレスポンス能力推定のための負荷調整力評価モデルの構築」，電学論 B, Vol.137, No.5, pp.372-380 (2017)

[65] 髙木健太郎, 浅野浩志, 坂東茂:「太陽光発電の大量連系を考慮した業務用空調機制御による系統調整力の経済性評価」，電学論 B, Vol.137, No.10, pp.678-686 (2017)

[66] 資源エネルギー庁：「バーチャルパワープラント(VPP)・ディマンドリスポンス(DR)とは」(URL: http://www.enecho.meti.go.jp/category/saving_and_new/advanced_systems/vpp_dr/about.html アクセス日:2018-11)

[67] 資源エネルギー庁：「エネルギー・リソース・アグリゲーション・ビジネスに関するガイドライン」(2017)

[68] 常盤昌広：「充電インフラを形成する大容量急速充電器「TQVC500M3」と CHAdeMO プロトコル」，NEC 技法，Vol.65，No.1 (2012)

[69] CHAdeMO 協議会：「急速充電設備の仕様、構造および設置について」第 1 回検討会資料

[70] 電気協同研究会: “配電系統における電力品質の現状と対応技術”, 電気協同研究第 60 巻第 2 号 (2005)

[71] 昭和電線ケーブルシステム株式会社ホームページ

(URL: http://www.swcc.co.jp/cs/index.htm アクセス日：2018-11)
[72] 電気学会 地域供給系統モデル 66kV 系統 架空・地中線混在系統負荷曲線データ
[73] 電気事業連合会：「一世帯あたり電力消費量の統計」
(URL: http://www.fepc.or.jp/enterprise/jigyou/japan/ アクセス日:2018-11)
[74] 一般社団法人次世代自動車振興センター：「平成 24 年度電気自動車・充電インフラ等の普及に関する調査報告書」(2013)
[75] 中央調査社「全国成人のコンビニエンスストア利用状況」
(URL: http://www.crs.or.jp/data/pdf/cvs.pdf アクセス:2018-11)
[76] 国土交通省：「自動車関係統計データ 自動車輸送統計調査(年報)」(2015)
[77] CHAdeMO 協議会：「電気自動車用急速充電器の設置・運用に関する手引書」(2014)
[78] 経済産業省「一般送配電事業者が行う調整力の公募調達に係る考え方」
(URL: http://www.meti.go.jp/press/2016/10/20161017002/20161017002-1.pdf アクセス:2018-11)
[79] 北海道ガス株式会社：「個人のお客さま／料金表一覧」
(URL: https://www.hokkaido-gas.co.jp/home/ryo-kin/menu/ryokinhyo.html アクセス:2018-11)
[80] 北海道電力：「従量電灯など基本的なご契約の単価表（電気供給約款）」
[81] 石田武志, 森俊介, 堂脇清志：「経済性制約と機器の部分負荷特性を考慮した業務建物の最適 CGS 導入決定支援システムの構築」，電学論 B，Vol.125，No.4, pp.373-380 (2005)
[82] 若園芳嗣，加藤丈佳，ウ カイ，横水康伸，岡本達希，鈴置保雄：「分オーダーの給湯負荷データを用いた住宅用マイクロコジェネの運転方法・蓄熱容量の検討」, 電学論 B, Vol.121, No.8, pp.373-380 (2001)
[83] 乾義尚, 武藤利英, 前田哲彦：「電力・給湯負荷の実測値に基づく集合住宅用 PEFC μCGS の容量と運転法の検討」，電学論 B，Vol.128，No.2， pp.451-458 (2008)

著者が発表した論文

本論文に関して著者が公表した論文

[1] 中村勇太, 原亮一, 北裕幸, 田中英一：「配電系統における電気自動車用急速充電器を用いた電圧制御」、電気学会論文誌 B 分冊，Vol.138，No.2，pp. 107-115 (2018)

[2] 中村勇太, 原亮一, 北裕幸, 武田清賢：「需給調整力提供を目的としたアグリゲータにより統括されるコージェネレーションシステムの最適運用」、電気学会論文誌 B 分冊, (掲載決定，2019 年 2 月に掲載予定)

本論文に関して著者が行った講演発表等

[1] Y.Nakamura, Y.Mitsukuri, Y.Mishima, R.Hara, H.Kita, K.Watanabe, K.Mori, Yasuhiro Kataoka, Eiji Kogure: "Study on Voltage Regulation in a Distribution System Using Electric Vehicles - Characteristic of Coordinated Control -", Proc. of Cigre SC/C6 Colloquium, Yokohama Japan, No. S6-3, pp.169-174 (2013)

[2] 中村勇太，三栗祐己，三島裕樹，原亮一，北裕幸：「電気自動車の無効電力調整による配電系統電圧低下補償時の経済制度の検討」，平成 26 年電気学会全国大会，6-115，6 分冊 pp.217-218 (2014)

[3] 中村勇太，三栗 祐己，井口傑，三島裕樹，北裕幸，原亮一：「電気自動車の有効・無効電力調整による配電系統電圧低下補償時の報奨金分配方法の検討」、平成 26 年電気・情報関係学会北海道支部連合大会、No.39、(2014)

[4] Y.Nakamura, Y.Mitsukuri, M.Iguchi, Y.Mishima, Ryoichi Hara, Hiroyuki Kita : ”Study of Economic System at Compensation for Voltage Drop Utilizing Charging Power Adjustment of Electric Vehicles”, IEEE PES Asia-Pacific Power and Energy Engineering Conference,0058,Hong Kong(2014)

[5] 中村勇太，原亮一，北裕幸，田中英一：「PV 出力による電圧上昇対策としての EV 用急速充電器の等価 SVC 容量の推定に関する基礎検討」，平成 27 年度電気・情報関係学会北海道支部連合大会，No.66 (2015)

[6] 中村勇太，原亮一，北裕幸，田中英一：「配電系統における EV 用急速充電器を用いた電圧制御に関する検討」，電気学会電力技術・電力系統技術・半導体電力変換技術合同研究会，PE-16-003／PSE-16-023／SPC-16-042 (2016)

[7] 中村勇太，原亮一，北裕幸，田中英一：「PV 出力による電圧上昇対策としての EV 用急速充電器の等価 SVC 容量の推定」，平成 28 年電気学会全国大会，6-202 (2016)

[8] Y.Nakamura, R.Hara, H.Kita, E.Tanaka : ”Study on Voltage Regulation Utilizing EV Rapid Charger in Distribution System”, The International Conference on Electrical Engineering, 90290 (2016)

[9] 中村勇太，原亮一，北裕幸，田中英一：「配電系統における EV 用急速充電器の無効電力を用いた電圧制御に関する検討」, 平成 28 年電気学会電力・エネルギー部門大会, 論文Ⅱ, No.182 (2016)

[10] 中村勇太，原亮一，北裕幸，田中英一：「配電系統における EV 用急速充電器を用いた電圧制御に関する検討－制御パラメータの最適設計と電圧変動抑制効果の検証－」平成 28 年電力技術・電力系統技術合同研究会，PE-16-116／PSE-16-136 (2016)

[11] 中村勇太，原亮一，北裕幸，田中英一：「配電系統における EV 用急速充電器を用いた電圧制御に関する検討―電圧不感帯に基づく電圧制御手法の提案―」，平成 28 年度電気・情報関係学会北海道支部連合大会，No.56 (2016)

[12] 中村勇太，原 亮一，北 裕幸，田中英一：「配電系統における EV 用急速充電器を用いた電圧制御に関する検討 - 電圧不感帯設計が制御効果に与える影響 - 」，平成 29 年電気学会全国大会，6-181 (2017)

[13] Y.Nakamura,R.Hara,H.Kita,E.Tanaka:"Study on Voltage Regulation Utilizing EV Rapid Chargers in Distribution System - Design of Voltage Dead-band for Long-term Control -",Proc. of International Conference on Electrical Engineering (ICEE2017), No.201702200000139, (2017)

[14] 中村勇太，原 亮一，北 裕幸，田中英一：「配電系統における電気自動車用急速充電器を用いた電圧制御」，平成 29 年電気学会電力・エネルギー部門大会，論文Ⅰ，No.32 (2017)

[15] 中村勇太,原 亮一,北 裕幸,田中英一,横川 誠,武田清賢：「電力需給調整力提供を目的とした複数のコージェネレーションシステムの運用計画に関する基礎検討－系統による影響を考慮した調整能力評価－」，平成 29 年電力技術・電力系統技術合同研究会，PE-17-060, PSE-17-060(2017)

[16] 中村勇太, 原亮一, 北裕幸, 横川誠, 武田清賢：「電力需給調整力提供を目的とした複数のコージェネレーションシステムの運用計画に関する基礎検討―調整力割り当て方法に関する一考察―」，平成 29 年度電気・情報関係学会北海道支部連合大会，No.54 (2017)

[17] 中村勇太，原亮一，北裕幸，横川誠，武田清賢：「需給調整力提供を目的とした複数コージェネレーションシステムの運用計画に関する検討」, 平成 30 年電気学会全国大会, 6-134 (2018)

[18] 中村勇太，原亮一，北裕幸，武田清賢：「調整力提供を目的としたアグリゲータにより統括されるコージェネレーションシステムの最適運用」，平成 30 年電気学会電力・エネルギー部門大会，論文Ⅰ，No.53 (2018)

[19] 中村勇太，原亮一，北裕幸，武田清賢：「電力需給調整に貢献するコージェネレーションシステムの最適構成に関する検討」，平成 30 年電力技術・電力系統技術合同研究会，PE-18-145，PSE-18-121 (2018)

[20] 中村勇太，原亮一，北裕幸，武田清賢：「電力需給調整に貢献するコージェネレーションシステムの運用に関する検討 −需要の不確実性を考慮した調整能力評価−」，平成 30 年度電気・情報関係学会北海道支部連合大会，No.53 (2018)

謝辞

本論文は，筆者が北海道大学大学院 情報科学研究科 システム情報科学専攻 システム融合学講座 電力システム研究室に在籍中の研究成果をまとめたものであり，本論文の執筆にあたり，多くの方々から多大なるご指導・ご支援を頂きました。

北海道大学大学院情報科学研究科電力システム研究室の　北　裕幸教授には，指導教官として，研究室ゼミ，論文執筆ならびに学会発表等，多岐にわたって，貴重なご指導頂きました。ここに心からお礼申し上げます。

北海道大学大学院情報科学研究科電力システム研究室の　原　亮一准教授には日ごろの研究打合せ，論文執筆，学会発表練習等の研究活動等，様々な場面で貴重な経験・ご指導を頂きました。ここに心からお礼申し上げます。

また，北海道大学大学院情報科学研究科電力システム研究室の　田中　英一助教（2018 年 3 月にご退官）にも日ごろの研究打合せや原稿添削等，細部に渡りご指導や助言を頂きました。ここに感謝の意を表します。

北海道大学大学院情報科学研究科　五十嵐　一教授ならびに小笠原　悟司教授にはご多忙な中本論文の副査を務めて頂きました。深く御礼申し上げます。

東京電力ホールディングス株式会社の佐野常世様，西田悠介様，東京電力パワーグリッド株式会社の小林直樹様，渡辺雅人様，北海道ガス株式会社武田清賢様をはじめとする企業の方にも，研究打合せ等で様々な方面・角度からの意見を頂きました。心より感謝申し上げます。

本研究の一部はパワーアカデミーの研究助成により実施されたものであり，関係各位にはここに深くお礼申し上げます。

筆者が在籍中に秘書として所属していた高橋緑様，出村澄世様，有馬理恵様には，学会の手続きから，普段からの研究室の生活まで大変お世話になりました。厚く御礼申し上げます。

また研究室内の学生の皆様から，研究内容をはじめとする助言を沢山頂き，さらには充実した研究室生活を送ることができました。

最後に，大学院進学まで生活や経済的な支援をして頂いた両親に深く感謝いたします。ありがとうございました。

2019 年 1 月

北海道大学大学院 情報科学研究科

システム情報科学専攻 電力システム研究室

中村 勇太

付録

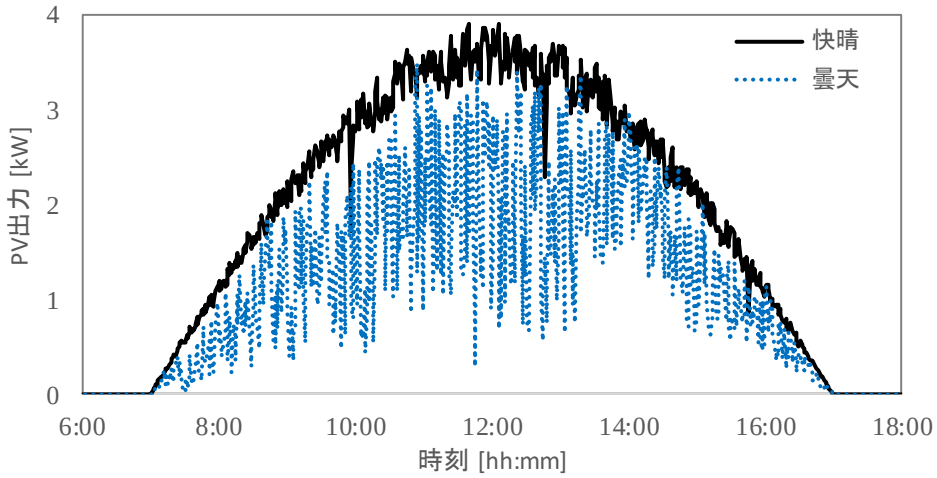
本論文の数値試算の仕様・フローチャートおよび使用したプログラムリストを添付する。
なお，本プログラムは MATLAB R2015b～2018a の実行環境で開発・実行を遂行しており，基本的にバージョンによらず動作する。

付録(電圧制御に貢献する急速充電器の運用)

ここでは，第 2 章で検討した電圧制御に貢献する充電器の運用において，制御パラメータが制御効果に与える影響について検証した結果を述べる。
また，数値試算に使用したプログラムの説明およびプログラムリストを掲載する。

制御パラメータの設計

ここでは，第 2 章で検討した電圧制御に貢献する充電器の運用において，制御パラメータが制御効果に与える影響について検証した結果を述べる。設計方法として，第 2 章で取り扱った系統モデル・充電器状況の下，付録_図 1 に示すような異なる PV 出力パターンを 100 日分作成し，制御パラメータを段階的に変えたときの電圧制御指標を算出し，その 100 日分の電圧制御指標値の分布(期待値・最大値)から制御パラメータの特性を明らかにする。本節で使用する各電圧制御指標値は，前節で取り扱った LRT タップ切替回数(1 日平均を意味する「平均 LRT タップ切替回数」)，電圧変動量および電圧逸脱量に加えて，100 日間のうち何日電圧逸脱が発生したかという「電圧逸脱日数」を用いる。

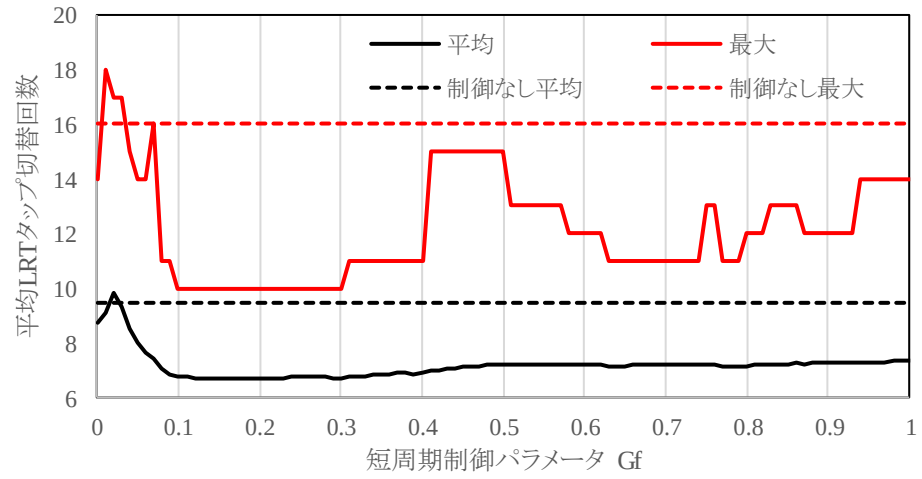


付録_図 1 代表的な PV 出力パターン

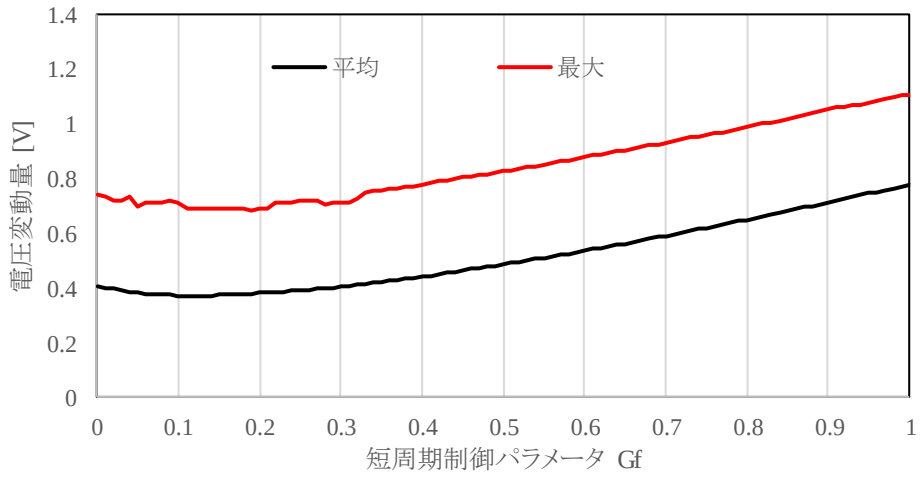
短周期制御パラメータ

短周期制御パラメータはフィードバックゲイン(G_f)である。G_f は 0～1 の範囲であることから，本試算ではG_f をこの範囲を 0.01 刻みで変更させ，各電圧制御指標値を算出する。

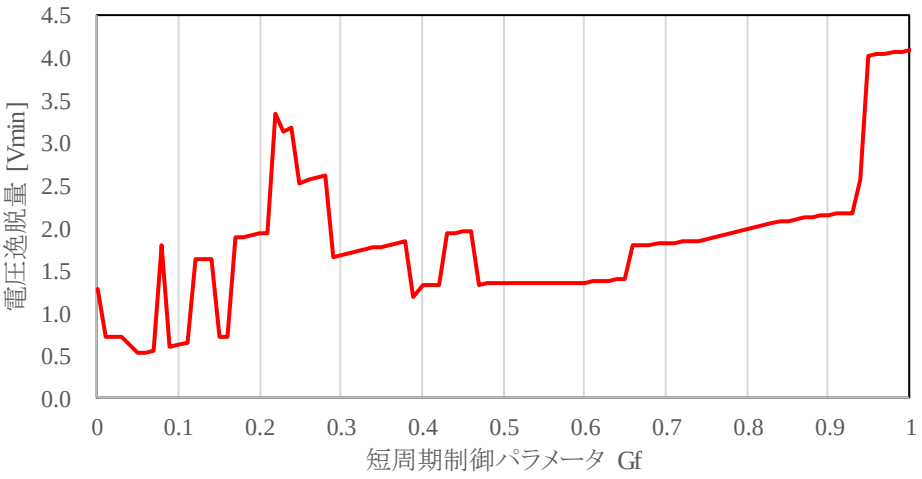
付録_図 2 は，実際の試算結果から算出した各電圧制御指標値を示している。同図(a)～(c)において，黒線が 100 日間の平均値(期待値)，赤線が 100 間の最大値を示し，実線が制御パラメータを変更したときの値，破線が制御しない場合(パラメータに依存しないため，x 軸に平行)の値を示している(破線が存在しない場合は y 軸最大値以上の値である)。同図(d)は逸脱量の平均(同図(c)の実線)を拡大したものと，青色が逸脱日数を同グラフに示している(実線・破線は他と同じ)。



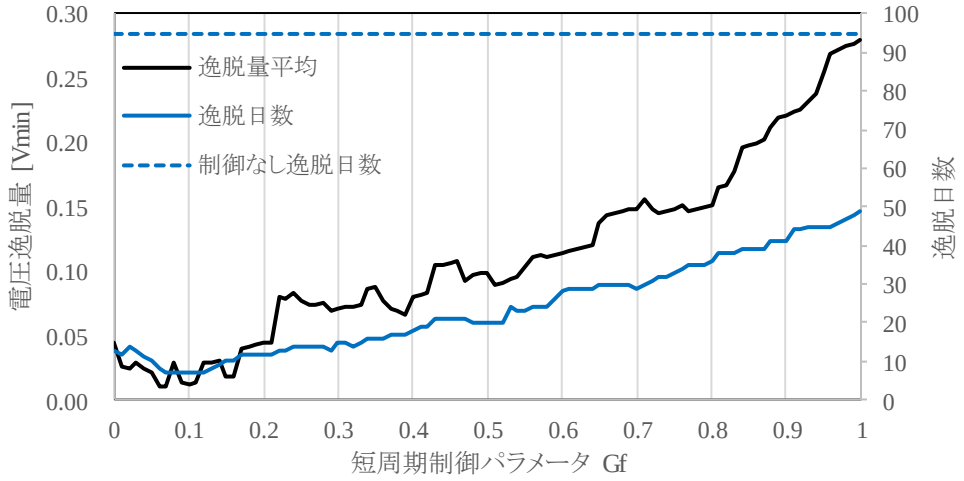
(a) 平均 LRT タップ切替回数



(b) 電圧変動量



(c) 電圧逸脱量(最大値)



(d) 電圧逸脱量(平均)と電圧逸脱日数

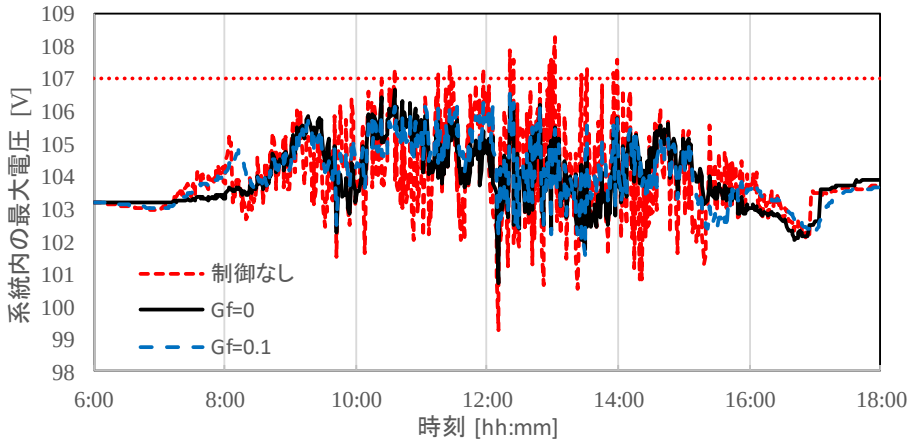
付録_図 2 短周期制御パラメータ変更時の電圧制御指標値

まず同図(a)の平均 LRT タップ切替回数を観察すると，G_f が 0.08 以下では平均値・最大値が比較的高く，G_f が 0.01 や 0.02 のときの最大値は制御なしの場合よりも高くなっている。一方で，G_f が 0.08 以上であれば平均値・最大値共に減少する傾向となっている。次に同図(b)に示す電圧変動量を観察すると，G_f が 0 よりも 0.1～0.2 付近にした方が平均値・最大値共に小さい。また，同図(c)に示す電圧逸脱量の最大量は突発的に増減する個所があるが，全体としてはG_f の増加に伴い増加する傾向にある。同図(d)の電圧逸脱量の平均値および電圧逸脱日数も G_f の増加に伴って増加傾向にある一方，わずかではあるが G_f が 0 よりも 0.1 の方が小さい。これらの結果を踏まえると，全体としてはG_f が増加すると各指標値も増加する傾向にあるが，G_f を 0 にすると平均 LRT タップ回数が多く，総合的にはG_f を 0.1 程度とした方が各電圧制御指標は小さくなり，G_f =0.1 が適切なパラメータ値になる，ということがいえる。

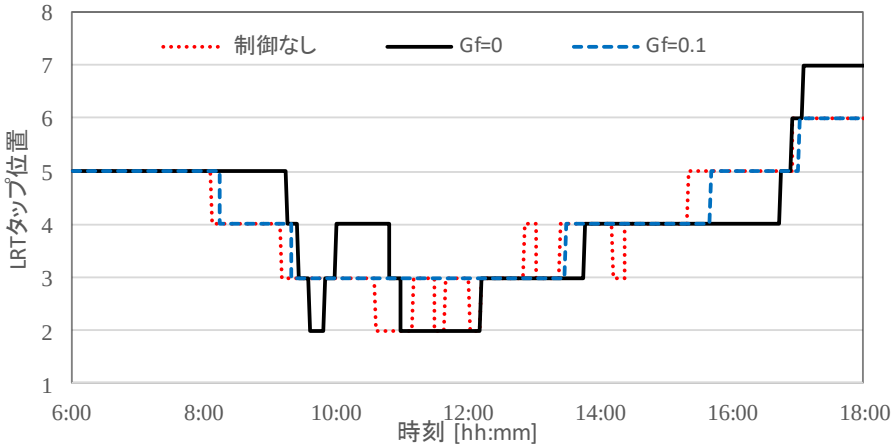
ここで，G_f の設定が制御結果に与える影響を詳細に観察するために，制御なしの場合と，G_f を 0 と 0.1(最適値)とした 3 つの場合において，100 日間のうち最も LRT タップ切替回数が多い日の系統内の最大電圧(V_{max})，LRT のタップ位置，LRT 不感帯逸脱積算量，末端ノード(#E)の接続されている充電器 1 台の無効電力補償量をそれぞれ付録_図 3 の(a)～(d)に示す。

同図(a)に示すように制御なしでは電圧逸脱が発生したものの，G_f =0 と G_f =0.1 設定して提案手法を実施することでその電圧逸脱は回避されている。同図(b)のタップ切替回数を観察すると，制御なしでは頻繁にタップ切替動作がされているが，提案手法によってその切替回数は低減されている。しかしながら，G_f =0 と G_f =0.1 を比較すると，G_f =0 では連続したタップ動作(9:14 から 9:37 や 16:44 から 17:10 等)がされており，制御なしおよびG_f =0.1 には行われないタップ位置にも切替している(10:00 から 10:48 のタップ位置 4,17:05 以降のタップ位置 7)。同図(c)に示す LRT 不感帯逸脱積算量を観察すると，G_f =0 の上記の時間帯では不感帯積算量が LRT のタップ動作を行う不感帯逸脱積算量(同図(c)の黒破線)を継続して超過しており，1 度 LRT タップ動作時限時間(10 分)後に再度タップ切替を行っている。また，同図(d)に示す無効電力補償量を見ると，G_f =0 の場合では上記の時間帯を中心に定常的に無効電力補償がされている。

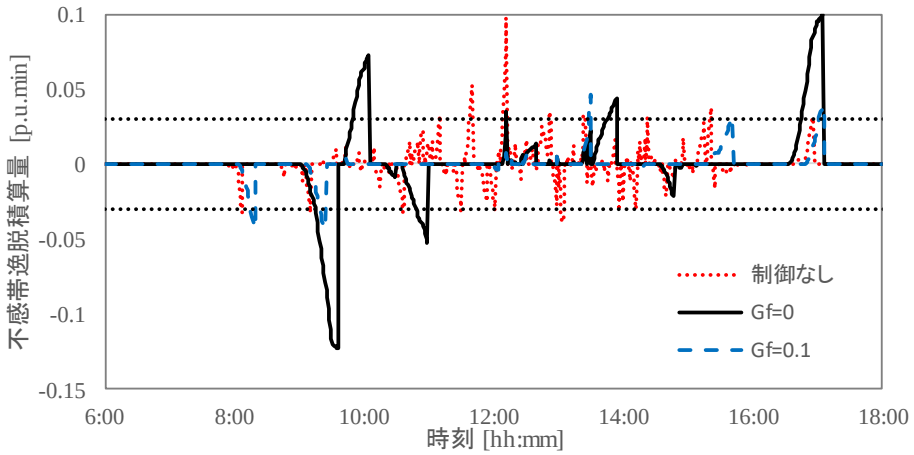
このように，各充電器は接続されている電圧に基づいて動作するため，PV 出力や負荷等の正味の負荷による電圧変動か LRT 等の電圧制御機器によるものなのか判断することができない。従って，電圧変動を全て抑制すると，電圧制御機器側ではさらに制御が必要だと判断し，更なる動作を誘発してしまう，すなわち電圧制御機器との協調が計れない可能性がある。一方でG_f=0.1 とすると，LRT のタップ切替動作直後にはその電圧変動を抑制する方向に無効電力補償を行うが，タップ切替後の PV 出力や負荷等の負荷による電圧変動が小さければその補償量が減衰されていく。これにより，LRT のタップ動作から数制御サンプリング時間後には LRT の電圧推定値は目標電圧値周辺に位置する。その結果，充電器の無効電力補償による LRT の不必要な動作はされず，PV 出力や負荷等の正味の負荷が大きく変化した際にタップ動作はされることから，LRT のタップ動作との協調が図ることができると考えられる。



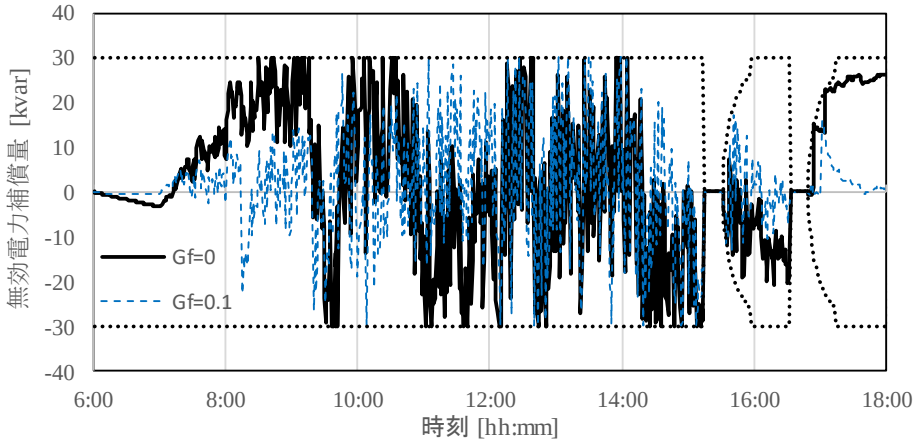
(a) 系統内の最大電圧



(b) LRT のタップ位置



(c) LRT 不感帯逸脱積算量



(d) 末端ノード(#E)の接続されている充電器 1 台の無効電力補償量

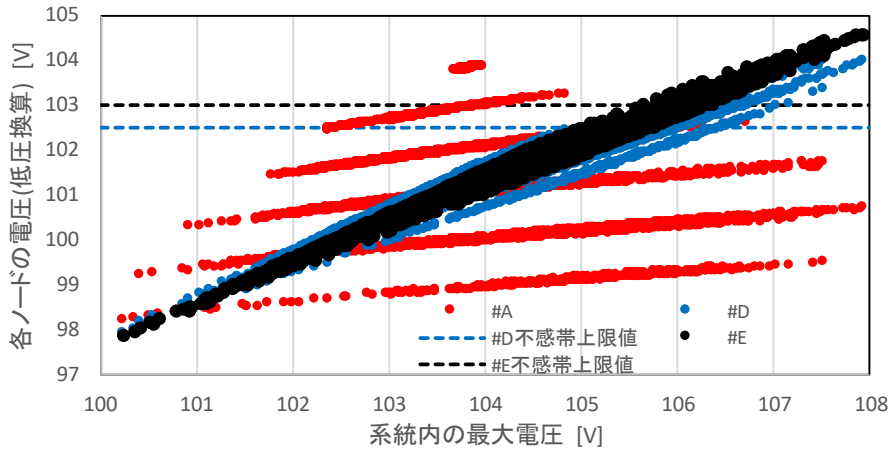
付録_図 3 短周期制御パラメータが電圧制御に与える影響の検証結果

長周期制御パラメータ

付録_図 4 の各プロット点は過去の電圧情報を入力情報として，オフラインで作成したグラフであり，横軸が系統内の最大電圧，縦軸が各ノード(#A，#D，#E)の電圧値を示している。赤色で示した#A ノードの関係をみると，いくつかの線上にわかれている。一方で，末端側(青色：#D,黒色：#E)のノードは，ほとんど一直線上に位置しており各ノードの電圧と系統内の最大電圧の相関は高いと考えられる。このように，各ノードと系統内の電圧の相関関係はノードの位置によって大きく変化する。このようにノードの変化による相関関係は，系統内の電力潮流と LRT のタップ位置(送出電圧)が要因となって変化する。同じ電力を送る場合でも，LRT のタップ切替による系統全体の電圧の変化に伴って潮流量も変化する。電源側ノードでは LRT のタップ切替による電圧変化が負荷や PV 等の潮流量変化に伴う電圧変化よりも大きく，タップ位置によって段階的に変化する。一方，末端側ノードはで LRT のタップ切替による電圧変化よりも，時々刻々変化する潮流量の変化が大きいため，直線状に変化する。

これより，末端側のノードでは受電点の電圧(縦軸)から系統内の電圧情報(横軸)をある程度推定することができ，各色の破線に示す高さ(受電点の電圧)以下であれば，系統内で電圧上限値(横軸

107V)の逸脱が発生しないと考えられる。よって，各色の破線を対応する充電器の上限不感帯上限値$V_i^{d,upper}$とし，不感帯の上限値以下に収めるように無効電力補償量を制御する。一方，電源側のノードでは受電点の電圧(縦軸)だけでは系統内の電圧情報(横軸)を推定できず，末端側のような破線を引くことが困難である。仮に同様の手法により決定(約 99V)にすれば，系統内の最大電圧が適正範囲上限値まで余裕がある時間にも無効電力補償を行ってしまい，逆に系統内の電圧が適正範囲の下限値逸脱や電圧制御機器等との動作方向の不一致が発生するといった影響が想定される。このような理由から長周期制御は，系統の末端側に接続された充電器だけが実施する。また同様の手順で下限側不感帯の下限値を決定する。

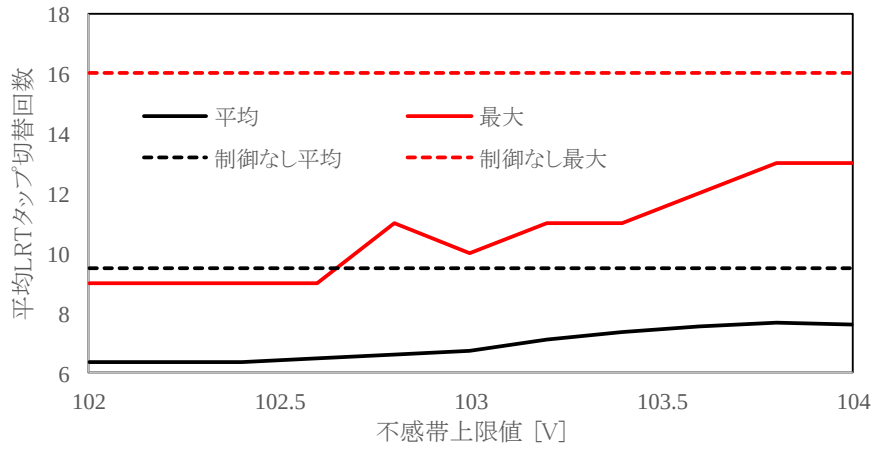


付録_図 4 過去の電圧情報から算出した不感帯の設計

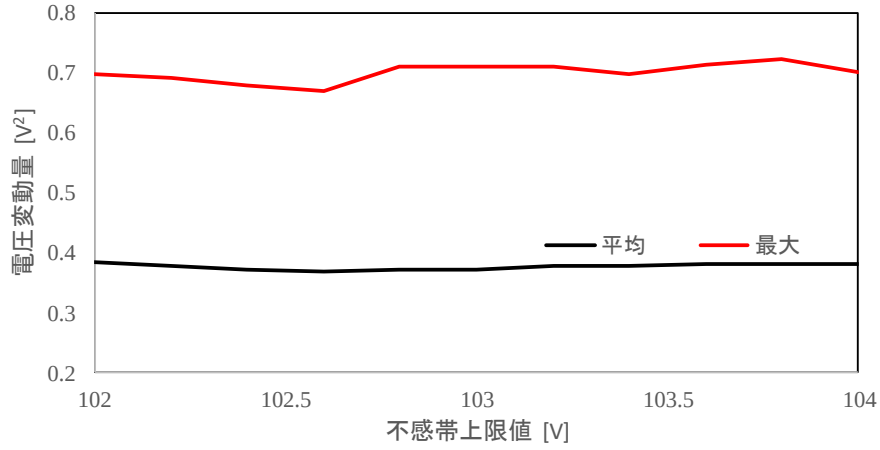
この設計方法の妥当性について検証するために，不感帯を段階的に変更させたときの各電圧制御指標を算出・比較を行う。不感帯は前述したように高圧ノードにより値が異なる(#D : 102.5V と #E : 103.0V)が，ここでは両ノードも同じだけ値を変更させる。また説明上#E の不感帯上限値を基準にし，#E を 102.0V から 104.0V まで 0.2V づつ変化させて試算する(#D は 101.5V から 103.5V まで変化)。不感帯幅については#E は 0.5V，#D は 0.3V とする。

以上の条件で試算した結果を付録_図 5(a)～(d)に示す。まず，同図(a)の平均 LRT タップ切替回数を観察すると，不感帯上限値が低いほどタップ切替回数の平均値・最大値共にが少なく，下限値が高くなるにつれてタップ切替回数も増加する。同図(b)の電圧変動量は最大値が多少ばらつきはあるものの，平均値はほとんど変化しない。同図(c)および(d)の電圧逸脱量および電圧逸脱日数については，上限値を 103V 付近の値に設定することで平均値・最大値共に低い傾向にあり，103V から離れるにつれて増加傾向にある。

