SSI-DT79135032

博士論文

マイクログリッドの運用・設計手法の開発と経済性の評価

下町健太朗

2016 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（工学）授与の要件として提出した博士論文である.

下町健太朗

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

マイクログリッドの運用・設計手法の開発と経済性の評価*

下町健太朗

概要

地球温暖化や化石燃料枯渇等の問題を解決するための一手段として，近年，太陽光発電(PV: Photovoltaic)などの再生可能エネルギー(RE: Renewable Energy)を利用した発電システムの導入が進められている．しかしながら，PV の出力は天候や日射量に大きく左右される電源であるため，他の供給手段や蓄電池などと組み合わせて運用することが必要である．マイクログリッド(MG: Microgrid)は，供給対象エリア(建物単位や地域)内の電力需要および RE・蓄電池の運転状態を監視・制御し，また予測される将来の状況に対して，MG 内部の機器を計画的に運用・制御することで，エネルギー利用の高効率化や RE の統合制御を可能とする技術であり，PV との親和性が高い．PV および蓄電池は原理的には直流で動作し，給電距離がそれほど長くない MG 等においては，従来の交流方式ではなく直流方式で給電することで，交直変換の段数を減らすことができ，高効率な MG を構築できる可能性がある．

本論文では MG，特に直流を用いた直流 MG に対して，その最適運用のために対象とする時間粒度の異なる 3 つの運用機能を搭載した EMS を開発すると共に, MG を構成する各種機器の容量設計手法を明らかにすることにより MG の経済性評価を行うことを目的としている．最適運用については，EMS において必要となる 3 つの機能のうち, 特に前日計画機能について検討を行い，具体的な計画アルゴリズムを開発する．また，当日運用機能と組み合わせた際の効果についても確認を行う．MG における設備容量設計においては，住宅需要家向け直流 MG における PV，蓄電池および EV の最適容量をそのコストが最小となるように設計を行う．さらに，本論文で提案する運用手法ならびに容量設計手法を用いて，従来のように交流で給電を行う交流 MG と直流 MG との経済性比較を行う．

これらの研究を通して, MG の運用, 設計, 給電方式と多岐に渡る包括的な試算を行うともに，MG における経済性の評価を行う．

キーワード：マイクログリッド，直流給電，EMS，経済性評価

*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-DT79135032, 2015 年 12 月 2 日

Study on Operating and Sizing Method and Economic Evaluation of Microgrid†

Kentaro Shimomachi

Abstract

In recent years, studies of renewable energy (RE) such as photovoltaic (PV) generation are increasing to solve serious environmental problems such as global warming,. However, the output of PV would be fluctuating because it depends on weather conditions such as solar radiation, ambient temperature and so on. Therefore, output should be compensated by energy storage systems such as a battery system to reduce that fluctuation. In demand-side, the local demand and supply system called microgrid (MG), becomes real possibility. Since PVs and batteries primitively work with DC, advantages of DC supply system over AC supply might be emphasized.

This paper proposes various algorithms for MG; optimal operating, optimal sizing and economic evaluation of supply system in MG. In optimal operating, day-ahead operation scheduling functions in energy management system for DCMG are developed and implied. The day-ahead operation and daily operation algorithms are combined and evaluated in operating simulation. Finally, these functions are implied to real facility.

In optimal sizing, capacities of PV system, battery system and EV system are calculated to minimize costs in DCMG for residence user.

In economic evaluation, costs for ACMG and DCMG are calculated and compared by using proposed methods.

In result, comprehensive calculation for MG is considered by proposed method and economic evaluation is proposed.

Key words: microgrid, DC supply, EMS, economic evaluation

† Doctoral Thesis, Division of Systems Science and Informatics, Graduate School of Information Science and Technology, Hokkaido University, SSI-DT79135032, 2nd, December, 2015

目 次

図目次

表目次

第 1 章　序論

1.1　近年の電力システムに関する動向

現代の生活においては，電力は欠かせないものであり，現在の我が国における電力化率（総合エネルギー統計の最終エネルギー消費量に占める電力消費量の割合）は 23.6%である[1]．情報化社会の進展なども相まって，電気への依存度は依然として高まることが予想され，電力システムの役割は今後さらに重要になるものと考えられる．このように社会基盤としての重要性が増す中で，電気事業を取り巻く状況は時代とともに変化しており，電力システムには安定供給を維持しつつ，環境の変化に順応するための技術革新を行う使命が課されている．近年における状況の変化の例として，特に注目されていることは，「地球温暖化対策」，「情報化社会の進展」および「福島第一原子力発電所の事故」が挙げられる．

地球温暖化対策が電気事業で重視されるようになった契機としては，1997 年の京都議定書の採択が挙げられる．その中では，温室効果ガス排出量削減の数値目標が設定されており，2002 年に施行された電気事業者による新エネルギー等の利用に関する特別措置法，通称 RPS 法によって，電気事業者は新エネルギーによる発電電力を一定量以上利用することが義務付けられた．太陽光発電(PV)においては，発電時に CO_2 を発生させることがなく，材料となるケイ素も化石燃料と比較して十分な埋蔵量を確保できると考えられている．このような背景から，PV の普及促進に向けて，発電電力の全量買取制度が平成 24 年 7 月に開始されるなどの政策が取られている．これに加えて，福島の震災以降，我が国は原発廃止の方向に政策を転換しており，その代替エネルギー源として，PV が期待されている．海外に目を向けると，風力発電(WT)に関しては，再生可能エネルギー(RE)を利用した電源としては発電コストが安価であるため，系統規模が大きい欧米での導入量が多く，系統安定化対策のための高度な発電量予測システムや過不足分を市場で取引可能な仕組みも構築されている．さらに，ドイツでフィードインタリフ制度が導入されたことを契機に欧州の導入が急速に拡大しており，太陽電池や出力予測の技術革新が各方面で進められている．

このような分散型電源(DG)の導入拡大は，発電部門における温室効果ガスの排出量削減，送電ロスの削減の点でメリットが大きく，今後も導入量の増加傾向が続くものと予想されるが，これまで発電側から需要家側への一方向であった電力の流れが双方向になることに伴う課題も数多い[2]．さらには，電気自動車(EV)大量導入などの将来的な動向も考えられる．

ところで，PV や EV，各種蓄電池等の DG は，本質的には直流電力を出力する装置であるが，既存の電力系統が交流システムであるために，出力を交流に変換して扱うことが必要となる．また，需要中にも LED 照明，PC 等の電子機器のような直流駆動するものが増えてきており，このような負荷においても，負荷毎に直流への変換が必要である．すなわち，既存の交流システムにおいては，直流電源の出力を直流負荷で電力を消費する際には，直流を交流に変換し，さらに再び直流に変換する必要がある．一方で，直流による配電を行うことにより，配電に合わせるだけ

のために交流に変換する処理が必要でなくなる．変換の際には，電力の損失が伴うため，DG や直流負荷の割合が大きな配電系統においては直流の方が交流よりも損失を低減することができることが示唆されている[3].

情報化社会の進展との関連して，現在ではマイクログリッド(MG)やスマートコミュニティ(SC)と呼ばれる概念が研究段階にある[4]，[5]．MG，SC の基本的な考え方としては，「ICT 技術を用いて電力システムを効率化する」というものであり，スマートメーター—やデマンドレスポンス(DR)がその代名詞となっているが，その技術範囲は多岐にわたり，導入の動機は信頼度向上や計量簡易化など国によって異なる．

最後に，福島第一原子力発電所の事故に関しては，今後の電力供給のあり方が根本的に見直される可能性があり，その後の動向に注視する必要がある．この原発事故においては，一つの原発が停止したことにより，関東地方においては，慢性的な電力不足が発生した．このことにより，現在の電力供給システムの根幹である，大規模な発電所による大量発電方式の弱点が露呈した．先の MG 技術においては，DG による電力の地産地消が可能となるため，このような事態を避けられる可能性が示唆されている．

1.2 マイクログリッドを取り巻く現状

1.2.1 マイクログリッドの概要と特徴

MG の定義は，前項で述べた通り多岐に渡るが，本論文では「DG と負荷を持つ小規模系統で，複数の電源および熱源が IT 関連技術を使って一括管理されて，既存の電力会社の商用系統から独立して運転可能なオンサイト型の電力供給システム」とする．

MG においては，ICT 技術を用いて DG 群と負荷群を制御する．DG は，種類によって様々な特性を有しており，複数種類の DG を組み合わせ適切に制御することで，互いの欠点を補完することが期待されている．例として，PV や WT は，季節や天候に出力が左右されるという欠点を有している．一方，ガスタービン(GT)や燃料電池(FC)は，安定した電力供給が可能であるが，発電時に燃料を消費し CO_2 を排出するという欠点がある．これらの電源を適切な容量導入して，運用スケジューリングと組み合わせることで，CO_2 排出量を削減しつつ，経済的で安定した電力供給を行うことが可能となる．

MG に期待される効果の一つとして，電力系統の信頼度向上が挙げられる．前節でも述べた通り，福島第一原発事故によって電力の大量生産・大量輸送という方式の弱点が露呈した．このように，電力会社が所有する系統において，大規模停電ないしはそれに準ずる事態が発生した場合には，MG を単独系統とすることで重要負荷に継続した電力供給が可能となる．特に，災害などにより，大量の負傷者が発生する事態においては，病院などの重要負荷における停電が問題となるため，MG の導入により多くの人命を救助できる可能性がある．

MG に期待されるもう一つの効果である CO_2 排出量削減の観点からは，MG 内にコージェネレーションシステムを導入することでさらに能力が向上することが考えられる．コージェネレーションシステムでは，ある小規模な DG の発電によって得られる排熱を，熱負荷の供給に用いることで，環境性および経済性の向上が期待できる．さらに，地域内での相互融通により，一層経済

性に優れたエネルギー供給システムの構築が可能となる．

さらに，離島や辺地向けの電力供給システムにおいては，MG が有利となる．既存の商用系統との連系には，ある程度の容量を持った送電線が必要となるが，離島のように発電所から極めて離れた場所に送電することは，電力会社の経済的な負担となることが考えられる．離島に MG を構築した場合には，このような経済的損失を回避できる．

一方で，現状の電力系統に対して MG を大量導入することは，系統に悪影響をおよぼす可能性も考えられる．MG 内に，適切でない容量の自然エネルギーを導入した場合には，その出力変動がそのまま電力系統へと影響する可能性が考えられる．また，単独運転時には，系統が小さい故に顕在化してくる各種の系統問題がある．これに対して十分な考慮を払った上で経済的なシステムを構築することが重要となる．

1.2.2　マイクログリッドの基本構成

MG の基本構成は，電源および熱源，電力及び熱貯蔵装置，電力及び熱負荷，電力及び熱輸送設備，情報ネットワーク及び需給制御システムなどから構成される．一例として，図 1.1 に MG の構成を示す．従来の電力ネットワークとの大きな違いは，

- エネルギー貯蔵装置（特に蓄電装置）
- 熱エネルギーの活用

の存在である．電力貯蔵装置については，商用の電力ネットワークにおいても，揚水発電など水の位置エネルギーに変換した形で電気エネルギーは蓄積されているが，MG の場合はさらに必要度が高い．これは，電力ネットワークの場合は，需要家が多いため大数の法則により需要カーブが平滑化されるが，MG の場合は需要家数が小さいためその変動が十分に平滑化されず，これに対応する必要があること，また，自然エネルギー（太陽光・風力など）の場合は，発電量自体が安定せずに変動し，これにも対応する必要があるためである．商用の電力ネットワークでは，その目的が電気エネルギーのみの供給であることと，ネットワーク電源は通常は需要地から遠方にあるため，熱エネルギーの活用は発電側でのみ発電効率向上という形で実施され，需要地においては活用できない．これに対し MG は，電熱源が需要地近接であるため，熱エネルギーの活用が物理的に可能であることから，熱エネルギーが利用される．

MG における装置には，その出力や状態の計測装置，制御装置，情報通信装置などが併設される．各種情報は，エネルギーのネットワークとは異なる情報ネットワーク上でやり取りがなされる．図 1.1 に示した MG では，各種情報は中央のエネルギーマネジメントシステム(EMS)によって収集され，何らかの処理がなされた後，各エネルギー装置に指令が与えられる．また，EMS はその対象とする設備の規模に応じて名称が変化することがあり，住宅用では Home EMS(HEMS)，少し規模の大きな商業施設やオフィスビルにおいては Building EMS(BEMS)，工場においては Factory EMS(FEMS), 市や地域などのコミュニティ全体を統括する場合は Community EMS(CEMS) などと呼ばれる．

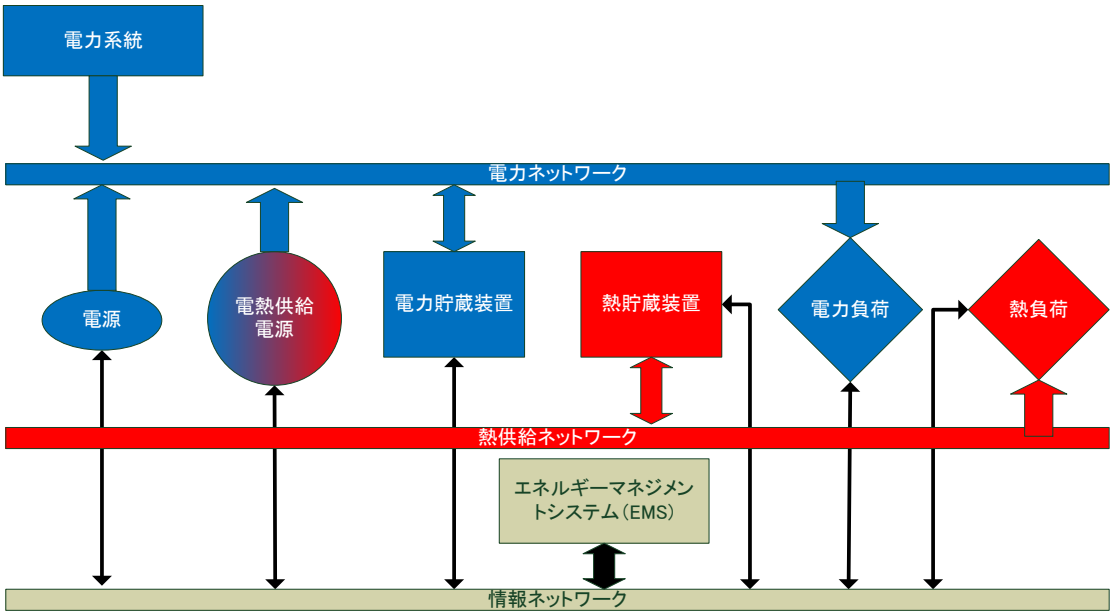

図 1.1 MG の主な構成要素

1.2.3　マイクログリッドの研究動向

I.　米国における事例

北米の電力供給における信頼性や品質が欧州や日本に比べると劣っていることに起因して，北米の MG 研究は電力の安定供給問題に対応することが最大の目的となっている．また，オバマ政権では，グリーンニューディール政策として，再生可能エネルギーへの 1500 億ドルの資金投入や，数百万人規模での雇用創出を挙げている．

カリフォルニア州，オハイオ州，フロリダ州，ハワイ州およびテキサス州などでは，各家庭にスマートメータを設置しており，MG 技術の本格的な導入がなされつつある．

一例として，コロラド州ボルダーにおける実証実験を紹介する[6]．ボルダーにおいては，スマートメータを各家庭に設置するとともに，送電線と光ファイバー通信ネットワークが整備された．送配電線におけるスマート化の例として，故障箇所の自動検知，電圧の自動管理，需要モニターによる電力変動への対応などが挙げられる．需要家のスマート化の一例として，スマートメータによる電力消費プランの紹介，家電製品とのコミュニケーション機能などが挙げられる．この実証試験においては，様々な知見が得られたが，光ファイバー通信ネットワークの敷設コストが膨大となるなどの問題が見られた．

II.　欧州における事例

EU では，エネルギーの上流から最終需要までのエネルギーチェーンを最適化するスマートグリッドプラットフォームの一環として DG の活用が位置づけられている．DG は，欧州共通の統一電力市場を発展させる上でも重要な役割を果たしており，更に，一部の電力系統に障害が起こった場合でも継続的に電力を供給することを可能にする．基本的に MG は配電系統と独立するのではなく，むしろ協調して将来の能動的な配電系統の一部を構成する役割を期待されている．ま

た，EU における電力供給の事情として，再生可能エネルギーの導入が活発であることが挙げられる．再生可能エネルギーの効率的な運用と，MG 技術とは密接に関係している．図 1.2 に EU 加盟各国における MG 導入のための予算分布を示す．