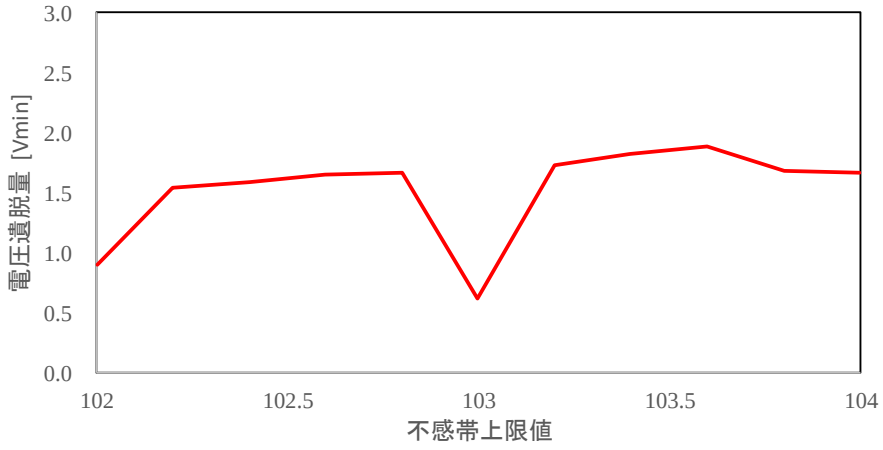
次に短周期制御と同様，制御パラメータの妥当性について議論するために，2 つのケースにおいて不感帯設定が電圧制御に与える影響を検証する。ケース 1 は提案手法によって設計した値(#D : 102.5V, #E : 103V)に設定したケースであり，ケース 2 は本試算で最も小さい値（#D : 101.5V, #E : 102V）に設定したケースである。



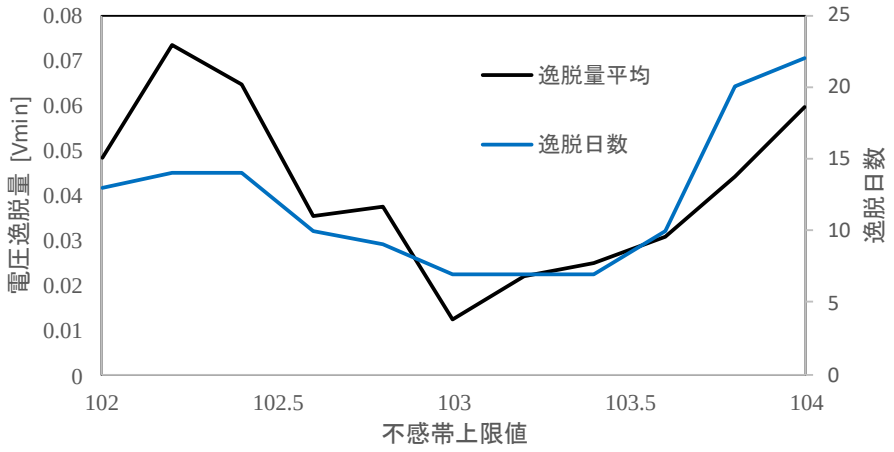
(a) 平均 LRT タップ切替回数



(b) 電圧変動量



(c) 電圧逸脱量



(d) 電圧逸脱量(平均)と電圧逸脱日数

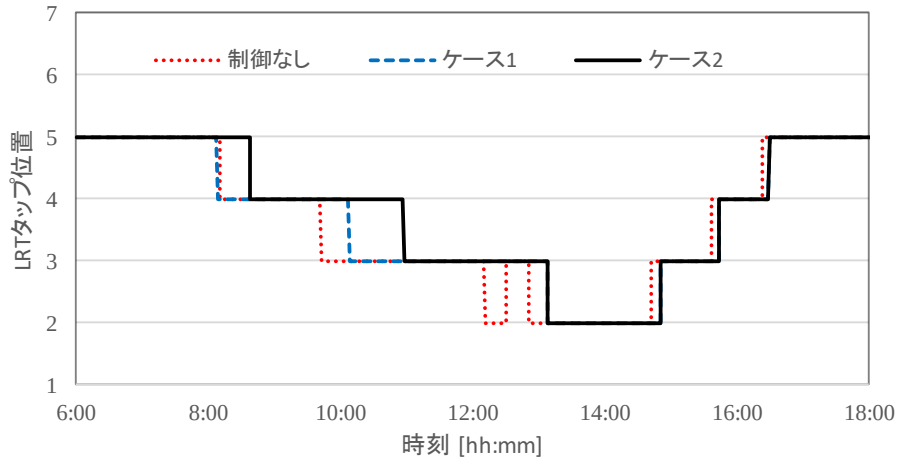
付録_図 5 長周期制御パラメータ変更時の電圧制御指標値

付録_図 6 (a)～(c)は 2 つケースにおける試算期間内のある 1 日の系統内の最大電圧，各充電器の長周期無効電力補償量の合計，LRT のタップ位置をそれぞれ示している。両ケース共に最大電圧が高い 9:00～13:00 を中心に無効電力補償がされているが，ケース 2 の方が，最大電圧があまり高くない時間帯である 7:30 付近から長周期無効電力補償がなされている。その結果として，本来動作すべき LRT のタップ切替が遅れ，10:58 で電圧逸脱が発生している。

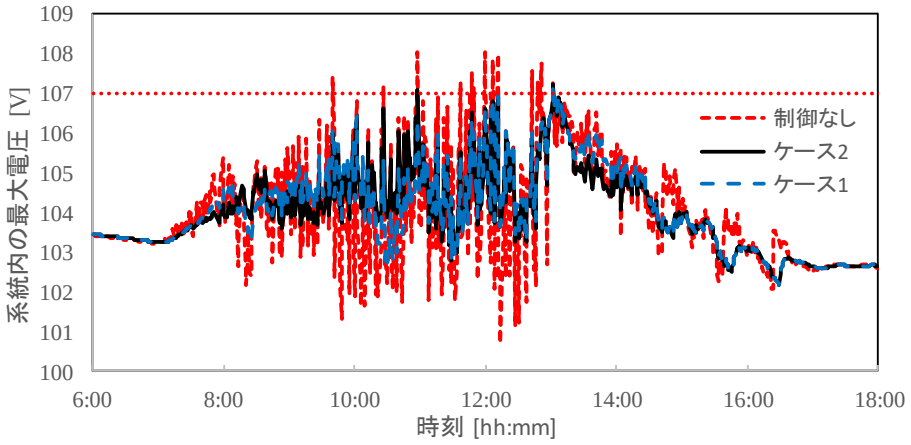
付録_図 7 に最大電圧と無効電力利用率(以後, 単に利用率)の関係を示す。ここで, 利用率は#D, #E の充電器 12 台の合計空き容量に対する合計の無効電力補償量の割合であり，正負は充電器の無効電力補償量の向きを表している。なお，同図には短周期制御および長周期制御の内訳も示している。ケース 1 では，最大電圧が 105V 付近で長周期制御の利用率が増加し始めるが，短周期制御は最大電圧の値に寄らずに補償しており，より広い範囲で電圧変動抑制がされている様子が観察できる。しかしながら，ケース 2 の長周期制御の利用率は最大電圧が 103.5V 付近から増加し始め，105.5V を超えると 100%に達しており，それ以上の電圧値では短周期制御は負方向しか補償できていない。

このように，不感帯を低く設定すると必要以上の長周期の無効電力補償がされ，短周期制御や LRT のタップ動作などの従来の電圧制御機器との協調が図れない可能性がある。また，不感帯を高くすると，電圧逸脱発生時に無効電力補償しない可能性がある。これらの結果を踏まえると，本試算のようにあらかじめ系統内で電圧逸脱が発生しない高圧ノードの電圧値を把握し，その値を基に設計するという設計手法は妥当であると考えられる。

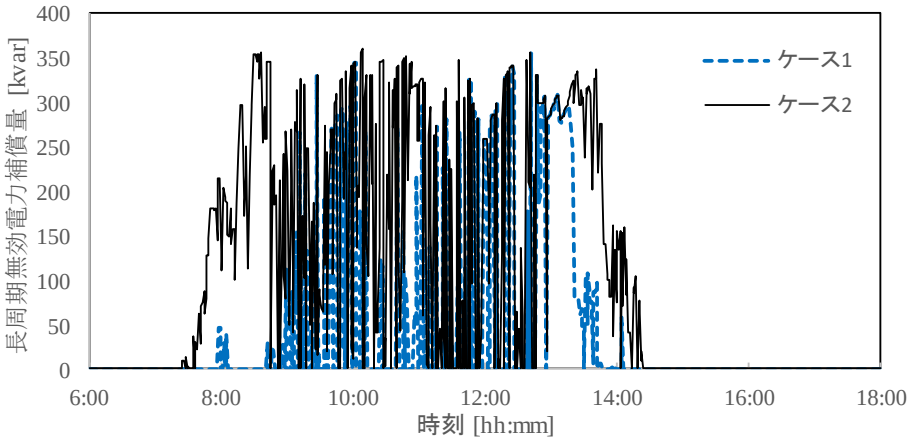
制御しない場合とこれらの最適なパラメータを用いた提案手法を実施した場合の制御指標を示している。なお，同表には比較のために各制御のみを実施した結果も示している。提案手法は電圧変動量の最大値以外は最小となっており，異なる PV 出力パターンにも有効であることがいえる。



(a) LRT のタップ位置

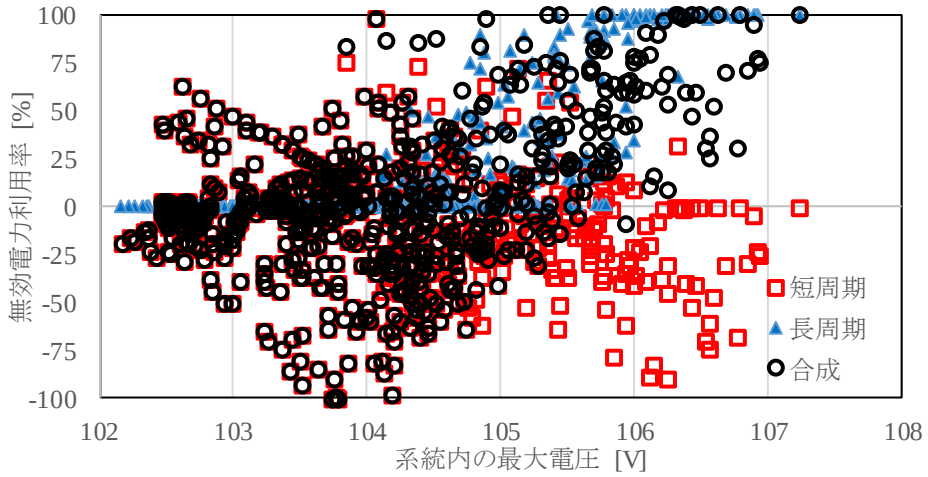


(b) 系統内の最大電圧

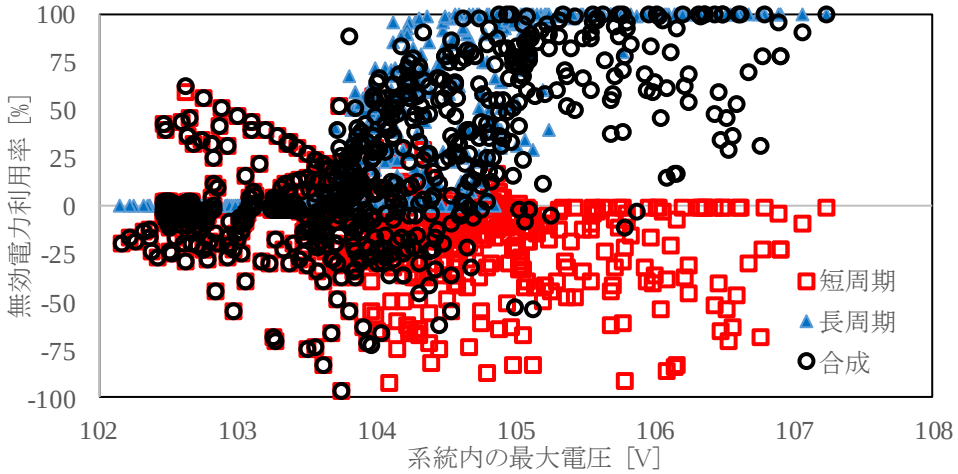


(c) 長周期無効電力補償量

付録_図 6 長周期制御パラメータが電圧制御結果に与える影響の検証結果



(a) ケース 1



(b) ケース 2

付録_図 7 系統内の最大電圧と無効電力利用率の関係

付録表 1 100 日間の試算における各種制御および提案手法の各電圧指標値

	タップ切替回数		電圧変動量 [V]		電圧逸脱量 [Vmin]		逸脱日数
	平均	最大	平均	最大	平均	最大	
制御なし	9.49	16	1.349	1.942	3.785	17.47	95
短周期制御	7.59	12	0.377	0.696	0.078	3.90	21
長周期制御	7.49	16	1.195	1.766	0.719	5.12	60
提案手法	6.77	10	0.371	0.710	0.012	0.613	7

数値試算に使用したプログラム

本論文の数値試算のフローチャートおよび使用したプログラムリストを添付する。
なお，本プログラムは MATLAB R2015b での実行環境で開発・実行していたが，MATLAB R2014b 以降であれば動作する。

プログラム全体の構成・概要

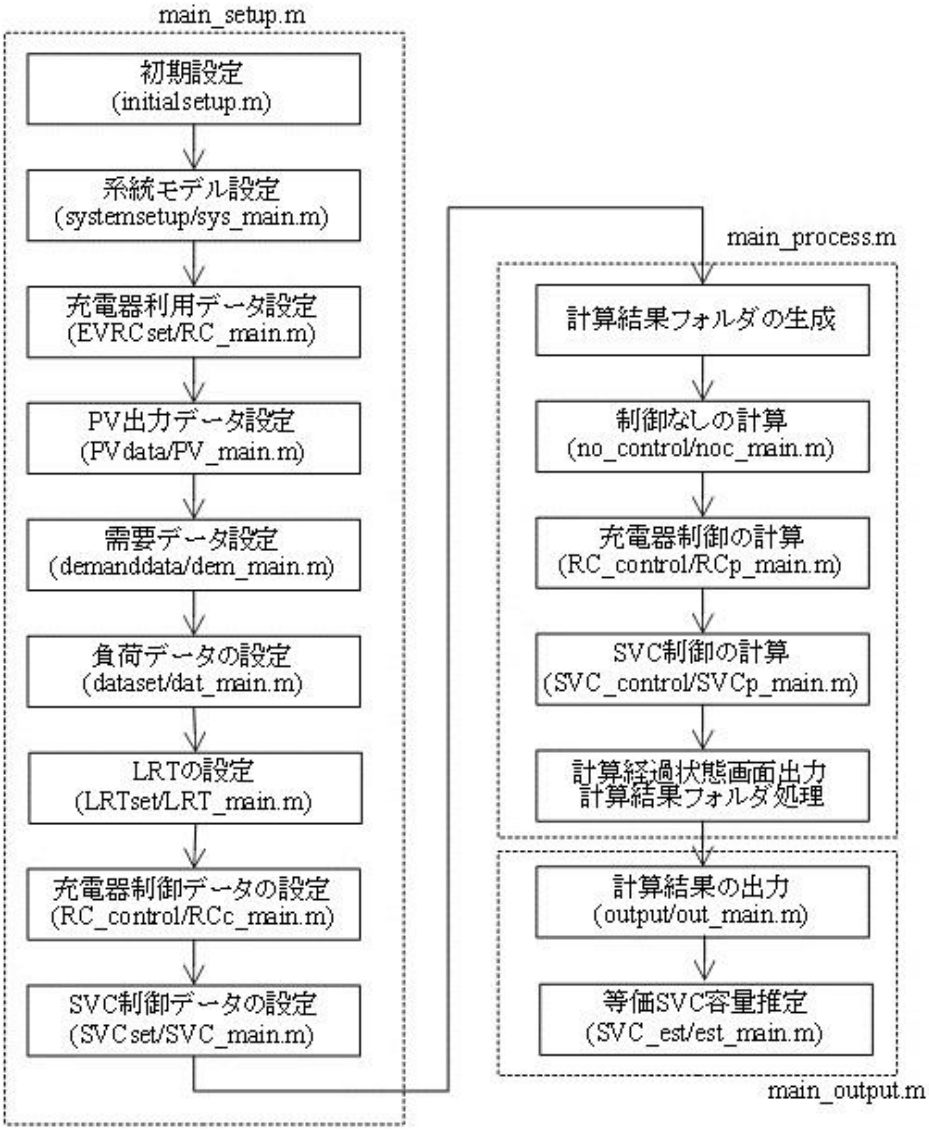
本プログラムのフォルダ構成を以下に示す。デフォルトで rootfolder の中に setup,process,takeover フォルダが存在する。rootfolder の main.m を実行することで result フォルダが生成される。
また、過去にこのプログラムで生成した（result 内の input フォルダ内の）mat データを”takeover”フォルダに格納することで設定せずにそのまま使用することができる。これは過去に生成した充電器パターンや PV 出力パターンを使用したい場合に用いる。ただし、日数やループ数が異なる場合、エラーとなる可能性があるため注意されたい。

```
rootfolder：プログラムが集約されているフォルダ(ルート)
├──setup              ：各種設定（負荷、PV、充電器利用パターンなど）フォルダ
│    └──systemsetup          ：系統モデルに関するデータを生成するフォルダ
│    ├──EVRCset   ：充電器利用データを生成するフォルダ
│    │    └──input   ：確率データが格納されているフォルダ
│    │         ├──EV          ：EV の特性に関する確率データが格納されているフォルダ
│    │         └──RC_num：充電器利用に関する確率データが格納されているフォルダ
│    ├──PVdata               ：PV 出力データを生成するフォルダ
│    │    └──input   ：PV 出力に関するデータが格納されているフォルダ
│    ├──demanddata           ：各種データを集約して正味のデータを生成するフォルダ
│    │    └──input   ：負荷データが格納されているフォルダ
│    ├──LRTset               ：LRT の設定に関するデータを生成するフォルダ
│    │    └──LDC_inp         ：LRT のインピーダンス整定値が格納されているフォルダ
│    ├──dataset              ：各種データを集約して正味のデータを生成するフォルダ
│    ├──RC_control           ：充電器制御に関するデータを生成するフォルダ
│    ├──SVCset               ：SVC 制御に関するデータを生成するフォルダ
│    ├──Otadataget  ：太田データの負荷データを生成するフォルダ
│    ├   └──inputdata        ：太田データ(csv)ファイルを格納するフォルダ
├──process                   ：実際の計算（プロセス）フォルダ
│    ├──common               ：各制御において共通フォルダ（潮流計算、LRT 動作等）
│    ├──no_control  ：制御なしの計算フォルダ
│    ├──output               ：出力(xlsx)プロセスフォルダ
│    ├──RC_control           ：充電器制御の計算フォルダ
│    └──SVC_control          ：SVC 制御の計算フォルダ
│    └──SVC_est              ：等価 SVC 容量の計算フォルダ
├──result：プログラム実行結果が集約されるフォルダ
├──takeover                  ：過去に生成した mat ファイルを引き継ぐ際に使用するフォルダ
```

1. プログラム内の各工程とフローチャート

各工程のデータ(mat)および工程間のデータの関係について本節で示す。なお、順番については、main.m(ルートフォルダ直下)の実行順に従う。

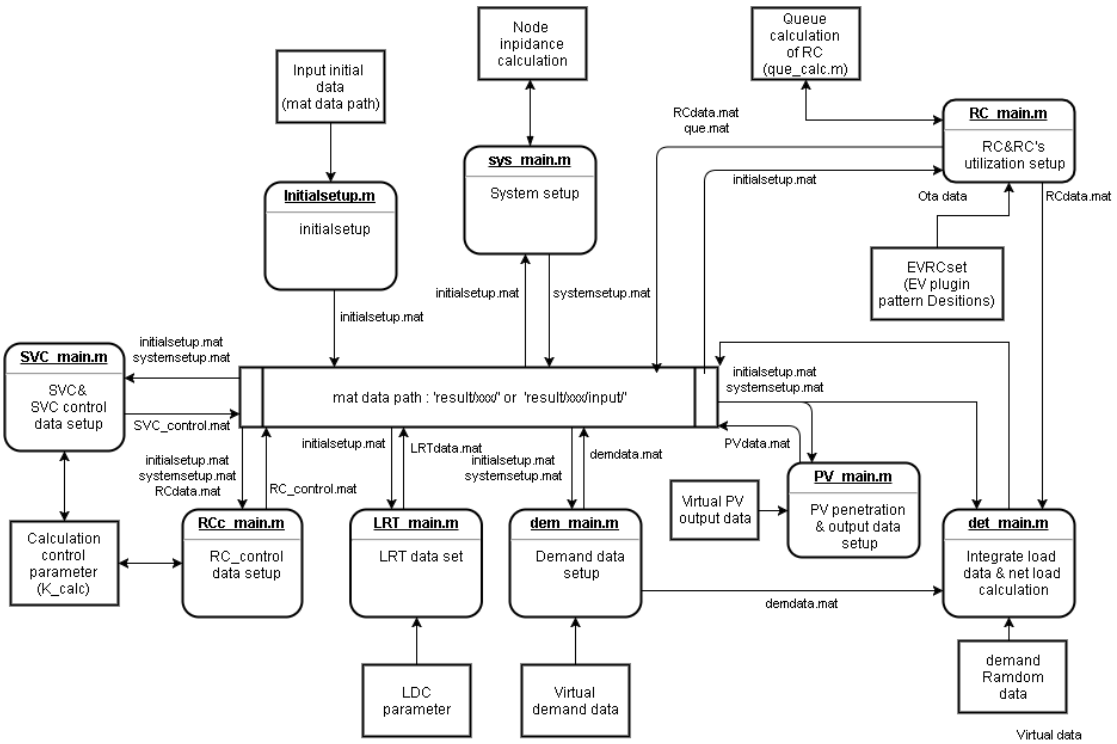
付録_図 8 は本プログラム(main.m)の主な構成について示している。同図の左側は setup フォルダ内の工程を示しており、右側が実際の計算プロセスを示している。なお、充電器または SVC 制御のどちらかしか使用しない場合は､使用しない制御器の設定を実行しないこと(コメントアウト)も可能である。



付録_図 8 プログラムのフローチャート

2. 設定(setup)フォルダ内の構成

Setup フォルダ内の工程の構成図を付録_図 9 に示す。同図は、データ(mat)ファイルの関係図について重点をおいており、各プロセスにやりとりする実体を出力フォルダ“result/xxx(initialsetup.m で入力)/input”としている。各プロセスを丸角の四角形で示しており、上部に各プロセスのメイン m ファイルを記載している（フォルダについては省略）。また、四角形はそのプロセスにおいて特筆すべき入出力データや内部工程（定数計算や設定等）を示している。また破線は負荷の取得方法について記載している部分である。



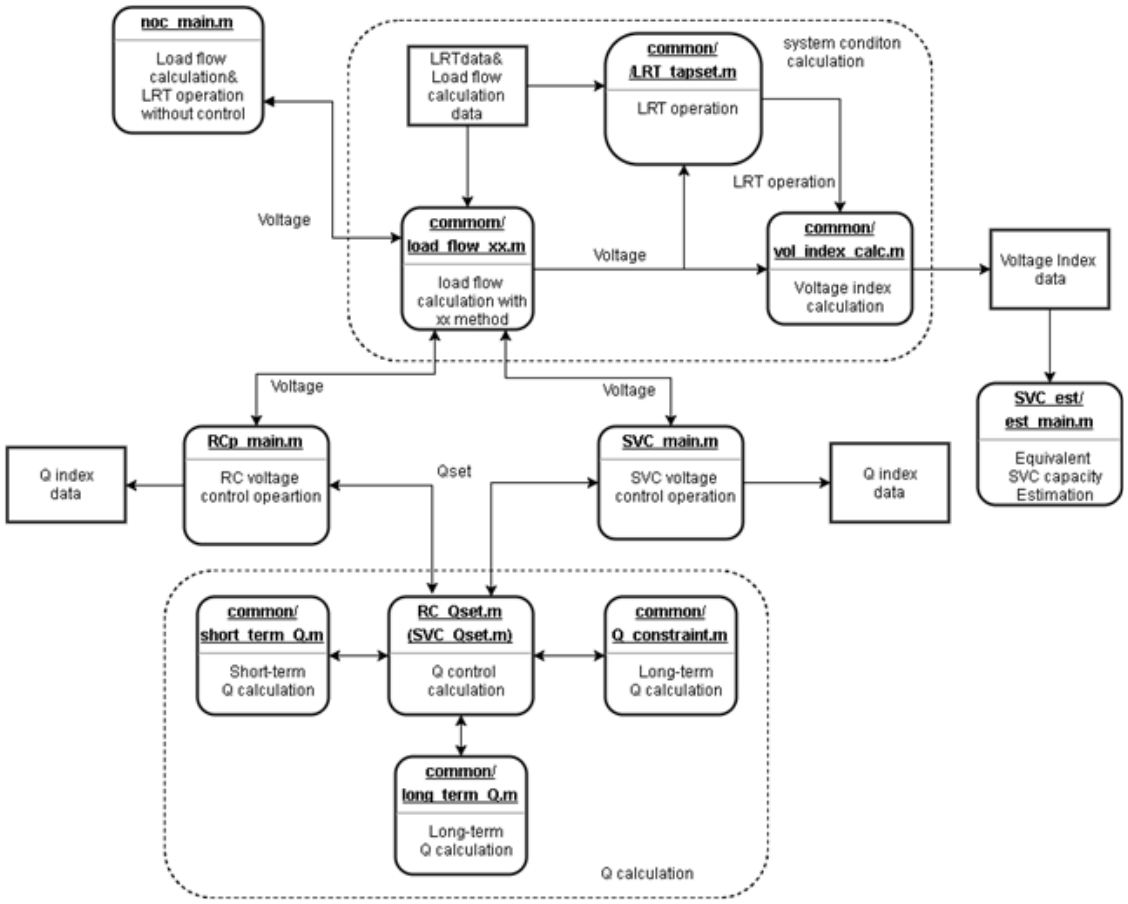
付録_図 9 Setup フォルダ内の工程の構成

3. 計算(process)フォルダ内の構成

process フォルダ内の工程の構成図を付録.図 1 に示す。なお、mat ファイルの入出力の関係については複雑になるため表 3.19 に記載する。

付録_図 9 に示すように本プログラムでは、潮流計算は全て、無効電力量の計算については充電器と SVC の場合で同じ common フォルダ内の関数を使用している。そのため、common フォルダにある関数を変更すると、ほかの制御時の関数も同様に変化するため、それぞれの関数で変更する必要はない。

また、文頭以外の関数（m ファイル）が同じ場合（noc_〇〇、RCp_〇〇と SVCp_〇〇）、内部のプログラムは基本的に同じ構文で構成されている。また、潮流計算は複数の計算方法があるため、表記上では”load_flow_xx.m”としており、”xx”には手法が入る場合もある。(Back Forward sweep 法の場合は”BFS”等)



付録.図 10 process フォルダ内の工程の構成図

4. mat ファイルの構成

本プログラムの mat ファイルの生成および使用の関係性を付録表 2 に示す。各工程で実行する際、'Use'と記載されている mat ファイルが無い場合は「変数が無い」というエラーが表示される可能性が高い。特に過去に生成した mat ファイルを使用する（mat ファイルを引き継ぐ）場合は、引継元（takeover）に mat ファイルがあるかどうか確認する必要がある（現状では、無い mat ファイルの工程を実行するように動作しているが、正しく動作しない可能性もある）。なお、"out_main"の欄は"*"記号をつけているが、これは mat ファイルがあれば実行（結果出力）する。引継でプログラムする際は、main_day_max が引継ファイルのより以下になっているか確認する必要がある。loop 回数についてはシミュレーション期間を繰り返すためエラーになる可能性は低いが、day(シミュレーション期間)の値が適切で無ければ読み取る負荷や PV データがなく、エラーとなる。

付録表 2 mat ファイルの生成・使用の関係性

matファイル名	setupフォルダ												processフォルダ				
	initialsetup	sys_main	RC_main	PV_main	dem_main	LRT_main	dat_main	Oka_main	RCc_main	SVC_main	noc_main	RCp_main	SVCp_main	out_main	est_main
initialsetup.mat	Create	Use	Use	Use	Use	Use	Use	Use	Use	Use	Use	Use	Use		
systemsetup.mat		Create		Use	Use		Use	Use	Use	Use	Use	Use	Use		
que.mat			Create												
RCdata.mat			Create				Use	Use	Use			Use			
PVdata.mat				Create			Use								
demdata.mat					Create		Use								
LRTdata.mat						Create					Use	Use	Use		
load.mat							Create	Create			Use	Use	Use		
RC_control.mat									Create			Use			
SVC_control.mat										Create			Use		
noc_result.mat											Create			Use*	
noc_volindex.mat											Create				
RCp_result.mat												Create			
RCp_Qindex.mat												Create			
RCp_volindex.mat												Create		Use*	Use
SVCp_result.mat													Create		
SVCp_Qindex.mat													Create		
SVCp_volindex.mat													Create	Use*	Use

5. mat ファイルの構成(setup フォルダ内のプロセス)

初期設定関連

mat_Table 1 initialsetup.mat

変数名	(配列の場合：配列数)	内容
main_loop_max	Value(自然数)	Main 文のループの最大数
main_day_max	Value(自然数)	Main 文のシミュレーション日数
tt_start(tt_stop)	Value(自然数)	各日における潮流計算範囲
tt_step	Value(自然数)	計算刻み幅
tt_max	Value(自然数)	計算回数
rootfolderpath	char	結果を出力するフォルダの文字列
foldername_input (foldername_output)	char	プログラム実行時の入力（出力）フォルダ

mat_Table 2 systemsetup.mat

変数名	値(配列の場合：配列数)	内容
sys_B(G)	ノード数×ノード数	ノード間のコンダクタンス、サセプタンス
sys_In(Pn,Vn,Zn)	Value	基準電流、電力、電圧、抵抗
sys_PV_node	PV 数	PV 番号とノード番号の紐付け
sys_b(r,x,bc)	ノード数×ノード数	各ノード間のサセプタンス、抵抗、リアクタンス
sys_dmnd_num	Value(自然数)	住宅件数(1800 軒)
sys_fre	Value	周波数(50)
sys_node_num	Value(自然数)	ノード数(56)
sys_pri_node_num	Value(自然数)	高圧ノード数(5)
sys_residence_node	住宅数	住宅番号とノード番号の紐付け
sys_tapr	ノード数	各ノードにおける柱上変圧器のタップ位置
sys_Z	ノード数×ノード数	ノード間のインピーダンス
sys_BFS	ノード数×ノード×3	潮流計算(BFS 法)に必要な配列(関数の受渡に使用)

EV プラグイン・充電器設定(EVRCset/RC_main)

mat_Table 3 RCdata.mat

変数名	値(配列の場合 : 配列数)	内容
RC_num	Value(自然数)	急速充電器数
RC_cap	充電器台数	充電器容量
EV_RC	Value(自然数)	1 日の顧客(EV)数
EV_battery_cp	EV 数	各 EV の充電容量[kWh]
EV_start(stop)	Value(時間)	充電器利用開始(終了)時間
RC_time_interval	Value(時間)	充電器データ時間刻み幅(単位：秒)
RC_settime_max	Value(自然数)	設定する時間帯の最大
RC_P	時間×充電器×日数	各日・各時間・各充電器の充電電力
RC_Q_max	時間×充電器×日数	各日・各時間・各充電器の最大補償可能な無効電力
RC_node	充電器台数	充電器が接続されているノード番号
EV_SOC_RC_P	時間×2	EV の SOC に対する充電電力(充電器容量を 1)

mat_Table 4 que.mat

変数名	値(配列の場合：配列数)	内容
que_resultlist	充電器台数×7×日数	待ち行列理論に関する計算結果まとめ(詳細は省略)
que_RC_time	充電器台数×日数	各充電器の利用回数
que_real_EVwaittime	日数	実際の平均待ち時間
que_RC_histgram	充電器台数×ヒストグラム範囲	時間毎の充電器利用台数のヒストグラム

PV データ生成(PVdata/PV_main)

mat_Table 5 PVdata.mat

変数名	値(配列の場合：配列数)	内容
PV_data	ノード数×時間×日数	PV 出力データ
PV_time_interval	Value(時間)	PV 出力データ時間刻み幅
PV_settime_max	Value(自然数)	設定する時間帯の最大

負荷設定(demanddata/dem_main)

mat_Table 6 demdata.mat

変数名	値(配列の場合：配列数)	内容
dem_time_imterval	Value(時間)	負荷データ時間刻み幅
dem_settime_max	Value(自然数)	設定する時間帯の最大
dem_loadP(Q)	ノード数×時間	負荷データ(有効・無効電力)

なお，現在は読み込み時間短縮のため，1 日毎に mat ファイル(〇loopload.m)を生成している。

LRT 設定(LRT/LRT_main)

mat_Table 7 LRTdata.mat

変数名	値(配列の場合：配列数)	内容
LRT_ep	Value	LRT 不感帯幅
LRT_Z_target	Value	LRT のインピーダンス整定値
LRT_Sd_th	Value	LRT タップ切替不感帯逸脱量
LRT_Vtarget	Value	目標電圧(低圧換算で 101[V])
LRT_V_max(min)	Value	LRT 最大(最小)送出電圧
LRT_limit_time	Value(時間)	動作時限
LRT_data_list	8 行	LRT の計算に用いる変数リスト

充電器の無効電力制御の設定(RC_control/RCc_main)

mat_Table 8 RC_control.mat

変数名	値(配列の場合：配列数)	内容
RCc_FeedbackGain	充電器台数×ループ数	フィードバックゲイン
RCc_FeedbankTime	充電器台数×ループ数	フィードバック時間(現在は 1)
RCc_Kl	充電器台数	長周期制御定数
RCc_Ks	充電器台数	短周期制御定数
RCc_dQ_limit	充電器台数×ループ数	補償量変化速度
RCc_down_deadband	充電器台数×ループ数	上限不感帯上限値
RCc_up_deadband	充電器台数×ループ数	下限不感帯下限値
RCc_deadband_d	充電器台数×ループ数	上下限不感帯幅
RCc_time_delay	充電器台数	制御遅れ時間
RCc_time_interval	充電器台数	制御量決定間隔
RCc_control_time	時間(+1)×充電器台数×ループ数	制御時間（単位：秒）
RCc_volm_time	時間(+1)×充電器台数×ループ数	電圧測定時間（単位：秒）

SVC の無効電力制御の設定(SVCset/SVC_main)

mat_Table 9 SVC_control.mat

変数名	値(配列の場合 : 配列数)	内容
SVC_num	Value(自然数)	SVC 台数
SVC_node	SVC 台数	SVC の接続ノード
SVC_cap	SVC 台数×ループ数	SVC 容量
SVCc_FeedbackGain	SVC 台数×ループ数	フィードバックゲイン
SVCc_FeedbankTime	SVC 台数×ループ数	フィードバック時間(現在は 1)
SVCc_Kl	Value(SVC 台数)	長周期制御定数
SVCc_Ks	Value(SVC 台数)	短周期制御定数
SVCc_dQ_limit	SVC 台数×ループ数	補償量変化速度
SVCc_down_deadband	SVC 台数×ループ数	上限不感帯上限値
SVCc_up_deadband	SVC 台数×ループ数	下限不感帯下限値
SVCc_deadband_d	SVC 台数×ループ数	上下限不感帯幅
SVCc_time_delay	SVC 台数	制御遅れ時間
SVCc_time_interval	SVC 台数	制御量決定間隔
SVCc_control_time	時間×SVC 台数	制御時間（単位：秒）
SVCc_volm_time	時間×SVC 台数	電圧測定時間（単位：秒）

負荷データの設定(dataset/dat_main)

mat_Table 10 load.mat

変数名	値(配列の場合：配列数)	内容
dset_netloadP(Q)	ノード数×時間×日数	正味の有効電力（無効電力）
dset_load_rand	日数	負荷需要のランダム性を模擬
dset_netloadP(Q)_before	ノード数×時間(60s)×日数	最初(tt=1)の短周期無効電力を計算するために使用

6. mat ファイルの構成(process フォルダ内のプロセス,試算全体の結果)

制御なしの電圧指標(no_control/noc_main)

mat_Table 11 noc_volindex.mat

変数名	値(配列の場合：配列数)	内容
volindex_noc_LRTtap	時間×日数×ループ数	各時間帯の LRT タップ位置
volindex_noc_LRTtap_num	日数×ループ数	各日の LRT タップ切替回数
volindex_noc_vol_cen	日数×ループ数	電圧逼迫度
volindex_noc_vol_fluc	日数×ループ数	電圧変動量
volindex_noc_vol_max	時間×日数×ループ数	系統内の最大電圧
volindex_noc_vol_min	時間×日数×ループ数	系統内の最小電圧
volindex_noc_vol_vio	日数×ループ数	電圧逸脱量

充電器を用いた場合

mat_Table 12 電圧指標(RCp_volindex.mat)

変数名	値(配列の場合：配列数)	内容
volindex_RCp_LRTtap	時間×日数×ループ数	各時間帯の LRT タップ位置
volindex_RCp_LRTtap_num	日数×ループ数	各日の LRT タップ切替回数
volindex_RCp_vol_cen	日数×ループ数	電圧逼迫度
volindex_RCp_vol_fluc	日数×ループ数	電圧変動量
volindex_RCp_vol_max	時間×日数×ループ数	系統内の最大電圧
volindex_RCp_vol_min	時間×日数×ループ数	系統内の最小電圧
volindex_RCp_vol_vio	日数×ループ数	電圧逸脱量

mat_Table 13 無効電力指標(RCp_Qindex.mat)

変数名	値(配列の場合：配列数)	内容
Qindex_RCp_P	時間×充電器×日数	各時間、各充電器の充電電力
Qindex_RCp_Qcapacity_factor	時間×日数×ループ数	無効電力出力範囲に対する利用率
Qindex_RCp_Qmax	時間×充電器×日数	各時間、各充電器の無効電力補償可能量
Qindex_RCp_Qmax_total	時間×日数×ループ数	合計無効電力補償可能量
Qindex_RCp_abssum	時間×日数×ループ数	合計無効電力補償(絶対値)
Qindex_RCp_capacity_factor	時間×日数×ループ数	充電器容量に対する利用率
Qindex_RCp_long_total	時間×日数×ループ数	合計長周期無効電力補償量
Qindex_RCp_short_total	時間×日数×ループ数	合計短周期無効電力補償量
Qindex_RCp_total	時間×日数×ループ数	合計無効電力補償

SVC を用いた場合

mat_Table 14 SVCp_volindex.mat

変数名	値(配列の場合：配列数)	内容
volindex_SVCp_LRTtap	時間×日数×ループ数	各時間帯の LRT タップ位置
volindex_SVCp_LRTtap_num	日数×ループ数	各日の LRT タップ切替回数
volindex_SVCp_vol_cen	日数×ループ数	電圧逼迫度
volindex_SVCp_vol_fluc	日数×ループ数	電圧変動量
volindex_SVCp_vol_max	時間×日数×ループ数	系統内の最大電圧
volindex_SVCp_vol_min	時間×日数×ループ数	系統内の最小電圧
volindex_SVCp_vol_vio	日数×ループ数	電圧逸脱量

mat_Table 15 無効電力の SVCp_Qindex.mat

変数名	値(配列の場合：配列数)	内容
Qindex_SVCp_Qcapacity_factor	時間×日数×ループ数	無効電力出力範囲に対する利用率
Qindex_SVCp_abssum	日数×ループ数	積算無効電力補償量
Qindex_SVCp_long_total	時間×日数×ループ数	合計長周期無効電力補償量
Qindex_SVCp_short_total	時間×日数×ループ数	合計短周期無効電力補償量
Qindex_SVCp_total	時間×日数×ループ数	合計無効電力補償

7. mat ファイルの構成(process フォルダ内のプロセス,1 日毎)

制御なしの場合

mat_Table 16 noc_result.mat

変数名	値(配列の場合：配列数)	内容
noc_LRT_Sd	時間	LRT の積算不感帯逸脱量
noc_LRT_current	時間(複素数)	LRT のバンク電流
noc_LRT_dV	時間	LRT の目標電圧値との偏差
noc_LRT_vol	時間	LRT の推定電圧
noc_vol	時間×ノード数	各時間の電圧

SVC 充電器を用いた場合

mat_Table 17 RCp_result.mat

変数名	値(配列の場合：配列数)	内容
RCp_LRT_Sd	時間	LRT の積算不感帯逸脱量
RCp_LRT_current	時間(複素数)	LRT のバンク電流
RCp_LRT_dV	時間	LRT の目標電圧値との偏差
RCp_LRT_vol	時間	LRT の推定電圧
RCp_Q	時間×充電器台数	各充電器の無効電力補償量
RCp_Q_long	時間×充電器台数	各充電器の長周期無効電力補償量
RCp_Q_short	時間×充電器台数	各充電器の短周期無効電力補償量
RCp_Q_total	時間×ノード数	各ノードにおける全無効電力補償量
RCp_dQ_long	時間×充電器台数	長周期無効電力補償の制御量
RCp_dQ_short_after	時間×充電器台数	短周期無効電力補償の制御量（修正前）
RCp_dQ_short_before	時間×充電器台数	短周期無効電力補償の制御量（修正後）
RCp_dQ_short_range_und	時間×充電器台数	短周期無効電力補償の制御の下限制約
RCp_dQ_short_range_upp	時間×充電器台数	短周期無効電力補償の制御の上限制約
RCp_vol	時間×ノード数	各時間の制御後の電圧
RCp_vol_Qbefore	時間×ノード数	各時間の制御前の電圧(RCp_Qset で使用)
RCp_vol_before	時間×ノード数	各時間の制御前の電圧(測定電圧)

RCp_vol_diff	時間×充電器台数	充電器が測定する電圧変動量
RCp_vol_long	時間×充電器台数	長周期制御を適用する電圧 (=RCp_vol_before)

SVC 充電器を用いた場合

mat_Table 18 SVCp_result.mat

変数名	値(配列の場合：配列数)	内容
SVCp_LRT_Sd	時間	LRT の積算不感帯逸脱量
SVCp_LRT_current	時間(複素数)	LRT のバンク電流
SVCp_LRT_dV	時間	LRT の目標電圧値との偏差
SVCp_LRT_vol	時間	LRT の推定電圧
SVCp_Q	時間×SVC 台数	各 SVC の無効電力補償量
SVCp_Q_long	時間×SVC 台数	各 SVC の長周期無効電力補償量
SVCp_Q_short	時間×SVC 台数	各 SVC の短周期無効電力補償量
SVCp_Q_total	時間×ノード数	各ノードにおける全無効電力補償量
SVCp_dQ_long	時間×SVC 台数	長周期無効電力補償の制御量
SVCp_dQ_short_after	時間×SVC 台数	短周期無効電力補償の制御量（修正前）
SVCp_dQ_short_before	時間×SVC 台数	短周期無効電力補償の制御量（修正後）
SVCp_dQ_short_range_und	時間×SVC 台数	短周期無効電力補償の制御の下限制約
SVCp_dQ_short_range_upp	時間×SVC 台数	短周期無効電力補償の制御の上限制約
SVCp_vol	時間×ノード数	各時間の制御後の電圧
SVCp_vol_Qbefore	時間×ノード数	各時間の制御前の電圧制御 (SVCp_Qset で使用)
SVCp_vol_before	時間×ノード数	各時間の制御前の電圧
SVCp_vol_diff	時間×SVC 台数	SVC が測定する電圧変動量
SVCp_vol_long	時間×SVC 台数	長周期制御を適用する電圧 (=SVCp_vol_before)

8. 各プログラム(m ファイル)の関係図

メインプログラム(root フォルダ)

main.m
├──main_setup.m
│　　├──initialsetup.m
│　　├── m_run.m(各プログラムフォルダへ移動)
├──main_process.m
├──main_output.m
└──takeover(ここに mat ファイルがあれば読み込み.txt)

setup/systemsetup フォルダ内

sys_main.m
├──branch_node.m
│　　├──con_cal.m
├──node_adm_matrix.m
├──tap_set.m

setup/EVRCset フォルダ内

RC_main.m
├──que_initialize.m
├──EVRCset.m
│　　├──ran_set.m
│　　├──ran_set2.m
│　　├──RC_P_set.m
│　　├──que_calc.m

setup/PVdata フォルダ内

PV_main.m
├──PV_insset.m
│　　├──PVdataset.m (PV_fixedset.m)

setup/ demanddata フォルダ内

dem_main.m

setup/ dataset フォルダ内

dat_main.m

setup/ LRTset フォルダ内

LRT_main.m

setup/ RC_control フォルダ内

RCc_main.m
├──K_calc.m

setup/ SVCset フォルダ内

SVC_main.m
├──K_calc.m

process/no_control フォルダ内 (括弧内はフォルダ位置を示す)

noc_main.m
├── (common/)load_flow_BFS.m
├── (common/)LRT_tapset.m
├── noc_process_output.m
├── noc_volindex_calc.m
│ ├──(common/) vol_index_calc.m

process/ RC_control フォルダ内 (括弧内はフォルダ位置を示す)

RCp_main.m
├──RCp_set.m
├──RCp_Qset.m
│ ├──(common/) short_term_Q.m
│ ├──(common/) long_term_Q.m
│ ├──(common/) Q_constraint.m
├── (common/) load_flow_BFS.m
├──RCp_Qset_after.m
├──RCp_volindex_calc.m
│ ├──(common/) vol_index_calc.m
├──RCp_Qindex_calc.m
├──RCp_process_output.m

process/ SVC_control フォルダ内 (括弧内はフォルダ位置を示す)

SVCp_main.m
├──SVCp_set.m
├──SVCp_Qset.m
│　　├──(common/) short_term_Q.m
│　　├──(common/) long_term_Q.m
│　　├──(common/) Q_constraint.m
├── (common/) load_flow_BFS.m
├──SVCp_Qset_after.m
├──SVCp_volindex_calc.m
│　　├──(common/) vol_index_calc.m
├──SVCp_Qindex_calc.m
├──SVCp_process_output.m