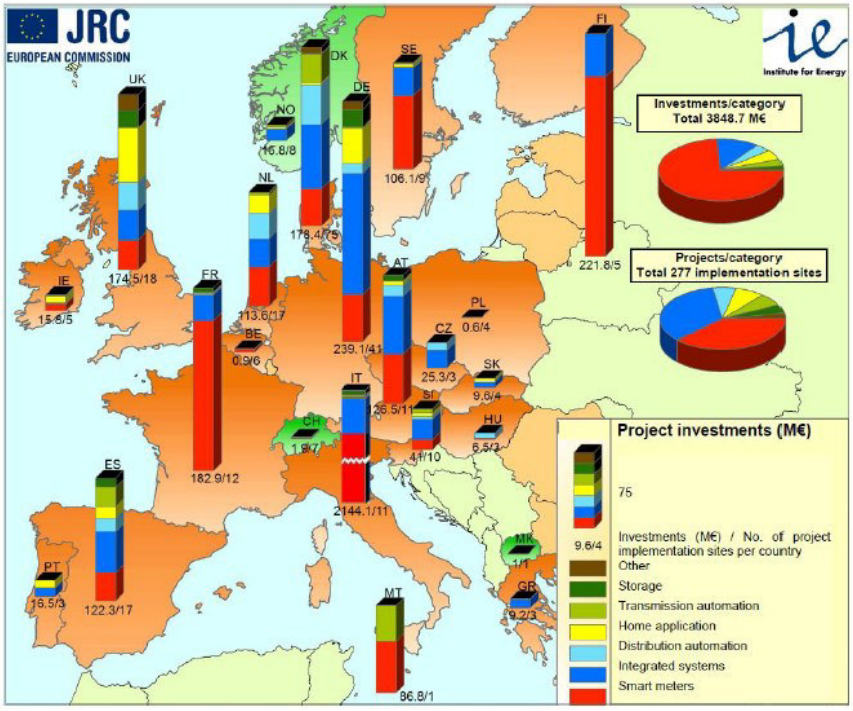

図 1.2 EU 加盟各国の MG 導入予算[7]

III. 我が国における事例

国内においては新エネルギー等地域集中実証研究事業の 3 プロジェクトが主な MG プロジェクトとして挙げられる．また，清水建設，東京ガスなど，様々な機関での研究事例がある．以下に，それぞれの事例の概要を示す．

- 愛知プロジェクト「2005 年日本国際博覧会・中部臨空都市における新エネルギー等地域集中実証研究」：愛知万博会場内の日本館等において，メタン発効ガスや高温ガス化ガスを活用した燃料電池を主体とし，太陽光発電，NAS 電池を用いたシステムによる MG 供給の実証研究[8]
- 京都プロジェクト「京都エコエネルギープロジェクト」：電力会社の既設の電力系統を使用した仮想 MG 上において，太陽光発電，風力発電の出力変動分を，バイオガス発電及び二次電池により賄う電力需給バランス制御技術の開発・実証[9]
- 八戸市プロジェクト「八戸市　水の流れを電気で返すプロジェクト」：下水汚泥消化ガスを燃料としたガスエンジンを主体として，太陽光発電，風力発電及び二次電池を用いた，再生可能エネルギーのみによる発電で構成されるシステム構成[10]
- 清水建設：技術研究所の実験棟に太陽光発電，マイクロガスタービン，ガスエンジン，二次電池を設置し，MG の実験を開始[11]
- 東京ガス：横浜研究所内に，ガスエンジンコージェネレーション，太陽光発電，風力発電，鉛蓄電池，ニッケル水素電池，三重効用吸収式冷温水器などを設置し，実負荷と実験用の模擬負

荷を用いて実証試験を実施[12]

- 直近では，横浜市，豊田市，けいはんな地区，北九州市の 4 つのなどでもスマートシティ実証プロジェクトが行われている[13]．詳細については後で述べる．

1.2.4　4 都市スマートシティ実証プロジェクトの概要

2009 年 11 月に我が国の経産相は省内横断的なプロジェクトチーム「次世代エネルギー・社会システム協議会」を設置し，2010 年 1 月には次世代のエネルギー流通および社会システムの在り方に対する中間的な取りまとめが発表された．この取りまとめを受けて，スマートグリッドならびにスマートシティの社会実証実験が公募され，選ばれたのが横浜市，豊田市，けいはんな地区，北九州市の 4 地域である．表 1.1 は各地域における実証プロジェクトの地域別の特徴ならびに導入目標等である．詳細については後述する．

表 1.1 各実証プロジェクトにおける地域の特徴ならびに導入目標等[13] (Japan Smart City Portal Web サイトより抜粋)

都市名	横浜市	豊田市	けいはんな学研都市	北九州市
面積	435.17 km^2（2014年1月時点）	918 km^2（2012年4月時点）	154.1 km^2（2012年4月時点）	488.78km^2（2011年10月時点）
人口	370万3258人（2014年1月時点）	42万2830人（2012年4月時点）	24万4872人（2012年4月時点）	97万1924人（2012年8月時点推計）
実証対象地区名	3エリア（みなとみらい21エリア、港北ニュータウンエリア、横浜グリーンバレーエリア）を中心とした横浜市全域	HEMS・EDMSを実証する東山地区・高橋地区および低炭素交通システム構築を実証する豊田市全域	精華・西木津地区（京都府京田辺市・木津川市・精華町）	八幡東区東田地区
実証対象地区面積	435.17 km^2（2014年1月時点）	918 km^2（市内全域）	7.737km^2（2012年4月時点）	1.2km^2
実証対象世帯数	社会実証用・技術実証用戸建・マンション：4000戸	新築住宅67戸、既設住宅160戸	HEMS導入：14世帯、エネルギーの見える化導入：約100世帯、電力デマンドレスポンス（DR）対応：約700世帯（いずれも2012年4月時点）	225世帯（2012年8月時点）
実証対象事業所数	業務ビル：4棟、商業ビル：3棟、マンション：8棟、大規模工場：1棟	商業施設2カ所、物流拠点1カ所、見える化拠点1カ所（とよたエコフルタ	BEMS導入：1棟（けいはんなプラザ、2012年4月時点）	50事業所（2012年8月時点）
実証対象EV/PHV台数，太陽光発電導入量等	デマンドレスポンス（DR）対応EV：25台（充放電対応EV6台を含む。PV・蓄電池を設置した充電スタンドは2カ所），太陽光発電（PV）：27MW、HEMS：4000戸、EV：	再生可能エネルギー導入比率61.2%、次世代自動車4000台	EV：60台（2012年4月時点）	スマートメータ：225台（2012年8月時点），蓄電池：約800kW、太陽光発電（PV）：約400kW、燃料電池：約110kW

I.　横浜スマートシティプロジェクト(YSCP)[14]

このプロジェクトにおいては，市民，民間企業，市の連携によってスマートシティモデルを構築し，その成功モデルを全国・海外に展開することを目的としている．対象地域である横浜は，大規模先進都市であり多様な地勢を有している．EMS を階層的に束ねることで，EMS 単位のエネルギー管理はもとより，システム全体として需要家サイドのエネルギーをマネジメントする．

EMS はそれぞれの環境に応じたエネルギーの見える化や管理を実現し，具体的には負荷の利用形態によって，戸建用 HEMS，集合住宅用 HEMS，マンション用 HEMS や，統合 BEMS，工場の

運転を最適制御する FEMS がある．統合 BEMS は，オフィスビル用や商業施設用の BEMS を束ね群管理する役割を担う．これらに加え，次世代交通システムの核となる充放電対応 EV（電気自動車）や充電ステーション，系統安定化に貢献する蓄電池 SCADA などを CEMS が集約し，コミュニティ全体でエネルギー管理の最適化を図っている．

CEMS を中心とした EMS 群の最適連携により，天候に左右される PV の不安定さを吸収するなど，再生可能エネルギーを大量導入しやすいインフラを構築している．併せて，大規模な DR 実証にも取り組んでおり，需要家にインセンティブ付きの電力使用制限依頼を送ることで，電力需要を抑制する行動をうながし，より低い社会コストによる CO_2 削減を実現するほか，PV 大量導入時の余剰電力を吸収するための DR も実証しながら全体最適なエネルギーマネジメントを実現する．具体的な導入目標は，PV が 27 MW，HEMS が 4000 戸，EV が 2000 台となっている．

II.　豊田市低炭素社会システム実証プロジェクト(Smart Mellit)[15**]**

このプロジェクトにおいては，再生可能エネルギー導入や各種省エネ/蓄エネ機器普及が進んだ 10 年後の家庭環境を想定している．次世代自動車を含む各種機器の電力授受パターンを HEMS により統合・制御し，生活者が「無理なく，無駄なく，便利で楽しい低炭素ライフ」を送れるようにすることが目的である．加えて，次世代モビリティの導入や公共交通インフラの整備，新しい交通利用形態の提供をセットで推進し,「クルマと人が世界一うまく共生するまち」を目指している．都市ガスやバイオマスなど多様なエネルギー源を活用し，熱・電気を面的に利用し，生活者は自らの低炭素行動に対する各種インセンティブを享受しながら，生活に身近な端末を介した「見える化」「行動支援」「制御」などにより，生活圏全体にとって最適な低炭素行動メニューを無理なく選択できる．具体的な導入目標は，再生可能エネルギーの導入比率が 61.2%，次世代 EV が 4000 台となっている．

III.　けいはんなエコシティ次世代エネルギー・社会システム実証プロジェクト[16**]**

けいはんな地区は研究機関，教育施設等が多く立地している他，現在大規模な宅地開発が進んでおり，先端技術や新しい社会システムの研究成果について住民を交えて実証することが期待できる．このような特徴から，このプロジェクトにおいては地域規模でのエネルギー需給の最適化を図ることを目的としている. そのために, 地域のエネルギーを統括的にマネジメントする CEMS や，住宅内のエネルギー需給をマネジメントする HEMS，大規模な DR などのエネルギーマネジメントを担う電力 DR，ビル内のエネルギーをマネジメントする BEMS，EV 充電管理システム，および V2X（Vehicle to X）などを構築しそれらを系統電力と連携させる．

具体的には，住宅やビルにおいて，CEMS が HEMS，BEMS および EV 充電管理センターと連携しながら DR を実行し，省エネ・省 CO_2 の効果を実証する．EV 充電管理システムでは，EV の位置や蓄電池残量を元に，充電場所・充電時間を誘導することでピークシフト効果を実証している．V2X では，工場内の電力需給に対する EV 蓄電池の利活用を実証している．これらの成果は「けいはんなエコシティモデル」としてビジネス化し，東北各都市の復興や海外諸国に展開される．具体的な導入目標は，EV および PHV が 60 台，HEMS 導入 14 世帯，エネルギーの見える化 100 世帯，電力 DR 対応 700 世帯となっている．

IV. 北九州スマートコミュニティ創造事業[17]

北九州市八幡東区東田地区は既に，環境施設の整備や多様な新エネルギー導入などにより，一般的な街と比べ30%のCO_2削減を達成している．そのため，このプロジェクトでは，新エネルギー導入の強化や地域エネルギーマネジメント，交通システムの整備などにより現状より，さらに20%の削減を目指し，CO_2削減量は市内一般街区の50%超を実現することを目的としている．これを達成するために，太陽光発電，燃料電池，小型風力など新エネルギーの導入率を10%以上に高めること，地域エネルギーマネジメントと協調が図れるHEMSやBEMSを開発し，一般家庭や各種ビルの省エネ効率を高めることが挙げられている. また, 先進的なエネルギー制御や, EV, 蓄電池などを組み合わせてエネルギー流通の全体最適を図る「地域節電所」を整備すること，EVなどの大量導入を可能にする充電設備の整備と並行し，自転車や公共交通機関が連携する次世代交通システムを構築することも挙げられている．具体的な導入目標は，蓄電池約800 kW，PV約400 kW，FC約110 kWとなっている．

1.3 直流給電を取り巻く現状

1.3.1 直流給電の概要と特徴

直流給電とは，一般に利用されている交流ではなく直流によって給配電を行う方式のことである．長距離送電においては，交流電力を用いることで変圧器を介して電圧の上昇を行うことが容易となるため，送電損失低減につながる．しかしながら，近年，PV，蓄電池およびEVと言った出力が直流であるDGの系統への積極的な導入に関する研究が盛んに行われており，また，電化製品についても直流の利用が増加している．このことから，長距離送電を伴わないような給配電においては，直流給電を行うことにより変換器段数の低減，ひいては変換損失の低減が期待される．加えて，変換電化製品の待機電力の減少も期待できる．家庭部門におけるエネルギー消費量は，この40年間に約5倍に増加しており，電力化率も1990年から2001年にかけて約40から約45%に増加している．その結果，家庭部門での電力消費は年々継続して上昇している．先に述べた通り，ライフスタイルの変化によって，PC，テレビ，スマートフォンといった電子機器の普及が進んでおり, 家庭における消費電力量のうち9.4%はこれらの待機時の消費電力量であると見積もられている．文献[18]では，充電器・アダプターなどの待機電力の削減効果として，原油換算で200万kl（2010年度推定）と見積もられている．

以上のような利点に加えて，電力供給信頼度の向上も期待できる．すなわち，変換器段数の減少に伴い，変換器の故障によって電力が供給ができなくなる可能性が少なくなる．現代社会は情報化が進んでおり，データセンターのような負荷には高いレベルの供給信頼度が必要となる．データセンターのサーバーコンピュータ機器へ電力を供給する場合，交流無停電システム（UPS）の信頼度に比べて，直流給電方式の信頼度は1桁高く，9Nineレベル（停止時間3.15ms/年）である[19]．これは，交流給電方式において，信頼度確保用のUPSシステム内に，整流装置，インバータおよびバイパス回路との切り替え用のスイッチが必要となり，機器構成が煩雑となることに起因している．

さらに，直流給電方式では，変換器段数の減少に伴う省スペース化およびコスト削減の効果も

もたらす．インターネットデータセンターの検討例では直流給電方式の構成は単純であることから，必要なスペースは交流方式に比べて半分に減少することも示されている．また，低損失であることから，1000m^2 規模のデータセンターでは電気料金を 1 年当たり 5000 万円程度削減できること，空調の設置コストを 10%低減できることも合わせて指摘されている．

このような利点を有する直流給電であるが，現在は研究・開発の段階であり一般には普及していない．また，直流給配電の技術課題としては，下記のものが挙げられる．

- 電圧範囲 : 直流給配電システムの電圧範囲は，発振や突入電流等の電圧変動に対する機器の耐力を踏まえて設定する必要がある．例えば，電圧範囲を低電圧に設定した場合は，消費電流が増大するため，配線が太くなり，施工性が問題となる．一方，電圧範囲を高電圧に設定した場合は，電圧範囲を広く設定でき，電圧変動に対して耐力を持たすことができる．さらに，消費電流が小さくなるため，配線を細くできる．しかし，高電圧は，短絡時のアーク電流が途切れ難いため，プラス端子とマイナス端子に適切な離隔をとり，アーク電流が切れやすい構造にする等の対策が必要となる．
- 電食 : 直流給配電システムは，機器の接地線等から迷走電流が流れ，電食が発生する恐れがある．電食とは，意図した回路以外のところを迷走電流が流れ，この迷走電流によって地中に埋設している金属体（管路やケーブル，鉄筋構造物等）が腐食することである．
- 負荷系統の短絡 : 直流給配電システムでは，負荷系統の一部で短絡事故等の障害が発生した場合，分電盤において電圧が大きく変動し，その影響を受けて他の機器の入力電圧も変動する．この電圧変動が他の機器の動作できる電圧範囲を逸脱したときに，他の機器が停止し，場合によっては破損する可能性がある．
- 発振 : 直流給配電システムでは，インピーダンス条件を満足しない場合に，直流が交流のように振幅する発振が発生する．特に，機器が負性抵抗特性（定電力で給電する特性）を有する場合，停電等により給電電圧が低下し，消費電流が増大すると発振する可能性は増大する．直流が発振することにより発振を発生させた機器だけでなく，他の機器の電源も入切を繰り返し，破壊される可能性がある．
- 突入電流 : 機器の中には，入力電圧の安定化や入力帰還雑音の抑制等の目的で，ノイズフィルタを接続することが考えられる．ノイズフィルタは，一般的にチョークコイルとコンデンサで構成される場合が多く，機器のスイッチ投入時やコンセント接続時に，チョークコイルよりも給電側に接続されるコンデンサの充電初期に過大な電流（突入電流）が発生する．この突入電流により分電盤において電圧変動が発生し，他の機器の入力電圧に悪影響を与える恐れがある．
- コンセント・プラグ・コネクタ等の形状 : 現在，直流給配電システムの最端末である直流コンセントや直流プラグは，交流のように形状や構造等の標準規格はない．特に，負荷への接続が直流となることから，接続回路を切り離す際のアーク対策が必要となる．コンセントからプラグを引き抜いた際，アーク放電が発生すると，引火や爆発，人体への火傷等の原因となる．交流の場合，物理的に引き抜かれたプラグとコンセントの間では，交流の特性（50 Hz であれば，0.01 秒に一度電圧値は 0 V となる）より，コンセントとプラグは直ちに電気的に絶縁の状態となる．直流の場合，時間の経過によって電圧値が 0V となることはないため，コンセントとプラグの間にアーク放電が発生した場合の消弧条件は交流に比べ容易でない．そのため，プラグを抜く際には接続されている電気用品の電源がオフとなるといったような配電系統と電気用品を安全に切り離す構造や抜け止めコンセントのように容易にはプラグが抜けない構造にす