process/ output フォルダ内

out_main.m
├──out_volindex.m
│　　├──calc_index.m
├──out_volindex.m

process/ SVC_est フォルダ内

est_main.m

```
%{
%%%%%%%%%%%%%%%%%%
メインプログラム
%%%%%%%%%%%%%%%%%%
%}

%% 各種設定(setup) main_setup;
main_setup;

%% 実際の制御・計算プロセス(process)
main_process; %メインプロセス

%% 出力プロセス
main_output;
```



```
%{
%%%%%%%%%%%%%%%%%%
main.setup(設定プロセス)
%%%%%%%%%%%%%%%%%%
%}

%% 各部分をコメントアウトすることで設定せずに実行可能

clear; %初期化
clc; %画面クリア

%% プログラム全体の初期設定
initialsetup();

%% 系統モデル設定
readtry_filename=strcat('takeover/systemsetup.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','systemsetup','sys_main.m'); %%系統モデル設定
m_run;%引継ファイルの有無チェックと実行

%% 充電器利用データ設定
fprintf('Start EVRCdataset ¥n');
readtry_filename=strcat('takeover/RCdata.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','EVRCset','RC_main.m'); % 充電器利用データの設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish EVRCdataset ¥n');

%% PV出力データ設定
readtry_filename=strcat('takeover/PVdata.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','PVdata','PV_main.m'); % PV出力データの設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish PVdataset ¥n');

%% 需要データ設定
readtry_filename=strcat('takeover/demdata.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','demanddata','dem_main.m'); % 需要データの設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish demandset ¥n');

%% 正味負荷データ設定
readtry_filename=strcat('takeover/load.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','dataset','dat_main.m'); % 負荷データの設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish dataset ¥n');

%% LRTの設定
readtry_filename=strcat('takeover/LRTdata.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','LRTset','LRT_main.m'); %  LRTの設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish LRTsetup ¥n');

%% 充電器制御データの設定
%
readtry_filename=strcat('takeover/RC_control.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','RC_control','RCc_main.m'); % RCによる電圧制御の設定
m_run;%引継ファイルの有無チェックと実行
```

```
fprintf('Finish RCcontrolset¥n');
%}

%% SVC制御データの設定
%
readtry_filename=strcat('takeover/SVC_control.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','SVCset','SVC_main.m'); % SVCによる電圧制御の設定
m_run;%引継ファイルの有無チェックと実行
fprintf('Finish SVCcontrolset¥n');
%}
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%系統初期設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力
clear; %初期化
load('../../initialsetup.mat'); %プログラム全体の初期設定

%% 系統初期設定

%系統関連
sys_node_num=56;
sys_fre=50;%周波数
sys_Pn=10;%基準容量[MVA]
sys_Vn=6.6;%基準電圧[kV]
sys_In=sys_Pn/sys_Vn/sqrt(3)*1000;%基準電流[A]
sys_Zn=sys_Vn/sys_In/sqrt(3)*1000;%基準インピーダンス[Ω]

%高圧負荷データの設定
sys_dmnd_num=1800;%需要家数
sys_pri_node_num=5;%高圧ノード数

save('systemsetup.mat'); %一時保存用

%% 系統データの作成
%{
接続しないノード間は抵抗無限とするため、初期値をinfとする
接続しないノード間はリアクタンス０とするため、初期値を０とする
接続しないノード間はサセプタンス０とするため、初期値を０とする
%}
sys_r=inf(sys_node_num,sys_node_num);%抵抗
sys_x=zeros(sys_node_num,sys_node_num);%インダクタンス
sys_b=zeros(sys_node_num,sys_node_num);%サセプタンス
sys_bc=zeros(sys_node_num,1);%容量サセプタンス

%低圧本線の長さ設定
temp_lle=0.04;%Low Length

%引込線の長さ設定system
temp_sal=0.005;%Short Aerial Length
temp_lal=0.04;%Long Aerial Length

%ブランチノード定数設定
branch_node;

%ノードアドミタンス行列計算
node_adm_matrix;

%タップ比の設定
tap_set;

%% BFSによる潮流計算の高速化に使用
```

```
sys_BFS=zeros(sys_node_num,sys_node_num,3);
sys_BFS(1,1,1)=sys_pri_node_num;
sys_BFS(2,1,1)=sys_Vn;
sys_BFS(3,1,1)=sys_node_num;
sys_BFS(:,2,1)=sys_tapr;
sys_BFS(:,:,2)=sys_r;
sys_BFS(:,:,3)=sys_x;

%% matファイルに保存

savematname=strcat(foldername_input,'systemsetup.mat');
save(savematname,'sys_*');

save('systemsetup.mat','sys_*');%一時保存用
```



```
%ノード設定

%{
con_cal(node1,node2,lsort,leng)
con_calのlsortについて
1.立上りケーブル
2.AL240の高圧線
3.AL120の高圧線
11.柱上変圧器（30,6600タップ、長さは１として入力すること。）
12.柱上変圧器（30,6450タップ、長さは１として入力すること。）
13.柱上変圧器（50,6600タップ、長さは１として入力すること。）
14.柱上変圧器（50,6450タップ、長さは１として入力すること。）
15.柱上変圧器（75,6600タップ、長さは１として入力すること。）
16.柱上変圧器（75,6450タップ、長さは１として入力すること。）
17.柱上変圧器（100,6600タップ、長さは１として入力すること。）
18.柱上変圧器（100,6450タップ、長さは１として入力すること。）
31.AL120の低圧線（6600タップ）
32.AL120の低圧線（6450タップ）
33.SV60の低圧線（6600タップ）
34.SV60の低圧線（6450タップ）
35.引込線DV3.2（6600タップ）
36.引込線DV3.2（6450タップ）

%低圧本線の長さ設定
temp_lle=0.05;%Low Length
%spe=1;special 電圧降下を大きくするためにノード7の部分の低圧線を3倍に

%引込線の長さ設定
temp_sal=0.005;%Short Aerial Length
temp_lal=0.04;%Long Aerial Length
%}

save('rxb.mat');

sys_residence_node=zeros(sys_dmnd_num,1); %住宅番号の特定
sys_PV_node=zeros(sys_dmnd_num,1); %PV番号の特定

%高圧線
con_cal(1,2,2,1);%#S～#A
con_cal(2,3,2,1);%#A～#sys_B
con_cal(3,4,2,1);%#B～#C
con_cal(4,5,3,1);%#C～#D
con_cal(5,6,3,1);%#D～#E

Primary_count=1;
 for node_num=7:9:7+9*4
            Primary_count=Primary_count+1;
            if Primary_count<=3
                con_cal(Primary_count,node_num,15,1);%#A～#Tr2
            else
                con_cal(Primary_count,node_num,16,1);%#A～#Tr2
            end
            con_cal(node_num,node_num+1,31,temp_lle);%#Tr1～#Tp1
            con_cal(node_num+1,node_num+2,31,temp_lle);%#Tp1～#Tp2
            con_cal(node_num,node_num+3,35,temp_sal);%#Tr1～#a-1
```

```
            con_cal(node_num,node_num+4,35,temp_lal);%#Tr1～#sys_b-1
            con_cal(node_num+1,node_num+5,35,temp_sal);%#Tp1～#c-1
            con_cal(node_num+1,node_num+6,35,temp_lal);%#Tp1～#d-1
            con_cal(node_num+2,node_num+7,35,temp_sal);%#Tp2～#e-1
            con_cal(node_num+2,node_num+8,35,temp_lal);%#Tp2～#f-1
 end

%% 住宅番号、PV番号とノードの関連付け

r_count=1;
PV_count=1;
Primary_count=2;
 for node_num=10:9:10+9*4
     for r_num=node_num:node_num+5 %実際に計算するノード
            sys_residence_node(r_count,1)=r_num;
            sys_PV_node(r_count,1)=r_num;
            r_count=r_count+1;
     end
     for r_num=node_num-3:node_num-1 %実際に計算するノード
            sys_residence_node(r_count,1)=r_num;
            sys_PV_node(r_count,1)=r_num;
            r_count=r_count+1;
            sys_residence_node(r_count,1)=r_num;
            sys_PV_node(r_count,1)=r_num;
            r_count=r_count+1;
     end
     for r_num=1:12*29 %その他のノード %r_numは一時変数
            sys_residence_node(r_count,1)=Primary_count;
            sys_PV_node(r_count,1)=Primary_count;
            r_count=r_count+1;
     end
     Primary_count=Primary_count+1;
 end

%% 計算確認のため、高圧ノードで負荷のみの計算も別フィーダで実施

con_cal(1,52,2,1);%#S～#A'
con_cal(52,53,2,1);%#A'～#sys_B'
con_cal(53,54,2,1);%#sys_B'～#C'
con_cal(54,55,3,1);%#C'～#D'
con_cal(55,56,3,1);%#D'～#E'
```



```
function con_cal(node1,node2,lsort,leng)

%変数設定
load('systemsetup.mat')
load('rxb.mat')

%グローバル変数宣言
%global sys_Zn sys_r sys_x sys_b sys_fre

%ブランチノード定数
CVT325r=0.07623;%CVT325の正相抵抗[Ω/km]
CVT325x=0.09540;%CVT325の正相リアクタンス[Ω/km]
CVT325c=0.61;%CVT325のキャパシタンス0.61[μF/km]
AL240r=0.12632;%AL240の正相抵抗[Ω/km]
AL240x=0.30928;%AL240の正相リアクタンス[Ω/km]
AL240c=0;%AL240のキャパシタンス[μF/km]
AL120r=0.25265;%AL120の正相抵抗は0.25265[Ω/km]
AL120x=0.34848;%AL120の正相リアクタンスは0.34848[Ω/km]
AL120c=0;%AL120のキャパシタンス[μF/km]
SV60r=0.36100;%SV60の正相抵抗
SV60x=0.07980;%SV60の正相リアクタンス
SV60c=0;%SV60のキャパシタンス[μF/km]
HK32r=2.3;%引込線の正相抵抗[Ω/km]
HK32x=0.094;%引込線の正相リアクタンス[Ω/km]
HK32c=0;%引込線のキャパシタンス[μF/km]
TR30r=0.02205;%柱上トランス30の抵抗[Ω]
TR30x=0.03131;%柱上トランス30の正相リアクタンス[Ω]
TR30c=0;%柱上トランス30のキャパシタンス[μF]
TR50r=0.01185;%柱上トランス50の抵抗[Ω]
TR50x=0.01566;%柱上トランス50の正相リアクタンス[Ω]
TR50c=0;%柱上トランス50のキャパシタンス[μF]
TR75r=0.00806;%柱上トランス75の抵抗[Ω]
TR75x=0.01699;%柱上トランス75の正相リアクタンス[Ω]
TR75c=0;%柱上トランス75のキャパシタンス[μF]
TR100r=0.00569;%柱上トランス100の抵抗[Ω]
TR100x=0.01203;%柱上トランス100の正相リアクタンス[Ω]
TR100c=0;%柱上トランス100のキャパシタンス[μF]

%タップ比
tap660=6600/210;
tap645=6450/210;

%単相なので3分の2倍
sph=2/3;%single phase

%ΔＹ変換の関係で3分の1倍
dlt_y=1/3;

%{
con_cal(node1,node2,lsort,leng)
con_calのlsortについて

1.立上りケーブル
2.AL240の高圧線
3.AL120の高圧線
```

```
11.柱上変圧器（30,6600タップ、長さは１として入力すること。）
12.柱上変圧器（30,6450タップ、長さは１として入力すること。）
13.柱上変圧器（50,6600タップ、長さは１として入力すること。）
14.柱上変圧器（50,6450タップ、長さは１として入力すること。）
15.柱上変圧器（75,6600タップ、長さは１として入力すること。）
16.柱上変圧器（75,6450タップ、長さは１として入力すること。）
17.柱上変圧器（100,6600タップ、長さは１として入力すること。）
18.柱上変圧器（100,6450タップ、長さは１として入力すること。）
31.AL120の低圧線（6600タップ）
32.AL120の低圧線（6450タップ）
33.SV60の低圧線（6600タップ）
34.SV60の低圧線（6450タップ）
35.引込線DV3.2（6600タップ）
36.引込線DV3.2（6450タップ）
%}

switch lsort
    case 1
%{
        sys_r(node1,node2)=CVT325r*leng/sys_Zn;
        sys_r(node2,node1)=CVT325r*leng/sys_Zn;
        sys_x(node1,node2)=CVT325x*leng/sys_Zn;
        sys_x(node2,node1)=CVT325x*leng/sys_Zn;
        sys_b(node1,node2)=2*pi*sys_fre*CVT325c*leng*sys_Zn/1000000;
        sys_b(node2,node1)=2*pi*sys_fre*CVT325c*leng*sys_Zn/1000000;
%}
    case 2
        sys_r(node1,node2)=AL240r*leng/sys_Zn;
        sys_r(node2,node1)=AL240r*leng/sys_Zn;
        sys_x(node1,node2)=AL240x*leng/sys_Zn;
        sys_x(node2,node1)=AL240x*leng/sys_Zn;
        sys_b(node1,node2)=2*pi*sys_fre*AL240c*leng*sys_Zn/1000000;
        sys_b(node2,node1)=2*pi*sys_fre*AL240c*leng*sys_Zn/1000000;
    case 3
        sys_r(node1,node2)=AL120r*leng/sys_Zn;
        sys_r(node2,node1)=AL120r*leng/sys_Zn;
        sys_x(node1,node2)=AL120x*leng/sys_Zn;
        sys_x(node2,node1)=AL120x*leng/sys_Zn;
        sys_b(node1,node2)=2*pi*sys_fre*AL120c*leng*sys_Zn/1000000;
        sys_b(node2,node1)=2*pi*sys_fre*AL120c*leng*sys_Zn/1000000;
    case 11
        sys_r(node1,node2)=tap660^2*TR30r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap660^2*TR30r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap660^2*TR30x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap660^2*TR30x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*TR30c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*TR30c*leng*sys_Zn/1000000/dlt_y;
    case 12
        sys_r(node1,node2)=tap645^2*TR30r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap645^2*TR30r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap645^2*TR30x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap645^2*TR30x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*TR30c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*TR30c*leng*sys_Zn/1000000/dlt_y;
    case 13
```



```
        sys_r(node1,node2)=tap660^2*TR50r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap660^2*TR50r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap660^2*TR50x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap660^2*TR50x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*TR50c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*TR50c*leng*sys_Zn/1000000/dlt_y;
    case 14
        sys_r(node1,node2)=tap645^2*TR50r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap645^2*TR50r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap645^2*TR50x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap645^2*TR50x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*TR50c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*TR50c*leng*sys_Zn/1000000/dlt_y;
    case 15
        sys_r(node1,node2)=tap660^2*TR75r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap660^2*TR75r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap660^2*TR75x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap660^2*TR75x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*TR75c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*TR75c*leng*sys_Zn/1000000/dlt_y;
    case 16
        sys_r(node1,node2)=tap645^2*TR75r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap645^2*TR75r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap645^2*TR75x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap645^2*TR75x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*TR75c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*TR75c*leng*sys_Zn/1000000/dlt_y;
    case 17
        sys_r(node1,node2)=tap660^2*TR100r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap660^2*TR100r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap660^2*TR100x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap660^2*TR100x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*TR100c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*TR100c*leng*sys_Zn/1000000/dlt_y;
    case 18
        sys_r(node1,node2)=tap645^2*TR100r*leng/sys_Zn*dlt_y;
        sys_r(node2,node1)=tap645^2*TR100r*leng/sys_Zn*dlt_y;
        sys_x(node1,node2)=tap645^2*TR100x*leng/sys_Zn*dlt_y;
        sys_x(node2,node1)=tap645^2*TR100x*leng/sys_Zn*dlt_y;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*TR100c*leng*sys_Zn/1000000/dlt_y;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*TR100c*leng*sys_Zn/1000000/dlt_y;
    case 31
        sys_r(node1,node2)=tap660^2*AL120r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap660^2*AL120r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap660^2*AL120x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap660^2*AL120x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*AL120c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*AL120c*leng*sys_Zn/1000000/sph;
    case 32
        sys_r(node1,node2)=tap645^2*AL120r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap645^2*AL120r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap645^2*AL120x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap645^2*AL120x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*AL120c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*AL120c*leng*sys_Zn/1000000/sph;
```

```
    case 33
        sys_r(node1,node2)=tap660^2*SV60r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap660^2*SV60r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap660^2*SV60x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap660^2*SV60x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*SV60c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*SV60c*leng*sys_Zn/1000000/sph;
    case 34
        sys_r(node1,node2)=tap645^2*SV60r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap645^2*SV60r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap645^2*SV60x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap645^2*SV60x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*SV60c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*SV60c*leng*sys_Zn/1000000/sph;
    case 35
        sys_r(node1,node2)=tap660^2*HK32r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap660^2*HK32r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap660^2*HK32x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap660^2*HK32x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap660^(-2)*2*pi*sys_fre*HK32c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap660^(-2)*2*pi*sys_fre*HK32c*leng*sys_Zn/1000000/sph;
    case 36
        sys_r(node1,node2)=tap645^2*HK32r*leng/sys_Zn*sph;
        sys_r(node2,node1)=tap645^2*HK32r*leng/sys_Zn*sph;
        sys_x(node1,node2)=tap645^2*HK32x*leng/sys_Zn*sph;
        sys_x(node2,node1)=tap645^2*HK32x*leng/sys_Zn*sph;
        sys_b(node1,node2)=tap645^(-2)*2*pi*sys_fre*HK32c*leng*sys_Zn/1000000/sph;
        sys_b(node2,node1)=tap645^(-2)*2*pi*sys_fre*HK32c*leng*sys_Zn/1000000/sph;
    otherwise
        disp('系統データ作成時にエラーが発生したため中断！');
        return

end
sys_Z(node1,node2)=sqrt(sys_r(node1,node2)*sys_r(node1,node2)+sys_x(node1,node2)*sys_x(node1,node2));

%変数保存
save('rxb.mat','sys_r','sys_x','sys_b','sys_Z')
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ノードアドミッタンス行列作成
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

load('rxb.mat');

temp_Y=zeros(sys_node_num,sys_node_num);%アドミッタンス行列
sys_G=zeros(sys_node_num,sys_node_num);%実数部
sys_B=zeros(sys_node_num,sys_node_num);%虚数部

for jj=1:sys_node_num
    for ii=1:sys_node_num

        summ=0;temp=0;%テンポラリ変数初期化

        if(ii==jj)%(i=j)の場合
            for jj2=1:sys_node_num
                if(ii~=jj2)%自ノード除去
                    temp=1/(sys_r(ii,jj2)+1i*sys_x(ii,jj2))+1i*sys_b(ii,jj2)/2;
                    summ=summ+temp;
                end
            end

            temp_Y(ii,jj)=summ+1i*sys_bc(ii);
        end

        if(ii~=jj)%(i≠j)の場合
            temp_Y(ii,jj)=-1/(sys_r(ii,jj)+1i*sys_x(ii,jj));
        end
    end
end


%ノードアドミタンス行列各成分
sys_G=real(temp_Y);
sys_B=imag(temp_Y);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 タップごとの低圧換算値設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%global sys_tapr

%定数設定
tap1=6600;
tap2=6450;
taplow=105;
sys_tapr=zeros(sys_node_num,1);%tap_ratio


%タップ比設定
for ii=1:6
    sys_tapr(ii,1)=taplow/tap1;
end

for ii=7:15
    sys_tapr(ii,1)=taplow/tap1;
end

for ii=16:24
    sys_tapr(ii,1)=taplow/tap1;
end

for ii=25:33
    sys_tapr(ii,1)=taplow/tap1;
end

for ii=34:42
    sys_tapr(ii,1)=taplow/tap2;
end

for ii=43:51
    sys_tapr(ii,1)=taplow/tap2;
end

for ii=52:sys_node_num
    sys_tapr(ii,1)=taplow/tap2;
end
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 急速充電器(Rapid Charger:RC)データの設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期設定と入力データ
clear;
load('../../initialsetup.mat');

%% 急速充電器(Rapid Charger:RC_num)データの設定

% データ時間関係
RC_time_interval=60;
temp_per_hour=RC_time_interval/(60*60);
RC_settime_max=1440;

% 充電器充電電力関係
RC_num=30;%急速充電器数1
RC_area_max=5; %エリア数(通常は5)
RC_node=zeros(RC_num,1); %急速充電器の接続ノード
RC_area=zeros(RC_num,1); %エリア情報

RC_cap=zeros(RC_num,1); %急速充電器の容量
RC_cap(:,1)=30;

EV_RC=150; %顧客(EV)数[台/日]

EV_battery_cp=zeros(EV_RC,1);
EV_battery_cp(:,1)=18; %各EVの充電容量[kWh]

EV_start=1; %EVの充電器利用開始可能時間
EV_stop=1440; %EVの充電器利用可能最終時間

%% 各充電器の配置・エリア分類

for ii=1:RC_num
    RC_node(ii,1)=floor((ii-1)/floor(RC_num/5))+2; %均等配置
    %RC_node(ii,1)=floor((ii-1)/floor(RC_num/2))+5; %#A,#Bに集中

    %RC_area(ii,1)=1; %パターンを固定させる場合
    RC_area(ii,1)=floor((ii-1)/floor(RC_num/5))+1; %通常
end

%% 各急速充電器のデータの抽出・保存

%各データの抽出
load(strcat(cd,'¥input¥EV¥EV_SOC_RC_P.csv'),'-ascii'); %EVのSOCと充電器の充電電力の関係
load(strcat(cd,'¥input¥EV¥s_EV_SOC.csv'),'-ascii');    %EVのSOC決定の累積確率密度関数
load(strcat(cd,'¥input¥RC_num¥s_patern.csv'),'-ascii');    %パターン決定の累積確率密度関数
load(strcat(cd,'¥input¥RC_num¥s_RC.csv'),'-ascii');        %各パターンの累積確率密度関数

%matファイルへの(一時)保存
save('EVRCdata.mat','RC_*','EV_*');
save('s_EVRCdata.mat','s_*');
```

```
RC_P=zeros(RC_settime_max,RC_num,main_day_max); %充電器の充電電力
RC_Q_max=zeros(RC_settime_max,RC_num,main_day_max); %充電器の無効電力補償可能量(絶対値)

%% 待ち行列計算の初期設定
que_initialize; %待ち行列の初期設定

%% 充電器の利用状況の模擬

for day=1:main_day_max
    [RC_P(:,:,day),RC_Q_max(:,:,day)]=EVRCset(day); %充電器利用状況設定
end

%% matファイルの保存

matname=strcat(foldername_input,'RCdata.mat');
save(matname,'RC_*','EV_*'); %出力
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 各EVの設定・充電器の有効および無効電力のプロファイルの設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

% EVの充電モデルに合わせてコメントアウトを切替して利用

function [ RC_P,RC_Q_max ] = EVRCset(day)
%% データ入力および初期設定
load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

load('EVRCdata.mat');
load('s_EVRCdata.mat');
load('que.mat');

%% 出力変数
RC_P=zeros(RC_settime_max,RC_num);
RC_Q_max=zeros(RC_settime_max,RC_num);

%% 一時変数
temp_EV_RCarrive_time=zeros(EV_RC,1); %EVの充電器到着時間
temp_RC_ave=zeros(RC_num,3); %充電器が接続されているEV番号(0:利用可能)
temp_EV_SOC=zeros(EV_RC,1);                 %各EVのSOC
temp_EV_SOC_t=zeros(RC_settime_max,EV_RC);  %各EVのSOCの推移
temp_RC_ave_t=zeros(RC_settime_max,RC_num); %各充電器の利用EV番号
temp_RC_num_used=zeros(RC_settime_max,1);       %各充電器の利用台数
temp_RC_sum_P=zeros(RC_settime_max,1);      %全充電器の充電電力
temp_RC_sum_Q=zeros(RC_settime_max,1);      %全充電器の補償可能無効電力
temp_EV_RC_info=zeros(EV_RC,5);             %各EVの充電結果一覧
temp_EV_pat=zeros(EV_RC,1);     %各EVのパターン
temp_RC_area_pat=zeros(RC_area_max,1);  %各充電器(エリア毎)のパターン
temp_RC_pat_area=ones(numel(s_patern(:,1)),1); %全てのパターンが選択されたかどうかを判断

%% エリア毎・充電器パターンの決定
%ランダム
%
for jj=1:RC_area_max
    c=1;
    s_RC_area_pat=zeros(1,2);
    for ii=1:numel(s_patern(:,1))
        if temp_RC_pat_area(ii,1)==1
            s_RC_area_pat(c,1)=ii;
            if c==1
                s_RC_area_pat(c,2)=s_patern(ii,1);
            else
                s_RC_area_pat(c,2)=s_RC_area_pat(c-1,2)+s_patern(ii,1);
            end
            c=c+1;
        end
    end
    s_RC_area_pat(:,2)=s_RC_area_pat(:,2)/max(s_RC_area_pat(:,2)); %規格化
```

```
    if jj>numel(s_patern(:,1))
        temp_RC_area_pat(jj,1)=ran_set(s_patern,numel(s_patern(:,1)));
    else
        temp_RC_area_pat(jj,1)=ran_set(s_RC_area_pat,numel(s_RC_area_pat(:,1)));
    end
    temp_RC_pat_area(temp_RC_area_pat(jj,1),1)=0;
end
%}

%固定
%temp_RC_area_pat(1,1)=2;

%% 各EVの充電器(もしくはエリア)先の決定

for ii=1:EV_RC
    %EV_mileage(ii,1)=ran_set(s_EV_mileage,numel(s_EV_mileage(:,1)));

    temp_EV_SOC(ii,1)=ran_set(s_EV_SOC,numel(s_EV_SOC(:,1)));

    %プラグインする充電器を先に決定する方法
    %{
    X=rand;
    temp_EV_RC_info(ii,1)=ceil(X*RC_num);
    temp_EV_RCarrive_time(ii,1)=ran_set2(s_RC(:,temp_RC_area_pat(RC_area(temp_EV_RC_info(ii)))),numel ↙
(s_RC(:,temp_RC_area_pat(RC_area(temp_EV_RC_info(ii))))));
    %}

    %エリアのみを先に決定する方法
    %
    %temp_EV_pat(ii,1)=ran_set(s_patern,numel(s_patern(:,1)));
    temp_EV_pat(ii,1)=2;
    temp_EV_RCarrive_time(ii,1)=ran_set2(s_RC(:,temp_EV_pat(ii,1)),numel(s_RC(:,temp_EV_pat(ii,1))));

    %}
end

%% 各EVの充電器の割当

for tt=EV_start:EV_stop

    % プラグインするEVの決定
    if min(temp_EV_RCarrive_time)<=tt
       for ii=1:EV_RC

            %% プラグインする充電器を先に決定する方法
            %{

            if temp_EV_SOC(ii,1)<1.0 && temp_RC_ave(temp_EV_RC_info(ii,1),1)==0 && ↙
temp_EV_RCarrive_time(ii,1)<=tt %充電器が空きかどうか
                temp_RC_ave(temp_EV_RC_info(ii,1),1)=ii; %充電器の割当
                temp_RC_ave(temp_EV_RC_info(ii,1),2)=temp_RC_area_pat(RC_area(temp_EV_RC_info(ii,1)));
                temp_EV_RC_info(ii,2)=tt;
                temp_EV_RC_info(ii,4)=tt-temp_EV_RCarrive_time(ii,1); %充電スタンド待ち時間
                temp_RC_ave(temp_EV_RC_info(ii,1),3)=temp_RC_ave(temp_EV_RC_info(ii,1),3)+1;
```



```
                if  temp_time_on<=tt && tt<=temp_time_off
                    temp_RC_num(temp_EV_RC_info(ii,1),day)=temp_RC_num(temp_EV_RC_info(ii,1),day) ↙
+1; %EV数(利用者数)
                end
            end
            %}

            %% エリアのみを先に決定する方法
            %

            if temp_EV_SOC(ii,1)<1.0 && min(abs(temp_RC_ave(:,1)))==0 && temp_EV_RC_info(ii,1)==0 %充電↙
器が空きかどうか
                % (エリア決定方法) 空いている充電器からランダムで決定
                c=1;
                s_RC_ave=zeros(1,2);
                for jj=1:RC_num
                    if temp_RC_ave(jj,1)==0 && temp_RC_area_pat(RC_area(jj,1))==temp_EV_pat(ii,1)

                        s_RC_ave(c,1)=jj;
                        if c==1
                            s_RC_ave(c,2)=s_RC(tt,temp_RC_area_pat(RC_area(jj,1)));
                        else
                            s_RC_ave(c,2)=s_RC_ave(c-1,2)+s_RC(tt,temp_RC_area_pat(RC_area(jj,1)));
                        end
                        c=c+1;
                    end
                end
                RC_ave_num=c-1;
                s_RC_ave(:,2)=s_RC_ave(:,2)/max(s_RC_ave(:,2)); %規格化

                if temp_EV_RCarrive_time(ii,1)<=tt && temp_EV_SOC(ii,1)<1.0 && s_RC_ave(1,1)~=0
                    RC_plagin_num = ran_set( s_RC_ave,RC_ave_num );
                    temp_RC_ave(RC_plagin_num,1)=ii; %充電器の割当
                    temp_RC_ave(RC_plagin_num,2)=temp_EV_pat(ii,1);


                    if  temp_time_on<=tt && tt<=temp_time_off
                        temp_RC_num(RC_plagin_num,day)=temp_RC_num(RC_plagin_num,day)+1; %EV数(利用者↙
数)
                    end

                    temp_EV_RC_info(ii,1)=RC_plagin_num;
                    temp_EV_RC_info(ii,2)=tt;
                    temp_EV_RC_info(ii,4)=tt-temp_EV_RCarrive_time(ii,1); %充電スタンド待ち時間
                    temp_RC_ave(RC_plagin_num,3)=temp_RC_ave(RC_plagin_num,3)+1;
                end
            end
            %}
        end
    end
    %% 充電電力および無効電力の決定

    for ii=1:RC_num
        if temp_RC_ave(ii,1)>0 %充電器が空きかどうか
            if  temp_time_on<=tt && tt<=temp_time_off
```

```
                que_RC_time(ii,day)=que_RC_time(ii,day)+1;
            end

            RC_P(tt,ii)=RC_P_set(temp_EV_SOC(temp_RC_ave(ii,1),1),RC_cap(ii,1),EV_SOC_RC_P); ↙
%充電器の充電電力設定
            temp_EV_SOC(temp_RC_ave(ii,1),1)=temp_EV_SOC(temp_RC_ave(ii,1),1)+RC_P(tt,ii) ↙
*temp_per_hour/EV_battery_cp(temp_RC_ave(ii,1),1); %SOCの変化量設定
            if temp_EV_SOC(temp_RC_ave(ii,1),1)>=1.0
                temp_EV_RC_info(temp_RC_ave(ii,1),3)=tt;
                temp_EV_RC_info(temp_RC_ave(ii,1),5)=tt-temp_EV_RC_info(temp_RC_ave(ii,1),2); %充電時間
                temp_RC_ave(ii,1)=-3; %待ち時間
            end
        elseif temp_RC_ave(ii,1)<0
            temp_RC_ave(ii,1)=temp_RC_ave(ii,1)+1; %待ち時間考慮
            RC_P(tt,ii)=0;
        else
            RC_P(tt,ii)=0;
        end
        RC_Q_max(tt,ii)=sqrt(RC_cap(ii,1)*RC_cap(ii,1)-RC_P(tt,ii)*RC_P(tt,ii));

    end

    temp_RC_ave_t(tt,:)=temp_RC_ave(:,1);
    temp_RC_num_used(tt,:)=sum(nnz(temp_RC_ave(:,1)));
    temp_EV_SOC_t(tt,:)=temp_EV_SOC(:,1);

    temp_RC_sum_P(tt,1)=sum(RC_P(tt,:));
    temp_RC_sum_Q(tt,1)=sum(RC_Q_max(tt,:));
end

%% 待ち行列理論の適用と充電器割当結果の出力
que_calc;
%
filename=strcat(foldername_input,num2str(day),'day/');
mkdir(filename);
EVRCset_output;
%}
end
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 待ち行列理論に関する初期設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 保存する変数の設定
que_resultlist=zeros(RC_num,7,main_day_max); %待ち行列理論に関する計算の結果まとめ
que_RC_time=zeros(RC_num,main_day_max); %各充電器の利用回数
que_real_EVwaittime=zeros(1,main_day_max); %実際の待ち時間

%% 一時変数
temp_time_off=841; % 最後の到着時間(待ち行列の計算範囲)              (resultlistの1)
temp_time_on=601;  % 最初の想定到着時間                              (resultlistの2)
temp_RC_num=zeros(RC_num,main_day_max); %EV数(利用者数)       (resultlistの3)
temp_ramuda=zeros(RC_num,main_day_max); %平均到着率           (resultlistの4)
temp_myu=zeros(RC_num,main_day_max);  %平均サービス(充電)時間 (resultlistの5)
temp_row=zeros(RC_num,main_day_max);  %平均利用率             (resultlistの6)
temp_Tw=zeros(RC_num,main_day_max);   %平均待ち時間            (resultlistの7)

temp_calc_dt=10;%実際の待ち時間の刻み幅
temp_calc_max=floor((temp_time_off-temp_time_on)/temp_calc_dt); %計算最大値
temp_calc_count=zeros(temp_calc_max,main_day_max+1);    %ヒストグラム作成のためのカウンタ

%% ヒストグラムの初期化と時間軸データの追加

que_RC_histgram=zeros(temp_calc_max,main_day_max+1);        %ヒストグラム
for tt=1:temp_calc_max
    que_RC_histgram(tt,1)=(tt-1)*temp_calc_dt+temp_time_on;
    temp_calc_count(tt,1)=(tt-1)*temp_calc_dt+temp_time_on;
end

save('que.mat','que_*','temp_*'); %データの保存
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 待ち行列理論の計算
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

% 各変数についてはque_initialize参照

temp_ramuda(:,day)=temp_RC_num(:,day)./(temp_time_off-temp_time_on);
temp_myu(:,day)=temp_RC_num(:,day)./que_RC_time(:,day);
temp_row(:,day)=temp_ramuda(:,day)./temp_myu(:,day);
temp_Tw(:,day)=temp_row(:,day)./(1-temp_row(:,day))./temp_myu(:,day);

%temp_Tw(isnan(temp_Tw(:,day)),day)=0;

que_resultlist(:,1,day)=temp_time_on;
que_resultlist(:,2,day)=temp_time_off;
que_resultlist(:,3,day)=temp_RC_num(:,day);
que_resultlist(:,4,day)=temp_ramuda(:,day);
que_resultlist(:,5,day)=temp_myu(:,day);
que_resultlist(:,6,day)=temp_row(:,day);
que_resultlist(:,7,day)=temp_Tw(:,day);

for ii=1:EV_RC
   if temp_EV_RC_info(ii,2)>=temp_time_on && temp_EV_RC_info(ii,2)<=temp_time_off
       que_real_EVwaittime(1,day)=que_real_EVwaittime(1,day)+temp_EV_RC_info(ii,4);
   end
   for tt=1:numel(que_RC_histgram(:,1))-1
       if temp_EV_RC_info(ii,2)>=que_RC_histgram(tt,1) && temp_EV_RC_info(ii,2)<=que_RC_histgram(tt+1, ↙
1)
            que_RC_histgram(tt,day+1)=que_RC_histgram(tt,day+1)+temp_EV_RC_info(ii,4);
            temp_calc_count(tt,day+1)=temp_calc_count(tt,day+1)+1;
       end
   end

end
que_real_EVwaittime(1,day)=que_real_EVwaittime(1,day)/sum(temp_RC_num(:,day));
que_RC_histgram(:,day+1)=que_RC_histgram(:,day+1)./temp_calc_count(:,day+1);

matname=strcat(foldername_input,'que.mat');
save(matname,'que_*'); %データの保存(calc_は一時変数のため削除)
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
充電器割当結果の出力
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 充電器割当結果の出力

filename1=strcat(foldername_input,num2str(day),'day/temp_EV_RC_info.csv');
filename2=strcat(foldername_input,num2str(day),'day/RC_Q_max.csv');
filename3=strcat(foldername_input,num2str(day),'day/RC_num.csv');

csvwrite(filename1,temp_EV_RC_info); % 計算結果の書き込み
csvwrite(filename2,RC_Q_max); % 計算結果の書き込み
csvwrite(filename3,temp_RC_ave); % 計算結果の書き込み
```

```
function output = ran_set( data,data_num )
%乱数設定
% 入力：乱数に用いる変数（１列目：値、2列目：累積確率）、出力：変数値
    X = rand;%乱数生成

    for ii=1:data_num
       if X<=data(ii,2)
            output = data(ii,1);
            break;
       end
    end
    if X>=data(data_num)
        output = data(data_num,1);
    end
end
```



```
function output = ran_set2( data,data_num )
%乱数設定
% 入力：乱数に用いる変数（１列目：値、2列目：累積確率）、出力：行番号
    X = rand;%乱数生成

    for ii=1:data_num
       if X<data(ii,1)
            output = ii;
            break;
       end
    end
    if X>=data(data_num)
        output = data_num;
    end
end
```

```
function [ output ] = RC_P_set( temp_EV_SOC,RC_cap,EV_SOC_RC_P )
%UNTITLED この関数の概要をここに記述
%   詳細説明をここに記述

    for ii=1:numel(EV_SOC_RC_P(:,1))
        if temp_EV_SOC<EV_SOC_RC_P(ii,1)
            output=RC_cap*EV_SOC_RC_P(ii,2);
            break;
        end
    end
    if temp_EV_SOC >= EV_SOC_RC_P(numel(EV_SOC_RC_P(:,1)),1)
            output=RC_cap*EV_SOC_RC_P(numel(EV_SOC_RC_P(:,1)),2);
    end
end
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 PVデータの設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ
clear;
load('../../initialsetup.mat');
loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

%% PV出力データの生成
PV_time_interval=60;    %1分しか対応していません
PV_settime_max=1440;    %1440分しか対応していません

PV_data=zeros(sys_node_num,PV_settime_max,main_day_max);
PV_insset; % PVの導入箇所の決定

for day=1:main_day_max
        temp_rand=1.0; %時間の経過によるランダム性
        temp_rand_cont=1.0;       %現時間のランダム性

        PV_fuc_t=floor((840-600)*rand+600); %出力が変動する時間帯
        Amp=1.0; %ピーク比(1.0)

        for tt=1:PV_settime_max

            %午前、午後でPV変動量が大きく変化させる場合
            %
            if tt==1 || tt==PV_fuc_t
                Ax=0.8*rand+0.2; %変動振幅
            end
            %}

            temp_rand_now=temp_rand_cont; %前時間とのランダムの継続性
            temp_rand=Ax*rand+(1-Ax); %時間によるランダム性
            temp_rand_cont=0.7*temp_rand+0.3*temp_rand_now; %現時間のランダム性

            if rand<0.01 %PV出力突然性を模擬
                if temp_rand_cont>0.5
                    temp_rand_cont=temp_rand_cont-0.3;
                else
                    temp_rand_cont=temp_rand_cont+0.3;
                end
            end
            temp_PVoutput(:,tt)=Amp*(temp_rand_cont*1.0+0)*PVoutput(:,tt);
        end

        PV_data(:,:,day)=temp_PVoutput(:,:);
end

matname=strcat(foldername_input,'PVdata.mat');
save(matname,'PV_data','PV_time_interval','PV_settime_max');
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PVの導入箇所を決定し、ノード換算のPV出力を出力
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}
function [ PVoutput ] = PV_fixedset( PV_ins_num )
%PVdataset PVの導入箇所を決定し、ノード換算のPV出力を出力
%   入力は導入数
%% 初期設定等
load('../../initialsetup.mat');
loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

    load('PV_data.mat');%PVデータの読み込み
    %PV導入数>残りPV導入箇所の場合：全てに導入
    if PV_ins_num>=numel(temp_PV_ins_ave)
        PV_ins_num=numel(temp_PV_ins_ave);
    end
    PVoutput=zeros(sys_node_num,PV_settime_max); %node換算のPV出力

    %% PVの導入箇所の決定(導入リストの生成)（固定リスト）
        temp_PV_ins_list(int16(numel(temp_PV_ins_list))+1:int16(numel(temp_PV_ins_list)+PV_ins_num),1) ↙
=temp_PV_ins_ave(1:int16(PV_ins_num),1); %リストの生成
        temp_PV_ins_ave(1:int16(PV_ins_num))=[];%PV導入可能変数から「導入が決定したPV番号」を取り除く
        temp_PV_ins_list=sort(temp_PV_ins_list); %リストの並べ替え

    %% PV出力値の決定
    for ii=2:numel(temp_PV_ins_list)
        PV_ins_primary=1+floor((temp_PV_ins_list(ii)-1)/360); %高圧ノードの特定
        PV_ins_secondary=rem(temp_PV_ins_list(ii),12); %低圧ノードの特定
        if(PV_ins_secondary==0)
            PV_ins_secondary=12;
        end
        PV_ins_Tr=floor(rem(temp_PV_ins_list(ii)-1,360)/12)+1;

        if PV_ins_Tr==max_Tr(PV_ins_primary) && PV_ins_secondary <= 6
            PVoutput((PV_ins_primary-1)*9+PV_ins_secondary+9,:) = transpose(PVoutputdata(:,1) ↙
/sys_Pn/1000);
        elseif PV_ins_Tr==max_Tr(PV_ins_primary) && PV_ins_secondary >= 7 && PV_ins_secondary <= 8
            PVoutput((PV_ins_primary-1)*9+7,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
        elseif PV_ins_Tr==max_Tr(PV_ins_primary) && PV_ins_secondary >= 9 && PV_ins_secondary <= 10
            PVoutput((PV_ins_primary-1)*9+8,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
        elseif PV_ins_Tr==max_Tr(PV_ins_primary) && PV_ins_secondary >= 11 && PV_ins_secondary <= 12
            PVoutput((PV_ins_primary-1)*9+9,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
        else
            PVoutput(PV_ins_primary+1,:) = PVoutput(PV_ins_primary+1,:) + transpose(PVoutputdata(:,1) ↙
/sys_Pn/1000);
        end
    end
    save('PV_data.↙
mat','PV_node','temp_PV_ins_info','temp_PV_ins_ave','PVoutputdata','sys_Pn','temp_PV_count','temp_PV_in↙
s_list');
end
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 PVの設置個所・出力の設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% PV出力データの初期設定

load(strcat(cd,'¥input¥PVoutputdata.csv'),'-ascii') %理想的なPV出力データ(kW)の読み込み
PVoutput=zeros(sys_node_num,PV_settime_max); %node換算のPV出力
temp_PV_max=1800; %PVの最大導入数(住宅1800軒と同じ)


%% PV導入箇所の決定の初期設定

temp_PV_count=zeros(30,5); %各高圧変圧器のPV導入数
temp_PV_ins_list=[]; %PV導入リスト

temp_PV_ins_info=zeros(5,12);
temp_PV_ins_ave=transpose(1:1800);

save('PV_data.mat','temp_*','PV*');

temp_PV_pen_in=80; %PV導入割合 [%]
temp_PV_ins_num=temp_PV_pen_in/100*1800; %PV導入数

[ PVoutput ] = PVdataset( temp_PV_ins_num ); %PV導入箇所設定
%[ PVoutput ] = PV_fixedset( temp_PV_ins_num ); %PV導入箇所設定（固定）昇順設定
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PVの導入箇所を決定し、ノード換算のPV出力を出力
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

function [ PVoutput ] = PVdataset( PV_ins_num )
%PVdataset PVの導入箇所を決定し、ノード換算のPV出力を出力
%   入力は導入数
%% 初期設定等
load('../../initialsetup.mat');
loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname,'sys_Pn','sys_node_num');  %系統データの読込

    load('PV_data.mat');%PVデータの読み込み

    %PV導入数>残りPV導入箇所の場合：全てに導入
    if PV_ins_num>=numel(temp_PV_ins_ave)
        PV_ins_num=numel(temp_PV_ins_ave);
    end
    PVoutput=zeros(sys_node_num,PV_settime_max); %node換算のPV出力

    %% PVの導入箇所の決定(導入リストの生成)

    for ii=1:PV_ins_num %1台毎に導入決定

        temp_PV_ins_ave_num=numel(temp_PV_ins_ave); %PV導入可能数
        X = rand;%乱数生成

        temp_PV_ins_temp=floor(X*temp_PV_ins_ave_num)+1; %PVの設置個所(PV_No)の決定

        if temp_PV_ins_temp>numel(temp_PV_ins_ave)
            temp_PV_ins_temp=numel(temp_PV_ins_ave);
        end
        temp_PV_ins_bum=temp_PV_ins_ave(temp_PV_ins_temp,1); %導入が決定したPV番号

        temp_PV_ins_ave(temp_PV_ins_temp)=[]; %PV導入可能変数から「導入が決定したPV番号」を取り除く

        %既に低圧ノードに接続されている場合を考慮した処理
        temp_PV_ins_primary=floor((temp_PV_ins_bum-1)/360)+1; %高圧ノードの特定
        temp_PV_ins_secondary=rem(temp_PV_ins_bum,12); %低圧ノードの特定

        if temp_PV_ins_secondary==0
            temp_PV_ins_secondary=12;
        end
        temp_PV_ins_Tr=floor(rem(temp_PV_ins_bum-1,360)/12)+1;
        temp_PV_count(temp_PV_ins_Tr,temp_PV_ins_primary)=temp_PV_count(temp_PV_ins_Tr, ↙
temp_PV_ins_primary)+1;

        [max_number,max_Tr]=max(temp_PV_count);

        temp_PV_ins_list(numel(temp_PV_ins_list)+1,1)=temp_PV_ins_bum; %リストの生成
        temp_PV_ins_list=sort(temp_PV_ins_list); %リストの並べ替え
    end
```



```
    %%  PV出力値の決定
        %ここの部分は、PV導入リストとPVノード情報(sys_PV_node)が一致しないようにしているため以下の処理↙
を行う。
        %導入リストにおいて一番厳しい電圧を算出するために、一番導入されている低圧系統を抽出させて潮流計↙
算の系統に組み込ませる。
        for ii=2:numel(temp_PV_ins_list)

            temp_PV_ins_primary=1+floor((temp_PV_ins_list(ii)-1)/360); %高圧ノードの特定
            temp_PV_ins_secondary=rem(temp_PV_ins_list(ii),12); %低圧ノードの特定
            if(temp_PV_ins_secondary==0)
                temp_PV_ins_secondary=12;
            end
            temp_PV_ins_Tr=floor(rem(temp_PV_ins_list(ii)-1,360)/12)+1;

            if temp_PV_ins_Tr==max_Tr(temp_PV_ins_primary) && temp_PV_ins_secondary <= 6
                PVoutput((temp_PV_ins_primary-1)*9+temp_PV_ins_secondary+9,:) = transpose(PVoutputdata ↙
(:,1) /sys_Pn/1000);
            elseif temp_PV_ins_Tr==max_Tr(temp_PV_ins_primary) && temp_PV_ins_secondary >= 7 && ↙
temp_PV_ins_secondary <= 8
                PVoutput((temp_PV_ins_primary-1)*9+7,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
            elseif temp_PV_ins_Tr==max_Tr(temp_PV_ins_primary) && temp_PV_ins_secondary >= 9 && ↙
temp_PV_ins_secondary <= 10
                PVoutput((temp_PV_ins_primary-1)*9+8,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
            elseif temp_PV_ins_Tr==max_Tr(temp_PV_ins_primary) && temp_PV_ins_secondary >= 11 && ↙
temp_PV_ins_secondary <= 12
                PVoutput((temp_PV_ins_primary-1)*9+9,:) = transpose(PVoutputdata(:,1) /sys_Pn/1000);
            else
                PVoutput(temp_PV_ins_primary+1,:) = PVoutput(temp_PV_ins_primary+1,:) + transpose ↙
(PVoutputdata(:,1) /sys_Pn/1000);
            end

        end
    save('PV_data.mat','temp_*','PV*');
end
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 負荷データの設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ
clear;
load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

load(strcat(cd,'¥input¥daydata.csv'),'-ascii')%配電線全体の住宅負荷
dem_time_imterval=60;
dem_settime_max=1440;
temp_PF=0.9;    %力率
dem_loadP=zeros(sys_node_num,dem_settime_max); %有効電力負荷
dem_loadQ=zeros(sys_node_num,dem_settime_max); %無効電力負荷

%% 負荷設定
% 各住宅、各時間でランダム
%
for ii=1:1800
    temp_rand=1.0; %場所によるランダム性を考慮する場合(現状は固定)
    dem_loadP(sys_residence_node(ii,1),:) = dem_loadP(sys_residence_node(ii,1),:) -  transpose( ↙
temp_rand*daydata(:,1)/sys_Pn/1000 ) ;
    dem_loadQ(sys_residence_node(ii,1),:) = dem_loadQ(sys_residence_node(ii,1),:) -  transpose( ↙
temp_rand*daydata(:,1)*sqrt(1-temp_PF*temp_PF) /sys_Pn/1000 ) ;
end

%別ブランチ設定(電圧確認用)

for ii=52:56
    dem_loadP(ii,:) = dem_loadP(ii,:) -200*(1.0)*transpose( daydata(:,1)/sys_Pn/1000 ) ;
    dem_loadQ(ii,:) = dem_loadQ(ii,:) -200*(1.0)*transpose( daydata(:,1)*sqrt(1-temp_PF*temp_PF) ↙
/sys_Pn/1000 ) ;
end

matname=strcat(foldername_input,'demdata.mat');
save(matname,'dem_*');
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 正味負荷データの設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ
clear;
load('../../initialsetup.mat');
loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込
loadmatname=strcat(foldername_input,'demdata.mat');
load(loadmatname);  %住宅負荷(P,Q)を入力
loadmatname=strcat(foldername_input,'PVdata.mat');
load(loadmatname);      %PVデータを入力
loadmatname=strcat(foldername_input,'RCdata.mat');
load(loadmatname);      %充電器データを入力

dset_netloadP_before=zeros(sys_node_num,60,main_day_max); %制御前の有効電力(tt=1よりも前)
dset_netloadQ_before=zeros(sys_node_num,60,main_day_max); %制御前の有効電力(tt=1よりも前)

dset_netloadP=zeros(sys_node_num,tt_max,main_day_max); %制御前の有効電力
dset_netloadQ=zeros(sys_node_num,tt_max,main_day_max); %制御前の無効電力

%% 各日のランダム値の引継または計算
readtrymatname=strcat('rand_list.mat');
if exist(readtrymatname,'file')==2
    load(readtrymatname,'dset_load_rand');
else
    dset_load_rand(1:main_day_max,1)=rand(main_day_max,1)*0.6+0.6; %ランダム性を考慮
    %dset_load_rand(day,1)=1;%負荷ランダム考慮しない場合
end

%% メインプロセス
for day=1:main_day_max

    %read_loop_day=87; %使用するデータを変えない場合(充電器データのみ変化)
    read_loop_day=day; %日付ループ毎に使用するデータを変える場合

    if day==1 && day~=read_loop_day
        fprintf ('Caution!! ¥n Only %d day demand&PV data is run. ¥n¥n pless any key¥n',read_loop_day);↙
%警告
    end

    node_RC_P=zeros(RC_settime_max,sys_node_num); %各ノードの充電電力

    for ii=1:RC_num
        node_RC_P(:,RC_node(ii,1))=node_RC_P(:,RC_node(ii,1))+RC_P(:,ii,day);
    end

    tt_ini=tt_start/tt_step; %時刻データ(tt_start)を数値に変換
    tt_end=tt_stop/tt_step; %時刻データ(tt_stop)を数値に変換

    temp_PV_tt=ceil(tt_start/PV_time_interval);
    temp_RC_tt=ceil(tt_start/RC_time_interval);
```

```
    timecount=1;

    for tt=1:tt_max
        % 電圧下限逸脱として扱う場合は負荷を3倍に設定する
        dset_netloadP(:,tt,day)=dset_load_rand(read_loop_day,1)*dem_loadP(:,temp_PV_tt);
        dset_netloadP(:,tt,day)=dset_netloadP(:,tt,day)+PV_data(:,temp_PV_tt,read_loop_day)-node_RC_P ↙
(temp_RC_tt,:)'/sys_Pn/1000;%要修正

        dset_netloadQ(:,tt,day)=dset_load_rand(read_loop_day,1)*dem_loadQ(:,temp_PV_tt);

        if rem(timecount,PV_time_interval)==1
            temp_PV_tt=temp_PV_tt+1;
        end

        if rem(timecount,RC_time_interval)==1
            temp_RC_tt=temp_RC_tt+1;
        end
        timecount=timecount+tt_step;
    end

    timecount=1;
    temp_PV_tt=ceil(tt_start/PV_time_interval);
    temp_RC_tt=ceil(tt_start/RC_time_interval);
    if temp_PV_tt>1 && temp_RC_tt>1
        for tt=1:60

            temp_dem_tt=floor((tt_start-(tt-1)*tt_step)/tt_step);


            dset_netloadP_before(:,tt,day)=dset_load_rand(read_loop_day,1)*dem_loadP(:,temp_dem_tt);
            dset_netloadP_before(:,tt,day)=dset_netloadP_before(:,tt,day)+PV_data(:,temp_PV_tt, ↙
read_loop_day)-node_RC_P(temp_RC_tt,:)'/sys_Pn/1000;%要修正

            dset_netloadQ_before(:,tt,day)=dset_load_rand(read_loop_day,1)*dem_loadQ(:,temp_dem_tt);

            if rem(timecount,PV_time_interval)==1
                temp_PV_tt=temp_PV_tt-1;
            end

            if rem(timecount,RC_time_interval)==1
                temp_RC_tt=temp_RC_tt-1;
            end
            timecount=timecount+tt_step;
        end

        dset_netloadP_before(2:6,:,day)=1.2*dset_netloadP_before(2:6,:,day);  %従来負荷を1.2倍
    end
    dset_netloadP(2:6,:,day)=1.2*dset_netloadP(2:6,:,day);  %従来負荷を1.2倍
end

matname=strcat(foldername_input,'load.mat');
save(matname,'dset_*'); %負荷データの保存

%% 日毎にmatファイルを分割・生成(〇dayload.mat)
```



```
for day=1:main_day_max
    dset_day_netloadP=dset_netloadP(:,:,day);
    dset_day_netloadP_before=dset_netloadP_before(:,:,day);
    dset_day_netloadQ=dset_netloadQ(:,:,day);
    dset_day_netloadQ_before=dset_netloadQ_before(:,:,day);

    matname=strcat(foldername_input,num2str(day),'dayload.mat');
    save(matname,'dset_day*');
end
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 LRTに関する初期設定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ

clear; %初期化
load('../../initialsetup.mat'); %プログラム全体の初期設定
load('LRTrxb.mat'); %系統インピーダンス読込

%% LRTに関する初期設定

LRT_ep=0.0125/2; %不感帯幅
LRT_Z_target=sys_r(1,2)+sys_r(2,3)+sys_r(3,4)+1i*(sys_x(1,2)+sys_x(2,3)+sys_x(3,4)); %整定値
%LRT_Z_target=sys_r(1,2)+sys_r(2,3)+sys_r(3,4)+sys_r(4,5)+1i*(sys_x(1,2)+sys_x(2,3)+sys_x(3,4)+sys_x ↙
(4,5)); %整定値

LRT_Sd_th=0.03; %LRTタップ切替不感帯逸脱量
LRT_Vtarget=0.962; %目標電圧(低圧換算で101[V]
LRT_V_initial=0.975; %LRTの初期送出電圧

LRT_V_max=1.05; %LRT最大送出電圧
LRT_V_min=0.90; %LRT最小送出電圧
LRT_limit_time=10*60;       %動作時限

%% LRTtapsetで用いる配列リスト

LRT_data_list=zeros(8,1);
LRT_data_list(1,1)=LRT_ep;
LRT_data_list(2,1)=LRT_Z_target;
LRT_data_list(3,1)=LRT_Sd_th;
LRT_data_list(4,1)=LRT_Vtarget;
LRT_data_list(5,1)=LRT_V_max;
LRT_data_list(6,1)=LRT_V_min;
LRT_data_list(7,1)=LRT_limit_time;

savematname=strcat(foldername_input,'LRTdata.mat');
save(savematname,'LRT_*');
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%
%RCの制御に関する初期設定
%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ
clear;
load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込
loadmatname=strcat(foldername_input,'RCdata.mat');
load(loadmatname);

%% RCの電圧制御に関する初期設定

RCc_time_interval(1,1:RC_num,1:main_loop_max)=60; %各RCの制御間隔
RCc_time_delay(1,1:RC_num,main_loop_max)=0;
temp_RCc_time_max=ceil((tt_stop-tt_start)/min(min(RCc_time_interval(1,:,:))));
RCc_volm_time=zeros(temp_RCc_time_max+1,RC_num,main_loop_max);
RCc_control_time=zeros(temp_RCc_time_max+1,RC_num,main_loop_max);

for loop=1:main_loop_max
    for ii=1:RC_num
        temp_RCc_time=ceil((tt_stop-tt_start)/RCc_time_interval(1,ii,loop));
        RCc_volm_time(1:temp_RCc_time,ii,loop)=(1:RCc_time_interval(1,ii,loop):tt_stop-tt_start)';
        RCc_control_time(1:temp_RCc_time,ii,loop)=RCc_volm_time(1:temp_RCc_time,ii,loop)+RCc_time_delay ↙
(1,ii,loop);
    end
end

%% 共通項
RCc_dQ_limit(1:RC_num,1:main_loop_max)=30; %出力変化制約

%% 短周期パラメータ
temp_RCc_s_node=RC_node; %短周期を実施する充電器のノード情報(通常は充電器の各ノードと同じ)

RCc_FeedbackGain=zeros(RC_num,main_loop_max); %フィードバックゲイン
RCc_FeedbackGain(:,:)=0.1; %一定

RCc_FeedbankTime(1:RC_num,1:main_loop_max)=1; %フィードバック時間(現在は1)

%% 長周期パラメータ
temp_RCc_l_node=RC_node; %長周期を実施する充電器のノード情報
temp_RCc_l_node(1:floor(3/5*RC_num))=0; %長周期を実施しない充電器は0に設定

temp_node_up_deadband=[inf;inf;inf;102.5;103];
temp_node_down_deadband=[0;0;0;96;95];
for ii=1:RC_num
    RCc_up_deadband(ii,1:main_loop_max)=temp_node_up_deadband(RC_node(ii)-1);
    RCc_down_deadband(ii,1:main_loop_max)=temp_node_down_deadband(RC_node(ii)-1);
end
RCc_deadband_d(1:RC_num,1:main_loop_max)=0.5; %不感帯幅
RCc_deadband_d(19:24,:)=0.3;
```

```
%% モード切替
temp_control_mode=3; %1:短周期のみ 2:長周期のみ　それ以外:両制御
if temp_control_mode==1 %長周期制御無効ノード
    temp_node_up_deadband=[inf;inf;inf;inf;inf];
    temp_node_down_deadband=[0;0;0;0;0];
elseif temp_control_mode==2
    temp_RCc_s_node(:)=0; %短周期制御無効モード
end

%% 各種制御定数の算出

temp_Vlow=0.95; %適正範囲下限値
RCc_Ks=zeros(RC_num,1); %短周期制御定数
[RCc_Ks]=K_calc(RC_num,temp_RCc_s_node,temp_Vlow);
RCc_Kl=zeros(RC_num,1); %長周期制御定数
[RCc_Kl]=K_calc(RC_num,temp_RCc_l_node,temp_Vlow);
savematname=strcat(foldername_input,'RC_control.mat');
save(savematname,'RCc_*');

%% 制御定数一覧表出力
temp_RC_label=(1:RC_num)';
temp_RCc_table=table(temp_RC_label,RCc_FeedbackGain,RCc_FeedbankTime,RCc_dQ_limit,RCc_up_deadband, ↙
RCc_down_deadband,...
    RCc_deadband_d,RCc_Ks,RCc_Kl,...
    'VariableNames', ↙
{'RC_No','FeedbackGain','FeedbackTime','dQ_limit','up_deadband','down_deadband','band_d',...
    'Ks','Kl'});
writetable(temp_RCc_table,strcat(foldername_input,'RCc_list.xlsx')); % 計算結果の書き込み
```



```
function [ K ] = K_calc( total_num,each_node,Vlow )
%   短周期および長周期制御定数の算出
%   RCにもSVCにも適用可能

load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname,'sys_Pn'); %系統データの読込

load('rxb.mat');

temp_eachnode_num=zeros(5,1); %制御実施台数カウント
for ii=2:6
    temp_eachnode_num(ii-1)=nnz(each_node==ii);
end
control_effect_num=zeros(6,6);
for ii=2:6
    control_effect_num(ii-1,ii)=0; %iiとii-1ノード間に影響を与える台数
    for jj=ii:6
        control_effect_num(ii-1,ii)=control_effect_num(ii-1,ii)+temp_eachnode_num(jj-1);
    end
end

K=zeros(total_num,1);
for ii=1:total_num
    temp_r=0;
    temp_x=0;
    if each_node(ii)~=0
        for jj=2:each_node(ii)
            temp_r=temp_r+control_effect_num(jj-1,jj)*sys_r(jj-1,jj);
            temp_x=temp_x+control_effect_num(jj-1,jj)*sys_x(jj-1,jj);
        end
        temp_r=1/Vlow*temp_r*105/sys_Pn/1000;
        temp_x=1/Vlow*temp_x*105/sys_Pn/1000;
        K(ii,1)=sqrt(temp_r^2+temp_x^2);
    else
        K(ii,1)=inf;
    end
end
end
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%
SVCの制御に関する初期設定
%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データ

clear;

load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname,'sys_Pn','sys_node_num');  %系統データの読込

%% SVC特有の設定
SVC_num=1;  %SVC台数
SVC_node=6; %SVCが接続されているノード
SVC_cap=zeros(SVC_num,main_loop_max);
%SVC_cap(:,:)=900; %容量
%SVCc_dQ_limit=SVC_cap; %出力変化制約

%% SVCの電圧制御に関する初期設定

SVCc_time_interval(1,1:SVC_num)=60; %各SVCの制御間隔
SVCc_time_delay(1,1:SVC_num)=0;

temp_SVCc_time=ceil((tt_stop-tt_start)/min(SVCc_time_interval(1,:,:)));
SVCc_volm_time=zeros(temp_SVCc_time,SVC_num);
SVCc_control_time=zeros(temp_SVCc_time,SVC_num);

for ii=1:SVC_num
    temp_SVCc_time=ceil((tt_stop-tt_start)/SVCc_time_interval(1,ii));
    SVCc_volm_time(1:temp_SVCc_time,ii)=(1:SVCc_time_interval(1,ii):tt_stop-tt_start)';
    SVCc_control_time(1:temp_SVCc_time,ii)=SVCc_volm_time(1:temp_SVCc_time,ii)+SVCc_time_delay(1,ii);
end

%% 短周期パラメータ
temp_SVCc_s_node=SVC_node; %短周期を実施する充電器のノード情報(通常は充電器の各ノードと同じ)

SVCc_FeedbackGain=zeros(SVC_num,main_loop_max);
SVCc_FeedbackGain(:,1:main_loop_max)=0.10;

SVCc_FeedbankTime(1:SVC_num,1:main_loop_max)=1;

%% 長周期パラメータ
temp_SVCc_l_node=SVC_node; %長周期を実施する充電器のノード情報
%temp_SVCc_l_node(1:floor(2/5*SVC_num))=0; %長周期を実施しない充電器は0に設定

temp_node_up_deadband=[inf;inf;inf;102.5;103];
temp_node_down_deadband=[0;0;0;96;95];

for ii=1:SVC_num
    SVCc_up_deadband(ii,1:main_loop_max)=temp_node_up_deadband(SVC_node(ii)-1);
    SVCc_down_deadband(ii,1:main_loop_max)=temp_node_down_deadband(SVC_node(ii)-1);
```



```
end
SVCc_deadband_d(1:SVC_num,1:main_loop_max)=0.5; %不感帯幅

%% ループ毎にパラメータを変更

SVC_cap(1,1:main_loop_max)=750/(main_loop_max)*(1:main_loop_max); %loop毎に容量を変更する場合
SVCc_dQ_limit=SVC_cap; %出力変化制約

%フィードバックゲイン変更する場合
%SVCc_FeedbackGain(:,1:main_loop_max)=1/(main_loop_max-1)*(0:main_loop_max-1);

%% モード切替

temp_control_mode=3; %1:短周期のみ 2:長周期のみ  それ以外:両制御
if temp_control_mode==1 %長周期制御無効モード
    SVCc_up_deadband=[inf;inf;inf;inf;inf];
    SVCc_down_deadband=[0;0;0;0;0];
elseif temp_control_mode==2
    temp_SVCc_s_node(:)=0; %短周期制御無効モード
end

%% 各種制御定数の算出

temp_Vlow=0.95; %適正範囲下限値

SVCc_Ks=zeros(SVC_num,1);   %短周期制御定数
[SVCc_Ks]=K_calc(SVC_num,temp_SVCc_s_node,temp_Vlow);

SVCc_Kl=zeros(SVC_num,1);   %長周期制御定数
[SVCc_Kl]=K_calc(SVC_num,temp_SVCc_l_node,temp_Vlow);

savematname=strcat(foldername_input,'SVC_control.mat');
save(savematname,'SVC*');

%% 制御定数一覧表出力

temp_SVC_label=(1:SVC_num)';
temp_SVCc_table=table(temp_SVC_label,SVC_cap,SVCc_FeedbackGain,SVCc_FeedbankTime,SVCc_dQ_limit, ↙
SVCc_up_deadband,SVCc_down_deadband,...
    SVCc_deadband_d,SVCc_time_delay',SVCc_time_interval',SVCc_Ks,SVCc_Kl,...
    'VariableNames',↙
{'SVC_No','SVC_cap','FeedbackGain','FeedbackTime','dQ_limit','up_deadband','down_deadband','band_d',...
    'time_delay','time_interval','Ks','Kl'});
writetable(temp_SVCc_table,strcat(foldername_input,'SVCc_list.xlsx')); % 計算結果の書き込み
```

```
function [ K ] = K_calc( total_num,each_node,Vlow )
%   短周期および長周期制御定数の算出
%   RCにもSVCにも適用可能(20161122)

load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname,'sys_Pn'); %系統データの読込

load('rxb.mat');

temp_eachnode_num=zeros(5,1); %制御実施台数カウント
for ii=2:6
    temp_eachnode_num(ii-1)=nnz(each_node==ii);
end

control_effect_num=zeros(6,6);
for ii=2:6
    control_effect_num(ii-1,ii)=0; %iiとii-1ノード間に影響を与える台数
    for jj=ii:6
        control_effect_num(ii-1,ii)=control_effect_num(ii-1,ii)+temp_eachnode_num(jj-1);
    end
end

K=zeros(total_num,1);
for ii=1:total_num
    temp_r=0;
    temp_x=0;
    if each_node(ii)~=0
        for jj=2:each_node(ii)
            temp_r=temp_r+control_effect_num(jj-1,jj)*sys_r(jj-1,jj);
            temp_x=temp_x+control_effect_num(jj-1,jj)*sys_x(jj-1,jj);
        end
        temp_r=1/Vlow*temp_r*105/sys_Pn/1000;
        temp_x=1/Vlow*temp_x*105/sys_Pn/1000;
        K(ii,1)=sqrt(temp_r^2+temp_x^2);
    else
        K(ii,1)=inf;
    end
end

end
```



```
%{
%%%%%%%%%%%%%%%%%%
main_process(プロセス部分)
%%%%%%%%%%%%%%%%%%
%}

%% プロセスに関する準備
clc; %画面クリア
warning('off','all') %tablewriteによるエラー警告の非表示
addpath(strcat(pwd,'/process/common')) %共通部分のフォルダを実行パスに追加

loop_time=tic;looptime=toc(loop_time);
loop_day_time=tic;loopdaytime=toc(loop_day_time);

%% 計算結果フォルダの生成
load('initialsetup.mat');
mkdir(rootfolderpath,'output/loop_result');

%% 計算プロセス
for loop=1:main_loop_max

    loopfoldername=strcat('loop',num2str(loop));%outputフォルダの名前(loop毎のデータ格納)
    fprintf('main process¥n %d loop¥n',loop);
    loop_day_time=tic; %計算時間計測

    for loop_day=1:main_day_max

        load('initialsetup.mat');
        dayfoldername=strcat(rootfolderpath,'output/',loopfoldername,'/',num2str(loop_day),'day'); %出↙
カフォルダの生成
        mkdir(dayfoldername);

       %% 制御なしの計算(loopで変わる要素がないため, 計算はloop=1のみ)

        if loop==1
            %
            fprintf('%dday no control ¥n',loop_day);
            m_run_name=fullfile('process','no_control','noc_main.m');
            run(m_run_name);
            %}
        else
            t1 = datetime('now','Format','yyyy/MM/dd HH:mm:ss'); %現在時間の取得
            t = t1 + seconds(main_loop_max-loop)*looptime/loop + seconds(main_day_max-loop_day) ↙
*loopdaytime/loop_day; %計算終了時間の推定
            disp(t);
        end

        %% 充電器制御の計算
        %
            % RC制御の計算
            fprintf('%dday RC control ¥n',loop_day);
            m_run_name=fullfile('process','RC_control','RCp_main.m');
            run(m_run_name);
        %}
```

```
        %% SVC制御の計算
        %
            fprintf('%dday SVC control ¥n',loop_day);
            m_run_name=fullfile('process','SVC_control','SVCp_main.m');
            run(m_run_name);
        %}

        %% 計算経過状態画面出力
        clc; %画面クリア
        loopdaytime=toc(loop_day_time);
        fprintf('¥n %d loop %d day finish ¥n',loop,loop_day);
        fprintf('about %5.1fseconds/loop¥n',loopdaytime/loop_day);
        %
        t1 = datetime('now','Format','yyyy/MM/dd HH:mm:ss'); %現在時間の取得
        t = t1 + seconds(main_day_max-loop_day)*loopdaytime/loop_day; %計算終了時間の推定
        disp(t);
        %}

        %% 各日フォルダの処理
        %
            dayfoldername=strcat(num2str(loop_day),'day');
            zipname=strcat(rootfolderpath,'output/',loopfoldername,'/',num2str(loop_day),'day'); %出力↙
フォルダの生成
            zipfoldername=strcat(rootfolderpath,'output/',loopfoldername,'/');

            if exist(strcat(zipfoldername,dayfoldername),'file')==7
                zip(zipname,dayfoldername,zipfoldername); %フォルダ圧縮(zip形式)
                rmdir(zipname,'s'); %フォルダ削除
            end
        %}
    end

    %% 表示関連
    looptime=toc(loop_time);
    fprintf('¥n %d loop finish ¥n',loop);
    fprintf('Estimated completion time¥n');
    t1 = datetime('now','Format','yyyy/MM/dd HH:mm:ss'); %現在時間の取得
    t = t1 + seconds(main_loop_max-loop)*looptime/loop; %計算終了時間の推定
    disp(t);

    %% 各ループフォルダの処理
    %
    zipname=strcat(rootfolderpath,'output/loop_result/',loopfoldername);
     zipfoldername=strcat(rootfolderpath,'output/');

     if exist(strcat(zipfoldername,loopfoldername),'dir')==7
         zip(zipname,loopfoldername,zipfoldername); %フォルダ圧縮(zip形式)
         rmdir(strcat(rootfolderpath,'output/',loopfoldername),'s'); %フォルダ削除
     end
    %}
end
warning('on','all') %tablewriteによるエラー警告の非表示の解除
rmpath(strcat(pwd,'/process/common')) %共通部分のフォルダを実行パスから削除
```



```
function [ dQ_short,dQ_short_range_und,dQ_short_range_upp,dQ_long] = Q_constraint( dQ_limit, ↙
RCp_Q_short,Q_short_before,Q_long,Q_long_before,Q_max )
%UNTITLED7 短周期無効電力制御量の修正
%   詳細説明をここに記述

%% 各制御の変化上下限制約
dQ_short_range_und=-dQ_limit;
dQ_short_range_upp=dQ_limit;

%% 短周期&長周期制御
dQ_short=RCp_Q_short-Q_short_before;
dQ_long=Q_long-Q_long_before;

%% 各制御の補償範囲の決定
if abs(Q_long)>=Q_max
    dQ_long=sign(Q_long)*Q_max-Q_long_before;
end
if abs(dQ_long)>=dQ_limit %長周期制御の変化速度制約
    dQ_long=sign(dQ_long)*dQ_limit;

    if sign(dQ_long)>0
        dQ_short_range_upp=0; %増加は認めない
    else
        dQ_short_range_und=0; %減少は認めない
    end
end

Q_s_upm=Q_max-(Q_long_before+dQ_long)-Q_short_before; %短周期成分の余裕分
Q_s_dwm=-Q_max-(Q_long_before+dQ_long)-Q_short_before; %短周期成分の余裕分
if dQ_short_range_upp>Q_s_upm
    dQ_short_range_upp=Q_s_upm;
end
if dQ_short_range_und<Q_s_dwm
    dQ_short_range_und=Q_s_dwm;
end

% 出力変化制約による範囲制約
if dQ_short_range_upp>dQ_limit-dQ_long &&  dQ_long>=0
    dQ_short_range_upp=dQ_limit-dQ_long;
end
if dQ_short_range_und<-dQ_limit-dQ_long &&  dQ_long<=0
    dQ_short_range_und=-dQ_limit-dQ_long;
end

%補償方向による制約
%
if dQ_short*dQ_long>0 %% 補償方向が同じ
    if dQ_long>0 && max(dQ_short,dQ_long)-dQ_long<dQ_short_range_upp
        dQ_short_range_upp=max(dQ_short,dQ_long)-dQ_long;
    end
    if dQ_long<0 && min(dQ_short,dQ_long)-dQ_long>dQ_short_range_und
        dQ_short_range_und=min(dQ_short,dQ_long)-dQ_long;
    end
end
%}
```

```
%各成分の補償量の決定
%短周期
if dQ_short>dQ_short_range_upp
    dQ_short=dQ_short_range_upp;
elseif dQ_short<dQ_short_range_und
    dQ_short=dQ_short_range_und;
end

end
```



```
function [ Out_shortQ ] = short_term_Q( dV,shortQ,Ks,FGain,FTime )
% 短周期制御

dQ=dV/Ks;

Out_shortQ=(1-FGain).*shortQ+dQ;

end
```

```
function [ LRTtap,LRTtap_num,vol_cen,vol_fluc,vol_max,vol_min,vol_vio ] = vol_index_calc( LRT_vol, ↙
node_vol )
%UNTITLED11 電圧指標値の計算

    load('../../initialsetup.mat');

    LRTtap=LRT_vol(1:tt_max,1);

    LRTtap_num=nnz(diff(LRT_vol(:)));

    vol_cen=mean(sqrt(mean((node_vol(1:tt_max,1:51)-101).*(node_vol(1:tt_max,1:51)-101),2)));

    V_diff_squre=diff(node_vol(:,2:6)).*diff(node_vol(:,2:6));

    vol_fluc=sum(mean(V_diff_squre(10:tt_max-1,:)));

    vol_max=max(node_vol(:,7:51),[],2);
    vol_min=min(node_vol(:,7:51),[],2);

    vol_vio=sum(max(max(vol_max-107,95-vol_min),0)*tt_step/60);

end
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 潮流計算
  ここでは、１スラックノード
  複数負荷ノードで計算
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

function [Out,LRT_current]=load_flow_BFS(netloadP1,netloadQ1,V_LRT,sys_BFS)
%% 潮流計算の初期設定

sys_pri_node_num=sys_BFS(1,1,1);
sys_Vn=sys_BFS(2,1,1);
sys_node_num=sys_BFS(3,1,1);
sys_tapr=sys_BFS(:,2,1);
sys_r=sys_BFS(:,:,2);
sys_x=sys_BFS(:,:,3);

%% 潮流計算
Vss =V_LRT*sys_Vn;
dV=ones(sys_node_num,1);
V=ones(sys_node_num,1);

nodeP=zeros(sys_node_num,1);
nodeQ=zeros(sys_node_num,1);

while(norm(dV,inf)>0.0001)
    %% 各計算の初期設定(電力)
    P=zeros(sys_node_num,1);
    Q=zeros(sys_node_num,1);
    dP=zeros(sys_node_num,sys_node_num);
    flowP=zeros(sys_node_num,sys_node_num);
    flowQ=zeros(sys_node_num,sys_node_num);

    nodeP(:)=-1.0*netloadP1(:); %各ノードで直接消費/注入される電力
    nodeQ(:)=-1.0*netloadQ1(:);

    beforeV=V;
    V=abs(V);

    %% 低圧ノードの計算(Backward)

    for ii=1:sys_pri_node_num
        l_n=15+9*(ii-1); %住宅ノード
        for jj=3:-1:1%低圧接点ノード
            h_n=6+9*(ii-1)+jj;
            for kk=1:2 %住宅ノード-低圧接点ノード間
                P(l_n)=nodeP(l_n);
                Q(l_n)=nodeQ(l_n);
                P(h_n)=P(l_n)+sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
                Q(h_n)=Q(l_n)+sys_x(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);  %%修正あり
                dP(h_n,l_n)=sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);

                flowP(h_n,l_n)=P(h_n);
                flowQ(h_n,l_n)=Q(h_n);
```

```
            l_n=l_n-1;
        end
        P(h_n)=sum(flowP(h_n,:))+nodeP(h_n);
        Q(h_n)=sum(flowQ(h_n,:))+nodeQ(h_n);

    end

    for jj=2:-1:1 %低圧接点ノード
        l_n=7+9*(ii-1)+jj;
        h_n=l_n-1;
        P(h_n)=P(l_n)+sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
        Q(h_n)=Q(l_n)+sys_x(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
        P(h_n)=P(h_n)+nodeP(h_n);
        Q(h_n)=Q(h_n)+nodeQ(h_n);
        flowP(h_n,l_n)=P(h_n);
        flowQ(h_n,l_n)=Q(h_n);

        dP(h_n,l_n)=sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    end

    %高圧低圧の変圧器
    l_n=7+9*(ii-1);
    h_n=ii+1;
    P(l_n)=P(l_n)+nodeP(l_n);
    Q(l_n)=Q(l_n)+nodeQ(l_n);
    P(h_n)=P(l_n)+sys_r(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    Q(h_n)=Q(l_n)+sys_x(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    flowP(h_n,l_n)=P(h_n);
    flowQ(h_n,l_n)=Q(h_n);

    dP(h_n,l_n)=sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
end

%save('PQ.mat','P','Q');

%% 高圧ノードの計算(Backward)
for ii=6:-1:2
    l_n=ii;
    h_n=l_n-1;
    P(l_n)=P(l_n)+nodeP(l_n);
    Q(l_n)=Q(l_n)+nodeQ(l_n);
    P(h_n)=P(l_n)+sys_r(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    Q(h_n)=Q(l_n)+sys_x(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    flowP(h_n,l_n)=P(h_n);
    flowQ(h_n,l_n)=Q(h_n);
    dP(h_n,l_n)=sys_r(h_n,l_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
end

for ii=56:-1:52
    l_n=ii;
    h_n=l_n-1;
    if ii==52
        h_n=1;
        P(l_n)=P(l_n)+nodeP(l_n);
        Q(l_n)=Q(l_n)+nodeQ(l_n);
```



```
        subP=nodeP(l_n)+P(l_n)+sys_r(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
        subQ=nodeQ(l_n)+Q(l_n)+sys_x(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    else
        P(l_n)=P(l_n)+nodeP(l_n);
        Q(l_n)=Q(l_n)+nodeQ(l_n);
        P(h_n)=nodeP(l_n)+P(l_n)+sys_r(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
        Q(h_n)=nodeQ(l_n)+Q(l_n)+sys_x(l_n,h_n)*(P(l_n)*P(l_n)+Q(l_n)*Q(l_n))/V(l_n)/V(l_n);
    end
end


%% 高圧ノードの計算(Forward)

V(1,1)=V_LRT;
for ii=2:6
    h_n=ii-1;
    l_n=ii;
    V(l_n)=V(h_n)*(1-(flowP(h_n,l_n)*sys_r(l_n,h_n)+flowQ(h_n,l_n)*sys_x(l_n,h_n))/V(h_n)/V(h_n) ↙
+1i*(flowQ(h_n,l_n)*sys_r(l_n,h_n)-flowP(h_n,l_n)*sys_x(l_n,h_n))/V(h_n)/V(h_n));
end

for ii=52:56
    h_n=ii-1;
    l_n=ii;
    if ii==52
        h_n=1;
        V(l_n)=V(h_n)*(1-(subP*sys_r(l_n,h_n)+subQ*sys_x(l_n,h_n))/V(h_n)/V(h_n)+1i*(subQ*sys_r ↙
(l_n,h_n)-subP*sys_x(l_n,h_n))/V(h_n)/V(h_n));
    else
        V(l_n)=V(h_n)*(1-(P(h_n)*sys_r(l_n,h_n)+Q(l_n)*sys_x(l_n,h_n))/V(h_n)/V(h_n)+1i*(Q(h_n) ↙
*sys_r(l_n,h_n)-P(h_n)*sys_x(l_n,h_n))/V(h_n)/V(h_n));
    end
end

%% 低圧ノードの計算(Forward)

for ii=1:sys_pri_node_num
    h_n=ii+1;
    l_n=7+9*(ii-1);

    V(l_n)=V(h_n)*(1-(flowP(h_n,l_n)*sys_r(l_n,h_n)+flowQ(h_n,l_n)*sys_x(l_n,h_n))/abs(V(h_n)*V ↙
(h_n))+1i*(flowQ(h_n,l_n)*sys_r(l_n,h_n)-flowP(h_n,l_n)*sys_x(l_n,h_n))/abs(V(h_n)*V(h_n)));

    for jj=1:2%低圧接点ノード
        low_n=6+9*(ii-1)+jj;
        h_n=low_n;
        l_n=low_n+1;
        V(l_n)=V(h_n)*(1-(flowP(h_n,l_n)*sys_r(l_n,h_n)+flowQ(h_n,l_n)*sys_x(l_n,h_n))/abs(V(h_n)*V ↙
(h_n))+1i*(flowQ(h_n,l_n)*sys_r(l_n,h_n)-flowP(h_n,l_n)*sys_x(l_n,h_n))/abs(V(h_n)*V(h_n)));
    end

    l_n=10+9*(ii-1);
    for jj=1:3%低圧接点ノード
        low_n=6+9*(ii-1)+jj;
        for kk=1:2 %住宅ノード-低圧接点ノード間
```

```
                h_n=low_n;
                V(l_n)=V(h_n)*(1-(flowP(h_n,l_n)*sys_r(l_n,h_n)+flowQ(h_n,l_n)*sys_x(l_n,h_n))/abs(V ↙
(h_n)*V(h_n))+1i*(flowQ(h_n,l_n)*sys_r(l_n,h_n)-flowP(h_n,l_n)*sys_x(l_n,h_n))/abs(V(h_n)*V(h_n)));

                l_n=l_n+1;
            end
        end
    end
    dV=abs(beforeV-V);
end

%% 電流計算
Inode_pu=zeros(sys_node_num,sys_node_num);
Inode_pu(1,2)=(V(1,1)-V(2,1))/(sys_r(1,2)+1i*sys_x(1,2));
LRT_current=Inode_pu(1,2);
V=abs(V);
%% 電圧の結果を低圧換算して出力
Out=zeros(1,sys_node_num);
for ii=1:sys_node_num
    Out(1,ii)=V(ii,1)*sys_tapr(ii,1)*sys_Vn*1000;
end
```



```
function [ out_Q_long ] = long_term_Q( Vol,up_deadband,down_deadband,deadband_d,Q_long,KI )
%UNTITLED5 長周期制御
%   詳細説明をここに記述
out_Q_long=Q_long;
if Vol>up_deadband %電圧不感帯上限よりも高い場合
    out_Q_long=(Vol-up_deadband)/KI+Q_long;

elseif Vol<down_deadband %電圧不感帯下限よりも低い場合
    out_Q_long=(Vol-down_deadband)/KI+Q_long;
elseif ((down_deadband+deadband_d)<Vol && Vol<(up_deadband-deadband_d)) && abs(Q_long)>0    %上下限不感↙
帯の中にある場合
    if Q_long>0
        out_Q_long =(Vol-(up_deadband-deadband_d))/KI+Q_long;
    elseif Q_long<0
        out_Q_long =(Vol-(down_deadband+deadband_d))/KI+Q_long;
    end
    if out_Q_long*Q_long<0  %符号が異なる場合
        out_Q_long=0;
    end
end

end
```

```
function [ out_LRT_vol,out_LRT_Sd,out_LRT_lag ] = LRT_tapset( LRT_vol,LRT_Sd,temp_LRT_lag,LRT_current, ↙
tt_step,LRT_data_list )
%UNTITLED LRTのタップ動作
LRT_ep=LRT_data_list(1,1);
LRT_Z_target=LRT_data_list(2,1);
LRT_Sd_th=LRT_data_list(3,1);
LRT_Vtarget=LRT_data_list(4,1);
LRT_V_max=LRT_data_list(5,1);
LRT_V_min=LRT_data_list(6,1);
LRT_limit_time=LRT_data_list(7,1);

out_LRT_vol=0; %outputする送り出し電圧

LRT_dV=LRT_Vtarget-abs(LRT_vol-LRT_Z_target*LRT_current);
    temp_LRT_lag=temp_LRT_lag-tt_step;

if abs(LRT_dV)-LRT_ep>0
    if temp_LRT_lag==temp_LRT_lag-tt_step
        LRT_Sd=sign(LRT_dV)*(abs(LRT_dV)-LRT_ep)*tt_step/60;
    else
        LRT_Sd=LRT_Sd+sign(LRT_dV)*(abs(LRT_dV)-LRT_ep)*tt_step/60;
    end

    if abs(LRT_Sd)>= LRT_Sd_th && temp_LRT_lag<=0
        if LRT_dV >0 && LRT_vol < LRT_V_max-0.001
            out_LRT_vol = LRT_vol + 0.0125;

            temp_LRT_lag=LRT_limit_time;

        elseif LRT_dV <0 && LRT_vol > LRT_V_min+0.001
            out_LRT_vol = LRT_vol - 0.0125;

            temp_LRT_lag=LRT_limit_time;
        end
    end
else
    LRT_Sd=0;
end

out_LRT_lag=temp_LRT_lag;
out_LRT_Sd=LRT_Sd;
if out_LRT_vol==0
    out_LRT_vol=LRT_vol;
end

end
```



```
%{
%%%%%%%%%%%%%%%%%%
制御なしの計算
%%%%%%%%%%%%%%%%%%
%}
%% 初期化と入力データの設定
clearvars -except loop*; %ループ数以外削除
load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

loadmatname=strcat(foldername_input,'LRTdata.mat');
load(loadmatname);  %LRTデータの読込

loadmatname=strcat(foldername_input,num2str(loop_day),'dayload.mat');
load(loadmatname);  %正味負荷データの読込

dset_netloadP(:,:,loop_day)=dset_day_netloadP;
dset_netloadP_before(:,:,loop_day)=dset_day_netloadP_before;
dset_netloadQ(:,:,loop_day)=dset_day_netloadQ;
dset_netloadQ_before(:,:,loop_day)=dset_day_netloadQ_before;

%% 変数の設定
noc_vol=zeros(tt_max,sys_node_num); %制御なしの場合の電圧
noc_LRT_vol(1:tt_max,1)=LRT_V_initial;  %制御なしの場合のLRT送出電圧
noc_LRT_current=zeros(tt_max,1);    %制御なしの場合のLRTのバンク電流
noc_LRT_dV=zeros(tt_max,1); %制御なしの場合のLRTの推定電圧と目標電圧の差
noc_LRT_Sd=zeros(tt_max,1); %制御なしの場合の不感帯逸脱積算量
temp_LRT_lag=0;
temp_tt_ini=tt_start/tt_step; %時刻データ(tt_start)を数値に変換
temp_tt_end=tt_stop/tt_step; %時刻データ(tt_stop)を数値に変換

%% メインプロセス
for tt=1:tt_max
        [noc_vol(tt,:),noc_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt,loop_day),dset_netloadQ ↙
(:,tt,loop_day),noc_LRT_vol(tt,1),sys_BFS);%潮流計算
        [noc_LRT_vol(tt+1,1),noc_LRT_Sd(tt+1,1),temp_LRT_lag]=LRT_tapset(noc_LRT_vol(tt,1),noc_LRT_Sd ↙
(tt,1),temp_LRT_lag,noc_LRT_current(tt,1),tt_step,LRT_data_list); %LRTのタップ動作
end

matname=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/noc_result.mat');
save(matname,'noc_*');

%% 電圧指標の計算
noc_volindex_calc;

%% 出力項目の設定と出力
noc_process_output;
```

```
%{
%%%%%%%%%%%%%%%%%%
制御なしの計算結果出力
%%%%%%%%%%%%%%%%%%
%}

output_time=duration(0,0,(tt_start-1:tt_step:tt_stop-1)'); %時刻
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/noc_result.xlsx');

output_noc_vol_table=table(output_time,noc_vol);
writetable(output_noc_vol_table,output_filename,'Sheet','node_voltage'); % 計算結果の書き込み

output_noc_table=table(output_time,noc_LRT_vol(1:tt_max,1),noc_LRT_current,noc_LRT_Sd(1:tt_max,1));
writetable(output_noc_table,output_filename,'Sheet','LRT'); % 計算結果の書き込み
```



```
%{
%%%%%%%%%%%%%%%%%%
制御なしの電圧制御指標計算・出力
%%%%%%%%%%%%%%%%%%
%}
% matファイルがあれば読込、無ければ生成
readtry_filename=strcat(foldername_output,'noc_volindex.mat');

if exist(readtry_filename,'file')==0
    %変数定義
    volindex_noc_LRTtap=zeros(tt_max,main_day_max,main_loop_max);    %送出電圧
    volindex_noc_LRTtap_num=zeros(main_day_max,main_loop_max);  %タップ切替回数
    volindex_noc_vol_cen=zeros(main_day_max,main_loop_max);      %電圧逼迫度
    volindex_noc_vol_fluc=zeros(main_day_max,main_loop_max);          %電圧変動量

    volindex_noc_vol_max=zeros(tt_max,main_day_max,main_loop_max);       %系統内の最大電圧
    volindex_noc_vol_min=zeros(tt_max,main_day_max,main_loop_max);       %系統内の最小電圧
    volindex_noc_vol_vio=zeros(main_day_max,main_loop_max);          %電圧逸脱量
else
    load(readtry_filename);
end

[volindex_noc_LRTtap(:,loop_day,loop),volindex_noc_LRTtap_num(loop_day,loop),volindex_noc_vol_cen ↙
(loop_day,loop),volindex_noc_vol_fluc(loop_day,loop),...
    volindex_noc_vol_max(:,loop_day,loop),volindex_noc_vol_min(:,loop_day,loop),volindex_noc_vol_vio ↙
(loop_day,loop)]...
    =vol_index_calc(noc_LRT_vol(1:tt_max,1),noc_vol);

save(readtry_filename,'volindex_noc_*');
```

```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の場合
%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データの設定
clearvars -except loop*; %ループ数以外削除

load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

loadmatname=strcat(foldername_input,'LRTdata.mat');
load(loadmatname);  %LRTデータの読込

loadmatname=strcat(foldername_input,num2str(loop_day),'dayload.mat');
load(loadmatname);  %正味負荷データの読込

dset_netloadP(:,:,loop_day)=dset_day_netloadP;
dset_netloadP_before(:,:,loop_day)=dset_day_netloadP_before;
dset_netloadQ(:,:,loop_day)=dset_day_netloadQ;
dset_netloadQ_before(:,:,loop_day)=dset_day_netloadQ_before;

loadmatname=strcat(foldername_input,'RCdata.mat');
load(loadmatname);  %RCデータの読込

loadmatname=strcat(foldername_input,'RC_control.mat');
load(loadmatname);  %RC制御データの読込

%% 初期化
RCp_set;

%% メインプロセス

%
tt=1;
[RCp_vol_before(tt,:),RCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP_before(:,tt,loop_day), ↙
dset_netloadQ_before(:,tt,loop_day),RCp_LRT_vol(tt,1),sys_BFS);%潮流計算
for ii=1:RC_num
    temp_RCp_control_vol(1,ii)=RCp_vol_before(tt,RC_node(ii));
end
%}

for tt=1:tt_max
    if rem(temp_timecount,min(RCc_time_interval(:,:,loop)))==1 || rem(temp_timecount,RC_time_interval) ↙
==1

        % tt時刻における制御前
        [RCp_vol_Qbefore(temp_RCQ_timecount,:),RCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt, ↙
loop_day),dset_netloadQ(:,tt,loop_day)+RCp_Q_total(temp_RCQ_timecount,:)',RCp_LRT_vol(tt,1),sys_BFS);%↙
潮流計算
        RCp_vol_before(tt,:)=RCp_vol_Qbefore(temp_RCQ_timecount,:);
```



```
        RCp_Qset;% 充電器の無効電力出力の決定

        [RCp_vol(tt,:),RCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt,loop_day),dset_netloadQ ↙
(:,tt,loop_day)+RCp_Q_total(temp_RCQ_timecount,:)',RCp_LRT_vol(tt,1),sys_BFS);%潮流計算

        RCp_Qset_after; % 充電器の無効電力決定後の処理

        temp_RCQ_timecount=temp_RCQ_timecount+1;      %充電器制御の時間カウント

        %充電器利用パターンの時間カウント(全RCで一律)
        if rem(temp_timecount,RC_time_interval)==1
            temp_RC_timecount=temp_RC_timecount+1;
        end
    else
        [RCp_vol(tt,:),RCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt,loop_day),dset_netloadQ ↙
(:,tt,loop_day)+RCp_Q_total(temp_RCQ_timecount,:)',RCp_LRT_vol(tt,1),sys_BFS);%潮流計算
    end
    [RCp_LRT_vol(tt+1,1),RCp_LRT_Sd(tt+1,1),temp_LRT_lag]=LRT_tapset(RCp_LRT_vol(tt,1),RCp_LRT_Sd(tt, ↙
1),temp_LRT_lag,RCp_LRT_current(tt,1),tt_step,LRT_data_list); %LRTのタップ動作

    %LRT_tapset; %LRTのタップ動作
    temp_timecount=temp_timecount+tt_step;
end

%
matname=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/RCp_result.mat');
save(matname,'RCp_*');
%}

%% 電圧指標の計算
RCp_volindex_calc;
RCp_Qindex_calc;

%% 出力項目の設定と出力
RCp_process_output;
```

```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の初期設定
%%%%%%%%%%%%%%%%%%
%}

%% 系統変数の設定
RCp_vol=zeros(tt_max,sys_node_num); %制御後の電圧
RCp_vol_before=zeros(tt_max,sys_node_num);  %制御前の電圧
RCp_LRT_vol(1:tt_max,1)=LRT_V_initial;  %制御なしの場合のLRT送出電圧
RCp_LRT_current=zeros(tt_max,1);    %制御なしの場合のLRTのバンク電流
RCp_LRT_dV=zeros(tt_max,1); %制御なしの場合のLRTの推定電圧と目標電圧の差
RCp_LRT_Sd=zeros(tt_max,1); %制御なしの場合の不感帯逸脱積算量
temp_LRT_lag=0;

%% RC関連の設定
temp_RC_tt_ini=ceil(tt_start/RC_time_interval); %RC利用状況の時間の初期tt
temp_RCp_time_max=tt_max;

RCp_vol_before=zeros(temp_RCp_time_max,sys_node_num);   %制御前の電圧
RCp_Q=zeros(temp_RCp_time_max,RC_num); %各充電器の無効電力出力
RCp_Q_long=zeros(temp_RCp_time_max,RC_num); %長周期制御のQ
RCp_Q_short=zeros(temp_RCp_time_max,RC_num); %短周期制御のQ
RCp_dQ_long=zeros(temp_RCp_time_max,RC_num); %短周期制御のQ
RCp_dQ_short_before=zeros(temp_RCp_time_max,RC_num); %短周期制御のQ
RCp_dQ_short_after=zeros(temp_RCp_time_max,RC_num); %短周期制御のQ
RCp_dQ_short_range_upp=zeros(temp_RCp_time_max,RC_num); %短周期制御調整範囲の上限
RCp_dQ_short_range_und=zeros(temp_RCp_time_max,RC_num); %短周期制御調整範囲の下限
RCp_vol_diff=zeros(temp_RCp_time_max,RC_num); %電圧変化
RCp_vol_long=zeros(temp_RCp_time_max,RC_num); %長周期制御用電圧
RCp_Q_total=zeros(temp_RCp_time_max,sys_node_num); %各ノードの無効電力

%時間カウント関連
temp_timecount=1;
temp_RC_timecount=temp_RC_tt_ini;
temp_RCc_timecount=ones(1,RC_num);
temp_RCc_volm_timecount=ones(1,RC_num);
temp_RCQ_timecount=1;
temp_RCc_control_timecount(1,1:RC_num)=1;
temp_RCc_intval_timecount(1,1:RC_num)=1;
temp_RCp_control_vol=zeros(1,RC_num);
```



```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の無効電力算出
%%%%%%%%%%%%%%%%%%
%}

%% 電圧測定

for ii=1:RC_num
    if RCc_volm_time(temp_RCc_intval_timecount(1,ii),ii,loop)==temp_timecount
        temp_RCc_volm_timecount(1,ii)=temp_RCQ_timecount;%電圧測定時間(s)の記録
        %短周期
        RCp_vol_diff(temp_RCc_volm_timecount(1,ii),ii)=RCp_vol_Qbefore(temp_RCQ_timecount,RC_node(ii))- ↙
temp_RCp_control_vol(1,ii);
        %長周期
        RCp_vol_long(temp_RCc_volm_timecount(1,ii),ii)=RCp_vol_Qbefore(temp_RCQ_timecount,RC_node(ii));
    else
        %短周期
        RCp_vol_diff(temp_RCQ_timecount,ii)= RCp_vol_diff(temp_RCc_volm_timecount(1,ii),ii);
        %長周期
        RCp_vol_long(temp_RCQ_timecount,ii)=RCp_vol_long(temp_RCc_volm_timecount(1,ii),ii);
    end
end

%% 引継と制御

if temp_RCQ_timecount==1 %最初の時間(前時間補償量を0に設定)
    for ii=1:RC_num
        RCp_Q_short(temp_RCQ_timecount,ii)=short_term_Q(RCp_vol_diff(temp_RCc_volm_timecount(1,ii),ii), ↙
0,RCc_Ks(ii,1),RCc_FeedbackGain(ii,loop),RCc_FeedbankTime(ii,loop));
        RCp_Q_long(temp_RCQ_timecount,ii)=long_term_Q(RCp_vol_long(temp_RCc_volm_timecount(1,ii),ii), ↙
RCc_up_deadband(ii,loop),RCc_down_deadband(ii,loop),RCc_deadband_d(ii,loop),0,RCc_Kl(ii,1));
    end
    RCp_Q(temp_RCQ_timecount,:)=RCp_Q_short(temp_RCQ_timecount,:)+RCp_Q_long(temp_RCQ_timecount,:);

else
    %% 引継
    RCp_Q_short(temp_RCQ_timecount,:)=RCp_Q_short(temp_RCQ_timecount-1,:);
    RCp_Q_long(temp_RCQ_timecount,:)=RCp_Q_long(temp_RCQ_timecount-1,:);


    %% 制御
    for ii=1:RC_num
        if RCc_control_time(temp_RCc_control_timecount(1,ii),ii,loop)==temp_timecount
            % 短周期
            RCp_Q_short(temp_RCQ_timecount,ii)=short_term_Q(RCp_vol_diff(temp_RCc_volm_timecount(1,ii), ↙
ii),RCp_Q_short(temp_RCQ_timecount-1,ii),RCc_Ks(ii,1),RCc_FeedbackGain(ii,loop),RCc_FeedbankTime(ii, ↙
loop));
            % 長周期
            RCp_Q_long(temp_RCQ_timecount,ii)=long_term_Q(RCp_vol_long(temp_RCc_volm_timecount(1,ii), ↙
ii),RCc_up_deadband(ii,loop),RCc_down_deadband(ii,loop),RCc_deadband_d(ii,loop),RCp_Q_long ↙
(temp_RCQ_timecount-1,ii),RCc_Kl(ii,1));
        end
        RCp_dQ_short_before(temp_RCQ_timecount,ii)=RCp_Q_short(temp_RCQ_timecount,ii)-RCp_Q_short ↙
(temp_RCQ_timecount-1,ii);
```

```
        %% 相互作用を考慮した上下限制約
        [RCp_dQ_short_after(temp_RCQ_timecount,ii),RCp_dQ_short_range_und(temp_RCQ_timecount,ii), ↙
RCp_dQ_short_range_upp(temp_RCQ_timecount,ii),RCp_dQ_long(temp_RCQ_timecount,ii)]=Q_constraint...
            (RCc_dQ_limit(ii,loop),RCp_Q_short(temp_RCQ_timecount,ii),RCp_Q_short(temp_RCQ_timecount-1, ↙
ii),RCp_Q_long(temp_RCQ_timecount,ii),RCp_Q_long(temp_RCQ_timecount-1,ii),RC_Q_max(temp_RC_timecount, ↙
ii,loop_day));

    end
    RCp_Q_long(temp_RCQ_timecount,:)=RCp_dQ_long(temp_RCQ_timecount,:)+RCp_Q_long(temp_RCQ_timecount- ↙
1,:);
    RCp_Q_short(temp_RCQ_timecount,:)=RCp_dQ_short_after(temp_RCQ_timecount,:)+RCp_Q_short ↙
(temp_RCQ_timecount-1,:);
    RCp_Q(temp_RCQ_timecount,:)=RCp_Q_short(temp_RCQ_timecount,:)+RCp_Q_long(temp_RCQ_timecount,:);

end

%% 無効電力量をノード毎に合算

RCp_Q_total(temp_RCQ_timecount,:)=0;
for ii=1:RC_num
    RCp_Q_total(temp_RCQ_timecount,RC_node(ii))=RCp_Q_total(temp_RCQ_timecount,RC_node(ii))-RCp_Q ↙
(temp_RCQ_timecount,ii)/sys_Pn/1000;%通常
end
RCp_Q_total(temp_RCQ_timecount+1,:)=RCp_Q_total(temp_RCQ_timecount,:);
```



```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の制御後の電圧計測
%%%%%%%%%%%%%%%%%%
%}

for ii=1:RC_num
    if RCc_volm_time(temp_RCc_intval_timecount(1,ii),ii,loop)==temp_timecount
        temp_RCc_intval_timecount(1,ii)=temp_RCc_intval_timecount(1,ii)+1;
   end

    if RCc_control_time(temp_RCc_control_timecount(ii),ii,loop)==temp_timecount
        temp_RCc_control_timecount(1,ii)=temp_RCc_control_timecount(1,ii)+1;

        % 制御後(次の制御点)の電圧測定
        temp_RCp_control_vol(1,ii)=RCp_vol(tt,RC_node(ii));
    end

    %end
end
```

```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の無効電力指標計算
%%%%%%%%%%%%%%%%%%
%}

%% matファイルがあれば読込、無ければ生成

readtry_filename=strcat(foldername_output,'RCp_Qindex.mat');
if exist(readtry_filename,'file')==0
    %変数定義
    Qindex_RCp_abssum=zeros(main_day_max,main_loop_max);                %積算無効電力量
    Qindex_RCp_total=zeros(temp_RCp_time_max,main_day_max,main_loop_max);           %各ループの各時間ご↙
との全充電器の無効電力量
    Qindex_RCp_long_total=zeros(temp_RCp_time_max,main_day_max,main_loop_max);      %各ループの各時間ご↙
との全充電器の長周期無効電力量
    Qindex_RCp_short_total=zeros(temp_RCp_time_max,main_day_max,main_loop_max);     %各ループの各時間ご↙
との全充電器の短周期無効電力量
    Qindex_RCp_Qmax_total=zeros(temp_RCp_time_max,main_day_max,main_loop_max);      %各ループの各時間ご↙
との全充電器の補償可能無効電力量
    Qindex_RCp_Qcapacity_factor=zeros(temp_RCp_time_max,main_day_max,main_loop_max); %各ループの各時間↙
ごとの全充電器の無効電力利用率
    Qindex_RCp_capacity_factor=zeros(temp_RCp_time_max,main_day_max,main_loop_max); %各ループの各時間ご↙
との全充電器利用率
    Qindex_RCp_Qmax=zeros(temp_RCp_time_max,RC_num,main_day_max); %各充電器の最大無効電力(時間間隔を換↙
算)
    Qindex_RCp_P=zeros(temp_RCp_time_max,RC_num,main_day_max); %各充電器の有効電力
else
    load(readtry_filename);
end

Qindex_RCp_abssum(loop_day,loop)=mean(mean(abs(RCp_Q)/30));
%計算高速化
%
Qindex_RCp_total(:,loop_day,loop)=sum(RCp_Q(:,:),2);
Qindex_RCp_long_total(:,loop_day,loop)=sum(RCp_Q_long(:,:),2);
Qindex_RCp_short_total(:,loop_day,loop)=sum(RCp_Q_short(:,:),2);

temp_timecount=1;
temp_RC_timecount=temp_RC_tt_ini;
temp_RCQ_timecount=1;

for tt=1:tt_max
    if min(rem(RCc_time_interval(:,:,loop)*(tt-1),tt_step))==1 || min(rem((RCc_time_interval(:,:,loop) ↙
+RCc_time_delay(:,:,loop))*(tt-1),tt_step))==1 || rem(temp_timecount,RC_time_interval)==1

        Qindex_RCp_Qmax(temp_RCQ_timecount,:,loop_day)=RC_Q_max(temp_RC_timecount,:,loop_day);
        Qindex_RCp_P(temp_RCQ_timecount,:,loop_day)=RC_P(temp_RC_timecount,:,loop_day);

        Qindex_RCp_Qmax_total(temp_RCQ_timecount,loop_day,loop)=sum(RC_Q_max(temp_RC_timecount,:, ↙
loop_day),2);
        Qindex_RCp_capacity_factor(temp_RCQ_timecount,loop_day,loop)=100*mean(sqrt(RCp_Q ↙
(temp_RCQ_timecount,:).*RCp_Q(temp_RCQ_timecount,:)+RC_P(temp_RC_timecount,:,loop_day).*RC_P ↙
(temp_RC_timecount,:,loop_day))/RC_cap(:,1)');%各ループの各時間ごとの全充電器の無効電力利用率
```



```
        temp_RCQ_timecount=temp_RCQ_timecount+1;
        if rem(temp_timecount,RC_time_interval)==1
            temp_RC_timecount=temp_RC_timecount+1;
        end

    end
    temp_timecount=temp_timecount+tt_step;
end
Qindex_RCp_Qcapacity_factor(:,loop_day,loop)=100*abs(Qindex_RCp_total(:,loop_day,loop)). ↙
/Qindex_RCp_Qmax_total(:,loop_day,loop); %各ループの各時間ごとの全充電器の無効電力利用率
%}
save(readtry_filename,'Qindex_RCp_*');
```

```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の電圧制御指標計算・出力
%%%%%%%%%%%%%%%%%%
%}

% matファイルがあれば読込、無ければ生成

readtry_filename=strcat(foldername_output,'RCp_volindex.mat');
if exist(readtry_filename,'file')==0
    %変数定義
    volindex_RCp_LRTtap=zeros(tt_max,main_day_max,main_loop_max);    %送出電圧
    volindex_RCp_LRTtap_num=zeros(main_day_max,main_loop_max);  %タップ切替回数
    volindex_RCp_vol_cen=zeros(main_day_max,main_loop_max);      %電圧逼迫度
    volindex_RCp_vol_fluc=zeros(main_day_max,main_loop_max);         %電圧変動量

    volindex_RCp_vol_max=zeros(tt_max,main_day_max,main_loop_max);       %系統内の最大電圧
    volindex_RCp_vol_min=zeros(tt_max,main_day_max,main_loop_max);       %系統内の最小電圧
    volindex_RCp_vol_vio=zeros(main_day_max,main_loop_max);          %電圧逸脱量

else
    load(readtry_filename);
end

[volindex_RCp_LRTtap(:,loop_day,loop),volindex_RCp_LRTtap_num(loop_day,loop),volindex_RCp_vol_cen ↙
(loop_day,loop),volindex_RCp_vol_fluc(loop_day,loop),...
    volindex_RCp_vol_max(:,loop_day,loop),volindex_RCp_vol_min(:,loop_day,loop),volindex_RCp_vol_vio ↙
(loop_day,loop)]...
    =vol_index_calc(RCp_LRT_vol(1:tt_max,1),RCp_vol);

save(readtry_filename,'volindex_RCp_*');
```



```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の結果出力
%%%%%%%%%%%%%%%%%%
%}

%% 無効電力関連
output_time=duration(0,0,(tt_start-1:tt_step:tt_stop-1)'); %時刻
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/RCp_Q_result.xlsx');

output_RCp_Q_table=table(output_time,RCp_Q);
writetable(output_RCp_Q_table,output_filename,'Sheet','RCp_Q'); % 計算結果の書き込み

output_RCp_Q_table=table(output_time,RCp_Q_short);
writetable(output_RCp_Q_table,output_filename,'Sheet','RCp_Q_short'); % 計算結果の書き込み

output_RCp_Q_table=table(output_time,RCp_Q_long);
writetable(output_RCp_Q_table,output_filename,'Sheet','RCp_Q_long'); % 計算結果の書き込み

output_RCp_Q_table=table(output_time,RCp_Q_total(1:temp_RCp_time_max,2:6));
writetable(output_RCp_Q_table,output_filename,'Sheet','RCp_Q_total'); % 計算結果の書き込み

%% 各充電器の無効電力出力
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/RCp_Q_eachRC.xlsx');
for ii=1:RC_num
    output_RCp_Q_RCtable=table(output_time,Qindex_RCp_P(:,ii,loop_day),Qindex_RCp_Qmax(:,ii,loop_day), ↙
RCp_Q(:,ii),...
        RCp_Q_short(:,ii),RCp_Q_long(:,ii),...
        'VariableNames',{'time','Charging_Power','Max_reactive_Power','Output','Short','Long'});
    sheetname=strcat('RC_',num2str(ii));
    writetable(output_RCp_Q_RCtable,output_filename,'Sheet',sheetname); % 計算結果の書き込み
end

%電圧関連
output_time=duration(0,0,(tt_start-1:tt_step:tt_stop-1)'); %時刻
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/RCp_result.xlsx');

output_RCp_vol_table=table(output_time,RCp_vol_before(:,:));
writetable(output_RCp_vol_table,output_filename,'Sheet','RCp_vol_before'); % 計算結果の書き込み

output_RCp_vol_table=table(output_time,RCp_vol);
writetable(output_RCp_vol_table,output_filename,'Sheet','node_vol'); % 計算結果の書き込み

output_RCp_table=table(output_time,RCp_LRT_vol(1:tt_max,1),RCp_LRT_current,RCp_LRT_Sd(1:tt_max,1));
writetable(output_RCp_table,output_filename,'Sheet','LRT'); % 計算結果の書き込み
```

```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の場合
%%%%%%%%%%%%%%%%%%
%}

%% 初期化と入力データの設定

clearvars -except loop*; %ループ数以外削除

load('../../initialsetup.mat');

loadmatname=strcat(foldername_input,'systemsetup.mat');
load(loadmatname);  %系統データの読込

loadmatname=strcat(foldername_input,'LRTdata.mat');
load(loadmatname);  %LRTデータの読込

loadmatname=strcat(foldername_input,num2str(loop_day),'dayload.mat');
load(loadmatname);  %正味負荷データの読込

dset_netloadP(:,:,loop_day)=dset_day_netloadP;
dset_netloadP_before(:,:,loop_day)=dset_day_netloadP_before;
dset_netloadQ(:,:,loop_day)=dset_day_netloadQ;
dset_netloadQ_before(:,:,loop_day)=dset_day_netloadQ_before;

loadmatname=strcat(foldername_input,'SVC_control.mat');
load(loadmatname);  %SVC制御データの読込

%% 初期化
SVCp_set;

%% メインプロセス
%
tt=1;
[SVCp_vol_before(tt,:),SVCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP_before(:,tt,loop_day), ↙
dset_netloadQ_before(:,tt,loop_day),SVCp_LRT_vol(tt,1),sys_BFS);%潮流計算
for ii=1:SVC_num
    temp_SVCp_control_vol(1,ii)=SVCp_vol_before(tt,SVC_node(ii));
end
%}

for tt=1:tt_max

    if min(rem(SVCc_time_interval*(tt-1),tt_step))==1 || min(rem((SVCc_time_interval+SVCc_time_delay)* ↙
(tt-1),tt_step))==1 || rem(temp_timecount,SVC_time_interval)==1

        % tt時刻における制御前
        [SVCp_vol_Qbefore(temp_SVCQ_timecount,:),SVCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:, ↙
tt,loop_day),dset_netloadQ(:,tt,loop_day)+SVCp_Q_total(temp_SVCQ_timecount,:)',SVCp_LRT_vol(tt,1), ↙
sys_BFS);%潮流計算
        SVCp_vol_before(tt,:)=SVCp_vol_Qbefore(temp_SVCQ_timecount,:);

        SVCp_Qset;% 充電器の無効電力出力の決定
        [SVCp_vol(tt,:),SVCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt,loop_day), ↙
```



```
dset_netloadQ(:,tt,loop_day)+SVCp_Q_total(temp_SVCQ_timecount,:)',SVCp_LRT_vol(tt,1),sys_BFS);%潮流計算

        SVCp_Qset_after; % 充電器の無効電力出力後の処理
        %充電器利用パターンの時間カウント(全SVCで一律)
        if rem(temp_timecount,SVC_time_interval)==1
            temp_SVC_timecount=temp_SVC_timecount+1;
        end
        temp_SVCQ_timecount=temp_SVCQ_timecount+1;      %充電器制御の時間カウント
    else
        [SVCp_vol(tt,:),SVCp_LRT_current(tt,1)]=load_flow_BFS(dset_netloadP(:,tt,loop_day), ↙
dset_netloadQ(:,tt,loop_day)+SVCp_Q_total(temp_SVCQ_timecount,:)',SVCp_LRT_vol(tt,1),sys_BFS);%潮流計算
    end

    [SVCp_LRT_vol(tt+1,1),SVCp_LRT_Sd(tt+1,1),temp_LRT_lag]=LRT_tapset(SVCp_LRT_vol(tt,1),SVCp_LRT_Sd ↙
(tt,1),temp_LRT_lag,SVCp_LRT_current(tt,1),tt_step,LRT_data_list); %LRTのタップ動作
    temp_timecount=temp_timecount+tt_step;
end

%
matname=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/SVCp_result.mat');
save(matname,'SVCp_*');
%}

%% 電圧指標の計算
SVCp_volindex_calc;
SVCp_Qindex_calc;

%% 出力項目の設定と出力
SVCp_process_output;
```

```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の初期設定
%%%%%%%%%%%%%%%%%%
%}

%% 系統変数の設定
SVCp_vol=zeros(tt_max,sys_node_num);     %制御後の電圧
SVCp_vol_before=zeros(tt_max,sys_node_num); %制御前の電圧

SVCp_LRT_vol(1:tt_max,1)=LRT_V_initial; %制御なしの場合のLRT送出電圧
SVCp_LRT_current=zeros(tt_max,1);    %制御なしの場合のLRTのバンク電流
SVCp_LRT_dV=zeros(tt_max,1);     %制御なしの場合のLRTの推定電圧と目標電圧の差
SVCp_LRT_Sd=zeros(tt_max,1);     %制御なしの場合の不感帯逸脱積算量
temp_LRT_lag=0;

%% SVC関連の設定
SVC_time_interval=SVCc_time_interval;
temp_SVC_tt_ini=ceil(tt_start/SVC_time_interval); %SVC利用状況の時間の初期tt
temp_SVCp_time_max=ceil(tt_max*tt_step/max(min(min(SVCc_time_interval),min(SVCc_time_delay)),tt_step));

SVCp_vol_before=zeros(temp_SVCp_time_max,sys_node_num); %制御前の電圧
SVCp_Q=zeros(temp_SVCp_time_max,SVC_num); %各充電器の無効電力出力
SVCp_Q_long=zeros(temp_SVCp_time_max,SVC_num); %長周期制御のQ
SVCp_Q_short=zeros(temp_SVCp_time_max,SVC_num); %短周期制御のQ
SVCp_dQ_long=zeros(temp_SVCp_time_max,SVC_num); %短周期制御のQ
SVCp_dQ_short_before=zeros(temp_SVCp_time_max,SVC_num); %短周期制御のQ
SVCp_dQ_short_after=zeros(temp_SVCp_time_max,SVC_num); %短周期制御のQ
SVCp_dQ_short_range_upp=zeros(temp_SVCp_time_max,SVC_num); %短周期制御調整範囲の上限
SVCp_dQ_short_range_und=zeros(temp_SVCp_time_max,SVC_num); %短周期制御調整範囲の下限
SVCp_vol_diff=zeros(temp_SVCp_time_max,SVC_num); %電圧変化
SVCp_vol_long=zeros(temp_SVCp_time_max,SVC_num); %長周期制御用電圧
SVCp_Q_total=zeros(temp_SVCp_time_max,sys_node_num);

if exist('BFSdata.mat','file')==0
    save('BFSdata.mat','sys_pri_node_num','sys_Vn','sys_node_num','sys_r','sys_x','sys_tapr'); %matファ↙
イルの読込の高速化ために作成
end

temp_timecount=1;
temp_SVC_timecount=temp_SVC_tt_ini;
temp_SVCc_timecount=ones(1,SVC_num);
temp_SVCc_volm_timecount=ones(1,SVC_num);
temp_SVCQ_timecount=1;
temp_SVCc_control_timecount(1,1:SVC_num)=1;
temp_SVCc_intval_timecount(1,1:SVC_num)=1;
temp_SVCp_control_vol=zeros(1,SVC_num);
```



```
%{
%%%%%%%%%%%%%%%%%%
充電器制御の無効電力算出
%%%%%%%%%%%%%%%%%%
%}
%% 電圧測定

for ii=1:SVC_num
    if SVCc_volm_time(temp_SVCc_intval_timecount(1,ii),ii)==temp_timecount
        temp_SVCc_volm_timecount(1,ii)=temp_SVCQ_timecount;%電圧測定時間(s)の記録
        %短周期
        SVCp_vol_diff(temp_SVCc_volm_timecount(1,ii),ii)=SVCp_vol_Qbefore(temp_SVCQ_timecount,SVC_node ↙
(ii))-temp_SVCp_control_vol(1,ii);
        %長周期
        SVCp_vol_long(temp_SVCc_volm_timecount(1,ii),ii)=SVCp_vol_Qbefore(temp_SVCQ_timecount,SVC_node ↙
(ii));
    else
        %短周期
        SVCp_vol_diff(temp_SVCQ_timecount,ii)= SVCp_vol_diff(temp_SVCc_volm_timecount(1,ii),ii);
        %長周期
        SVCp_vol_long(temp_SVCc_volm_timecount(1,ii),ii)=SVCp_vol_long(temp_SVCc_volm_timecount(1,ii), ↙
ii);
    end
end

%% 引継と制御

if temp_SVCQ_timecount==1 %最初の時間(前時間補償量を0に設定)
    for ii=1:SVC_num
        SVCp_Q_short(temp_SVCQ_timecount,ii)=short_term_Q(SVCp_vol_diff(temp_SVCc_volm_timecount(1,ii), ↙
ii),0,SVCc_Ks(ii,1),SVCc_FeedbackGain(ii,loop),SVCc_FeedbankTime(ii,loop));
        SVCp_Q_long(temp_SVCQ_timecount,ii)=long_term_Q(SVCp_vol_long(temp_SVCc_volm_timecount(1,ii), ↙
ii),SVCc_up_deadband(ii,loop),SVCc_down_deadband(ii,loop),SVCc_deadband_d(ii,loop),0,SVCc_Kl(ii,1));
    end
    SVCp_Q(temp_SVCQ_timecount,:)=SVCp_Q_short(temp_SVCQ_timecount,:)+SVCp_Q_long ↙
(temp_SVCQ_timecount,:);
else
    %% 引継
    SVCp_Q_short(temp_SVCQ_timecount,:)=SVCp_Q_short(temp_SVCQ_timecount-1,:);
    SVCp_Q_long(temp_SVCQ_timecount,:)=SVCp_Q_long(temp_SVCQ_timecount-1,:);

    %% 制御
    for ii=1:SVC_num
        if SVCc_control_time(temp_SVCc_control_timecount(1,ii),ii)==temp_timecount
            % 短周期
            SVCp_Q_short(temp_SVCQ_timecount,ii)=short_term_Q(SVCp_vol_diff(temp_SVCc_volm_timecount(1, ↙
ii),ii),SVCp_Q_short(temp_SVCQ_timecount-1,ii),SVCc_Ks(ii,1),SVCc_FeedbackGain(ii,loop), ↙
SVCc_FeedbankTime(ii,loop));
            % 長周期
            SVCp_Q_long(temp_SVCQ_timecount,ii)=long_term_Q(SVCp_vol_long(temp_SVCc_volm_timecount(1, ↙
ii),ii),SVCc_up_deadband(ii,loop),SVCc_down_deadband(ii,loop),SVCc_deadband_d(ii,loop),SVCp_Q_long ↙
(temp_SVCQ_timecount-1,ii),SVCc_Kl(ii,1));
        end

        %% 相互作用を考慮した上下限制約
```

```
        [SVCp_dQ_short_after(temp_SVCQ_timecount,ii),SVCp_dQ_short_range_und(temp_SVCQ_timecount,ii), ↙
SVCp_dQ_short_range_upp(temp_SVCQ_timecount,ii),SVCp_dQ_long(temp_SVCQ_timecount,ii)]=Q_constraint...
            (SVCc_dQ_limit(ii,loop),SVCp_Q_short(temp_SVCQ_timecount,ii),SVCp_Q_short ↙
(temp_SVCQ_timecount-1,ii),SVCp_Q_long(temp_SVCQ_timecount,ii),SVCp_Q_long(temp_SVCQ_timecount-1,ii), ↙
SVC_cap(ii,loop));

    end
    SVCp_Q_long(temp_SVCQ_timecount,:)=SVCp_dQ_long(temp_SVCQ_timecount,:)+SVCp_Q_long ↙
(temp_SVCQ_timecount-1,:);
    SVCp_Q_short(temp_SVCQ_timecount,:)=SVCp_dQ_short_after(temp_SVCQ_timecount,:)+SVCp_Q_short ↙
(temp_SVCQ_timecount-1,:);
    SVCp_Q(temp_SVCQ_timecount,:)=SVCp_Q_short(temp_SVCQ_timecount,:)+SVCp_Q_long ↙
(temp_SVCQ_timecount,:);

end

%% 無効電力量をノード毎に合算
SVCp_Q_total(temp_SVCQ_timecount,:)=0;
for ii=1:SVC_num
    SVCp_Q_total(temp_SVCQ_timecount,SVC_node(ii))=SVCp_Q_total(temp_SVCQ_timecount,SVC_node(ii))- ↙
SVCp_Q(temp_SVCQ_timecount,ii)/sys_Pn/1000;%通常
end
SVCp_Q_total(temp_SVCQ_timecount+1,:)=SVCp_Q_total(temp_SVCQ_timecount,:);
```



```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の制御後の電圧計測
%%%%%%%%%%%%%%%%%%
%}

for ii=1:SVC_num
    if SVCc_volm_time(temp_SVCc_intval_timecount(1,ii),ii)==temp_timecount
        temp_SVCc_intval_timecount(1,ii)=temp_SVCc_intval_timecount(1,ii)+1;
    end

    if SVCc_control_time(temp_SVCc_control_timecount(ii),ii)==temp_timecount
        temp_SVCc_control_timecount(1,ii)=temp_SVCc_control_timecount(1,ii)+1;

        % 制御後(次の制御点)の電圧測定
        temp_SVCp_control_vol(1,ii)=SVCp_vol(tt,SVC_node(ii));
    end
end
```

```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の無効電力指標計算
%%%%%%%%%%%%%%%%%%
%}


%% matファイルがあれば読込、無ければ生成

readtry_filename=strcat(foldername_output,'SVCp_Qindex.mat');
if exist(readtry_filename,'file')==0
    %変数定義
     Qindex_SVCp_abssum=zeros(main_day_max,main_loop_max);                %積算無効電力量
    Qindex_SVCp_total=zeros(temp_SVCp_time_max,main_day_max,main_loop_max);             %各ループの各時間↙
ごとの全充電器の無効電力量
    Qindex_SVCp_long_total=zeros(temp_SVCp_time_max,main_day_max,main_loop_max);        %各ループの各時間↙
ごとの全充電器の長周期無効電力量
    Qindex_SVCp_short_total=zeros(temp_SVCp_time_max,main_day_max,main_loop_max);       %各ループの各時間↙
ごとの全充電器の短周期無効電力量
    Qindex_SVCp_Qcapacity_factor=zeros(temp_SVCp_time_max,main_day_max,main_loop_max); %各ループの各時↙
間ごとの全充電器の無効電力利用率
else
    load(readtry_filename);
end
Qindex_SVCp_abssum(loop_day,loop)=mean(mean(abs(SVCp_Q)/900));

Qindex_SVCp_total(:,loop_day,loop)=sum(SVCp_Q(:,:),2);
Qindex_SVCp_long_total(:,loop_day,loop)=sum(SVCp_Q_long(:,:),2);
Qindex_SVCp_short_total(:,loop_day,loop)=sum(SVCp_Q_short(:,:),2);

temp_timecount=1;
temp_SVC_timecount=temp_SVC_tt_ini;
temp_SVCQ_timecount=1;

Qindex_SVCp_Qcapacity_factor(:,loop_day,loop)=100*mean(abs(SVCp_Q(:,:)),2)./SVC_cap(:,loop); %各ループ↙
の各時間ごとの全充電器の無効電力利用率
Qindex_SVCp_Qcapacity_factor(isnan(Qindex_SVCp_Qcapacity_factor))=100;

save(readtry_filename,'Qindex_SVCp_*'); %Qindex_SVCp_* から変更(保存時間の短縮)
```



```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の電圧制御指標計算・出力
%%%%%%%%%%%%%%%%%%
%}
% matファイルがあれば読込、無ければ生成
readtry_filename=strcat(foldername_output,'SVCp_volindex.mat');
if exist(readtry_filename,'file')==0
    %変数定義
    volindex_SVCp_LRTtap=zeros(tt_max,main_day_max,main_loop_max);  %送出電圧
    volindex_SVCp_LRTtap_num=zeros(main_day_max,main_loop_max);     %タップ切替回数
    volindex_SVCp_vol_cen=zeros(main_day_max,main_loop_max);        %電圧逼迫度
    volindex_SVCp_vol_fluc=zeros(main_day_max,main_loop_max);       %電圧変動量

    volindex_SVCp_vol_max=zeros(tt_max,main_day_max,main_loop_max);     %系統内の最大電圧
    volindex_SVCp_vol_min=zeros(tt_max,main_day_max,main_loop_max);     %系統内の最小電圧
    volindex_SVCp_vol_vio=zeros(main_day_max,main_loop_max);            %電圧逸脱量
else
    load(readtry_filename);
end

[volindex_SVCp_LRTtap(:,loop_day,loop),volindex_SVCp_LRTtap_num(loop_day,loop),volindex_SVCp_vol_cen ↙
(loop_day,loop),volindex_SVCp_vol_fluc(loop_day,loop),...
    volindex_SVCp_vol_max(:,loop_day,loop),volindex_SVCp_vol_min(:,loop_day,loop),volindex_SVCp_vol_vio ↙
(loop_day,loop)]...
    =vol_index_calc(SVCp_LRT_vol(1:tt_max,1),SVCp_vol);

save(readtry_filename,'volindex_SVCp_*');
```

```
%{
%%%%%%%%%%%%%%%%%%
SVC制御の結果出力
%%%%%%%%%%%%%%%%%%
%}
%% 無効電力関連
output_time=(tt_start-1:tt_step:tt_stop-1)'; %時刻
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/SVCp_Q_result.↙
xlsx');

%% 各充電器の無効電力出力
for ii=1:SVC_num
    output_SVCp_Q_SVCtable=table(output_time,Qindex_SVCp_total(:,loop_day,loop),SVCp_Q(:,:),...
    SVCp_Q_short(:,:),SVCp_Q_long(:,:),...
        'VariableNames',{'time','total','Output','Short','Long'});

    sheetname=strcat('SVC_',num2str(ii));
    writetable(output_SVCp_Q_SVCtable,output_filename,'Sheet',sheetname); % 計算結果の書き込み
end

%電圧関連
output_time=duration(0,0,(tt_start-1:tt_step:tt_stop-1)'); %時刻
output_filename=strcat(foldername_output,loopfoldername,'/',num2str(loop_day),'day/SVCp_result.xlsx');

output_SVCp_vol_table=table(output_time,SVCp_vol_before(:,:));
writetable(output_SVCp_vol_table,output_filename,'Sheet','SVCp_vol_before'); % 計算結果の書き込み

output_SVCp_vol_table=table(output_time,SVCp_vol);
writetable(output_SVCp_vol_table,output_filename,'Sheet','node_vol'); % 計算結果の書き込み

output_SVCp_table=table(output_time,SVCp_LRT_vol(1:tt_max,1),SVCp_LRT_current,SVCp_LRT_Sd(1:tt_max,1));
writetable(output_SVCp_table,output_filename,'Sheet','LRT'); % 計算結果の書き込み
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%
計算結果出力
%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 計算結果出力

clearvars -except loop* ; %ループ数以外削除

load('../../initialsetup.mat');

%% 電圧制御結果の出力
out_volindex;

%% 無効電力補償量の出力
out_Qindex;
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
無効電力に関する出力
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期設定（結果出力のラベル）

loop_day_label=(1:main_day_max+3)';
resulttable=table(loop_day_label,'VariableNames',{'day'});
output_filename=strcat(foldername_output,'Qindex_list.xlsx');
% タイトル(平均値・最大値・最小値)の入力
xlRange = sprintf('A%d:A%d', main_day_max+2,main_day_max+4);
title={'Average';'Max';'Min'};


%% 充電器の結果
readtry_filename=strcat(foldername_output,'RCp_Qindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);

    Qindex_RCp_abssum=calc_index(Qindex_RCp_abssum);

    RCp_table=table(loop_day_label,Qindex_RCp_abssum,'VariableNames',{'day','RCp_Q_sum'});
    resulttable=outerjoin(resulttable,RCp_table,'MergeKeys',true);

end

%% SVCの結果
readtry_filename=strcat(foldername_output,'SVCp_Qindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);

    Qindex_SVCp_abssum=calc_index(Qindex_SVCp_abssum);

    SVCp_table=table(loop_day_label,Qindex_SVCp_abssum,'VariableNames',{'day','SVCp_Q_sum'});
    resulttable=outerjoin(resulttable,SVCp_table,'MergeKeys',true);

end

writetable(resulttable,output_filename,'sheet','summary'); % 計算結果の書き込み
xlswrite(output_filename,title,'summary',xlRange);
```



```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
電圧に関する出力
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

%% 初期設定（結果出力のラベル）
loop_day_label=(1:main_day_max+3)';
resulttable=table(loop_day_label,'VariableNames',{'day'});
output_filename=strcat(foldername_output,'volindex_list.xlsx');
% タイトル(平均値・最大値・最小値)の入力
xlRange = sprintf('A%d:A%d', main_day_max+2,main_day_max+4);
title={'Average';'Max';'Min'};

%% 制御なしの場合（結果出力）

readtry_filename=strcat(foldername_output,'noc_volindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);

    volindex_noc_vol_cen=calc_index(volindex_noc_vol_cen);
    volindex_noc_LRTtap_num=calc_index(volindex_noc_LRTtap_num);
    volindex_noc_vol_fluc=calc_index(volindex_noc_vol_fluc);
    volindex_noc_vol_vio=calc_index(volindex_noc_vol_vio);

    noc_table=table(loop_day_label,volindex_noc_vol_cen,volindex_noc_LRTtap_num,volindex_noc_vol_fluc, ↙
volindex_noc_vol_vio,...
        'VariableNames',{'day','noc_vol_cen','noc_vol_LRTtap_num','noc_vol_fluc','noc_vol_vio'});
    resulttable=outerjoin(resulttable,noc_table,'MergeKeys',true);

    writetable(resulttable,output_filename,'sheet','noc'); % 計算結果の書き込み
    output_filename=strcat(foldername_output,'volindex_list.xlsx');
    resulttable=table(loop_day_label,'VariableNames',{'day'});
    xlswrite(output_filename,title,'noc',xlRange);
end

%% 充電器による制御の場合（結果出力）

readtry_filename=strcat(foldername_output,'RCp_volindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);

    volindex_RCp_vol_cen=calc_index(volindex_RCp_vol_cen);
    volindex_RCp_LRTtap_num=calc_index(volindex_RCp_LRTtap_num);
    volindex_RCp_vol_fluc=calc_index(volindex_RCp_vol_fluc);
    volindex_RCp_vol_vio=calc_index(volindex_RCp_vol_vio);

    RCp_table=table(loop_day_label,volindex_RCp_vol_cen,volindex_RCp_LRTtap_num,volindex_RCp_vol_fluc, ↙
volindex_RCp_vol_vio,...
        'VariableNames',{'day','RC_vol_cen','RC_vol_LRTtap_num','RC_vol_fluc','RC_vol_vio'});
    resulttable=outerjoin(resulttable,RCp_table,'MergeKeys',true);

    writetable(resulttable,output_filename,'sheet','RC'); % 計算結果の書き込み
    output_filename=strcat(foldername_output,'volindex_list.xlsx');
    resulttable=table(loop_day_label,'VariableNames',{'day'});
```

```
    xlswrite(output_filename,title,'RC',xlRange);
end

%% SVCによる制御の場合（結果出力）

readtry_filename=strcat(foldername_output,'SVCp_volindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);

    volindex_SVCp_vol_cen=calc_index(volindex_SVCp_vol_cen);
    volindex_SVCp_LRTtap_num=calc_index(volindex_SVCp_LRTtap_num);
    volindex_SVCp_vol_fluc=calc_index(volindex_SVCp_vol_fluc);
    volindex_SVCp_vol_vio=calc_index(volindex_SVCp_vol_vio);

    SVCp_table=table(loop_day_label,volindex_SVCp_vol_cen,volindex_SVCp_LRTtap_num, ↙
volindex_SVCp_vol_fluc,volindex_SVCp_vol_vio,...
        'VariableNames',{'day','SVC_vol_cen','SVC_vol_LRTtap_num','SVC_vol_fluc','SVC_vol_vio'});
    resulttable=outerjoin(resulttable,SVCp_table,'MergeKeys',true);

    writetable(resulttable,output_filename,'sheet','SVC'); % 計算結果の書き込み
    output_filename=strcat(foldername_output,'volindex_list.xlsx');
    resulttable=table(loop_day_label,'VariableNames',{'day'});
    xlswrite(output_filename,title,'SVC',xlRange);
end
```



```
function [ index ] = calc_index( index )
% 各指標の平均値・最大値・最小値を計算

    [row_num,com_num,page_num]=size(index);
    index(row_num+1,:)=mean(index);
    index(row_num+2,:)=max(index);
    index(row_num+3,:)=min(index);
end
```

```
%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
等価SVC容量の推定
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}
clear;
load('../../initialsetup.mat');
loadmatname=strcat(foldername_input,'SVC_control.mat');
load(loadmatname);  %SVC制御データの読込

%% 初期化
temp_SVCloop_index=zeros(1,main_loop_max);
est_SVCcap_fulc=zeros(main_day_max+3,1);
est_SVCcap_vio=zeros(main_day_max+3,1);
diff_volindex=zeros(main_loop_max,main_day_max);
readtry_filename=strcat(foldername_output,'RCp_volindex.mat');
if exist(readtry_filename,'file')==2
    load(readtry_filename);
    readtry_filename=strcat(foldername_output,'SVCp_volindex.mat');
    if exist(readtry_filename,'file')==2
        load(readtry_filename);

        %% SVC容量の推定・最大値・最小値の計算

        diff_volindex=abs(volindex_SVCp_vol_fluc'-ones(main_loop_max,1)*volindex_RCp_vol_fluc(:,1)'); %↙
日付を列、ループを行になるように転置
        [~,temp_SVCloop_index]=min(diff_volindex); %indexを取得
        est_SVCcap_fulc(1:main_day_max,1)=SVC_cap(1,temp_SVCloop_index(1,:));
        est_SVCcap_fulc(main_day_max+1,1)=mean(est_SVCcap_fulc(1:main_day_max,1));
        est_SVCcap_fulc(main_day_max+2,1)=max(est_SVCcap_fulc(1:main_day_max,1));
        est_SVCcap_fulc(main_day_max+3,1)=min(est_SVCcap_fulc(1:main_day_max,1));

        diff_volindex=abs(volindex_SVCp_vol_vio'-ones(main_loop_max,1)*volindex_RCp_vol_vio(:,1)'); %日↙
付を列、ループを行になるように転置
        [~,temp_SVCloop_index]=min(diff_volindex); %indexを取得
        est_SVCcap_vio(1:main_day_max,1)=SVC_cap(1,temp_SVCloop_index(1,:));
        est_SVCcap_vio(main_day_max+1,1)=mean(est_SVCcap_vio(1:main_day_max,1));
        est_SVCcap_vio(main_day_max+2,1)=max(est_SVCcap_vio(1:main_day_max,1));
        est_SVCcap_vio(main_day_max+3,1)=min(est_SVCcap_vio(1:main_day_max,1));

        %% 推定結果出力
        output_filename=strcat(foldername_output,'SVCest_result.xlsx');
        day_label=(1:main_day_max+3)';
        estSVC_table=table(day_label,est_SVCcap_fulc,est_SVCcap_vio,'VariableNames',↙
{'day','vol_fulc','vol_vio'});
        writetable(estSVC_table,output_filename);

        % タイトル(平均値・最大値・最小値)の入力
        xlRange = sprintf('A%d:A%d', main_day_max+2,main_day_max+4);
        title={'Average';'Max';'Min'};
        xlswrite(output_filename,title,xlRange);
    end
end
```