る必要がある．

1.3.2 直流給電の研究動向

I. 海外の動向

直流給電を対象とした実証実験として，米国エネルギー省のプロジェクトが行われている[20]．このプロジェクトにおいては，データセンターにおけるエネルギー需要のピークカットを目的として直流給電制御が行われている．この設備には，直流の DG として蓄電池や燃料電池，風力発電装置などが設置され，これらの DG と系統との連系についての評価が行われている．同様の研究がスウェーデン，フランスなどの各国で盛んに行われており，様々な DG の組み合わせによるエネルギー効率向上の検証，運用性や保守性の評価などが研究の対象となっている[21]．

このような活動を受けて，直流給電の国際標準化が進んでいる．国際電気標準会議(IEC)において，直流給電に関する戦略グループが 2009 年に発足し，標準化の対象範囲の明確化と重複・漏れの防止，直流給電の分野における技術・市場情報収集，ロードマップ作成などの役割が期待されている．このような活動を受けて，さらに，国際電気通信連合電気通信標準化部門(ITU-T)や欧州電気通信標準化機構(ETSI)も通信施設やデータセンターに適用する直流システム国際標準の検討を始めている．

II. 我が国の動向

我が国においても，直流給電における様々な研究がなされている．文献[22]では，直流配電システムと PV との親和性について論じられている．この文献では，住宅密集地において，配電系統内の電圧を維持しつつ 100 軒程度の負荷を集中制御する直流配電システムを構成している．直流配電においては，線路と変圧器の設置コストの面では交流と比較して不利であるが，変換器損失も含めた配電損失の低減につながっていることが確認されている．この特徴は，PV の導入量が増加するほど顕著となる．

1.4 本論文の目的と構成

以上まで述べてきた様に，MG の導入には再生可能エネルギー導入の促進ならびに環境負荷の低減，供給信頼度の向上，エネルギー利用効率の向上というような利点がある．また，直流給電においては，小さな系統に限って見れば変換損失の低減，供給信頼度の向上，低コスト化，省スペース化など利点がある．この MG と直流給電を組み合わせた直流 MG(DCMG)においては，それぞれの利点を高め合う可能性が示唆される．しかしながら，直流 MG によって享受できうるメリットを示すためには，実際に直流 MG 設備そのものを構築すること，直流 MG 運用手法を確立すること，安全性を確保することなど解決するべき問題がまだ多く，実用化には至っていない．MG 設備の構築においては既存のシステムが交流であるために，すでに構築されている設備に対して MG を導入する際には，交流用の設備と直流用の設備とを交換して新たに構築する必要があり，費用がかかることが問題として考えられる．また，交流システムと比較して共通バスにおける電圧維持が困難であり，このことも直流 MG のコストを高める一因となっている．これに付随する形で，直流 MG における EMS には独自に電圧維持を行う機能も組み込む必要があり，従来の交流システム向けの運用手法をそのまま利用することは好ましくない．また，先述のように実

際の直流 MG 設備の構築が進んでいないために，ある運用手法を開発したとしても，実際の直流 MG に適用できるかを検証することが困難である．安全面についても，交流電力のように電圧，電流がゼロとなる時間帯がないことや，具体的な安全性の規格がまだ定められていないことが原因で，既存の交流システムよりも安全に留意する必要があり，これに伴って対策費用が多くなるものと考えられる．

このようなことを受けて，著者は直流 MG の実現可能性とその効果についての実証プロジェクトである「平成 24 年～26 年地球温暖化対策技術開発・実証研究事業　自立・分散型エネルギー社会の実現に向けた直流方式による地域間相互エネルギー融通システムの開発」の一環として EMS 開発に携わり，直流 MG の性能および効果について検証を行ったので，本論文で述べる．また，このプロジェクトで得られた知見より，本論文では MG に導入する DG のコストに着目した最適な容量設計，ならびに電力供給方式を適切に選択することが MG の経済性に与える影響についても試算を行ったため，これについても述べる．

第 2 章では，先述のプロジェクトにおいて，著者が実際に携わった EMS 開発について述べる．EMS においては，翌日の負荷および PV の出力予測を利用して，蓄電池や EV の充放電計画を策定する．この際，PV の出力に関しては厳密なものを使用するのではなく，比較的用意に入手できるデータを用いて出力シナリオを生成し，天候の変化に対してロバストな計画を得られるような手法について示す．また，計画時では 30 分ごとの充放電計画を策定するが，実際の直流 MG 運用においては 1 秒ないしさらに短い時間間隔での機器制御が必要となる．本論文では，1 秒値での運用試算を行い，計画段階での性能が短い時間間隔で試算においても発揮されていることを確認した．

第 3 章では，直流 MG 内に導入する DG の最適容量について設計を行う．この際，CO_2 排出量削減を目標とした MG を想定し，これを達成しつつコストが最小となるような DG 容量を決定する．なお，本論文では実際にメーカーが公表している製品を組み合わせて MG を設計することを想定し，導入量を整数として扱うような手法を提案する．

第 4 章では，これまでに考えられてきた交流による給電を行う交流 MG と直流 MG とのコスト比較を行う．すなわち，最適容量設計において，需要内の交流/直流比率がコストに与える影響を試算する．

第 5 章では，全体を通して得た知見と今後の展望について述べる．

第 2 章　マイクログリッドにおけるエネルギーマネジメントシステム前日計画機能の開発

2.1　概要

本論文では MG の運用試算を行うにあたり，前日計画，当日運用およびリアルタイム制御の 3 段階制御を行う EMS を開発した．特に，これに搭載される前日計画機能を開発した．前日計画においては，1 日に直流 MG から排出される CO_2 排出量が最小となるような問題を解くことで設備の 30 分ごとの運用計画を得た．運用計画策定にあたっては，負荷や PV 出力の予測値が必要となるが，特に PV 出力に関しては気象条件によって大きな誤差が発生することが問題となる．そこで本論文では，ある 1 つの予測値だけを用いて運用計画を策定するのではなく，ある発生確率を持ったシナリオとして PV 出力予測を扱い，CO_2 排出量の期待値が最小となるように計画を策定する手法を開発した．これにより，天候変動による誤差に対してロバストな計画を得ることができた．さらに，得られた前日計画を 1 秒単位での当日運用に適用することで，さらに短い時間帯幅での計画の効果や妥当性を検証した．なお，本論文は北海道帯広市に構築された直流 MG 実証実験の一環として行われた側面があり，検証にはこのプロジェクトで得られたデータを利用した．

2.2　開発したエネルギーマネジメントシステムの機能

従来の MG における EMS では，最適化によって 30 分から 1 時間程度の機器の出力を決定するものや，最適化は行わずに 1 秒から 10 秒程度の瞬間的な MG の状態から機器の出力を決定するものなどが提案されていた．しかしながら，前者だけでは計画段階よりもさらに短い時間帯幅における予測誤差に対応することが困難であること，後者だけでは出力に最適性がないため MG に期待されるパフォーマンスが低下することなどが考えられる．これを受けて本論文で開発した EMS は，図 2.1 のように対象とする時間粒度の異なる 3 つの機能を搭載した．このことにより，予測誤差対応と最適性の確保のどちらの能力についても性能が向上することが期待できる．3 つの機能はそれぞれ 30 分程度の時間帯幅における各設備の出力を決定する前日計画機能，1 秒程度の時間帯幅における各設備の出力を決定する当日運用機能，さらに短い数十 msec 程度の時間帯幅における各設備の出力を決定するリアルタイム制御機能である．図 2.1 のように，対象とする時間帯幅が短くなるほど計算にかけることのできる時間が短くなり，最適性よりもリアルタイム性が重視される．

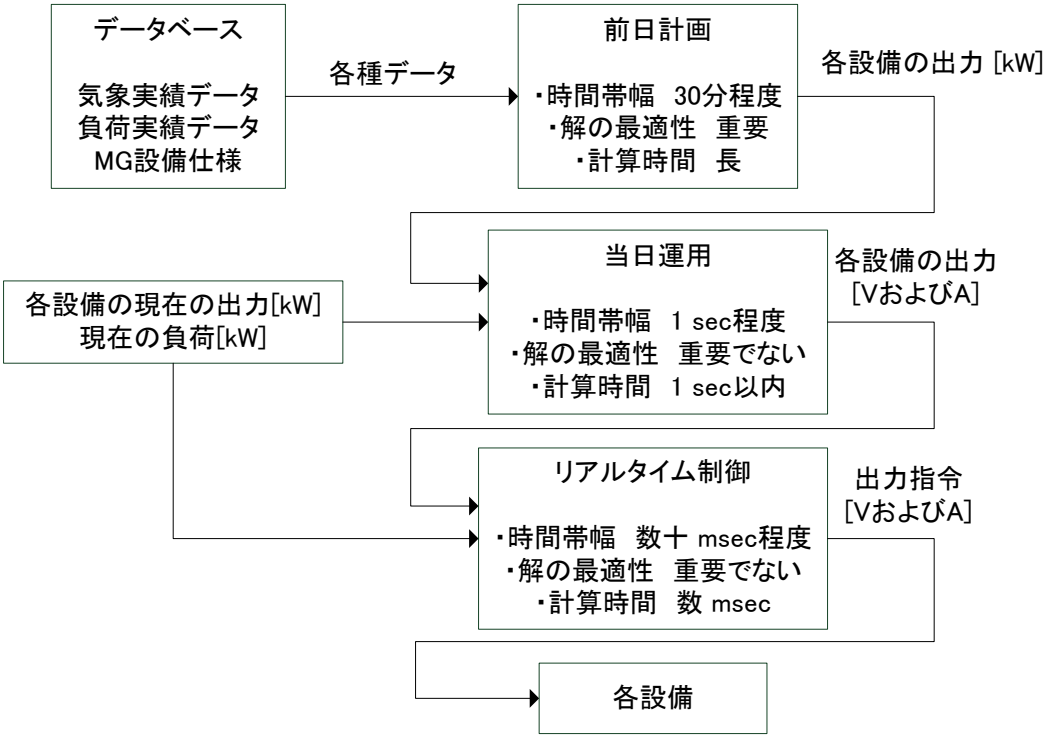

図 2.1 EMS に搭載する機能の時間粒度イメージ

2.2.1　前日計画機能の概要

一般に，前日計画においては複雑な最適化を行うに際して十分な時間が用意されている．そのため，リアルタイム性に関しては問題にならない．本論文では，ある予測データに基づいた前日計画を行うことを想定しており，おおむね予測発表，計算開始，求解の手順が 1 時間程度で完了することを想定している．また，予測に基づいた計画となるため，実際の運用において問題となる予測誤差への対応については，当日に行うこととなる．すなわち，前日計画では後述の当日運用やリアルタイム制御が誤差に対して対応する際に，実現可能なレンジでの各設備での最適運用を与えることが重要となる．これを受けて本論文で提案する前日計画では，気象庁から発表される気象情報および MG サイトで計測されたデータを利用して，対象とする MG 設備の CO_2 排出量が最小となるように，MG 内の各設備の 30 分単位での運用を決定するものとした．また，30 分単位の出力は W ないし kW で表現し，電力としての出力を決定するものとした．

本論文では，図 2.2 に示すような直流 MG を対象とした前日計画機能を開発した．この施設では，自地内電源として PV，蓄電池および EV を有しているものとした．これらは，MG 内の共通直流バスに接続されており，直流負荷へ電力供給を行う．また，MG と既存の電力系統とはコンバータ(以下 CONV)を介して一点で連系しており，MG 内の設備だけで電力需要を満足できない場合には，系統から電力を購入する．一方で，本論文では直流 MG を導入する目的の一つとして，MG がエネルギーの地産地消を行うことを想定している．そのため，この CONV は逆潮流ができない仕様とした．この制約を満足するためには，PV の余剰電力等により直流 MG 内の電力供給

が過多になる場合には，蓄電池による余剰電力の吸収や，PV の出力抑制といった制御を行う必要がある．

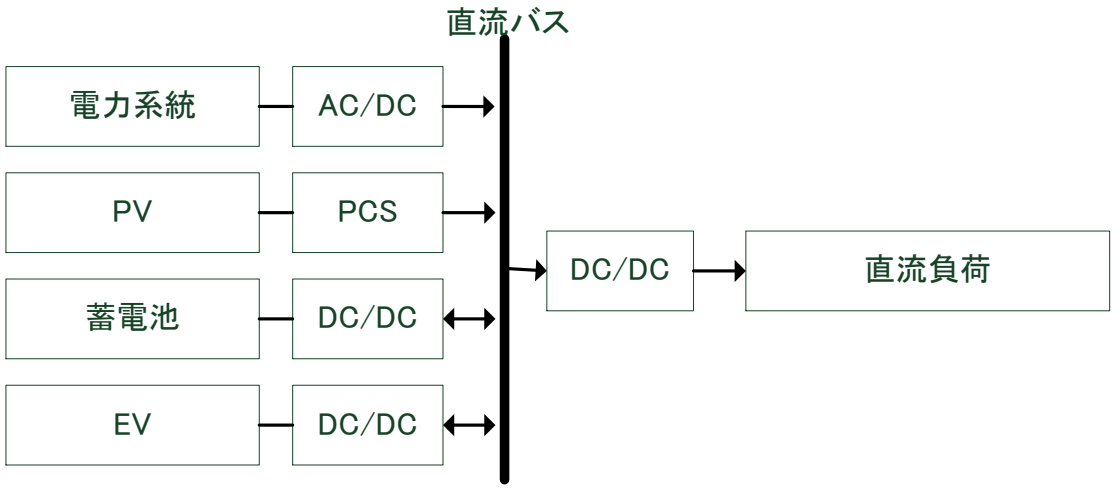

図 2.2 想定している直流 MG の構成図

2.2.2　当日運用機能の概要

当日運用においては，得られた前日計画を元にして各設備の 1 秒間の出力を決定する．前日計画においては，出力が W ないし kW で決定されたが，当日運用においてはこれを元にして各設備の出力電圧および出力電流を決定する．前日計画と比してさらに短い時間帯幅，本論文では 1 秒を対象として出力を決定するが，1 秒という短い時間帯幅の中で現在の MG の各設備の状態を確認し，設備に指令を与えるということが必要となる．そのため，本論文では当日運用においては最適化計算を行わない．もし予測誤差がなければ，出力は計画通りのものとなるが，予測誤差がある場合は様々な設備の制約に応じてフレキシブルに出力を変化させる必要がある．

2.2.3　リアルタイム制御機能の概要

直流 MG 内の各設備を当日運用通りに動作させるために, PCS や DC/DC コンバータなどの電力変換装置を瞬時瞬時に制御する機能である．過渡的な電圧変動や突入電流などが発生しないようなソフトスイッチング動作が求められる．

2.3　提案する前日計画機能

前日計画における入力データおよび出力データのイメージ図を図 2.3 に示す．図中における細線ブロックはデータを意味し，太線ブロックはデータ処理を意味する．前日計画においては，天気予報と保持しているデータを用いて最適運用計画を得る．すなわち，対象とする設備において

これまでに得られた，ある程度の計測期間がある日射量データベース，負荷データベースを利用する．このデータベースにおいては，最適計画における時間帯幅と同じかそれよりも短い時間帯幅での計測データが必要となる．これに加えて，対象とする設備におけるこれまでの天気予報実績とこれに対応して実際の天候がどのようになったかというデータベースに関しても予め構築する必要がある．これに関してもより時間帯幅が短い方が有効であると考えられるが，特に予報に関しては 30 分毎のような細かいデータを得ることは困難であると考えられる．そのため，本論文では 3 時間毎という計画における時間帯幅よりも長いデータを利用した．対象とする設備の容量や効率等の仕様，連系している電力系統が排出するであろう時間帯別 CO_2 排出量に関しても予めデータとして保持する必要がある．計画を行うに際して用意するデータとしては，翌日の天気予報，翌日の日付，翌日の EV 運用予約ならびに計画時点での蓄電池および EV の充電量(SOC)情報である．

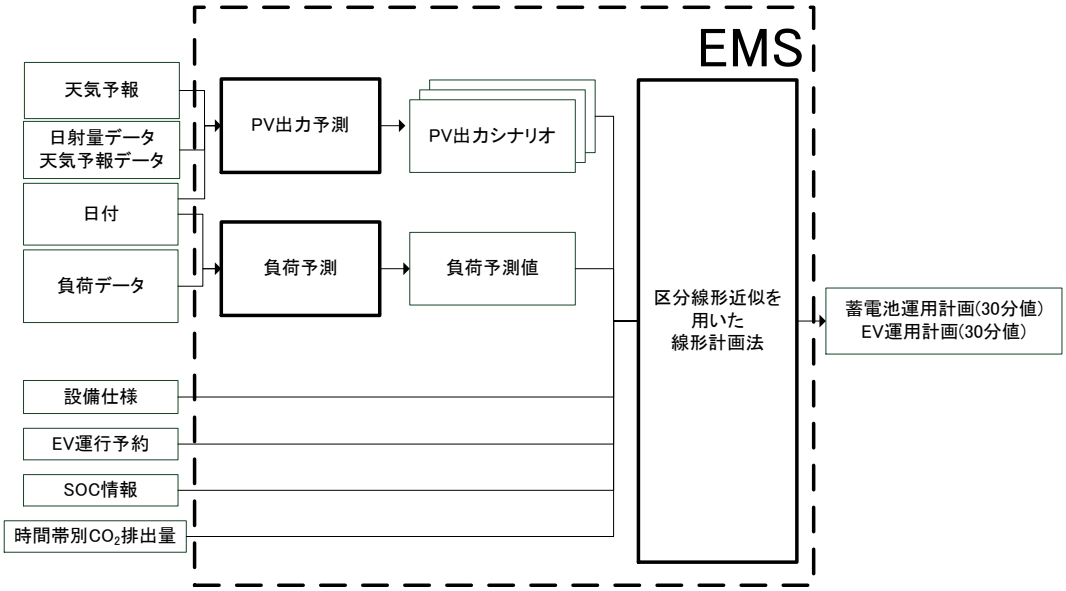

図 2.3 前日計画における入力データと出力データのイメージ

2.3.1 太陽光発電出力予測におけるシナリオ生成

本論文では PV 出力をシナリオとして扱うが，これを生成するためには天気予報，日射量データおよび天気予報データならびに日付を利用する．本論文では図 2.4 に示すような気象庁発表の 3 時間毎天気予報データ[23]を利用することを想定している．この天気予報では，3 時間毎の天気予報が晴，曇，雨(雪)の 3 段階で示される．また，午前 5 時，午前 11 時，午後 5 時の 3 回更新が行われている．

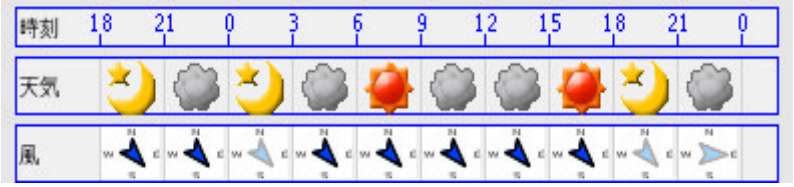

図 2.4 気象庁から発表される天気予報の一例

本論文では，これにならい天気予報を晴(F)，曇(C)，雨(R)の 3 種類の情報として扱い，データベースを構築した．一方で，気象庁発表の過去の気象データにおいては，天気の概況はさらに細かく分類されている．本論文では，特に頻出している 5 種類の天気，すなわち，快晴(CL)，晴(F)，薄曇(SC)，曇(C)，雨(R)として扱い，気象庁から公開されているすべてのデータを用いて天気実績データベースを構築した．この 2 種類のデータベースを組み合わせることにより，ある天気予報に対して，実際の天候がどのようになったのかという実績を知ることが可能となる．加えて，その情報を用いて，天気予報を入力した際にその時間帯の天候がどのようになるかという確率を求めることができる．

本論文では，得られた全てのデータのうち，PV 出力が発生する時間帯である午前 6 時から午後 6 時までの時間帯のデータのみを利用した．すなわち，3 時間毎の天気予報では 1 日につき 4 時間帯分のデータを利用した．また，遷移確率は季節や時間に依存しないものと仮定し，(2.1)式のように全データに対する該当データ数から確率 $p^{f,a}{}_{w}$ を計算した．

$$p^{f,a}{}_{w}=\frac{D^{f,a}}{\sum_{f}\sum_{a}D^{f,a}},\quad \begin{matrix} f\in 1,2,3 \\ a\in 1,2,3,4,5 \end{matrix} \tag{2.1}$$

ここに f は天気予報の種類を示し，1, 2, 3 の数値はそれぞれ F,C,R に対応している．a は天気実績の種類を示し，1, 2, 3, 4, 5 はそれぞれ CL, F, SC, C, R に対応している．D は時間帯を表している．表 2.1 に本論文で構築したデータベースから求められた天候の遷移確率の一例として，2013 年 5 月 17 日から 2014 年 9 月 5 日までのうち，データ記録が正常に行われた 174 日間のデータを用いた北海道帯広市におけるデータを示す．

表 2.1 データベースより算出した天候の遷移確率の一例(帯広市)

予報	実績				
	CL	F	SC	C	R
F	0.3605	0.2517	0.1769	0.1769	0.0340
C	0.0519	0.1556	0.0370	0.5778	0.1778
R	0.0556	0.1481	0.0556	0.2407	0.5000

次に，この遷移確率と翌日の天気予報データを用いて，対象とする日の全シナリオの発生確率を計算する．なお，シナリオそのものの発生確率は，1 時間帯の発生確率を乗算したものである．例として，4 時間帯の天気予報が”F F F F”として与えられた場合，”CL CL CL CL”というシナリオが発生する確率は，0.3605×0.3605×0.3605×0.3605=0.0169 となる．確率を計算する場合と同様に，PV 出力のある 4 時間帯だけに限定して考えると，考えうるすべてのシナリオは 5^4=625 通りとなる. しかしながら, 全てのシナリオに適用して解くには計算機の制約上困難な数であるため，次に示すシナリオ削減アルゴリズムを用いて，処理可能な個数までシナリオを削減する．

本論文では，文献[24]で示されるシナリオ削減アルゴリズムを用いて，PV 出力予測シナリオ個数を削減する．これは，繰り返し計算によってシナリオを 1 つずつ統合していくことにより，利用者の望ましい数までシナリオを削減することができるアルゴリズムである．まず，前処理とし

て，先述の天気のシナリオを PV 出力予測シナリオに変換する必要がある．本論文では入力された日付を利用して，過去5年間のデータのうちその日付の前後15日間の日射量データを利用した．また，検証の際には入力された日付より先のデータは，データベースに存在していたとしても，未来のデータとして扱うため利用しない．例として，2014 年 4 月 1 日が入力された際には，2008 年 4 月 1 日から 2014 年 3 月 31 日のデータまでを利用する．この日射量データの日付，時間帯，日射量と天気実績データベースの日付，時間帯，天候実績とを比較し，各時間帯，各天候状況における日射量の平均値を求める．さらに，これに PV 定格容量と変換効率を乗じることで，PV 出力予測シナリオを生成する．PV 出力がない時間帯については，一律 0 とする．次に，(2.2)式を用いて，PV 出力予測値の差が最大となる値から，各シナリオ同士の距離 c_{kj} を計算する．

$$c_{kj} = \max_{t\in\{1,\cdots,Tfin\}} \left| P^{k}_{PV\_F}(t) - P^{j}_{PV\_F}(t) \right|, k, j, = 1,\cdots, N \tag{2.2}$$

ここに k および j はシナリオ番号，t は時間帯数，T_{fin} は時間帯数の最大値，N は全シナリオ数，$P_{PV\_F}(t)$ は該当シナリオにおける時刻 t の PV 出力予測値[kW]である．これをすべての組合せに対して計算し，その最小値とそのシナリオ自体の発生確率を乗ずることでそのシナリオの確率距離を計算する．すなわち，1 回目の計算における確率距離 $z_l^{[1]}$ は，

$$z_l^{[1]} := p_l \cdot \min_{j\neq l} c_{lj}, l = 1,\cdots, N \tag{2.3}$$

ここに，上添字の 1 は 1 回目の繰り返しであることを示している． p_l はシナリオ l の発生確率である．すべてのシナリオについて確率距離を計算した後，最も確率距離の短いものを削減対象として記録する．確率距離が小さいということは，シナリオ距離が短い，すなわち他のシナリオと類似度が高く，かつ発生確率そのものも小さいということである．(2.4)式のように，1 回目の計算において削減対象となったシナリオ l_1 は削減リスト $\mathbf{J}^{[1]}$ に記録される．

$$\text{Choose } l_1 \in arg\min_{l\in\{1,\cdots,N\}} z_l^{[1]},\ \ set\ \ \mathbf{J}^{[1]} := \{l_1\} \tag{2.4}$$

同様の手順を繰り返すことによって，シナリオ削減リストを生成する．今，シナリオ削減後に n 個のシナリオだけを扱いたい場合，繰り返し回数は N-n+1 回となる．s ステップ目におけるシナリオ距離，確率距離，削減対象シナリオおよび削減リストは以下のようになる．なお，それぞれの数値の計算においては，リストに記載されたシナリオに関しては無視して計算を行う．

$$c_{kl}^{[s]} := \min_{j\notin \mathbf{J}^{[s-1]}\cup\{l\},} c_{kj}, \text{for } l \notin \mathbf{J}^{[s-1]}, k \in \mathbf{J}^{s-1} \cup \{l\} \tag{2.5}$$

$$z_l^{[s]} := \sum_{k\in \mathbf{J}^{[s-1]}\cup\{l\}} p^k \cdot \min c_{kl}^{[s]} l \notin \mathbf{J}^{[s-1]} \tag{2.6}$$

$$\text{Choose } l_s \in arg\min_{s\notin \mathbf{J}^{[s-1]}} z_l^{[s]}, \mathbf{J}^{[s]} := \mathbf{J}^{[s-1]} \cup \{l_s\} \tag{2.7}$$

N-n+1 回の計算終了後，リストに記載されたシナリオの発生確率は，それぞれ最もシナリオ距離の短いシナリオに再分配される．すなわち，再分配後のシナリオ j の発生確率を $\overline{q^j}$ とすると，

$$\overline{q^j} = p^j + \sum_{i\in \mathbf{J}(j)} p^i \tag{2.8}$$

$$\mathbf{J}_j = \{i \in \mathbf{J} : j = j(i)\} \text{ and } j(i) \in arg\min_{j \notin \mathbf{J}} \max_{t \in \{1,\cdots,T\}} \left| P^i_{PV\_F}(t) - P^j_{PV\_F}(t) \right|, \forall i \in \mathbf{J} \quad (2.9)$$

となる．ここに **J** は N-n+1 回の計算終了後における完成された削減リストである．

以上の操作により，*n* 個の PV 出力予測シナリオが得られ，次項に示す区分線形近似を用いた線形計画法に適用される．

2.3.2　区分線形近似を用いた線形計画法における定式化

前日計画では，先に求めた PV 出力予測シナリオ，翌日の MG 負荷需要予測値，EV の運行計画等をデータとして与え，MG 内の CO_2 排出量最小化を目的とした最適化計算によって翌日の 30 分ごとの蓄電池，EV の充放電電力量(kW 値)を決定する．負荷需要予測値に関しては，PV 出力予測とは異なり平日か休日か程度の差異しか存在しないため，本論文では対象とする MG の負荷データベースおよび日付から，それぞれの平均値を負荷需要予測値として与える．なお，MG 内において CO_2 排出を伴うものは系統からの購入電力のみであることから，CO_2 排出量の計算は購入電力に対して CO_2 排出量原単位を乗ずることで求めることができる．

先述の通り，本論文では PV 出力予測はある発生確率を持ったシナリオ群として扱うため，MG 内の需給アンバランスを系統からの購入電力を調整することで補償することとすれば，系統からの購入電力，ひいては CO_2 排出量もシナリオの数だけ考慮しなくてはならない．したがって， PV 出力のシナリオ毎に CO_2 排出量が異なることから，シナリオ発生確率を乗じた総 CO_2 排出量の期待値を最小化する運用計画となる．すなわち，目的関数を次式のように定義する．

$$min\ f = \sum_j \overline{q^j} \left(\sum_t E(t) \cdot P^j_{conv}(t) \cdot \Delta T \right) \quad (2.10)$$

ここに，$E(t)$：時間帯 t における系統購入電力の CO_2 排出原単位 [kg-CO_2/kWh]，ΔT ：時間帯幅 [h]，$P^j_{conv}(t)$：シナリオ j に対する時間帯 t の購入電力 [kW]，$\overline{q^j}$ ：シナリオ j の発生確率である．

本論文では，図 2.2 に示した MG を
図 2.5 に示すような電気回路としてモデル化すると共に，以下の制約条件を考慮する．

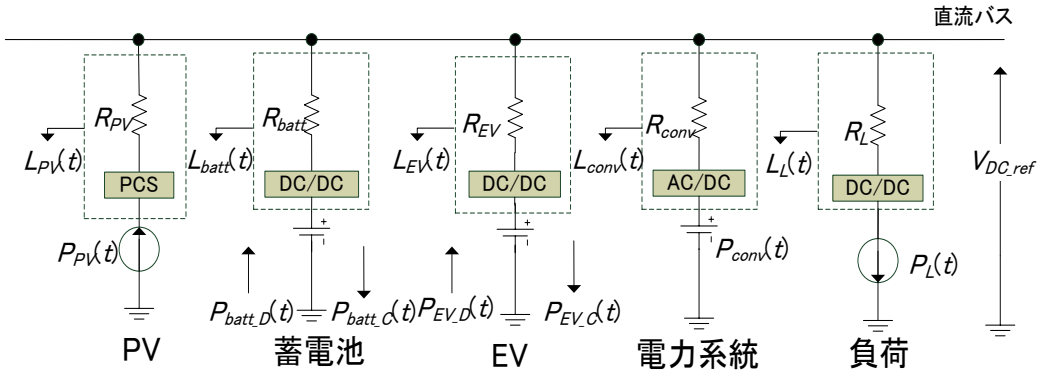

図 2.5 電気回路としてモデル化した MG

a) 需給平衡制約

$$\eta_L \cdot P_L(t) + L_L(t) = \left\{\eta_{conv} \cdot P_{conv}^{j}(t) - L_{conv}^{j}(t)\right\} + \left\{\eta_{batt} \cdot P_{batt\_D}(t) - \frac{P_{batt\_C}(t)}{\eta_{batt}} - L_{batt}(t)\right\} + \left\{\eta_{EV} \cdot P_{EV\_D}(t) - \frac{P_{EV\_C}(t)}{\eta_{EV}} - L_{EV}(t)\right\} + \left\{\eta_{PV} \cdot P_{PV}^{j}(t) - L_{PV}^{j}(t) - \eta_{PV} \cdot Rjc^{j}(t)\right\} \quad (2.11)$$

ここに，η：各装置に付随する変換器の変換効率，$P(t)$：時間帯 t における各機器の出力 [kW]，$L(t)$：時間帯 t における線路抵抗損失 [kW]，$Rjc(t)$：時間帯 t における PV の出力抑制量 [kW]である．なお，下添字は図 2.5 に示されている各機器の記号を表しており，上添字は予測シナリオを表している．また，前述のように PV 予測出力を複数のシナリオとして扱うため，系統からの購入電力やその他の関係部位の電力損失もシナリオ毎に変化するものとして取り扱う．このように定式化することで，想定するどのシナリオが起こっても制約を満たすことのできる計画を得ることができる．なお，各線路抵抗における損失は以下の手順で導出する．

ここでは，例として CONV と直流バス間の線路抵抗における損失と変換損失の総和である $L_{conv}(t)$ を導出する．今，直流バスにおける電圧が一定であると考えると，$L_{conv}(t)$ は以下の式で表現することができる．

$$L_{conv}(t) = (1-\eta_{conv}) \cdot P_{conv}(t) + \frac{V_{DC\_ref}^{2}}{2R_{conv}} - \frac{1}{2}\sqrt{\left(2\eta_{conv} \cdot P_{conv}(t) + \frac{V_{DC\_ref}^{2}}{R_{conv}}\right)^{2} - 4\eta_{conv}^{2} \cdot P_{conv}(t)^{2}} \quad (2.12)$$

他の構成設備についても，同様に電力の 2 次式で表現することができる．しかしながら，本論文における提案手法では，線形計画法を用いて運用計画を得るため，このままでは適用することができない．したがって，本論文では図 2.6 に示すような区分線形近似によって各損失を表現する．この例では，損失を電力 P が 10 W よりも小さな領域では L1，それ以外の領域では L2 という 2 本の線形式で表現している．