付録(需給調整に貢献する CGS の運用)

本論文の数値試算のフローチャートおよび使用したプログラムリストを添付する。
なお，本プログラムは MATLAB R2018b での実行環境で開発・実行していたが，MATLAB R2014b 以降であれば動作する。

1. プログラム全体の構成・概要

本プログラムのフォルダ構成を以下に示す。デフォルトで rootfolder の中に setup,process,takeover フォルダが存在する。rootfolder の main.m を実行することで result フォルダが生成される。
また、過去にこのプログラムで生成した（result 内の input フォルダ内の）mat データを”takeover”フォルダに格納することで設定せずにそのまま使用することができる。これは過去に生成した PV 出力パターンを使用したい場合に用いる。ただし、日数やループ数が異なる場合、エラーとなる可能性があるため注意されたい。

rootfolder：プログラムが集約されているフォルダ(ルート)
├──setup　　：各種設定（負荷、CGS の設定）フォルダ
│　└──AGset　　：アグリゲータに関するデータを生成するフォルダ
│　├──CGSsetup ：EV に関する設定フォルダ
│　├──resi_data　：各種データを集約して正味のデータを生成するフォルダ
│　├　└──resi_mat　　：需要データが格納されているフォルダ
├──process　　：実際の計算（プロセス）フォルダ
│　├──common　　：計算に必要な共通フォルダ（CPLEX、潮流計算、LRT 動作等）
│　├──sch　　：運用計画策定のフォルダ
│　├　└──sch_AG　　：運用計画策定時のアグリゲータの設定フォルダ
│　├　└──sch_opt　　：運用計画策定時の最適化の制約条件設定フォルダ
│　├　└──resi_sch　　：住宅毎に運用計画策定結果をまとめ、エクセル出力を行うフォルダ
│　├──ope　　：当日運用フォルダ（現在作成中）
├──takeover　　：過去に生成した mat ファイルを引き継ぐ際に使用するフォルダ

＜プログラム実行後＞

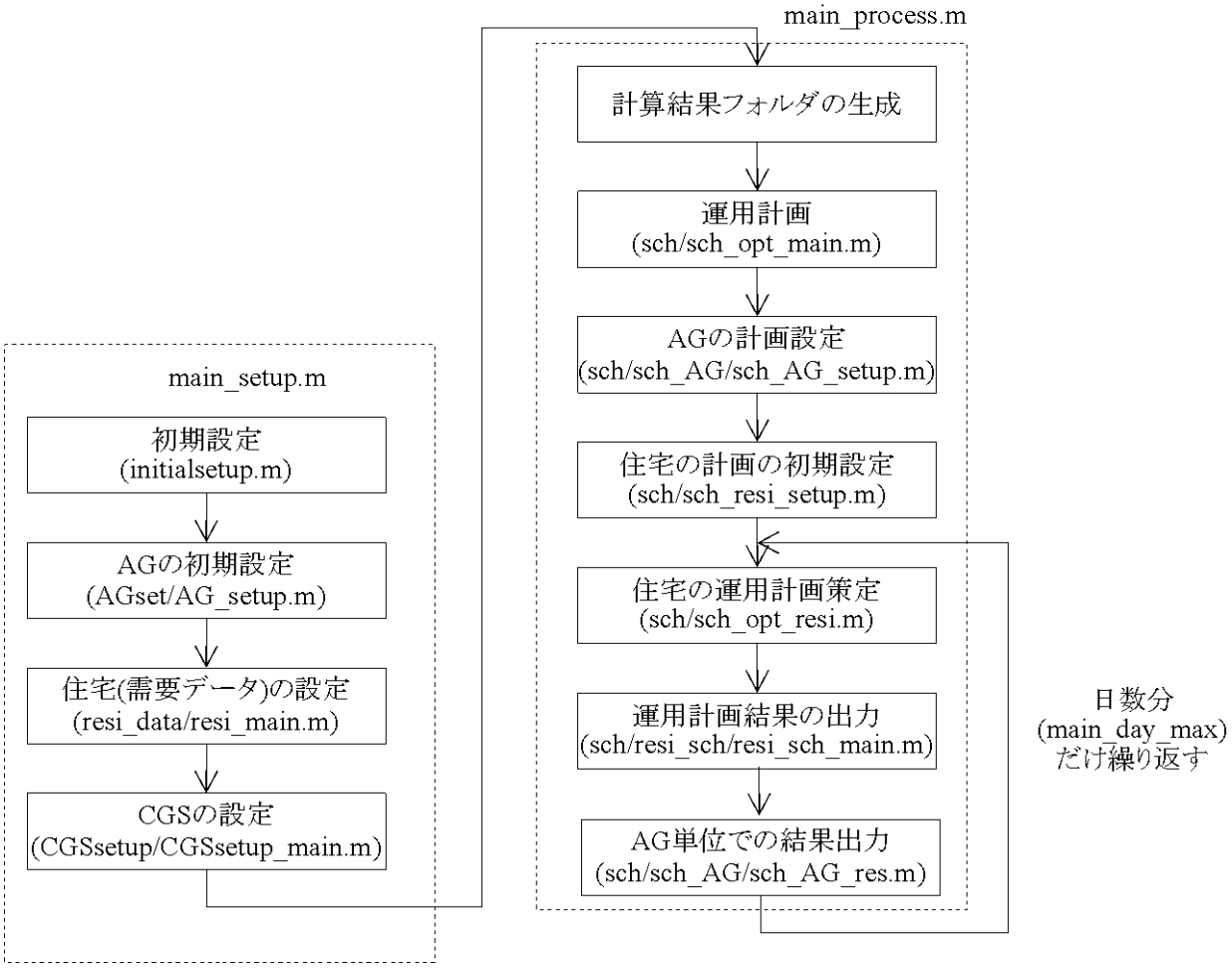
├──result　　：プログラム実行結果が集約されるフォルダ（自動生成）
│　├──”foldername“　　：”foldername”には任意の名前を設定できる
（キー入力で設定、何も入力しない場合は日付が入る）
│　├　└──input ：初期設定などを格納するフォルダ
│　├　└──output　　：試算結果(mat&エクセルファイル)を格納するフォルダ
│　├　　└──sch_result：住宅毎に運用計画策定結果(エクセル)を保続するフォルダ

2. プログラム内の各工程とフローチャート

各工程のデータ(mat)および工程間のデータの関係について本節で示す。なお、順番については、main.m(ルートフォルダ直下)の実行順に従う。

付録 CGS_図 1 は本プログラム(main.m)の主な構成について示している。図の左側は setup フォルダ内の工程を示しており、右側が実際の計算プロセスを示している。

なお，本プログラムは各工程(以下の図の各項目)において，”loop ”で始まる変数を除くすべての変数の初期化を行っている。



付録 CGS_図 1 main.m のフローチャート

3. プログラム内のデータと工程間のデータ依存の関係性

本プログラムの mat ファイルの生成および使用の関係性を示す。各工程で実行する際、'Use'と記載されている mat ファイルが無い場合は「変数が無い」というエラーが表示される可能性が高い。なお、"*"記号をつけているが、これは必要に応じて使用するものである。

mat_Table 19 mat ファイルの生成・使用の関係性(CGS)

		setupフォルダ				process/schフォルダ				
		initialsetup	AG_setup	resi_main	CGSsetup_main	sch_opt_main	sch_opt_f_resi.m	sch_opt_cons_resi.m	sch_AGope.m	resi_sch_main
Input フォルダ	initialsetup.mat	Create	Use	Use	Use	Use	Use	Use	Use	Use
	AG_data.mat		Create		Use	Use			Use	Use
	resi_data.mat			Create	Use	Use	Use	Use		Use
	CGS_data.mat				Create		Use*	Use*		Use*
Output フォルダ	sch_AG_data.mat					Create/Use	Use	Use	Update	Use
	sch_data.mat					Create/Use	Use	Use	Use	Use
	resi_sch.mat									Create/Use

4. 各工程の処理・変数一覧

入力フォルダ内の mat ファイル(setup フォルダ内)

初期設定(initialsetup)

mat_Table 20 initialsetup.mat

変数名	(配列の場合：配列数)	内容
main_loop_max	Value(自然数)	Main 文のループの最大数
main_day_max	Value(自然数)	Main 文のシミュレーション日数（試算日数）
tt_step	Value(自然数)	計算刻み幅
tt_max	Value(自然数)	計算回数
sch_tt_step	Vaule(自然数) ※24h×60min で割り切れるようにする	運用計画刻み時間 [min] ※現状では 10 分固定。
sch_tt_mat	1440/ sch_tt_step	運用計画時間帯数
rootfolderpath (rootabspath_main)	char	結果を出力するフォルダの文字列 (main ファイルの絶対参照)
foldername_input (foldername_output)	char	プログラム実行時の入力（出力）フォルダ

AG の初期設定(AG_set/AGset_main)

mat_Table 21 AG_data.mat

変数名	値(配列の場合：配列数)	内容
AG_LFC_T	Value	調整力の制御周期（継続時間の 2 倍）[min]
AG_resi_sch_num	Value(自然数)	実際に計画する住宅数
AG_resi_num	Value(自然数)	AG が統括する住宅数（計画数の整数倍） ※本プログラムでは(AG_resi_sch_num と同一)

需要家の各種データ設定(resi_data/resi_main)

mat_Table 22 resi_data.mat

変数名	値(配列の場合：配列数)	内容
resi_num	Value(自然数)	想定する住宅数(AG_resi_sch_num と同一)
resi_data_device	6×住宅数	需要家の所有機器リスト（持っている場合：1） 1 行目:共通(必ず 1),2 行目:CGS,3 行目:EV, 4 行目:ESS,5 行目:未確定, 6 行目:PV に対応可能 例.CGS を所持する住宅[1;1;0;0;0;0]
resi_unit_rev_allow	Unit8 (0or 1) 住宅数×1	住宅内機器の発電電力による逆潮流許可のバイナリ(1：許可、0：禁止)
resi_xlsfile_list	Value 住宅数×1	使用する需要データ番号 全住宅で同一の番号にすると、需要が同一
resi_data_day_list	試算日数×3（年、月、日）	各試算日(loop_day)で使用するデータの日
resi_Cost_Pow	Value	電力購入単価 [¥/Wh]
resi_Price_Esell	Value	電力売電単価 [¥/Wh]
resi_Cost_GAS	Value	ガス購入単価 [¥/Wh]
resi_cost_boiler	Value	補助熱源の効率を考慮した熱供給単価[¥/Wh]
resi_Heat_COP	Value	熱系機器の成績係数(1.0)
resi_Pow_fix	Value(住宅数×1)	各時間帯の最低買電電力 [kW] （通常は設定不要） ただし、計算中では需要以上に変換される。
resi_data_day_HW (Heat,Pow,Yuha)	運用計画時間帯数×住宅数×試算日数	各時間帯・各住宅・各日の（給湯・熱・電力・湯張り）データ [kW]

※の単位は

CGS 機器設定(CGSsetup)

mat_Table 23 CGS_data.mat

変数名	(配列の場合：配列数)	内容
CGS_cap	Value	CGS 定格容量
CGS_P_min (max)	Value	CGS 最低/最大出力
CGS_HST_max	Value	CGS 蓄熱量上限
CGS_Heat_COP	Value	CGS の熱交換器の成績係数（1.0）
CGS_cost_quad (liniear,cons)	Vaule	CGS のガス消費単価の 2 次 (線形,定数項) ※の単位は[¥/kWh/sch_tt_step]
CGS_Heat_linear (cons)	Vaule	CGS の排熱量の線形,定数項 [kWh]
CGS_LFC_deltaT	Vaule	CGS の調整力提供の継続時間
CGS_HSplus_perkW	Value	CGS 出力調整時の排熱量の変化 [kWh/kW/sch_tt_step] (≠kW・h)

出力フォルダ内の mat ファイル(process/sch フォルダ内)

計画結果(sch) 主に sch_resi_setup.m 内

mat_Table 24 sch_data.mat

変数名	(配列の場合：配列数)	内容
sch_m_calctime	試算日数	各試算日の計算時間 (全住宅分)
sch_m_opt_resi_num	Value(自然数)	1 回の最適化で解く住宅数
sch_m_opt_daymax	Value(自然数)	1 回の最適化で解く最大日数
sch_m_opt_N_dev	6 行(自然数)	各機器の変数の数のリスト 各行は resi_data_device と同一
sch_m_opt_ctype_per_dev	6 行(Char)	各機器の変数の型リスト(詳細は後述) ‘C’が連続値、’B’がバイナリを示す。
Sch_m_opt_N_PER_resi	住宅数×6	各住宅・各機器の開始を指定するための番号 （各変数の”開始-1”を示す）
sch_m_solve_para_resi	最大機器変数×計画時間帯数×住宅数	最適化された変数 x を一時的に格納しておく変数（項目毎に分けたのが resi_sch.mat）
sch_m_HST_init (sch_m_HST_last)	住宅数×試算日数	CGS を所持している住宅の計画開始・終了値の蓄熱量 [kWh]
sch_m_HST_maxdaylast	Value	試算最終日の最終蓄熱量[kWh]
sch_m_LFC_resi	時間帯数×住宅数×調整力の各方向(2)	（一時保存）方向別の調整力の保存 sch_AG_data.mat における処理に使用※
sch_m_totalcost_resi	住宅数×試算日数	各住宅・試算日の運用コスト [¥]※
sch_m_incen	住宅数×試算日数	各住宅・試算日の報奨金 [¥]※
sch_m_cost_resi	住宅数×試算日数	各住宅・試算日の運転コスト [¥]※

AG 内の計画結果(sch_AG) 主に sch_AG_setup.m 内

mat_Table 25 sch_AG_data.mat

変数名	(配列の場合：配列数)	内容
sch_AG_LFC_req_tar	計画時間帯数×試算日数	必要調整量(必要調整力×半周期) ※評価時に使用
sch_AG_LFC_req_up(dn)	計画時間帯数	必要調整力
sch_AG_LFC_req_resi_up(dn)	計画時間帯数×住宅数	各需要家に提示する必要調整力の残存分
sch_AG_resi_list	住宅数×試算日数	AG が需要家に提示する順番リスト （プログラム内では計画する順番リスト）
sch_AG_reg_totalrec	住宅数×試算日数	試算日の前日までの報奨金総額(順番リスト更新時に使用)
sch_AG_inc_DkW_up (dn)	計画時間帯数×試算日数	各時間帯の調整力報奨金単価 [¥/kW・h]
sch_AG_res_LFC_dkW_up(dn)	計画時間帯数×住宅数	各時間・住宅から調達した調整力[¥/kW・h]
sch_AG_res_LFC_list	15×試算日数	AG 単位の各日の試算結果 行数は以下を示す。 調整量(1:増加、2:減少、3:合計) [kW・h] 調整力提供時間(4:増加、5:減少、6:両方向) [min], 調整力調達率(7:増加、8:減少、9:両方向)[%] 10:計算時間、11:全需要家供給コスト、12:報奨金合計 13:全需要家トータルコスト、14:最小、15 最大

住宅単位の計画結果主に sch_res_setup.m 内

mat_Table 26 sch_resi.mat

変数名	(配列の場合：配列数)	内容
sch_resi_cost_list	6 行×住宅数	各需要家のコストの内訳 [¥]
sch_resi_ene_list	6 行×住宅数	各需要家のエネルギー供給の内訳 [kWh] (6 行目は CGS 稼働時間)
sch_resi_Epur(Esell)	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の買電(売電)量 [Wh]
sch_resi_boiler(boilerHout)	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の補助熱源の出力（熱出力）[Wh]
sch_resi_dPup(dn)_dT	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の調整量 [kW・h]
sch_resi_Heat (Pow)	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の給湯・電力需要 [Wh]
sch_resi_Epur(Esell)_unit	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の売電・買電バイナリ
sch_resi_Heater_unit	計画時間帯数×住宅数×試算日数	逆潮流禁止時の電気ヒータ使用バイナリ
sch_resi_CGS_dP	計画時間帯数×住宅数×試算日数	CGS の出力変動(2 段階目の最適化で最小化をしている)
sch_resi_CGS_unit	計画時間帯数×住宅数×試算日数	CGS の稼働バイナリ
sch_resi_CGSstartup(stop)_unit	計画時間帯数×住宅数×試算日数	CGS の起動(停止)バイナリ
sch_resi_CGSoutput	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の CGS(発電)出力 [%] (定格が 100%)
sch_resi_HST	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の蓄熱量[Wh]
sch_resi_HST_out	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の蓄熱利用量[Wh]
sch_resi_LFC_CGS_dPup(dn)	計画時間帯数×住宅数×試算日数	各住宅･各時間帯の CGS の kW 調整代[kW]
sch_resi_LFC_CGS_dPup(dn)dT	計画時間帯数×住宅数×試算日数	各住宅・各時間帯の調整量 [kW・h] (sch_resi_dPup(dn)_dT と同一)
sch_resi_table_name_com(CGS)	各機器の変数の数×文字数	各変数の名前（エクセル出力値の名前になります)
sch_resi_table_list	機器の変数の最大数×機器の数	エクセルに出力するパラメータを指定 1 で出力。多すぎると全て出力されない。

5. 最適化とソルバーについて

本研究では、最適化ツールである CPLEX を用いて、最適化問題の求解を行っている。
その中でも特に使用しているのが、混合整数線形計画化問題(MILP)の求解ツールであり、**cplexmilp 関数を使用している。**
この関数に関する部分は IBM の公式サイトに記載しているため、適宜参照すると良い。
https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/refmatlabcplex/html/cplexmilp-m.html

プログラムを使用するにあたり、変更すべき点としては、
目的関数：opt_f
不等式制約：Aineq、bineq
等式制約：Aeq,beq
決定変数 x の上下限：ub,lb
決定変数 x の型；ctype
であり、これらを変更すると解 x が変化する。
以下の表は、各関数の設定している m ファイルおよびフォルダの一覧を示している。

付録表 最適化問題設定ファイル一覧

	変数名	内容
目的関数、決定変数 x の型	opt_f、ctype	sch_opt_f_resi.m
不等式制約	Aineq、bineq	取りまとめ関数は sch_opt_cos_resi.m 各変数への代入は sch_opt 内のフォルダで各項目（電力・熱需要や各機器）単位で設定。
等式制約	Aeq,beq	
決定変数 x の上下限	ub,lb	
CPLEX のオプション		sch_opt_process.m

決定変数 x の規則について

このプログラムでは、住宅に設置する機器を自由に追加したり、住宅毎に変更したりできるようになっているため、やや複雑になっている。

まず、x の変数の数は[各計画時間帯の配列数(後述)×全計画時間数×1 回で計画する日数×1 回で計画する住宅数]である。x の変数の順序としては、上記の式がそのまま優先順位に対応している。

例えば、計画時間帯を 10 分とし、1 回の最適化で 2 日、2 軒の住宅（住宅 1：各時間帯の変数が 20 個、住宅 2：各時間帯の変数が 10 個）の場合には

x=[1 1 1 1 1 1 1 … 1 1 1 1 1 1 1 1 1 1… 1 1 1 1 1 1 1 1 1 1… 1 1 1 1 1 1 1 1 1 1….1]

1 軒目の最初の時間帯の変数(20)

1 軒目の 1 日分の変数(1 から 2880:20×144)

1 軒目の 2 日目の変数(2881 から 5760)

2 軒目の最初の時間帯の配列数 10 個(5761 から 5770)

（以下、同様）

変数の総数は住宅 1(5760)+住宅 2(2880)の 8640 個

というような並び順となるが、前述したように、複数日・複数住宅を一回の最適化では解くことは計算時間がかかってしまうため、おすすめはできない。

各時間帯の変数は、需要家の所持している機器の数によって変化する。そのため、本プログラムでは、機器別の変数の数を用いて、住宅別に変数を自由に変更できるようにしている。

これらの機器の対応関係が正しく取れるように、最適化問題の設定関連のファイル内で、

バイアス用の変数(inc/prev + com/ CGS)を設定している。これは、各住宅・各時間帯・各日付で変化するinc/prev変数に各機器の変数(固定値：com/CGS)を加算することで、対応する変数の番号を指定することができる。

例えば、上記の例で2軒目の住宅の2日目の0:00～0:10の開始番号は

現在時間：inc = 5760(住宅1)+10(住宅2の1時間帯の変数の数)×144(1日目)=7200

1時間前：prev= 5760(住宅1)+10(住宅2の1時間帯の変数の数)×(144-1) = 7190

となる。

バイアス用の変数(inc/prev)の値(上記の例の7200)は自動的に計算されるため、このバイアス用の変数を基準として、何の機器、何番目の数字が何に対応しているかだけ把握すれば良い。

※例.「inc+CGS+1」が「現在時間のCGSに関する変数の1番目」(CGS出力),

「prev+com+2」が「過去時間のcomに関する変数の2番目」(売電), など

次節では、各機器、各数字が何の要素を示しているかの一覧表をまとめている。

決定変数 x の変数リスト

本プログラムでは、各機器の変数で開始番号を特定させており、変数+番号で各要素を示している。各機器(共通:com,CGS,EV,ESS)の変数の数は以下の表の通りである。

付録表 各変数の一覧と変数の数

変数名	変数の数	内容
com（必須の項目）	8	買電・売電などの住宅共通の項目
CGS	12	CGS

各要素のリストと結果(sch_resi.mat)の変数との対応関係を以下の表で示す。

付録表 com の変数一覧

変数	sch_resi.mat の変数	変数型 (C/B)	内容
com+1	sch_resi_Epur	C	買電電力 [Wh]
com+2	sch_resi_Esell	C	売電電力 [Wh]
com+3	sch_resi_Epur_unit	B	買電バイナリ(1:買電,0:売電)
com+4	sch_resi_Epur_unit	B	売電バイナリ ※定式化上で 3 と同時禁止
com+5	sch_resi_boiler	C	補助熱源の出力 [Wh]
com+6	sch_resi_boilerHout	C	補助熱源の暖房出力 [Wh]
com+7	sch_resi_dPup_dT	C	住宅内の全機器の出力増加方向の調整量 [kW・h]
com+8	sch_resi_dPdn_dT	C	住宅内の全機器の出力減少方向の調整量 [kW・h]

付録表 CGS の変数一覧

変数	sch_resi.mat の変数	変数型 (C/B)	内容
CGS+1	sch_resi_CGSoutput	C	CGS 発電電力 [%]
CGS+2	sch_resi_CGS_unit	B	CGS の起動停止バイナリ(1:起動,0:停止)
CGS+3	sch_resi_Heater_unit	B	CGS の逆潮流禁止時の電気ヒータ使用バイナリ
CGS+4	x_map_CGS_dP	C	CGS 出力変動
CGS+5	sch_resi_HST_out	C	蓄熱槽の蓄熱利用量 [Wh]
CGS+6	sch_resi_HST	C	蓄熱槽内の蓄熱量 [Wh]
CGS+7	sch_resi_CGSstartup_unit	B	CGS の停止→起動動作バイナリ
CGS+8	sch_resi_CGSstop_unit	B	CGS の起動→停止動作バイナリ
CGS+9	sch_resi_reg_CGS_dPup	C	CGS の出力増加方向の調整代 [kW]
CGS+10	sch_resi_reg_CGS_dPdn	C	CGS の出力減少方向の調整代 [kW]
CGS+11	sch_resi_LFC_CGS_dPupdT	C	CGS の出力増加方向の調整量 [kW・h]
CGS+12	sch_resi_LFC_CGS_dPdndT	C	CGS の出力減少方向の調整量 [kW・h]

6. 各プログラム(m ファイル)の関係図

メインプログラム(root フォルダ)

main.m
├──main_setup.m
│　　├──initialsetup.m
│　　├── m_run.m(各プログラムフォルダへ移動)
├──main_process.m
└──takeover

設定(setup フォルダ)

main.m
├──AGset
│　　├──AG_setup.m
├──resi_data
│　　├──resi_main.m
│　　├──resi_readwrite.m
│　　├──resi_mat (需要データ格納)
├──CGSsetup
│　　├── CGSsetup_main.m

計算(process フォルダ)