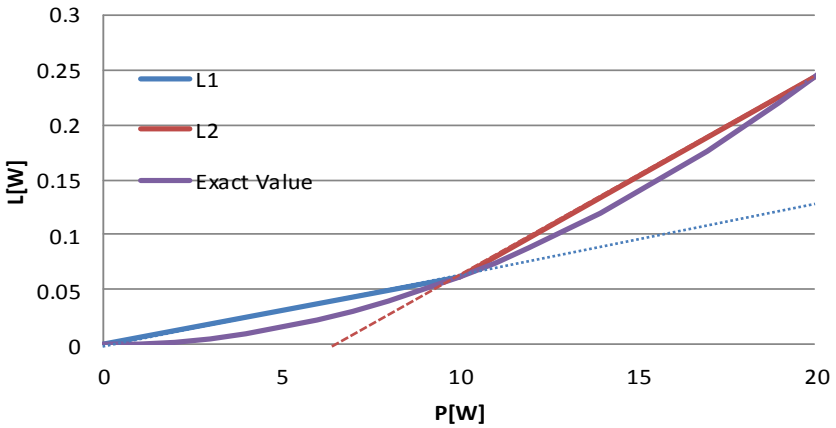

図 2.6 損失の区分線形近似のイメージ

細かく線形近似を行うことでより精度の良い解が得られることから，本論文では各損失を 100 本の線形式で表現した．また，線路抵抗を通過する電力 P に関しては，各機器の仕様が明確であることから，0 W から最大値までを 100 分割した．本論文における目的関数は，電力の損失が多くなるほど悪化するため，(2.13)式のように，損失がすべての線形式以上という制約を考慮することで，該当する区間の制約だけが有効となる．

$$L^{j}(t) \geq L_1, L_2, \cdots, L_{100} \tag{2.13}$$

b) 蓄電池と EV の充電残量（SOC）の時間推移

$$SOC_{batt}(t+1) = SOC_{batt}(t) + \{\mu_{batt} \cdot P_{batt\_C}(t) - P_{batt\_D}(t)\} \cdot \Delta T \tag{2.14}$$

$$SOC_{EV}(t+1) = SOC_{EV}(t) + \{\mu_{EV} \cdot P_{EV\_C}(t) - P_{EV\_D}(t)\} \cdot \Delta T \tag{2.15}$$

ここに，$SOC_{batt}(t)$および$SOC_{EV}(t)$：時間帯 t 内の最初の時点における蓄電池および EV の SOC [kWh]，μ_{batt}およびμ_{EV}：蓄電池および EV の往復充放電効率である．

c) 蓄電池・EV の日間利用制約

蓄電池は，一日の最初の時点で十分に充電されていれば，放電可能な電力量が多くなるが，反面，充電方向では使い勝手が悪くなってしまう．このように，蓄電池の初期 SOC は計画断面において運用の自由度と密接に関係する．このことは EV に対しても同様に考えられる．そこで本 EMS では，蓄電池ならびに EV の SOC の最終値に関して，翌々日の天気予報情報をもとに制約を設ける．すなわち，翌々日の午前 6 時から午前 9 時の天候が晴(F)ならば満充電の 30 %，曇(C)ならば同 50 %，雨(R)ならば同 70 %となるように SOC 最終値を決める．

$$SOC_{batt}(T^{fin}+1) = (0.3 \text{ or } 0.5 \text{ or } 0.7) \cdot SOC_{batt}^{max} \tag{2.16}$$

$$SOC_{EV}(T^{fin}+1) = (0.3 \text{ or } 0.5 \text{ or } 0.7) \cdot SOC_{EV}^{max} \tag{2.17}$$

ここに，T^{fin}：前日計画における最終時間帯，$T^{fin}+1$：計画日翌日の最初の時間帯，SOC_{batt}^{max}：蓄電池 SOC の最大値 [kWh]，SOC_{EV}^{max}：EVSOC の最大値 [kWh]である．

d) 各設備の容量制約

$$0 \leq P_{batt\_C}(t) \leq P_{batt}^{max} \tag{2.18}$$

$$0 \leq P_{batt\_D}(t) \leq P_{batt}^{max} \tag{2.19}$$

$$0 \leq P_{EV\_C}(t) \leq P_{EV}^{max} \tag{2.20}$$

$$0 \leq P_{EV\_D}(t) \leq P_{EV}^{max} \tag{2.21}$$

$$0 \leq P_{conv}(t) \leq P_{conv}^{max} \tag{2.22}$$

$$0.3 \cdot SOC_{batt}^{max} \leq SOC_{batt}(t) \leq SOC_{batt}^{max} \tag{2.23}$$

$$0.3 \cdot SOC_{EV}^{max} \leq SOC_{EV}(t) \leq SOC_{EV}^{max} \tag{2.24}$$

ここに，各変数の上添字の max：当該変数の最大値である．

なお，本論文では蓄電池および EV に関しては，事故等で系統から電力を供給できない場合に備え，常に 30 %以上の SOC を確保するように運用を行うことを想定している．

e) EV の走行に関する制約

EV が走行中で MG に接続されていない予定である時間帯には，EV の充放電を行うことができないため，当該時間帯の EV の充放電電力は 0 とする．

$$P_{EV\_C}\left(T^{running}\right)=P_{EV\_D}\left(T^{running}\right)=0 \qquad (2.25)$$

ここに，各変数の上添字の $T^{running}$ ：EV が走行している時間帯である．

また，走行時に消費されるエネルギーは，計算の便宜上，EV が出発した時点にまとめて消費されるものとみなして SOC を減じる．

$$SOC_{EV}\left(T^{run}+1\right)=SOC_{EV}\left(T^{run}\right)-\eta_{run}\cdot D \qquad (2.26)$$

ここに，η_{run} ：EV の km あたりの走行効率[kWh/km]，D ：走行予定距離 [km]，T^{run} ：EV の走行開始予定時間帯である．

実際の運用においては，EV 走行開始予定時刻までに，十分な SOC を確保しておく必要がある．そこで，出発までの SOC が満充電の 90 %となるように次の制約を課すこととする．

$$SOC_{EV}\left(T^{run}\right)\geq 0.9\cdot SOC_{EV}^{max} \qquad (2.27)$$

2.4 帯広市マイクログリッドを対象とした数値試算

第 1 章で述べたように，我が国の環境省は「平成 24 年～26 年地球温暖化対策技術開発・実証研究事業　自立・分散型エネルギー社会の実現に向けた直流方式による地域間相互エネルギー融通システムの開発」というプロジェクトを立ち上げ，MG に関する包括的な実証実験を行った．このプロジェクトでは，大きく 2 つのエリアに分けて実証実験が行われた．一方は山形市において，エリア間のエネルギー融通に関する実証実験が行われた．もう一方は帯広市において直流給電を行う MG に関するプロジェクトであり，本論文はこのプロジェクトの一環として行われた．したがって，本節ではこの帯広市 MG で得られたデータを利用して，EMS 機能の検証を行った．

2.4.1 帯広市マイクログリッドの設備構成

帯広市直流 MG 実証施設は，帯広市清掃センター内という実際のオフィスビルを対象として，ここに各設備を導入することで構築された．2013 年にプロジェクトが施行され，2015 年にすべての設備の実装が行われた．現在はすべての設備が撤去されている．これらの工事は，株式会社 NTT ファシリティーズが主体となって行われた．帯広市清掃センターの施設外観を図 2.7 に示す．

図 2.7 帯広市清掃センター外観

図 2.8 に帯広市 MG の設備構成イメージを示す．MG 内には電源として PV システム，蓄電池システムおよび EV が導入された．また，負荷はオフィスで実際に利用される直流負荷が新たに導入され，内訳は，ノートパソコン，LED 照明，モニタおよび冷蔵庫であった．これらの設備はすべて共通直流バスに接続され，供給電圧は 380 V 一定となるように制御されていた．

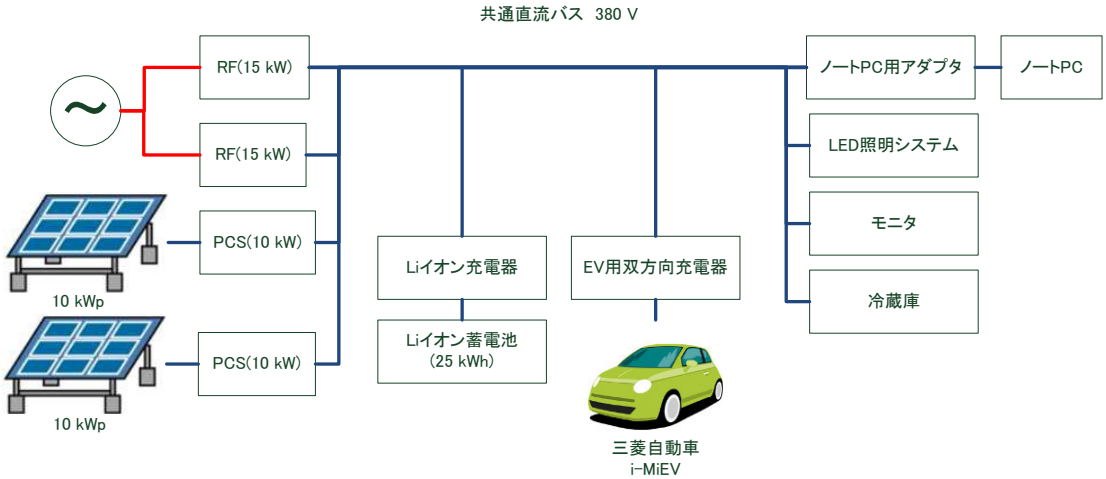

図 2.8 帯広市 MG の設備構成イメージ

実際に導入された PV システム，蓄電池システムおよび EV システムの外観を図 2.9，図 2.10 および図 2.11 に示す.

図 2.9 帯広市 MG に建設された PV システム

図 2.10 帯広市 MG に建設された蓄電池システムの外観

図 2.11 帯広市 MG に導入された EV の外観

表 2.2 に各設備におけるパラメータを示す．なお，これらのデータは実際に構築された設備において計測されたデータである．本論文ではこれらの数値を用いて試算を行った．

表 2.2 各設備におけるパラメータ

変数	意味	値
η_{conv}	連系コンバータ(CONV)の変換効率	0.95
η_{batt}	蓄電池用充電器の変換効率(往復)	0.94
η_{EV}	EV用双方向充電器の変換効率	
η_{run} [25]	EVの走行燃費	0.11 [kWh/km]
η_{PV}	PV用PCSの変換効率	0.965
η_{L}	負荷用コンバータの変換効率	0.9
μ_{batt}	蓄電池の充放電効率(往復)	0.95
μ_{EV}	EVの充放電効率(往復)	0.95
P_{conv}^{max}	CONVの最大電力	30 [kW]
P_{PV}	PVの定格	20 [kWp]
$P_{batt\_C}^{max}$	蓄電池の最大充電電力	10 [kW]
$P_{batt\_D}^{max}$	蓄電池の最大放電電力	
$P_{EV\_C}^{max}$ [25]	EVの最大充電電力	3 [kW]
$P_{EV\_D}^{max}$ [25]	EVの最大放電電力	
SOC_{batt}^{max}	蓄電池の最大SOC	20 [kWh]
SOC_{EV}^{max} [25]	EVの最大SOC	12.8 [kWh]
$R_{conv}, R_{batt}, R_{EV}, R_{PV}, R_{L}$	各設備の線路抵抗	0.1 [Ω]

2.4.2　最適化計算による前日計画の効果

帯広市清掃センターでは，新たに導入された直流負荷のトレンドが計測され，30 分値のデータとして蓄積された．本論文では，この負荷データに加えて気象庁発表の帯広市における気象データを用いて各種数値シミュレーションを行った．まず，提案手法であるシナリオを考慮した最適化計算で得られた前日計画と，特に最適化計算を用いずに系統から購入する電力に起因する CO_2 排出原単位の多寡から計画を決定する手法との比較を行った．

本論文では，系統から購入する電力には，それを発電する際に CO_2 の排出が伴うものと考えており，この CO_2 は MG から排出されたものとして計上する．CO_2 排出量は時間帯毎に変化するものと考え, 本論文では大きく昼(午前 6 時～午後 6 時)と夜(それ以外の時間帯)の 2 段階で考慮する．北海道電力㈱が公表している CO_2 排出原単位実績[26][27]をもとに, 昼の高負荷運転時には多くの火力機が運転していることを想定し 2013 年度実績の 0.681 [kg-CO_2/kWh]を，夜については将来的に原子力発電機が稼働することを見越して，2011 年度実績の 0.485 [kg-CO_2/kWh]を与えた．

最適化を用いない前日計画は次のように定義する．このようなシステムでは，予測情報を利用した最適運用計画を行わず，運用計画においてはタイマーによる単純な充放電のみを行えるようにする．すなわち，昼と夜とを分け，それぞれの時間帯で一定の動作をするような計画である．例えば，PV の余剰電力を見越して昼に充電，夜に放電とルールを設定したならば，各時間帯の充放電計画は以下のように決定される．

$$P_{batt\_C}(t) = \left(0.8 \cdot SOC_{batt}^{max} - SOC_{batt\_init}\right)/12 \qquad \text{(昼の時間帯)} \quad (2.28)$$

$$P_{batt\_D}(t) = \left(0.8 \cdot SOC_{batt}^{max} - 0.3 \cdot SOC_{batt}^{max}\right)/12 \qquad \text{(夜の時間帯)} \quad (2.29)$$

すなわち，情報としては SOC の初期値だけを利用し，最大値とのギャップを昼に充電，夜の時間帯ではすべての SOC を使い切るように放電するというものである．EV に関しても同様の動作を想定する．この計画策定方法を方法①とする．また，もう一つのルールとして，CO_2 排出原単位の低い夜に充電し，昼に放電することも考えられる．こちらに関しては，計画①の充放電方向を真逆にしたものを想定し，方法②として扱う．実際の運用シミュレーションでは，EMS ありの場合と同じルールで各設備の出力を調整する．

図 2.12 に方法①および方法②ならびに提案するアルゴリズムにおけるそれぞれの四季の CO_2 排出量を示す．なお，本試算では保持している予報データのうち各季 4 週間(春季：2013 年 5 月 17 日～2013 年 6 月 29 日，2014 年 4 月 10 日～2014 年 4 月 24 日，夏季：2013 年 7 月 4 日～2013 年 8 月 31 日，2014 年 7 月 2 日～2014 年 7 月 11 日，秋季：2013 年 9 月 15 日～2013 年 11 月 28 日，冬季：2013 年 12 月 22 日～2014 年 2 月 22 日)を代表日として選出し，シミュレーションを行った. ただし, 秋季のデータに関しては, データ欠損があるため 23 日分のみとなっている. また，EV に関してはオフィス用の使われ方をすることを想定し，平日のみ自動車として運用され，休日は常時 MG に接続されているものとした．平日は午前 9 時から自動車として稼働し，40 km を走行したのち午後 3 時に再び接続されるものとした．

方法①，方法②ともに導入前と比較した場合は削減効果があるものの，提案する最適化計算ありの場合と比較すると CO_2 排出量削減効果は小さいことが確認できる．方法①に関しては，冬のシミュレーションにおいては EMS ありの場合よりも排出量削減効果があるが，四季の平均では EMS ありの場合より方法①では約 16.1 %, 方法②では約 21.9 %排出量が増加することがわかった. 以上のことから，前日計画においては提案手法のように最適化計算によって直流 MG の CO_2 排出

量削減効果が向上することが示された.

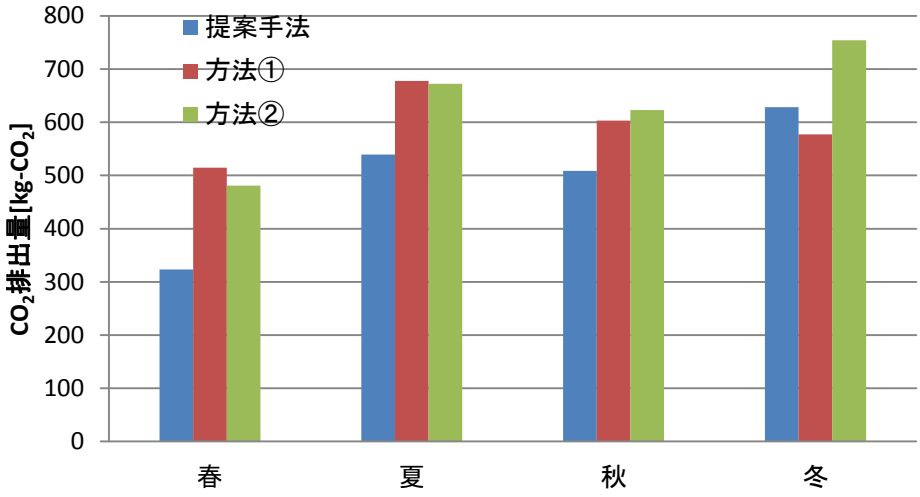

図 2.12 各前日計画における CO_2 排出量

2.4.3　シナリオを考慮した前日計画シミュレーション

本論文で提案する前日計画手法において，シナリオを考慮することの有効性を検証するため，前日計画を本アルゴリズムで策定した場合と，シナリオを考慮せず天気予報情報のみを用いて計画を行った場合とを比較した．試算条件は前節でと同様のものである．図 2.13 に 2013 年 6 月 26 日のデータに対して得られた，シナリオを考慮しない場合の前日計画を示す．各時間帯の積み上げ棒グラフは予測された MG の負荷曲線に合わせて，系統からの購入電力，PV，EV および蓄電池をどのように分担して供給しているかを示している．なお，それぞれの電力は各種変換器を介して共通バスでやり取りする電力であるため，需要と供給は一致している．また，PV 出力は天気予報（この日の天気予報は，”RRCC”であった）に基づいた予測値である．図 2.13 の場合はシナリオを考慮せず，天気予報と同じシナリオのみを採用している．この場合は天気予報に忠実に計画を策定しているため，午前 6 時から午前 9 時の間は PV 出力はそれほど大きくないことが見込まれており，この時間帯では蓄電池で吸収しなければならないほどの大きな PV 余剰電力が発生しないため，EV にのみ余剰電力と購入電力が充電される計画となっている．計画時点での CO_2 排出量は，34.9 [kg-CO_2]であった.

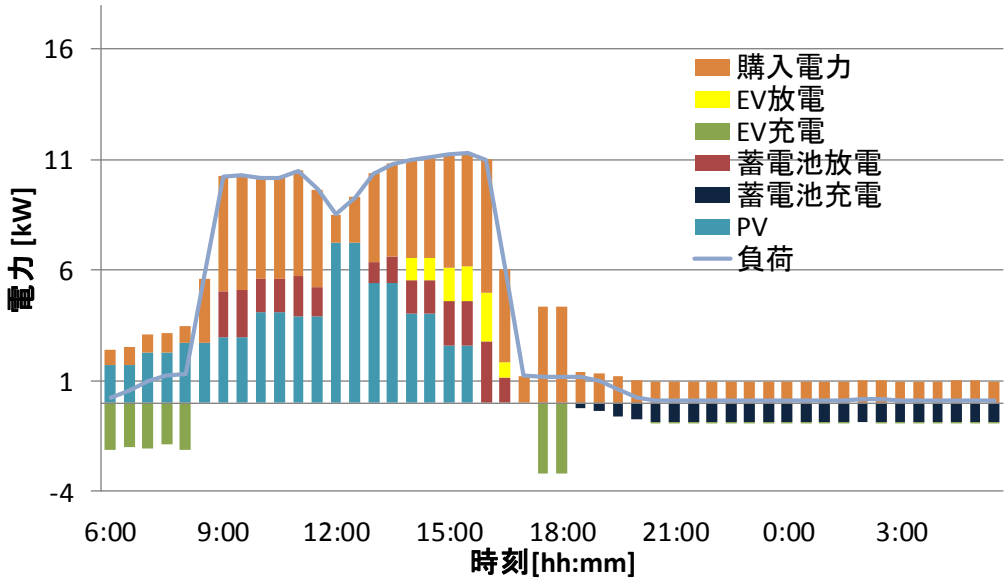

図 2.13 シナリオを考慮しない場合の MG 運用計画

図 2.14～図 2.23 に図 2.13 と同じ 6 月 26 日のデータに対して, シナリオ数を 10 個として提案手法を用いて得られた各シナリオにおける前日計画を図 2.13 と同様の方式で示す.

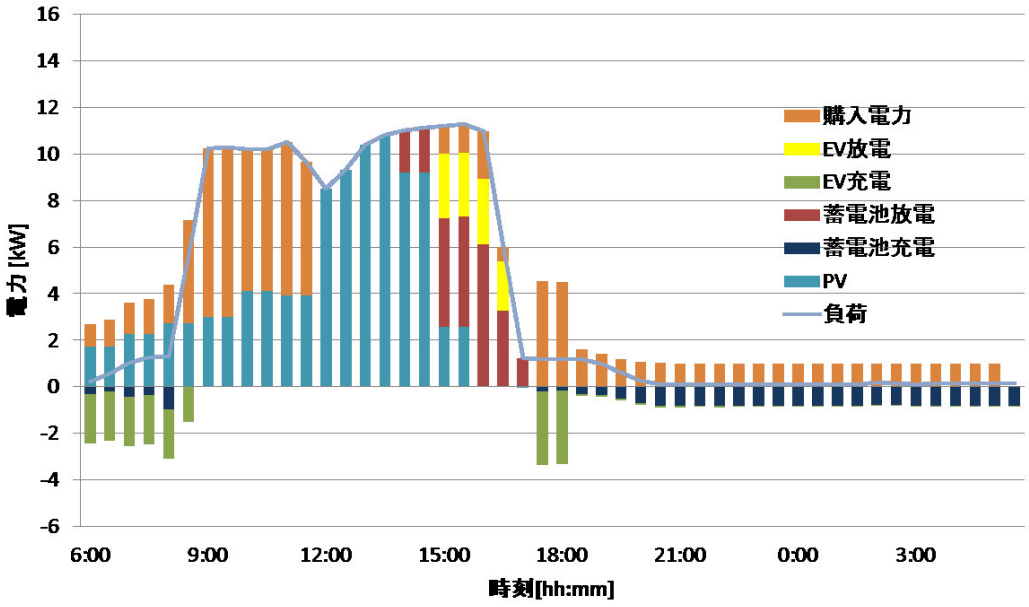

図 2.14 シナリオを考慮した場合の MG 運用計画(シナリオ“RRFC”)

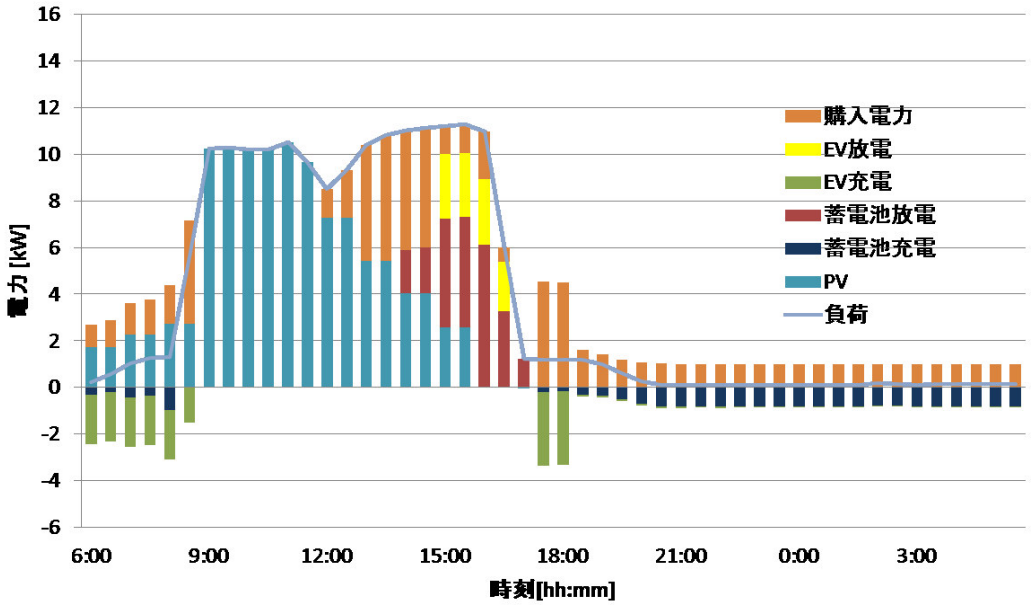

図 2.15 シナリオを考慮した場合の MG 運用計画(シナリオ“RFCC”)

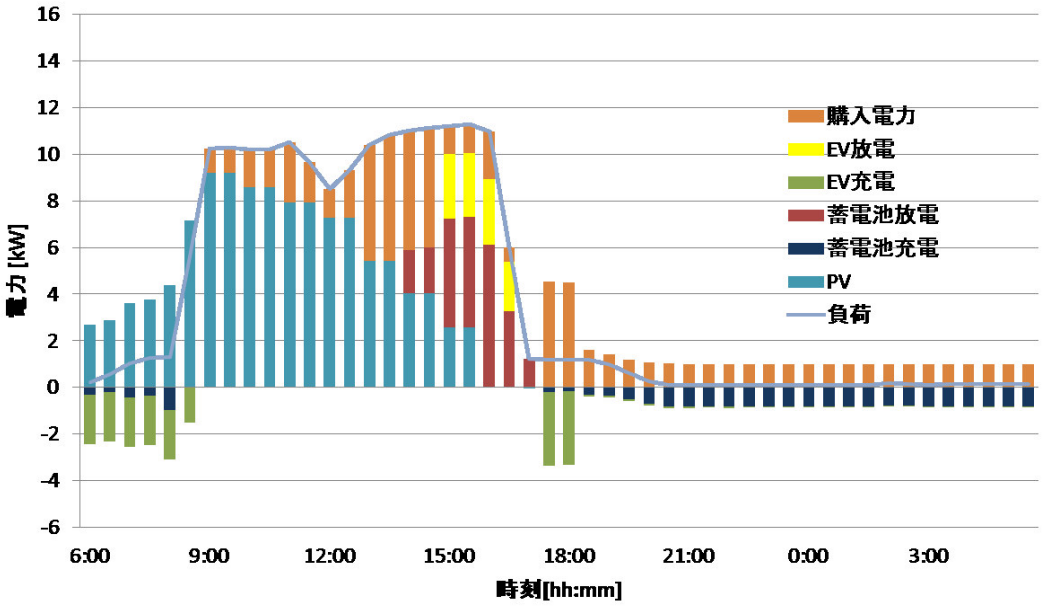

図 2.16 シナリオを考慮した場合の MG 運用計画(シナリオ“CCCC”)

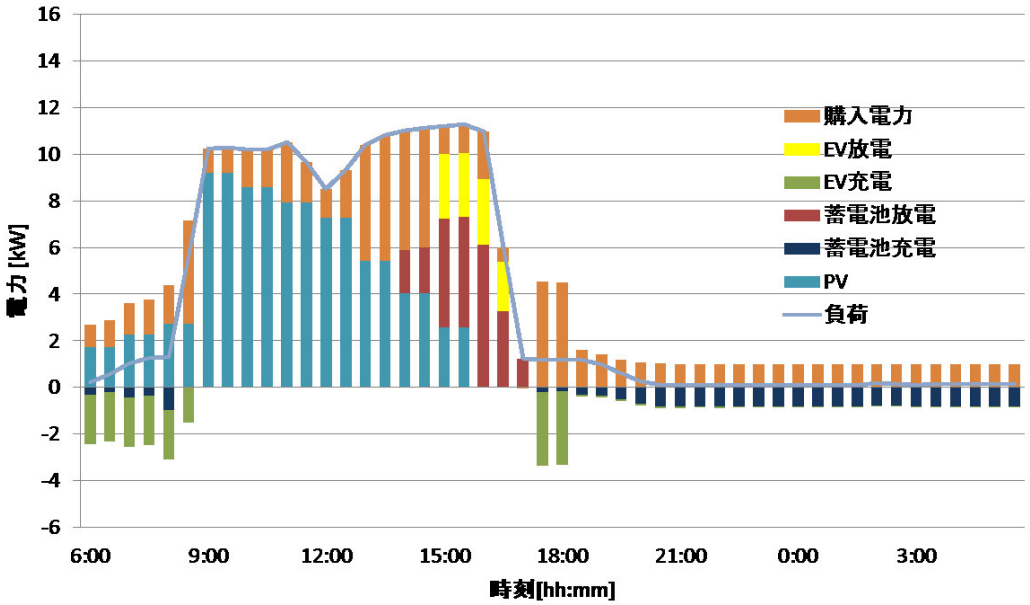

図 2.17 シナリオを考慮した場合の MG 運用計画(シナリオ“RCCC”)

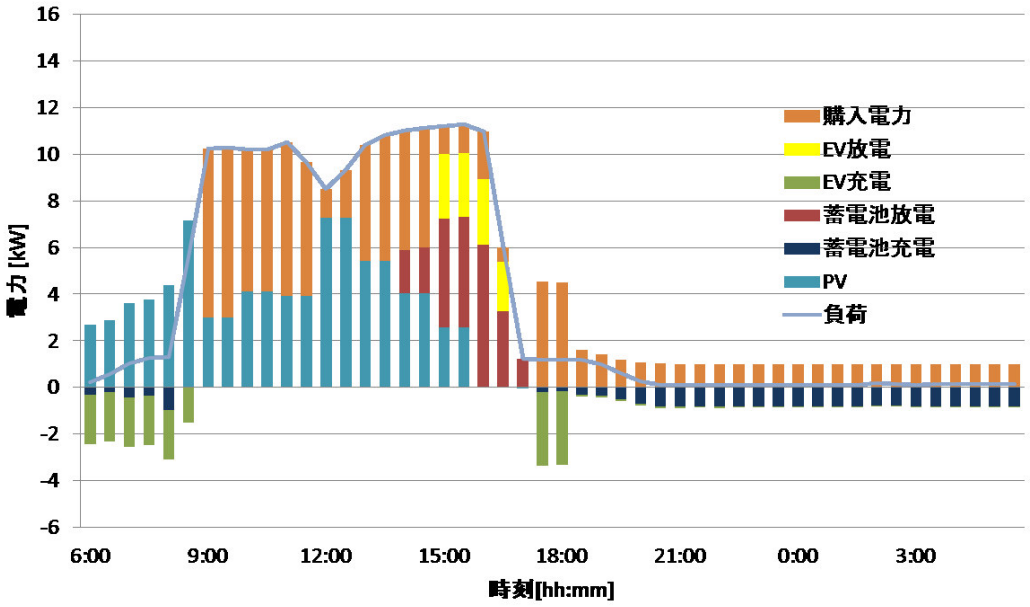

図 2.18 シナリオを考慮した場合の MG 運用計画(シナリオ“FRCC”)

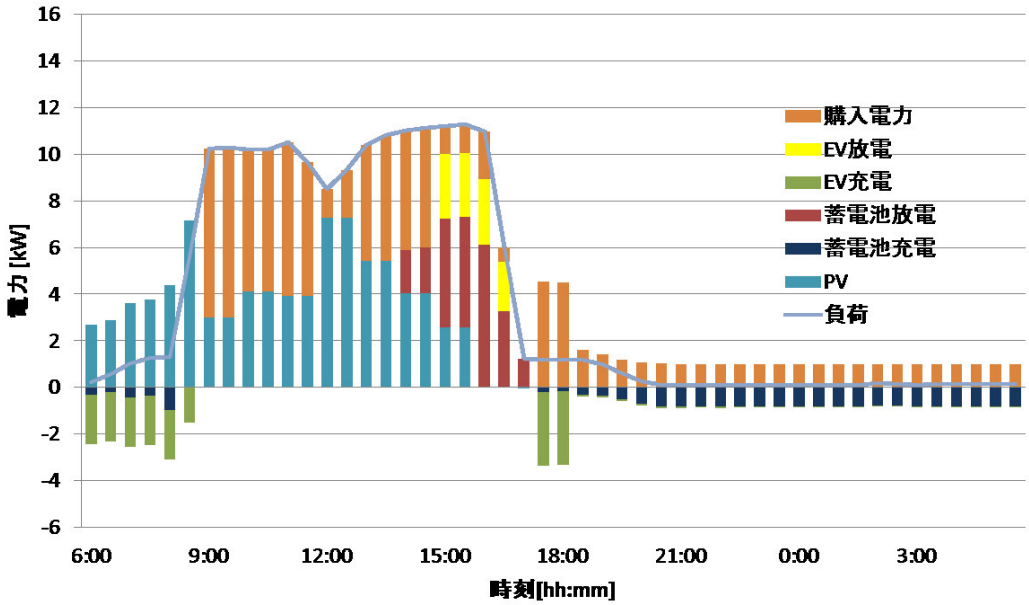

図 2.19 シナリオを考慮した場合の MG 運用計画(シナリオ“CRCC”)

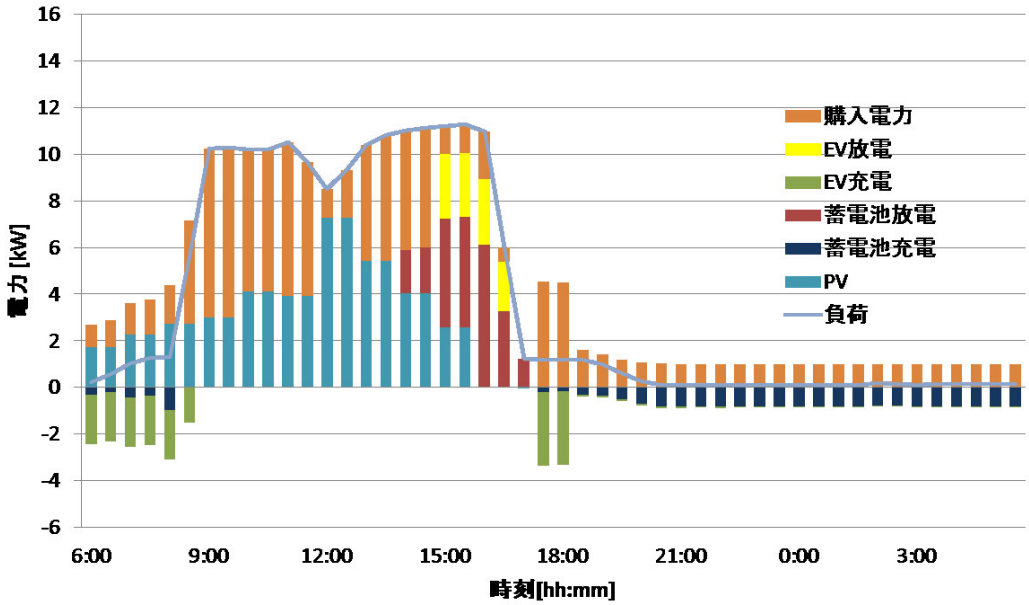

図 2.20 シナリオを考慮した場合の MG 運用計画(シナリオ“RRCC”)