process

```
├── common
│   ├──CPLEX
├── sch
│   ├──sch_opt_main.m
│   ├──sch_opt_ini.m
│   ├──sch_setup.m
│   ├──sch_disp.m
│   ├──sch_opt_f_resi.m
│   ├──sch_opt_cons_resi.m
│   ├──sch_opt_process.m
│   ├──sch_opt_res.m
│   ├──sch_AGope.m
│   ├──sch_AG
│       ├──sch_AG_setup.m
│       ├──sch_AG_res.m
│   ├──sch_opt (制約式の設定フォルダ)
│       ├──inc_and_perv_con.m
│       ├──sch_opt_Heat.m
│       ├──sch_opt_Pow.m
│       ├──sch_opt_CGS.m
│       ├──sch_opt_HST.m
│       ├──sch_opt_reg_HST.m
│       ├──sch_opt_reg.m
│       ├──sch_opt_CGSope.m
│       ├──sch_opt_cost.m
│       ├──sch_opt_reg_m.m
```

```
%{
%%%%%%%%%%%%%%%%%%
需給調整に貢献するCGSの運用のプログラム
%%%%%%%%%%%%%%%%%%
%}

%% 各種設定(setup) main_setup;
main_setup;

%% 実際の制御・出力(process)
main_process; %メインプロセス
```



```
%% main.mの設定部分
%% 各部分をコメントアウトすることで設定せずに実行可能

clear; %初期化
clc; %画面クリア

%% プログラム全体の初期設定
initialsetup();

%% AGの初期設定
readtry_filename=strcat('takeover/AG_data.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','AGset','AG_setup.m'); %%系統モデル設定
m_run;%引継ファイルの有無チェックと実行

%% 住宅の設定
readtry_filename=strcat('takeover/resi_data.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','resi_data','resi_main.m'); %%系統モデル設定
m_run;%引継ファイルの有無チェックと実行

%CGS
readtry_filename=strcat('takeover/CGS_data.mat'); %繰越フォルダにmatファイルがあるかどうか
m_run_name=fullfile('setup','CGSsetup','CGSsetup_main.m'); %%系統モデル設定
m_run;%引継ファイルの有無チェックと実行
```

```
%{
 AGに関係する変数設定
%}

%% AGに関する設定
clear; %初期化
load('../../initialsetup.mat'); %初期設定ファイルの読み込み

AG_resi_sch_num =20; %計画需要家数（resi_num以下）
AG_resi_num =1*AG_resi_sch_num; %統括する需要家数

AG_LFC_T=10; %LFCの周期（分）

%% 出力

matname=strcat(foldername_input,'AG_data.mat');
save(matname,'AG_*');
```



```
%%%%負荷データ読み込み（10分値データ）
clear; %初期化
load('../../initialsetup.mat'); %初期設定ファイルの読み込み

%% 初期設定
resi_num=20; %想定する住宅数
temp_resi_day_Serno= datenum(2008,3,12);     %試算1日目のシリアル値
temp_resi_day_int=1;                        %データ取得日の間隔（ここを0にすることで同一の日の試算が可能）
resi_data_day_list=uint16(zeros(resi_num,3));

%%  需要家の所有機器リスト
resi_data_device=uint8(zeros(6,resi_num)); %CGS,EV,HP,Air_conのリスト
resi_data_device(1,:)=1; %共通(整合をとるために設定, 必ず1に設定)
resi_data_device(2,:)=1; %CGS

% 各種需要データ
resi_data_day_Pow   =zeros(144,resi_num,main_day_max);
resi_data_day_HW    =zeros(144,resi_num,main_day_max);
resi_data_day_Heat  =zeros(144,resi_num,main_day_max);
resi_data_day_Yuha  =zeros(144,resi_num,main_day_max);

%負荷特性のファイルが入っているフォルダ
temp_dir = 'load/home/' ;
temp_matdir = 'resi_mat/';

temp_xls_file=''; %需要負荷データのexcelファイル名
for temp_no=1:20
    if temp_no<=10
        temp_xls_file(temp_no,:)=sprintf('load06%02d-2008',temp_no);
    else
        temp_xls_file(temp_no,:)=sprintf('load07%02d-2008',temp_no-10);
    end
end
%}

%使用する負荷データリスト（需要家毎に設定）
resi_xlsfile_list = uint16(ones(resi_num,1)); %住宅間の比較をする場合
resi_xlsfile_list(1:resi_num,1) = 1:resi_num;

% CGSやPV等の分散電源による逆潮流許可のバイナリ
resi_unit_rev_allow=uint8(ones(resi_num,1)); %1で許可,0で不許可

%% 需要データ作成

for temp_resi_loop=1:resi_num

    matname=strcat(temp_matdir,temp_xls_file(resi_xlsfile_list(temp_resi_loop,1),:),'.mat');
    resi_readwrite; %需要データ(matファイル)の生成・読み込み

    for temp_day_loop=1:main_day_max

        temp_day_Serno_loop=temp_resi_day_Serno+temp_resi_day_int*(temp_day_loop-1); %シリアル値のまま↙
日付計算
        [temp_year,temp_month,temp_day,~,~,~]=datevec(temp_day_Serno_loop);          %シリアル値を月と↙
日付に分解
```

```
        resi_data_day_list(temp_day_loop,1:3)=[temp_year,temp_month,temp_day]; %試算日ループとデータの↙
紐づけ

        resi_data_day_Pow(:,temp_resi_loop,temp_day_loop) = resi_data_year_Pow(:,temp_day,temp_month);
        resi_data_day_HW(:,temp_resi_loop,temp_day_loop) = resi_data_year_HW(:,temp_day,temp_month);
        resi_data_day_Heat(:,temp_resi_loop,temp_day_loop) = resi_data_year_Heat(:,temp_day, ↙
temp_month);
        resi_data_day_Yuha(:,temp_resi_loop,temp_day_loop) = resi_data_year_Yuha(:,temp_day, ↙
temp_month);
    end

    clearvars resi_data_year_* %年間データを削除
end

%% 全需要家に共通する設定
resi_Cost_Pow = 29.72 / 1000;           %電力購入単価(¥/Wh)
resi_Price_Esell = 13 / 1000;           %電力売電単価(¥/Wh)
resi_Cost_GAS = 68.07 / 1000 / 12.5;    %ガス購入単価(¥/Wh)

resi_Heat_COP=1; %各機器の成績係数

temp_boiler_effe = 0.9;                     %ボイラの効率

resi_cost_boiler = resi_Cost_GAS / temp_boiler_effe; %ボイラー効率項

resi_Pow_fix=zeros(resi_num,1);
resi_Pow_fix(1,1)=0; %最低購入電力(負に設定すると逆潮流（売電）許可)

%% matファイル出力
matname=strcat(foldername_input,'resi_data.mat');
save(matname,'resi_*');
```



```
%% 需要データ(matファイル)の生成・読み込み
if exist(matname,'file')==0
    %負荷需要生成(xlsファイル読み込み)
    %需要負荷フォルダの参照（いじらない）
    temp_xls_filepass=strcat(temp_dir,temp_xls_file(resi_xlsfile_list(temp_resi_loop,1),:),'.xls');

    %% データ作成関連
    %---上から順に，「電力」「給湯」「暖房」「湯張り」のデータ
    resi_data_year_Pow = zeros(144,34,12); %24時間*6，日数（31日）+3（前2つが時，分，後ろ１つが合計），↙
月数
    resi_data_year_HW   = zeros(144,34,12); %24時間*6，日数（31日）+3（前2つが時，分，後ろ１つが合↙
計），月数
    resi_data_year_Heat = zeros(144,34,12); %24時間*6，日数（31日）+3（前2つが時，分，後ろ１つが合↙
計），月数
    resi_data_year_Yuha = zeros(144,34,12); %24時間*6，日数（31日）+3（前2つが時，分，後ろ１つが合↙
計），月数
    for temp_M=1:12
        %電力（4行目から147行目）
        resi_data_year_Pow(:,:,temp_M) = xlsread(temp_xls_filepass,temp_M,'4:147');
        %給湯（152行目から295行目）
        resi_data_year_HW(:,:,temp_M) = xlsread(temp_xls_filepass,temp_M,'152:295');
        %暖房（300行目から443行目）
        resi_data_year_Heat(:,:,temp_M) = xlsread(temp_xls_filepass,temp_M,'300:443');
        %湯張り（448行目から591行目）
        resi_data_year_Yuha(:,:,temp_M) = xlsread(temp_xls_filepass,temp_M,'448:591');
    end

    %前から2列（時刻データ）を削除
    resi_data_year_Pow(:,1:2,:) = [];
    resi_data_year_HW(:,1:2,:) = [];
    resi_data_year_Heat(:,1:2,:) = [];
    resi_data_year_Yuha(:,1:2,:) = [];

    % データサイズ削減（単精度へ変換）
    resi_data_year_Pow=single(resi_data_year_Pow);
    resi_data_year_HW=single(resi_data_year_HW);
    resi_data_year_Heat=single(resi_data_year_Heat);
    resi_data_year_Yuha=single(resi_data_year_Yuha);

    save(matname,'resi_data_year_*');
else
    load(matname);
end
```

```
% CGSに関する設定

%% データの入力・初期化
clear; %初期化
load('../../initialsetup.mat'); %初期設定ファイルの読み込み

matname=strcat(foldername_input,'AG_data.mat'); %AGデータ読み込み
load(matname);

matname=strcat(foldername_input,'resi_data.mat'); %負荷データ読み込み
load(matname);

%% 設定

CGS_cap = 700 / (60 / sch_tt_step); %定格出力(単位W)※1時間帯分！
CGS_P_min = 28.5714;                    %最低負荷率(%)
CGS_P_max = 100;                        %最高負荷率(%)

CGS_HST_max = 140000 * 20 * 0.001162;   %蓄熱槽の容量(Wh)

%%%各種成績係数
%熱交換器の成績係数
CGS_Heat_COP = 1;

%% 初期値からの計算部分
%%CGSのガス消費量(0～2次項)
CGS_cost_quad = resi_Cost_GAS * 2 * 0;
CGS_cost_linear = resi_Cost_GAS * 3.0997;
CGS_cost_cons = resi_Cost_GAS * 18.171;


%排熱に関する線形近似式の係数項（単位は「W」）
%y=CGS_Heat_linear*x+CGS_Heat_cons
CGS_Heat_linear =1.8491;
CGS_Heat_cons =-19.195;

CGS_LFC_deltaT = AG_LFC_T/2;
CGS_HSplus_perkW = 1*CGS_LFC_deltaT / sch_tt_step * (CGS_Heat_linear)/0.7*100;   %調整対応周期単位の熱調↙
整容量(kWh/kW)

%% 出力
matname=strcat(foldername_input,'CGS_data.mat');
save(matname,'CGS_*');
```



```
%{
メイプロセス(main_process)
%}
%% 初期設定
clc; %画面クリア
clear; %初期化
warning('off','all') %tablewriteによるエラー警告の非表示
addpath(strcat(pwd,'/process/common/CPLEX/x64_win64')) %共通部分のフォルダを実行パスに追加

%% 計算結果フォルダの生成
load('initialsetup.mat');
mkdir(rootfolderpath,'output/loop_result');

loop_day_time=tic;loopdaytime=toc(loop_day_time);

%% 計算プロセス
for loop=1:main_loop_max

    loopfoldername=strcat('loop',num2str(loop));%outputフォルダの名前(loop毎のデータ格納)
    fprintf('main process¥n %d loop¥n',loop);
    loop_day_time=tic; %計算時間計測

    for loop_day=1:main_day_max
        load('initialsetup.mat');
        %% 各需要家の運用計画
        fprintf('%dday shedule ¥n',loop_day);
        m_run_name=fullfile('process','sch','sch_opt_main.m');
        run(m_run_name);

        %% 計算経過状態画面出力
        clc; %画面クリア
        loopdaytime=toc(loop_day_time);
        fprintf('¥n %d loop %d day finish ¥n',loop,loop_day);
        fprintf('about %5.1fseconds/loop¥n',loopdaytime/loop_day);
        t1 = datetime('now','Format','yyyy/MM/dd HH:mm:ss'); %現在時間の取得
        t = t1 + seconds(main_day_max-loop_day)*loopdaytime/loop_day; %計算終了時間の推定
        disp(t);
        %}
    end

end
```

```
%最適化による運用計画
%% 初期設定
clearvars -except loop*; %ループ数以外初期化(削除)
load('../../initialsetup.mat');
matname=strcat(foldername_input,'AG_data.mat');     %AGの初期設定
load(matname);
matname=strcat(foldername_input,'resi_data.mat');   %需要家データ
load(matname);
matname=strcat(foldername_output,'sch_AG_data.mat'); %アグリゲータの計画設定
if loop_day==1 %初日に設定
    m_run_name=fullfile('sch_AG','sch_AG_setup.m');
    run(m_run_name);
else
    load(matname);
end
matname=strcat(foldername_output,'sch_data.mat');   %計画の初期設定
if loop_day==1 %初日に設定
    sch_setup;
else
    load(matname);
end

%% 最小運転コスト算出（調整力最大化時に使用,functype=0)
sch_disp; %表示関連
fprintf('%d day cost calculation¥n',loop_day);
functype=0;
%{
for temp_sch_opt_time=1:AG_resi_sch_num %最適化回数
    sch_opt_ini; %temp_HST_last_unitバイナリの決定・計画する需要家の決定
    [opt_f,ctype,~]=sch_opt_f_resi(temp_sch_opt_time,loop_day,functype,temp_opt_day_num); %目的関数とc-typeの設定
    trytime=1;
    %制約条件の設定
    [ Aeq,Aineq,beq,bineq,lb,ub ] = sch_cons_resi ( temp_sch_opt_time,loop_day,temp_HST_last_unit,Inf(AG_resi_sch_num,1),zeros(AG_resi_sch_num,1),temp_opt_day_num,functype,trytime );

    sch_opt_process;
    temp_cost_optday(sch_AG_resi_list(temp_sch_opt_time,loop_day),1)=fval;
    [opt_f,x]=sch_opt_res(temp_sch_opt_time,opt_f,x,loop_day,calc_time,functype,temp_opt_day_num); %目的関数と結果の処理
end
%}

%% 運用コスト最小化(functype=1)もしくは調整力最大化(functype=2), CGS出力変動最小化(functype=3)
sch_disp; %表示関連
fprintf('%d day schedule¥n',loop_day);
for temp_sch_opt_time=1:AG_resi_sch_num/sch_m_opt_resi_num %最適化回数
    functype=1; %最適化タイプ

    sch_opt_ini; %temp_HST_last_unitバイナリの決定・計画する需要家・試算日数に合わせて計画する日付を決定
    [opt_f,ctype,opt_H]=sch_opt_f_resi(temp_sch_opt_time,loop_day,functype,temp_opt_day_num); %目的関数とc-typeの設定 （調整力最大化）

    % 浮動小数点などの理由で解けない場合があるため，意図的に制約条件（コスト）を緩和（解けない場合のみ繰
```



```
り返し）
    x=[];trytime=1;
    while (isempty(x) == 1 && trytime<5)
        %制約条件の設定
        [ Aeq,Aineq,beq,bineq,lb,ub ] = sch_cons_resi ( temp_sch_opt_time,loop_day,temp_HST_last_unit,temp_cost_optday,temp_incen,temp_opt_day_num,functype,trytime );
        sch_opt_process;
    end
    %{
    %調整力最大化時にfunctype3を実行（ここのコメントを外す）
    if abs(fval)>0
        if functype==1
            temp_cost_optday(sch_AG_resi_list(temp_sch_opt_time,loop_day),1)=fval;
        elseif functype==2
            temp_incen(sch_AG_resi_list(temp_sch_opt_time,loop_day),1)=fval;
        end
        functype=3;
        %temp_incen(sch_AG_resi_list(temp_sch_opt_time,loop_day),1)=fval;

        [opt_f,ctype,opt_H]=sch_opt_f_resi(temp_sch_opt_time,loop_day,functype,temp_opt_day_num); %目的関数とc-typeの設定 （調整力最大化）

        % 浮動小数点などの理由で解けない場合があるため，意図的に制約条件（コスト）を緩和（解けない場合のみ繰り返し）
        x=[];trytime=1;
        while (isempty(x) == 1 && trytime<5)
            %制約条件の設定
            [ Aeq,Aineq,beq,bineq,lb,ub ] = sch_cons_resi ( temp_sch_opt_time,loop_day,temp_HST_last_unit,temp_cost_optday,temp_incen,temp_opt_day_num,functype,trytime );
            sch_opt_process;
        end
    end
    %}
    for temp_opt_no=1:sch_m_opt_resi_num %1回の最適化の住宅数
        temp_sch_no=temp_opt_no+sch_m_opt_resi_num*(temp_sch_opt_time-1);     %temp_sch_no:計画している優先順位
        temp_no=sch_AG_resi_list(temp_sch_no,loop_day);                       %temp_no:計画する需要家

        [opt_f,x]=sch_opt_res(temp_no,opt_f,x,loop_day,calc_time,functype,temp_opt_day_num); %最適化結果の格納
        sch_AGope(temp_no,loop_day,temp_sch_no); %AGの動作:必要調整力の計算
    end
end

%% 出力
m_run_name=fullfile('sch_res','sch_res_main.m');%各需要家の結果とエクセル出力
run(m_run_name);

m_run_name=fullfile('sch_AG','sch_AG_res.m');%AG（全体）の結果
run(m_run_name);
```

```
function [ inc,prev ] = inc_and_perv_con( sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi )
%%  変数行列のバイアス設定(詳細はプログラム仕様に記載)

% 需要家バイアス
if temp_opt_no>1
    inc_resi=sum(N_COD_resi(1:temp_opt_no-1,1));
else
    inc_resi=0;
end

% 日付バイアス
if temp_day_num>1
    inc_day=(temp_day_num-1)*N_COD_day;
else
    inc_day=0;
end

%現時刻・対象需要家の開始変数番号
inc = N_PER * (sch_tt-1)+inc_day+inc_resi;

if sch_tt==1 && temp_day_num==1
    prev = 0+inc_day+inc_resi; %このときinc_dayは0であるが，わかりやすくするために記載
else
    prev = N_PER * (sch_tt-2)+inc_day+inc_resi; %2日目の最初時間(0:00)のprevが1日前の最終時間になる
end
end
```



```
%% 熱に関する制約

%熱に関する制約
Aplus = zeros(1*sch_tt_max,N_COD); %補助熱源の下限制約
bplus = zeros(1*sch_tt_max,1);
Aeqplus = zeros(3*sch_tt_max,N_COD);
beqplus = zeros(3*sch_tt_max,1);
%1:H_heat_demandに関する制約式 2:%H_HW_demandに関する制約式 3:CGSの蓄熱槽に関する制約式

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイ↙
アスの計算

    %1:H_heat_demandに関する制約式
    Aeqplus(sch_tt,inc+com+6) = resi_Heat_COP; beqplus(sch_tt) = resi_data_day_Heat(sch_tt,temp_no, ↙
loop_day);

    %2:H_HW_demandに関する制約式
    Aeqplus(sch_tt+sch_tt_max,inc+com+5) = resi_Heat_COP; Aeqplus(sch_tt+sch_tt_max,inc+com+6) = - ↙
resi_Heat_COP;
    beqplus(sch_tt+sch_tt_max) = resi_data_day_HW(sch_tt,temp_no,loop_day)+resi_data_day_Yuha(sch_tt, ↙
temp_no,loop_day);

    Aplus(sch_tt,inc+com+5)=-1;Aplus(sch_tt,inc+com+6)=1; %補助熱源の下限制約

    if resi_data_device(2,temp_no)==1 %CGS
        Aeqplus(sch_tt+sch_tt_max,inc+CGS+5) = CGS_Heat_COP;%2:H_HW_demandに関する制約式

        %3:蓄熱槽のエネルギーに関する制約式
        Aeqplus(sch_tt+2*sch_tt_max,prev+CGS+6) = 1;
        Aeqplus(sch_tt+2*sch_tt_max,inc+CGS+6) = -1;
        Aeqplus(sch_tt+2*sch_tt_max,inc+CGS+5) = -1;
        if sch_tt == 1&& temp_day_num==1 %最初の時間の場合、下限分をあらかじめ足しておく
            HST_init=sch_m_HST_init(temp_no,loop_day);
            beqplus(sch_tt+2*sch_tt_max) = beqplus(sch_tt+2*sch_tt_max) - HST_init;
        end
        Aeqplus(sch_tt+2*sch_tt_max,inc+CGS+1) = CGS_Heat_linear;
        Aeqplus(sch_tt+2*sch_tt_max,inc+CGS+2) = CGS_Heat_cons;
    end
end

%CPLEXの変数への追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);

Aeq = vertcat(Aeq,Aeqplus);
beq = vertcat(beq,beqplus);
```

```
%% 電力に関する定式化
Aplus = zeros(6*sch_tt_max,N_COD);
bplus = zeros(6*sch_tt_max,1);
Aeqplus = zeros(sch_tt_max,N_COD);
beqplus = zeros(sch_tt_max,1);
%Aeq 1:電力の需給平衡に関する制約,
%1:購入電力に関する制約,2:買電電力バイナリ,3:売電に関する制約,4:買電バイナリ、5・6:電気ヒータに関する制約,

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算

    %最低購入電力量(逆潮流を認める場合は0にしないと矛盾が生じる)
    temp_Pow_fix=min(resi_Pow_fix(temp_no,1)/(60/sch_tt_step),resi_data_day_Pow(sch_tt,temp_no, loop_day));

    %1:電力の需給平衡制約
    Aeqplus(sch_tt,inc+com+1) = 1;   Aeqplus(sch_tt,inc+com+2) = -1*temp_rev_unit; %逆潮流許可しない場合は0になる
    beqplus(sch_tt) = resi_data_day_Pow(sch_tt,temp_no,loop_day);

    % 購入電力に関する制約
    Aplus(sch_tt,inc+com+1) = -1;bplus(sch_tt,1)=-1.0*temp_Pow_fix;                    %1:買電
    Aplus(sch_tt+sch_tt_max,inc+com+1) = 1;Aplus(sch_tt+sch_tt_max,inc+com+3) = -1000;  %2:買電 (1000は非常に大きい数値として設定)
    Aplus(sch_tt+2*sch_tt_max,inc+com+2) = 1;Aplus(sch_tt+2*sch_tt_max,inc+com+4) = -1000;  %3:売電
    Aplus(sch_tt+3*sch_tt_max,inc+com+3)=1; Aplus(sch_tt+3*sch_tt_max,inc+com+4)=1;bplus(sch_tt+3*sch_tt_max)=1; %4:同時売電・買電禁止制約

    if  resi_data_device(2,temp_no)==1 %CGS
        Aeqplus(sch_tt,inc+CGS+1) = CGS_cap / 100; %需給平衡に関する制約

        %% 逆潮流禁止時に適用される定式化
        %
        if temp_rev_unit==0
            Aeqplus(sch_tt,inc+CGS+3) = ((resi_data_day_Pow(sch_tt,temp_no,loop_day) -temp_Pow_fix - CGS_P_min * (CGS_cap / 100)));

            %CGS用熱ヒータの利用バイナリ (3)
            %修正 20170717
            Aplus(sch_tt+4*sch_tt_max,inc+CGS+3) = 1; Aplus(sch_tt+4*sch_tt_max,inc+CGS+1) = 1 / (CGS_P_max - CGS_P_min);
            bplus(sch_tt+4*sch_tt_max) = CGS_P_max / (CGS_P_max - CGS_P_min );

            Aplus(sch_tt+5*sch_tt_max,inc+CGS+3) = -((CGS_P_min) * ( CGS_cap / 100 )  +temp_Pow_fix - resi_data_day_Pow(sch_tt,temp_no,loop_day));
        end
        %}
    end
end

%% CPLEXの変数に追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);
```



```
Aeq = vertcat(Aeq,Aeqplus);
beq = vertcat(beq,beqplus);
```

```
% CGS出力上下限と調整力・調整量に関する制約
Aplus = zeros(5*sch_tt_max,N_COD);
bplus = zeros(5*sch_tt_max,1);
%1: 出力増加方向の調整力,2:減少方向 3:増加方向の逆潮流の考慮,4:調整幅と調整力の変換（減少方向） 5:減少方向

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算

    %1:増加方向の調整力
    Aplus(sch_tt,inc+CGS+2) = -CGS_P_max;
    Aplus(sch_tt,inc+CGS+9) = 1/(CGS_cap*(60/sch_tt_step)/1000)*100; %調整幅_単位は[kW]であり，%に換算
    Aplus(sch_tt,inc+CGS+1)  = 1;

    %2:減少方向の調整力
    Aplus(sch_tt+sch_tt_max,inc+CGS+2) = CGS_P_min;
    Aplus(sch_tt+sch_tt_max,inc+CGS+10) = 1/(CGS_cap*(60/sch_tt_step)/1000)*100; %調整幅_単位は[kW]であり，%に換算
    Aplus(sch_tt+sch_tt_max,inc+CGS+1)  = -1;

    %3:上げ代分を確保するときは需要を超えてはいけない（逆潮流禁止時）
    %
    if temp_rev_unit==0
        Aplus(sch_tt+2*sch_tt_max,inc+com+1) = -1;
        Aplus(sch_tt+2*sch_tt_max,inc+com+7) = 1000/6; %ここだけコメントアウトすると調整力発動時のみ逆潮流許可
        ub(inc+com+2,1)=0;
        bplus(sch_tt+2*sch_tt_max,1)     = -temp_Pow_fix;
    end
    %]

    % 調整代(kW)と調整量(kW・h)の関係式
    %4:増加方向の調整力（量）変換
    Aplus(sch_tt+3*sch_tt_max,inc+CGS+11) = 1;  %増加方向の調整量
    Aplus(sch_tt+3*sch_tt_max,inc+CGS+9) = -1*CGS_LFC_deltaT/60;  %LFC量（kW・hに換算）

    %5:減少方向の調整力（量）変換
    Aplus(sch_tt+4*sch_tt_max,inc+CGS+12) = 1;  %増加方向の調整量
    Aplus(sch_tt+4*sch_tt_max,inc+CGS+10) = -1*CGS_LFC_deltaT/60;  %LFC量（kW・hに換算）
end

% CPLEXの変数への追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);
```



```
%% 蓄熱槽(HST)に関する制約(1時間断面)
Aplus = sparse(2*sch_tt_max+1,N_COD);
bplus = zeros(2*sch_tt_max+1,1);
%1:出力増加方向の調整時の蓄熱量,2:出力減少方向の調整時の蓄熱量の制約

for sch_tt=1:sch_tt_max

    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算

    %1:出力増加方向の調整時の蓄熱量
    Aplus(sch_tt,inc+CGS+9)=CGS_HSplus_perkW;
    Aplus(sch_tt,inc+CGS+6)=1;
    bplus(sch_tt)=CGS_HST_max;

    %2:出力減少方向の調整時の蓄熱量の制約
    Aplus(sch_tt+sch_tt_max,inc+CGS+10)=CGS_HSplus_perkW;
    Aplus(sch_tt+sch_tt_max,inc+CGS+6)=-1;
    bplus(sch_tt+sch_tt_max)=0;

    %蓄熱槽・貯湯槽の容量に関する制約
    ub(inc+CGS+6) = CGS_HST_max;
    lb(inc+CGS+6) = 0;
end

%% 蓄熱槽の最終値に関する制約

if HST_last_unit==1 && loop_day == main_day_max %最後の日は最初の最初の蓄熱量以上になるようにする
    [inc,prev]=inc_and_perv_con(sch_tt_max,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算
    HST_last=sch_m_HST_maxdaylast;
    Aplus(2*sch_tt_max+1,inc+CGS+6) = -1; bplus(2*sch_tt_max+1) = -HST_last;
end

% CPLEX変数へ追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);
```

```
% CGS稼働・停止に関する制約
Aplus = zeros(3*sch_tt_max+3,N_COD); %1～sch_tt_max 起動・停止の同時禁止制約,sch_tt_max+1:起動・停止の↙
最低回数に関する制約
bplus = zeros(3*sch_tt_max+3,1);      %sch_tt_max+2:起動・停止の上限制約,sch_tt_max+3:起動時間の制約
Aeqplus = zeros(3*sch_tt_max+3,N_COD); %起動・停止回数の算出
beqplus = zeros(3*sch_tt_max+3,1);

%% CGS出力変動(逆潮流を禁止する場合は使用しない)

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイ↙
アスの計算

    Aplus(sch_tt,inc+CGS+4)=-1; Aplus(sch_tt,inc+CGS+1)=1;Aplus(sch_tt,prev+CGS+1)=-1;
    Aplus(sch_tt+sch_tt_max,inc+CGS+4)=-1; Aplus(sch_tt+sch_tt_max,prev+CGS+1)=1;Aplus ↙
(sch_tt+sch_tt_max,inc+CGS+1)=-1;
end
%}

%% CGSの稼働バイナリと起動・停止バイナリの関係式
Aeqplus(3*sch_tt_max+1,CGS+7) = 1; beqplus(sch_tt_max+1) = 0;%最初のdelta_onは0
Aeqplus(3*sch_tt_max+2,CGS+8) = 1; beqplus(sch_tt_max+2) = 0;%最初のdelta_offは0

for sch_tt=1:sch_tt_max-1
    [increment,previous]=inc_and_perv_con(sch_tt+1,temp_day_num,N_PER,N_COD_day,temp_opt_no, ↙
N_COD_resi); %時間バイアスの計算
    Aeqplus(sch_tt+2*sch_tt_max,increment+CGS+2) = 1;    Aeqplus(sch_tt+2*sch_tt_max,previous+CGS+2) = ↙
-1;
    Aeqplus(sch_tt+2*sch_tt_max,increment+CGS+7) = -1;    Aeqplus(sch_tt+2*sch_tt_max,increment+CGS+8) ↙
= 1;
end

for sch_tt=1:sch_tt_max
    [increment,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時↙
間バイアスの計算
    % CGSの起動・停止操作は2回までとする
    Aplus(3*sch_tt_max+2,increment+CGS+7) = 1;  Aplus(3*sch_tt_max+2,increment+CGS+8) = 1;   bplus ↙
(3*sch_tt_max+2) = 2;

    %CGSの起動時間を1日22時間までとする（1日に2時間は停止している）
    Aplus(3*sch_tt_max+3,increment+CGS+2) = 1;  bplus(3*sch_tt_max+3) = 22*60/sch_tt_step;
end

%CPLEX変数への追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);

Aeq = vertcat(Aeq,Aeqplus);
beq = vertcat(beq,beqplus);
```



```
%調整力発動時の影響を考慮(複数時間)
Aplus = zeros(2*sch_tt_max,N_COD);
bplus = zeros(2*sch_tt_max,1);

%1:出力増加方向の調整時の蓄熱量,2:出力減少方向の調整時の蓄熱量の制約
[inc_s,~]=inc_and_perv_con(1,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %sch_tt=1のバイアス↙
の計算

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイ↙
アスの計算

    %1:調整時の蓄熱の上限制約(sch_ttから計算)
    Aplus(sch_tt,(inc_s+CGS+11):N_PER:(inc_s+N_PER*sch_tt)) =  1*60/CGS_LFC_deltaT*CGS_HSplus_perkW; ↙
%増加方向 kW・h→kW→熱量(kWh)変換
    Aplus(sch_tt,(inc_s+CGS+12):N_PER:(inc_s+N_PER*sch_tt))=  -1*60/CGS_LFC_deltaT*CGS_HSplus_perkW; ↙
%減少方向 kW・h→kW→熱量(kWh)変換
    Aplus(sch_tt,inc+CGS+6) = 1;
    bplus(sch_tt,1) = CGS_HST_max ;

    %2:調整時の蓄熱の下限制約(sch_ttから計算)
    Aplus(sch_tt+sch_tt_max,(inc_s+CGS+11):N_PER:(inc_s+N_PER*sch_tt)) =  -1 ↙
*60/CGS_LFC_deltaT*CGS_HSplus_perkW;    %増加方向 kW・h→kW→熱量(kWh)変換
    Aplus(sch_tt+sch_tt_max,(inc_s+CGS+12):N_PER:(inc_s+N_PER*sch_tt))= ↙
1*60/CGS_LFC_deltaT*CGS_HSplus_perkW;    %減少方向 kW・h→kW→熱量(kWh)変換
    Aplus(sch_tt+sch_tt_max,inc+CGS+6) = -1;

end
% CPLEX変数へ追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);

%% 最後の時間は発動時の影響を含めた蓄熱量が最初の蓄熱量以上になるようにする

if HST_last_unit==1 && loop_day == main_day_max
    Aplus = zeros(1,N_COD);
    bplus = zeros(1,1);
    [inc,prev]=inc_and_perv_con(sch_tt_max,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間↙
バイアスの計算

    HST_last=sch_m_HST_maxdaylast; %蓄熱量の最終値

    Aplus(1,(inc_s+CGS+11):N_PER:(inc_s+N_PER*sch_tt_max)) =  -1*60/CGS_LFC_deltaT*CGS_HSplus_perkW; ↙
%増加方向 kW・h→kW→熱量(kWh)変換
    Aplus(1,(inc_s+CGS+12):N_PER:(inc_s+N_PER*sch_tt_max))=  1*60/CGS_LFC_deltaT*CGS_HSplus_perkW;    %↙
減少方向 kW・h→kW→熱量(kWh)変換

    Aplus(1,inc+CGS+6) = -1; bplus(1) = -HST_last;
    Aineq = vertcat(Aineq,Aplus);
    bineq = vertcat(bineq,bplus);
end
```

```
%% 調整量に関する制約
Aeqplus = zeros(2*sch_tt_max,N_COD);
beqplus = zeros(2*sch_tt_max,1);
%Aeq 1・2:各機器の調整力と全機器・全住宅の調整力の等式制約

for sch_tt=1:sch_tt_max
    [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算

    %各機器の調整力と全機器の調整力の等式制約
    Aeqplus(sch_tt,inc+com+7) = 1; %増加方向
    Aeqplus(sch_tt+sch_tt_max,inc+com+8) = 1; %減少方向

    if  resi_data_device(2,temp_no)==1 %CGS
        Aeqplus(sch_tt,inc+CGS+11) =-1;  %Aeq 1:増加方向CGS調整力 (kW・hに換算)
        Aeqplus(sch_tt+sch_tt_max,inc+CGS+12) = -1;  %Aeq:2減少方向CGS調整力 (kW・hに換算)
    end
end
Aeq = vertcat(Aeq,Aeqplus);
beq = vertcat(beq,beqplus);

%% CGS出力変動最小化時に適用(調整量の維持)
if functype==3 && temp_day_num==1
    Aplus = zeros(1,N_COD);
    bplus = zeros(1,1);
    for sch_tt=1:sch_tt_max
        [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算
        Aplus(1,inc+com+7)=-sch_AG_inc_DkW_up(sch_tt,loop_day);
        Aplus(1,inc+com+8)=-sch_AG_inc_DkW_dn(sch_tt,loop_day);

    end
    bplus(1) = incen(temp_no,1);
    Aineq = vertcat(Aineq,Aplus);
    bineq = vertcat(bineq,bplus);
end
```



```
%% 各需要家のコスト制約
Aplus = zeros(1,N_COD);
bplus = zeros(1,1);
%1:各種コスト

for temp_day_num=1:opt_day_num
    loop_day=sch_day+temp_day_num-1; %制約の日にち

    for sch_tt=1:sch_tt_max
        [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi); %時間バイアスの計算

        Aplus(1,inc+com+1) = resi_Cost_Pow; Aplus(1,inc+com+2) = -resi_Price_Esell; %買電・売電
        Aplus(1,inc+com+5) = resi_cost_boiler;                                      %補助熱源
        if functype==3
        Aplus(1,inc+com+7) = -sch_AG_inc_DkW_up(sch_tt,loop_day);                   %報奨金(増加)
        Aplus(1,inc+com+8) = -sch_AG_inc_DkW_dn(sch_tt,loop_day);                   %報奨金(減少)
        end

        if resi_data_device(2,temp_no)==1 %CGS
            Aplus(1,inc+CGS+1) = CGS_cost_linear; Aplus(1,inc+CGS+2) = CGS_cost_cons;%CGSの燃料費(固定・可変費)
        end
    end
end
bplus(1) = cost(temp_no,1); %調整力最大化or出力変動最大化時は∞

%CPLEXの変数への追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);
```

```
% 需要家が提供する調整力に関する制約(必要調整力以下に制約)
Aplus = zeros(2*sch_tt_max,N_COD);
bplus = zeros(2*sch_tt_max,1);
%1:出力増加方向の調整力提供の上限制約,2:出力減少方向の調整力提供の上限制約

%% 調整量に関する制約(制約は1日目のみ)
for sch_tt=1:sch_tt_max

    for temp_opt_no=1:temp_resi_num %複数需要家を対象とした最適化の場合はここが繰り返し
        temp_sch_no=temp_opt_no+temp_resi_num*(temp_sch_opt_time-1);     %計画している優先順位
        temp_no=sch_AG_resi_list(temp_sch_no,sch_day);                  %計画する需要家

        N_PER=double(sch_m_opt_N_PER_resi(temp_no,5));
        N_COD_day=double(N_PER*24*(60/sch_tt_step));

        [inc,prev]=inc_and_perv_con(sch_tt,temp_day_num,N_PER,N_COD_day,temp_opt_no,N_COD_resi);  %時間↙
バイアスの計算

        %1:出力増加方向の調整力提供の制約
        Aplus(sch_tt,inc+com+7) = 1;

        %2:出力減少方向の全調整力の制約
        Aplus(sch_tt+sch_tt_max,inc+com+8) = 1;

    end
    temp_opt_set=1:temp_resi_num;
    temp_sch_no=temp_opt_set+temp_resi_num*(temp_sch_opt_time-1);     %計画している優先順位”群”
    temp_no_set=sch_AG_resi_list(temp_sch_no,sch_day);                 %計画する需要家”群”
    bplus(sch_tt,1) = sum(sch_AG_LFC_req_resi_up(sch_tt,temp_no_set,sch_day));                %対象需要↙
家”群”の必要調整力（増加）
    bplus(sch_tt+sch_tt_max,1) =sum(sch_AG_LFC_req_resi_dn(sch_tt,temp_no_set,sch_day)) ;    %対象需要↙
家”群”の必要調整力（減少）
end

%CPLEXの変数への追加
Aineq = vertcat(Aineq,Aplus);
bineq = vertcat(bineq,bplus);
```



```
%% 入力
clearvars -except loop*; %ループ数以外削除

load('../../../initialsetup.mat');

matname=strcat(foldername_input,'resi_data.mat');   %需要データ
load(matname);

matname=strcat(foldername_input,'CGS_data.mat');   %需要データ
load(matname);

matname=strcat(foldername_input,'AG_data.mat'); %需要家データ読み込み
load(matname);

matname=strcat(foldername_output,'sch_data.mat');   %計画の初期設定
load(matname);

matname=strcat(foldername_output,'sch_AG_data.mat');   %計画の初期設定
load(matname);

matname=strcat(foldername_output,'sch_resi.mat');

if loop_day==1
    sch_res_setup;
else
    load(matname);
end

%% 計画結果の格納
sch_res_resi; %各住宅の結果まとめ
sch_res_day_xlswrite;     %1日結果のエクセル出力
```

```
%  結果の変数設定&変数リスト(loop_day==1のときのみ実行)
matname=strcat(foldername_output,'sch_resi.mat');

sch_resi_cost_list=zeros(6,AG_resi_sch_num,main_day_max);                    %供給コスト概要
sch_resi_ene_list=zeros(6,AG_resi_sch_num,main_day_max);                     %エネルギーバランス量一覧

%以下，各最適化の決定変数の格納(詳細は仕様に記載)
sch_resi_Epur       =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-1
sch_resi_Esell      =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-2
sch_resi_Epur_unit  =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-3
sch_resi_Esell_unit =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-4
sch_resi_boiler     =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-5
sch_resi_boilerHout =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-6
sch_resi_dPup_dT    =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-7
sch_resi_dPdn_dT    =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);         %com-8

sch_resi_Heat   =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);             %last
sch_resi_Pow    =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);             %last
% CGS関連
if max(resi_data_device(2,:))==1 %CGS
    sch_resi_CGSoutput  =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-1
    sch_resi_CGS_unit   =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-2
    sch_resi_Heater_unit=zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-3
    sch_resi_CGS_dP     =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-4
    sch_resi_HST_out    =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-5
    sch_resi_HST        =zeros(sch_tt_max,AG_resi_sch_num,main_day_max);     %CGS-6
    sch_resi_CGSstartup_unit=zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-7
    sch_resi_CGSstop_unit   =zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-8
    sch_resi_LFC_CGS_dPup   =zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-9
    sch_resi_LFC_CGS_dPdn   =zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-10
    sch_resi_LFC_CGS_dPupdT =zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-11
    sch_resi_LFC_CGS_dPdndT =zeros(sch_tt_max,AG_resi_sch_num,main_day_max); %CGS-12
end

%各変数の名前
sch_resi_table_name_com=char↙
('Epur','Esell','Epur_unit','Esell_unit','boiler','boilerHout','dPup_dT','dPdn_dT','Heat','Pow');

sch_resi_table_name_CGS=char('CGSoutput','CGS_unit','Heater_unit','HST_in','CGS_dP','HST',...
    'CGSstartup_unit','CGSstop_unit','CGS_dPup','CGS_dPdn','CGS_dPupdT','CGS_dPdndT');

%% エクセルに出力するパラメータ（多すぎるとテーブル出力できない）
sch_resi_table_list=ones(max(sch_m_opt_N_dev(:,1)),length(sch_m_opt_N_dev(:,1)));
sch_resi_table_list(3:4,1)=0;sch_resi_table_list(7:8,2)=0;       %書込しない

%% 保存
save(matname,'sch_resi_*');
```



```
%% 最適化結果の格納
for temp_no=1:length(sch_m_solve_para_resi(1,1,:))
    temp_N_PER=sch_m_opt_N_PER_resi(temp_no,6);
    temp_com=0;
    temp_CGS=sch_m_opt_N_PER_resi(temp_no,1);
    % 共通
    sch_resi_Epur(:,temp_no,loop_day)        =sch_m_solve_para_resi(temp_com+1,:,temp_no)';        %↙
com-1
    sch_resi_Esell(:,temp_no,loop_day)       =sch_m_solve_para_resi(temp_com+2,:,temp_no)';        %↙
com-2
    sch_resi_Epur_unit(:,temp_no,loop_day)   =sch_m_solve_para_resi(temp_com+3,:,temp_no)';        %↙
com-3
    sch_resi_Esell_unit(:,temp_no,loop_day)  =sch_m_solve_para_resi(temp_com+4,:,temp_no)';        %↙
com-4
    sch_resi_boiler(:,temp_no,loop_day)      =sch_m_solve_para_resi(temp_com+5,:,temp_no)';        %↙
com-5
    sch_resi_boilerHout(:,temp_no,loop_day)     =sch_m_solve_para_resi(temp_com+6,:,temp_no)';     %↙
com-6
    sch_resi_dPup_dT(:,temp_no,loop_day)     =sch_m_solve_para_resi(temp_com+7,:,temp_no)';        %↙
com-7
    sch_resi_dPdn_dT(:,temp_no,loop_day)     =sch_m_solve_para_resi(temp_com+8,:,temp_no)';        %↙
com-8

    sch_resi_Heat(:,temp_no,loop_day)=resi_data_day_HW(:,temp_no,loop_day)+resi_data_day_Yuha(:, ↙
temp_no,loop_day);
    sch_resi_Pow(:,temp_no,loop_day)=resi_data_day_Pow(:,temp_no,loop_day);

if resi_data_device(2,temp_no)==1 %CGS
    sch_resi_CGSoutput(:,temp_no,loop_day)  =sch_m_solve_para_resi(temp_CGS+1,:,temp_no)';         %↙
CGS-1
    sch_resi_CGS_unit(:,temp_no,loop_day)   =sch_m_solve_para_resi(temp_CGS+2,:,temp_no)';         %↙
CGS-2
    sch_resi_Heater_unit(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+3,:,temp_no)';         %↙
CGS-3
    sch_resi_CGS_dP(:,temp_no,loop_day)     =sch_m_solve_para_resi(temp_CGS+4,:,temp_no)';         %↙
CGS-4
    sch_resi_HST_out(:,temp_no,loop_day)    =sch_m_solve_para_resi(temp_CGS+5,:,temp_no)';         %↙
CGS-5
    sch_resi_HST(:,temp_no,loop_day)        =sch_m_solve_para_resi(temp_CGS+6,:,temp_no)';         %↙
CGS-6
    sch_resi_CGSstartup_unit(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+7,:,temp_no)';     %↙
CGS-7
    sch_resi_CGSstop_unit(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+8,:,temp_no)';        %↙
CGS-8
    sch_resi_LFC_CGS_dPup(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+9,:,temp_no)';        %↙
CGS-9
    sch_resi_LFC_CGS_dPdn(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+10,:,temp_no)';       %↙
CGS-10
    sch_resi_LFC_CGS_dPupdT(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+11,:,temp_no)';     %↙
CGS-11
    sch_resi_LFC_CGS_dPdndT(:,temp_no,loop_day)=sch_m_solve_para_resi(temp_CGS+12,:,temp_no)';     %↙
CGS-12
end

%% コストの内訳計算
```

```
    % 1:購電, 2:売電, 3:暖房需要, 4:給湯需要, 5:CGS, 6:総計
    sch_resi_cost_list(1,temp_no,loop_day)=sum(sch_resi_Epur(:,temp_no,loop_day))*resi_Cost_Pow;
    sch_resi_cost_list(2,temp_no,loop_day)=(-1)*sum(sch_resi_Esell(:,temp_no,loop_day)) ↙
*resi_Price_Esell;
    sch_resi_cost_list(3,temp_no,loop_day)=sum(sch_resi_boilerHout(:,temp_no,loop_day)) ↙
*resi_cost_boiler;
    sch_resi_cost_list(4,temp_no,loop_day)=sum(sch_resi_boiler(:,temp_no,loop_day)-sch_resi_boilerHout ↙
(:,temp_no,loop_day))*resi_cost_boiler;
    sch_resi_cost_list(5,temp_no,loop_day)=sum(sch_resi_CGS_unit(:,temp_no,loop_day))*CGS_cost_cons+sum ↙
(sch_resi_CGSoutput(:,temp_no,loop_day))*CGS_cost_linear;

        % 1:購電, 2:売電, 3:暖房需要, 4:給湯需要, 5:CGS, 6:CGS稼働時間(分)
    sch_resi_ene_list(1,temp_no,loop_day)=sum(sch_resi_Epur(:,temp_no,loop_day))/1000;
    sch_resi_ene_list(2,temp_no,loop_day)=(-1)*sum(sch_resi_Esell(:,temp_no,loop_day))/1000;
    sch_resi_ene_list(3,temp_no,loop_day)=sum(sch_resi_boilerHout(:,temp_no,loop_day)/1000);
    sch_resi_ene_list(4,temp_no,loop_day)=sum(sch_resi_boiler(:,temp_no,loop_day)-sch_resi_boilerHout ↙
(:,temp_no,loop_day))/1000;
    sch_resi_ene_list(5,temp_no,loop_day)=sum(sch_resi_CGSoutput(:,temp_no,loop_day))/100*0.7/ ↙
(60/sch_tt_step);
    sch_resi_ene_list(6,temp_no,loop_day)=numel(find(sch_resi_CGSoutput(:,temp_no,loop_day)>0.1)) ↙
*sch_tt_step;
end
sch_resi_cost_list(6,:,loop_day)=sum(sch_resi_cost_list(1:5,:,loop_day),1);
save(matname,'sch_resi_*');
```



```
%% データ入力
temp_xlsfilename=strcat(num2str(loop_day),'day_sch_result_resi.xlsx');
outputfoldername=strcat(foldername_output,'sch_result');
mkdir(outputfoldername);
sch_solve_parameter=sch_m_solve_para_resi;

%% 最適化テーブル作成とxls出力
output_filename=strcat(outputfoldername,'/',temp_xlsfilename);
time=duration(0,(0:sch_tt_step:sch_tt_step*(sch_tt_max-1))',0); %時刻
output_time=datetime(resi_data_day_list(loop_day,1),resi_data_day_list(loop_day,2),resi_data_day_list ↙
(loop_day,3),0,(0:sch_tt_step:sch_tt_step*(sch_tt_max-1))',0);

%各住宅
for temp_no=1:length(sch_solve_parameter(1,1,:))

    temp_N_PER=sch_m_opt_N_PER_resi(temp_no,6);
    temp_com=0;
    temp_CGS=sch_m_opt_N_PER_resi(temp_no,1);

    HW_table=table(output_time,resi_data_day_HW(:,temp_no,loop_day)+resi_data_day_Yuha(:,temp_no, ↙
loop_day),'VariableNames',cellstr(char('time',sch_resi_table_name_com(sch_m_opt_N_dev(1)+1,:,1)))');
    Pow_table=table(output_time,resi_data_day_Pow(:,temp_no,loop_day),'VariableNames',cellstr(char ↙
('time',sch_resi_table_name_com(sch_m_opt_N_dev(1)+2,:,1)))');
    sch_output_table=innerjoin(HW_table,Pow_table);

    %共通
    for x_term=1:sch_m_opt_N_dev(1)
        if sch_resi_table_list(x_term,1)==1
        output_add_table=table(output_time,sch_solve_parameter(temp_com+x_term,:, ↙
temp_no)','VariableNames',cellstr(char('time',sch_resi_table_name_com(x_term,:)))');
        sch_output_table=innerjoin(sch_output_table,output_add_table);
        end
    end

    if resi_data_device(2,temp_no)==1 %CGS
        for x_term=1:sch_m_opt_N_dev(2)
            if sch_resi_table_list(x_term,1)==1
            output_add_table=table(output_time,sch_solve_parameter(temp_CGS+x_term,:, ↙
temp_no)','VariableNames',cellstr(char('time',sch_resi_table_name_CGS(x_term,:)))');
            sch_output_table=innerjoin(sch_output_table,output_add_table);
            end
        end
    end
    Sheetname=strcat('No',num2str(temp_no));
    writetable(sch_output_table,output_filename,'Sheet',Sheetname); % 計算結果の書き込み
end
```

```
%% 初期設定
clearvars -except loop*; %ループ数以外削除

load('../../../initialsetup.mat');
matname=strcat(foldername_input,'AG_data.mat');
load(matname);
matname=strcat(foldername_output,'sch_data.mat');
load(matname);

%% 調整力の線路損失分の計算

matname=strcat(foldername_output,'sch_AG_data.mat'); %読みこみ
load(matname);

sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+1,loop_day)=sum(sch_AG_res_LFC_dkW_up(:,1:AG_resi_sch_num, ↙
loop_day),2);
sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+1,loop_day)=sum(sch_AG_res_LFC_dkW_dn(:,1:AG_resi_sch_num, ↙
loop_day),2);

sch_AG_LFC_req_resi_up(:,AG_resi_sch_num+1,loop_day) = max(0,sch_AG_LFC_req_up(:,loop_day)- ↙
sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+1,loop_day));
sch_AG_LFC_req_resi_dn(:,AG_resi_sch_num+1,loop_day) = max(0,sch_AG_LFC_req_dn(:,loop_day)- ↙
sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+1,loop_day));

sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2,loop_day)=sum(sch_AG_res_LFC_dkW_up(:,1:AG_resi_sch_num, ↙
loop_day),2);
sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2,loop_day)=sum(sch_AG_res_LFC_dkW_dn(:,1:AG_resi_sch_num, ↙
loop_day),2);

sch_AG_LFC_req_resi_up(:,AG_resi_sch_num+2,loop_day) = max(0,sch_AG_LFC_req_up(:,loop_day)- ↙
sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2,loop_day));
sch_AG_LFC_req_resi_dn(:,AG_resi_sch_num+2,loop_day) = max(0,sch_AG_LFC_req_dn(:,loop_day)- ↙
sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2,loop_day));

% AG単位のLFCに関する結果まとめ
sch_AG_res_LFC_list(1,loop_day)=sum(sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2,loop_day));    %出力増加方↙
向の調整量
sch_AG_res_LFC_list(2,loop_day)=sum(sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2,loop_day));    %出力減少方↙
向の調整量
sch_AG_res_LFC_list(3,loop_day)=sch_AG_res_LFC_list(1,loop_day)+sch_AG_res_LFC_list(2,loop_day); %調整↙
量
sch_AG_res_LFC_list(4,loop_day)=sch_tt_step*numel(find(sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2, ↙
loop_day)>=sch_AG_LFC_req_tar(:,loop_day)-0.000001)); %調整力提供時間(増加)
sch_AG_res_LFC_list(5,loop_day)=sch_tt_step*numel(find(sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2, ↙
loop_day)>=sch_AG_LFC_req_tar(:,loop_day)-0.000001)); %調整力提供時間(減少)
sch_AG_res_LFC_list(6,loop_day)=sch_tt_step*numel(find(min(sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2, ↙
loop_day),sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2,loop_day))>=sch_AG_LFC_req_tar(:,loop_day) ↙
-0.000001)); %調整力提供時間(両方向)
sch_AG_res_LFC_list(7,loop_day)=100*sum(sch_AG_res_LFC_dkW_up(:,AG_resi_sch_num+2,loop_day))/sum ↙
(sch_AG_LFC_req_tar(:,loop_day)); %調整力提供達成率
sch_AG_res_LFC_list(8,loop_day)=100*sum(sch_AG_res_LFC_dkW_dn(:,AG_resi_sch_num+2,loop_day))/sum ↙
(sch_AG_LFC_req_tar(:,loop_day)); %調整力提供達成率
sch_AG_res_LFC_list(9,loop_day)=(sch_AG_res_LFC_list(7,loop_day)+sch_AG_res_LFC_list(8,loop_day))/2; %↙
調整力提供達成率
sch_AG_res_LFC_list(10,loop_day)=sch_m_calctime(loop_day,1);              %計算時間
```



```
sch_AG_res_LFC_list(11,loop_day)=sum(sch_m_cost_resi(:,loop_day));       %総供給コスト
sch_AG_res_LFC_list(12,loop_day)=sum(sch_m_inc_resi(:,loop_day));        %報奨金総額
sch_AG_res_LFC_list(13,loop_day)=sum(sch_m_totalcost_resi(:,loop_day)); %総トータルコスト
sch_AG_res_LFC_list(14,loop_day)=min(sch_m_totalcost_resi(:,loop_day)); %トータルコスト最小需要家
sch_AG_res_LFC_list(15,loop_day)=max(sch_m_totalcost_resi(:,loop_day)); %トータルコスト最大需要家

save(matname,'sch_AG_*');
```