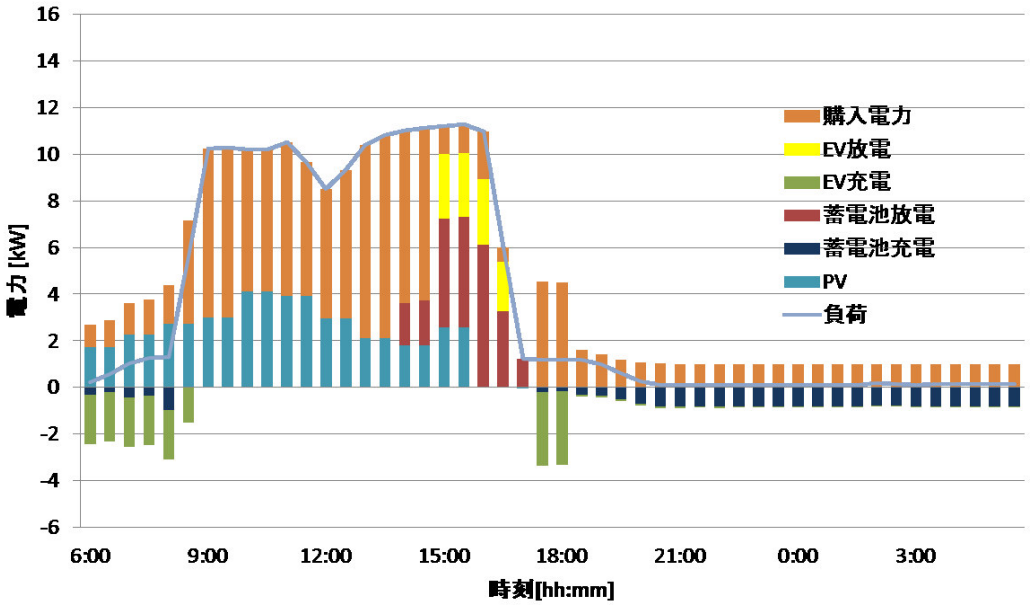

図 2.21　シナリオを考慮した場合の MG 運用計画(シナリオ“RRRC”)

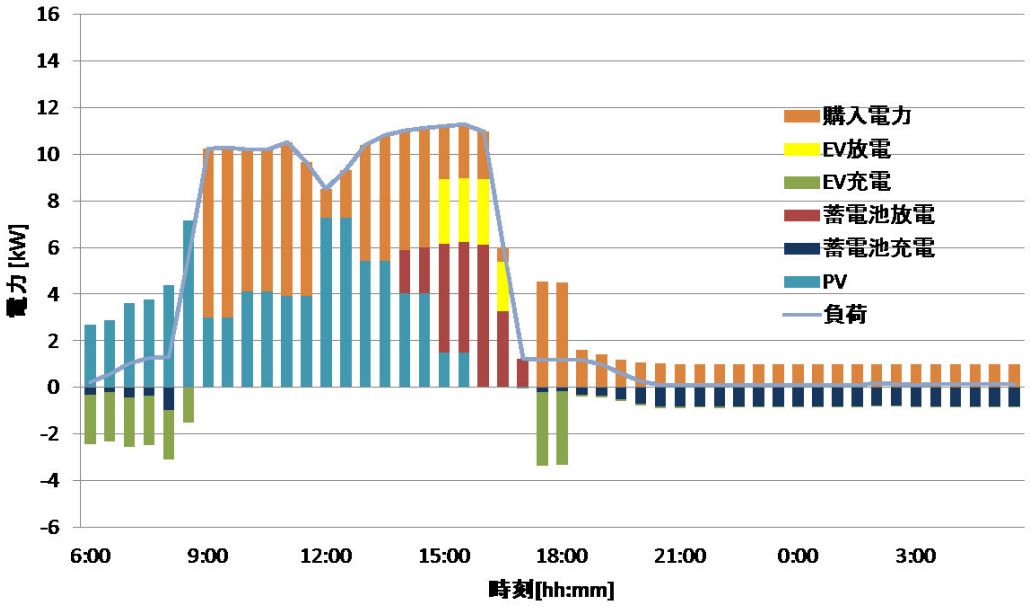

図 2.22　シナリオを考慮した場合の MG 運用計画(シナリオ“CRCR”)

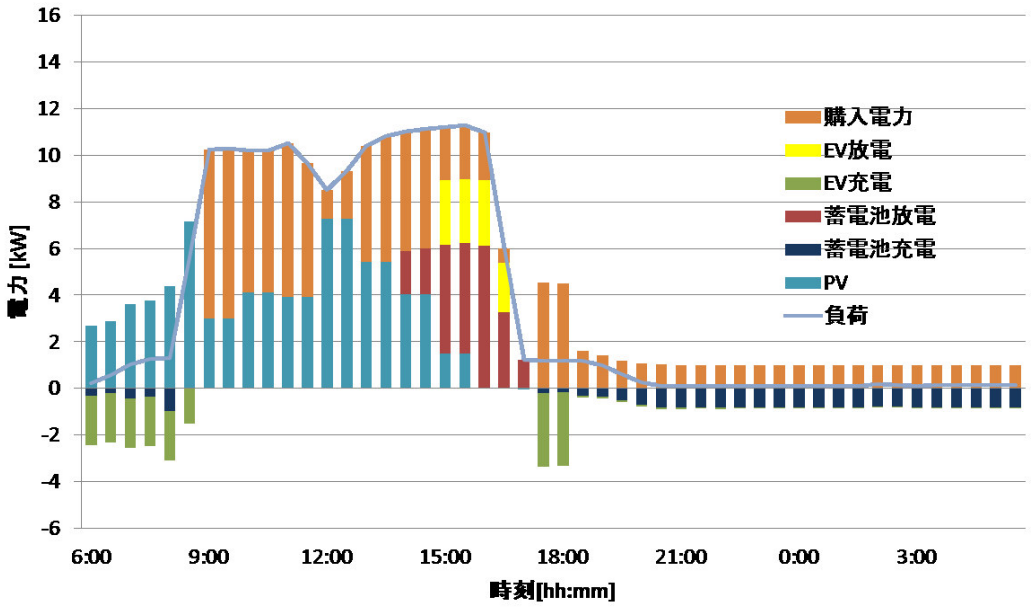

図 2.23 シナリオを考慮した場合の MG 運用計画(シナリオ“RRCR”)

シナリオを考慮した場合は採用された 10 個のシナリオのうち，図 2.16 に示した””CCCC”のように午前 6 時から午前 9 時の PV 出力が大きく見込まれるシナリオが最も発生確率が高いと見込まれたため，この時間帯では系統から電力を購入せず，蓄池および EV の充電がなされる計画が得られた．また，このようなシナリオでは日中は PV から十分な電力が供給されるため，蓄電池は放電を行わない計画となった. 計画時点での CO_2 排出量は, 選択されたシナリオの傾向として，図 2.13 の場合よりも PV 出力が大きいことが期待されたため，25.8 [kg-CO_2](ただし，期待値)であった.

表 2.3 に選択されたシナリオ，対応する CO_2 排出量および発生確率を示す.

表 2.3 得られたシナリオと対応する CO_2 排出量ならびに発生確率

	CO_2排出量 [kg-CO_2]	シナリオ発生確率
シナリオなし “R R C C”	34.9	
"R R F C"	27.3	0.13
"R F C C"	22	0.25
“C C C C”	21.4	0.3
“R C C C”	25.3	0.097
“F R C C”	31.6	0.043
“C R C C”	31.6	0.04
● “R R C C”	35.6	0.083
“R R R C”	42.4	0.026
“C R C R”	32.4	0.012
“R R C R”	36.3	0.026
期待値	25.8	

提案手法では，すべてのシナリオでの期待値が最小となる計画が選択されるため，シナリオを考慮しない図 2.13 と同じシナリオ(表中の赤丸のシナリオ)が選ばれた際には 35.6 [kg-CO_2]の排出量となる．

次に，提案手法によって計算された前日計画が，実際の運用においてどの程度ロバストなものであるかを当日運用シミュレーションによって検証した．シミュレーションにおいては，運用当日の実際の PV 出力(30 分平均値)と，図 2.24 に示す運用ルールに基づき当日運用を模擬した．

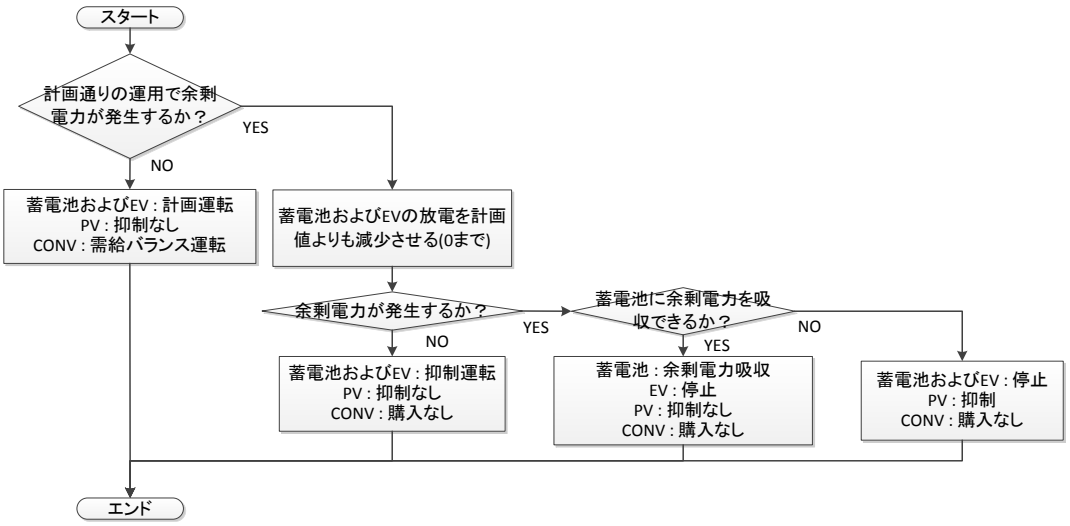

図 2.24 当日運用シミュレーションにおける MG の制御ルール

この運用ルールでは，基本的に前日に決定した充放電計画は変更せず，PV 出力も抑制しないというポリシーのもと，MG 内で余剰電力が発生しない場合には，需給のアンバランスを購入電力の変更によって調整する．次に，購入電力を 0 としても余剰電力が発生する場合には，蓄電池および EV の放電量を減らすことによって需給をバランスさせる．この場合でもなお余剰電力が発生する場合には，蓄電池は計画にない充電を行うことで見かけの負荷を増加させる．ただし EV に関しては，走行中で系統から切り離されている場合も多く，また，予定にない充電を行うことが機器の制約上困難であることから，計画にない充電は行わないこととする．さらに，蓄電池の容量制約に抵触し，これ以上充電を行えなくなった場合には蓄電池の運転を停止し PV 出力を抑制することで需給をバランスさせる．

図 2.25 に 6 月 26 日の当日の日射量データを用いた，シナリオを考慮しない場合の 30 分当日運用シミュレーションの結果を示す．また，図 2.26 に同様のデータを用いた，提案手法における 30 分シミュレーションの結果を示す．なお，この日は天気予報”RRCC”に反して”CCCC”の天候であった．

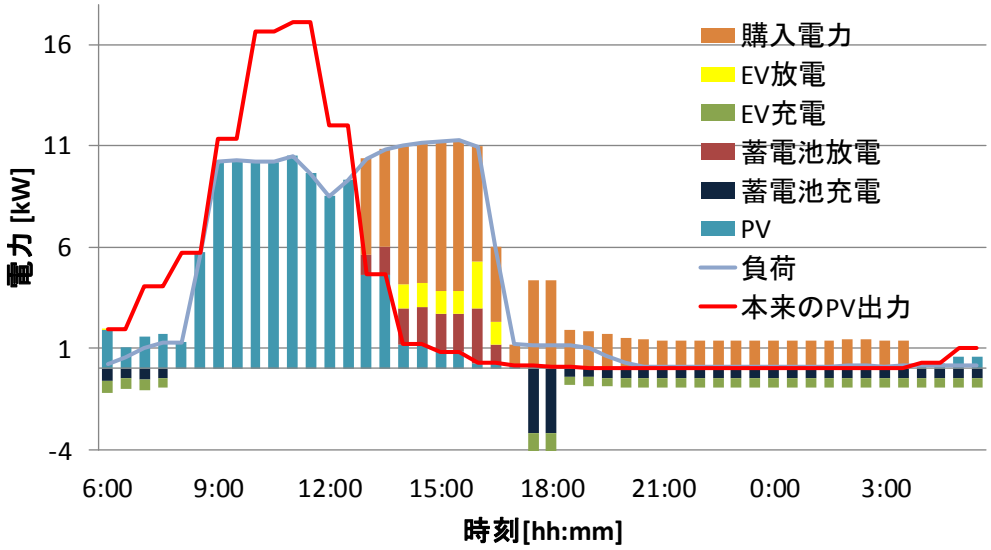

図 2.25 シナリオを考慮しない場合の当日運用

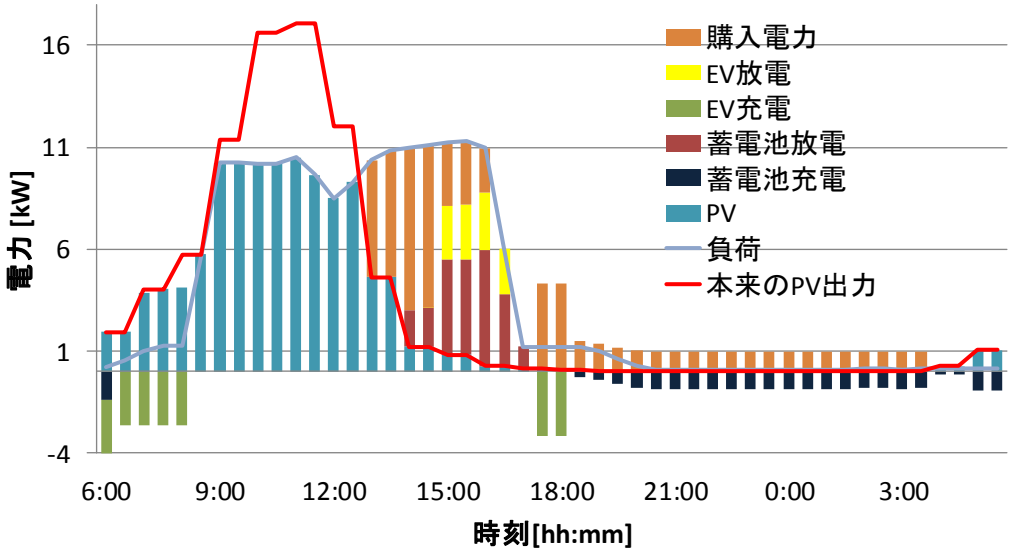

図 2.26 シナリオを考慮した場合の当日運用

この日は午前中の天気予報が外れた日であったため，9 時から 12 時までの時間帯の PV 出力が予測値よりも大きく，シナリオを考慮しない場合，本来はその時間帯に蓄電池が放電する計画であったが蓄電池は停止している．また，蓄電池はこの時間帯の放電に向けて充電を行っており，SOC が満充電となっているために PV の余剰電力は充電できず，PV の出力が一部抑制されてしまっていることがわかった．一方で，シナリオを考慮した提案手法の場合は，図 2.26 で問題となった 9 時から 12 時までの時間帯において，そもそも蓄電池の放電が計画されておらず，PV は抑制されているものの蓄電池および EV に関しては概ね計画通りの運転がなされていることが確認できた．また，蓄電池および EV の放電が 15 時付近に集中して計画されているため，放電機会を失っていないことも確認できた．さらに，わずかではあるが午前 6 時付近の時間帯で蓄電池の充

電が計画されていたため，シナリオを考慮しない場合と比較して PV を有効に利用できていることが確認できる．この日の CO_2 排出量は，シナリオを考慮しない場合では 27.2 [kg-CO_2]，提案手法では 20.1 [kg-CO_2]であり，シナリオを考慮することによって CO_2 排出量は 26.1 %削減できることが確認できた．

このように，天気予報が外れ PV 出力予測誤差が大きくなってしまうような日にはシナリオを考慮することの有効性が示唆されたが，誤差の大きさに対してどの程度有効かをより多くのデータで検証する必要がある．そこで，保持しているすべてのシミュレーション日に対して，図 2.27 のように横軸に PV 出力の 1 日積算値における予測誤差(天気予報のみを利用した場合の予測値と実際の PV 出力との差)と縦軸に CO_2 排出量削減量(シナリオを考慮することによって増減した分)を取ったものを算出した．

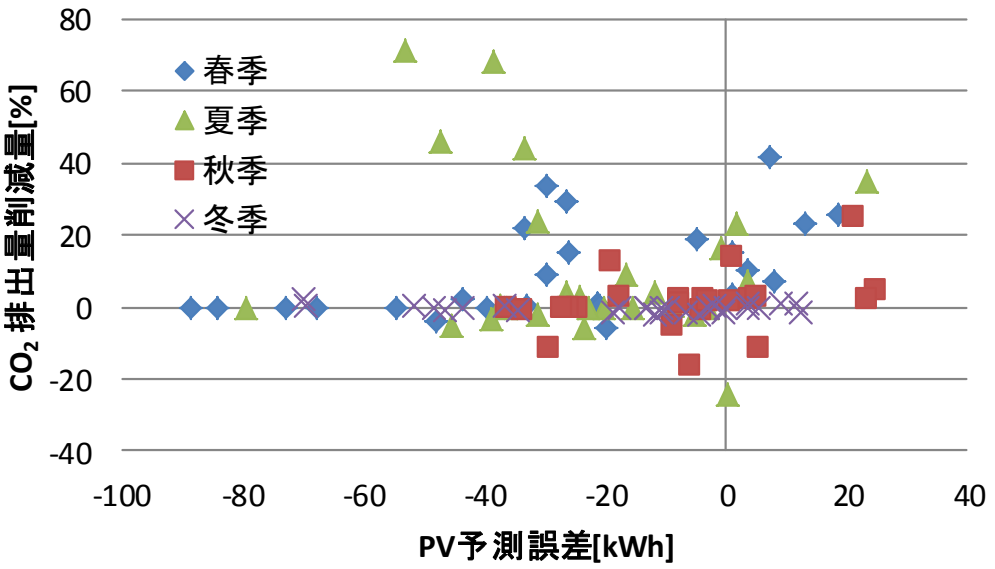

図 2.27 PV 予測誤差と CO_2 排出量削減量の関係

図 2.27 より，春季，夏季，秋季に関しては，提案手法によって CO_2 排出量が改善する日が多く見られた. 特に春季において CO_2 排出量削減効果が向上していることが確認でき, 平均では 7.77 %削減できることがわかった．しかしながら，1 日を通しての予測誤差の絶対値が 60 kWh を超えるほどに大きな場合には，提案手法の効果は薄いことがわかった．なお，このオフィスでの 1 日の平均負荷が約 90.3 kWh であるため，60 kWh の誤差とは平均負荷に対して 66.4%の値である．今回の試算条件では正の予測誤差は 60 kWh を越す日はなく, すべて負の方向の予測誤差であった．このような日では，天気予報がすべての時間帯で外れたような発生確率の非常に低い日であり，想定シナリオとして考慮されなかったことが原因である．また，冬季に関しては日射量が少ない日が多いために，提案手法による改善は小さいことが確認できた．試算における全データの平均では提案手法によって，2.31 %の CO_2 排出量削減ができることを確認した．

2.4.4　当日運用試算による前日計画の効果の検証

ここでは前項までに得られた前日計画機能の効果を当日運用機能と組合せた試算によって検証する．

まず，当日運用機能の概要について述べる．運用当日においては，基本的に前日計画に基づいて直流 MG 内の各機器を運用する．すなわち，蓄電池および EV は前日計画にしたがって充放電を行うとともに，実際の負荷や PV 出力に応じて，系統からの購入電力を直流 MG 内の需給バランスが維持されるように調整する．ただし，想定外の大きな PV 出力変動が発生すると，系統購入電力の調整だけでは対応できない可能性がある．また，前日計画の時間粒度は 30 分であり，30 分より短い周期の変動は前日計画では考慮されていない. 以上のことから, 当日運用においては，直流 MG 内の状態量をより短い時間粒度で（1 秒毎）監視し，直流バスの電圧が常に一定値となるように系統購入電力を逐一調整すると共に，その調整だけでは対応できない場合には蓄電池や EV の充放電量も前日計画値から適切に修正し運用を行う必要がある．以上のことから本論文では，直流 MG 内の需給バランスをどの機器に分担させるかに依存して直流 MG の運転モードを定義し，状況に応じて制御モードを図 2.28 のように適宜切り替える手法を提案する．以下では各制御モードの詳細を記述するが，直流 MG 内の需給バランスを担う機器を電圧源と呼び，それ以外の機器を電流源と呼ぶことにする．

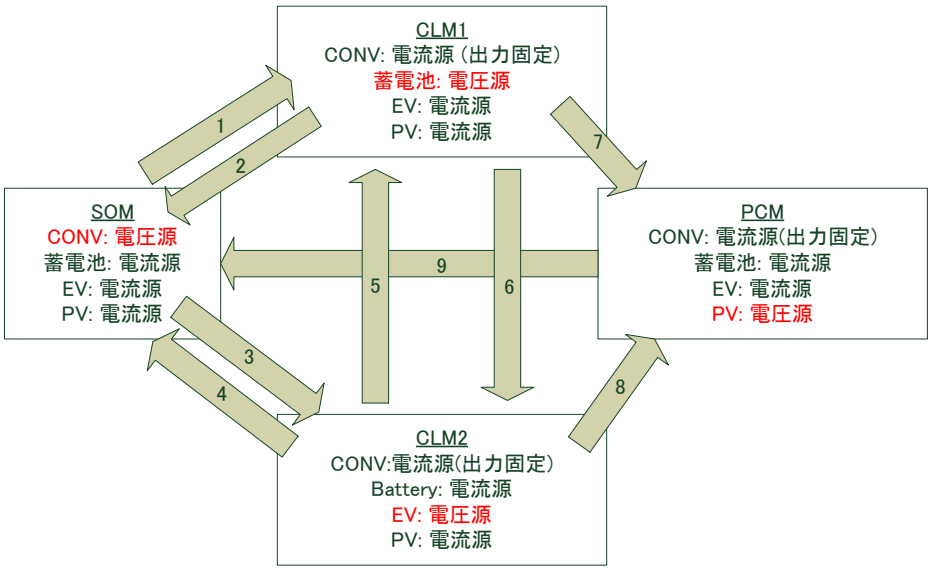

図 2.28 本論文で想定するモード遷移図

1)　計画運転モード　計画運転モード（Scheduled Operation Mode，以下 SOM）では，CONV を電圧源として選択し，蓄電池と EV は電流源として前日計画で策定された計画通りの充放電を行う．すなわち，計画運転モードにおいては各種予測誤差を CONV で対応させることを念頭に置いている．出力に対する時間遅れと線路抵抗を考慮すると，直流バス電圧を目標電圧 V_{DC_ref}（380 V）とするための CONV の出力電圧 $V_{conv}(t)$ は以下の制御則に基づいて決定される．

$$V_{conv}(t) = V_{DC\_ref} + R_{conv} \cdot I_{conv}(t - \Delta t) \qquad (2.30)$$

ここに，$I_{conv}(t-\Delta t)$：時刻 $t-\Delta t$ における CONV の出力電流［A］，R_{conv}：CONV の出力端から直流バスまでの給電線の線路抵抗［Ω］，Δt：サンプリングタイム［s］である．

各種予測誤差の影響が大きく，CONV の過負荷や逆潮流の可能性がある場合，すなわち

$$P_{conv}(t) \geq P_{conv}^{max} \tag{2.31}$$

$$P_{conv}(t) < 0 \tag{2.32}$$

の条件を満足した場合には，後述のコンバータロックモード 1 または 2 へ遷移する(それぞれ図 2.28 の遷移 1 および 3).

2) コンバータロックモード 1　コンバータロックモード 1（Converter Lock Mode 1，以下 CLM1）では，蓄電池を電圧源として運転する．この際，CONV の出力は最大値もしくは 0 で固定する（どちらに固定するかは，本モードへ遷移する直前の CONV の運転状態に依存する）．本モードにおける，CONV の出力電力 $P_{conv}(t)$と，蓄電池の出力電圧 $V_{batt}(t)$は，以下の制御則に基づいて決定される．

$$P_{conv}(t) = \begin{cases} P_{conv}^{max} & \text{出力上限に固定} \\ 0 & \text{出力下限に固定} \end{cases} \tag{2.33}$$

$$V_{batt}(t) = V_{DC\_ref} + R_{batt} \cdot I_{batt}(t-\Delta t) \tag{2.34}$$

ここに，R_{batt}：蓄電池から直流バスまでの給電線の線路抵抗［Ω］，$I_{batt}(t-\Delta t)$：時刻 $t-\Delta t$ における蓄電池の出力電流［A］である．

3) コンバータロックモード 2　コンバータロックモード 2（Converter Lock Mode 2，以下 CLM2）では，EV を電圧源として運転する．本モードでは，CLM1 と同様に，CONV の出力は固定する．本モードにおける EV の出力電圧 $V_{EV}(t)$は，以下の制御則に基づき決定される．

$$V_{EV}(t) = V_{DC\_ref} + R_{EV} \cdot I_{EV}(t-\Delta t) \tag{2.35}$$

ここに，R_{EV}：EV から直流バスまでの給電線の線路抵抗［Ω］，$I_{EV}(t-\Delta t)$：時刻 $t-\Delta t$ における EV の出力電流［A］である．

なお，CLM において電圧源となっている機器が満充電になった，もしくは定格で充電をしても余剰電力を吸収しきれなかった場合には，電流源動作している機器（CLM1 ならば EV，CLM2 ならば蓄電池）を電圧制御に切り替える（それぞれ図 2.28 の遷移 5 および 6）．しかし，どちらの機器も満充電になっている等の理由により余剰電力を吸収することができる機器がない場合には，PV 用 PCS によって PV の出力を抑制する PCS 制御モードに遷移する(それぞれ図 2.28 の遷移 7 および 8)．また，EMS の内部では仮に SOM に戻った場合の CONV 出力が計算されており，これが(2.31)式および(2.32)式の範囲内になった場合には再び SOM に戻る(図 2.28 の遷移 2 および 4).

4) PCS 制御モード　PCS 制御モード（PCS Control Mode，以下 PCM）では，PV 用 PCS を電圧源として運転する．すなわち本モードは，直流 MG 内の需給平衡を達成するために PV 出力を抑制することになる．本モードにおける PCS の出力電圧 $V_{PV}(t)$は，以下の制御則に基づき決定される．

$$V_{PV}(t) = V_{DC\_ref} + R_{PV} \cdot I_{PV}(t-\Delta t) \tag{2.36}$$

ここに，R_{PV}：PCS から直流バスまでの給電線の線路抵抗［Ω］，$I_{PV}(t-\Delta t)$：時刻 $t-\Delta t$ おける PCS の出力電流［A］である．

本モードにおいて PCS の出力電力が，その時間帯において PV が出力可能である最大電力を上回った場合は，PCS は負荷に対して十分な電力を供給することができない．その際は，直流 MG は供給力不足となり，直流バス電圧が 380 V から低下する．この状態になったとき，運転モードは PCM から SOM へ戻る(図 2.28 の遷移 9).

当日のデータを保持している 14 日間を対象として，モード遷移を考慮した運用シミュレーションを行った．なお，利用したデータは 2014 年 8 月に実証設備の直流 MG で実際に計測されたデータであり，前項の前日計画機能検証に用いたデータとは異なることに留意されたい．このような事情から，まず，検証に用いるデータの違いおよび機器制御方法の違いによって，前日計画検証と同様の結果が現れるかを検証する必要がある．図 2.29 に，シナリオの有無による各日の CO_2 排出量を示す．なお，この CO_2 排出量は計測データを元に算出した値である．図中の赤い点がある日においては，天気予報が外れた日であり，CO_2 排出量は提案手法の方がそれぞれ 18.9 %，10.0 %改善していることを確認した．このことから，前項で得られた 30 分単位での当日運用試算結果と，1 秒単位での当日運用試算結果とは同様の傾向を示すことが確認できた．

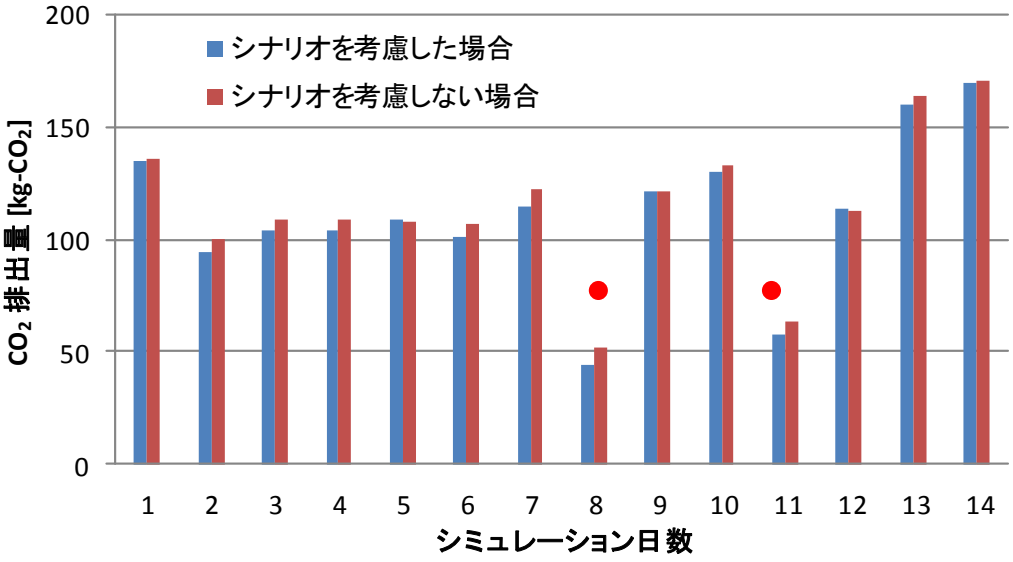

図 2.29 1 秒値シミュレーションにおける CO_2 排出量

図 2.30 に期間中の 1 日目の各機器のモード遷移を示す．なお，図 2.30 における縦軸の数字はそれぞれの機器がどのような状態にあるかを示しており，それぞれ，1=電圧源として，2=電流源(計画通り運転)，3=ロック(最大出力あるいは最小出力で固定)，4=停止(EV が直流 MG に接続されていない場合)に対応している．図 2.30 より，15 時付近まではコンバータが電圧制御を行っており，他の機器は計画通りの運用が行われていることがわかる．また，9 時から 15 時までは EV が直流 MG から切り離されて自動車として運用されているため，モード 4 に滞在している．15 時付近は PV の出力が大きいためにコンバータでの買電を中止し，蓄電池が電圧制御を行っている．それ以降は再びコンバータが電圧制御を行い，計画運転がなされている．このことから，当日の予測誤差に対応するために CONV だけが電圧源となるのではなく，他の電源

が必要に応じて電圧源を担う必要性が示唆された．図 2.31 にこの日の電圧の時間推移を示す．モード遷移が行われた各時点においても，直流バスの目標電圧である 380 V を維持できていることが確認できた．

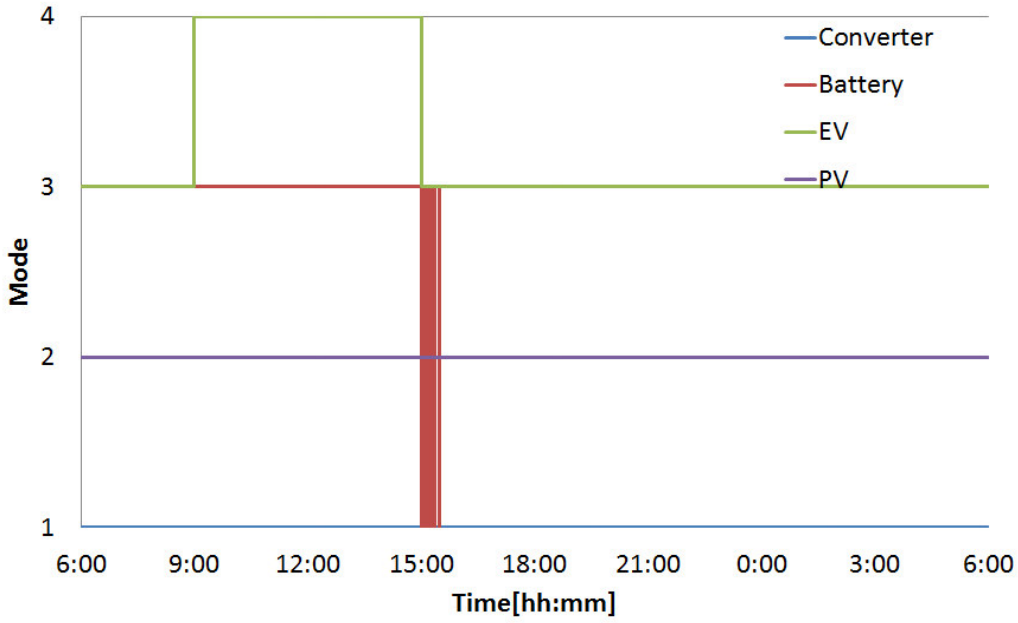

図 2.30 1 秒値シミュレーションにおける各設備のモード

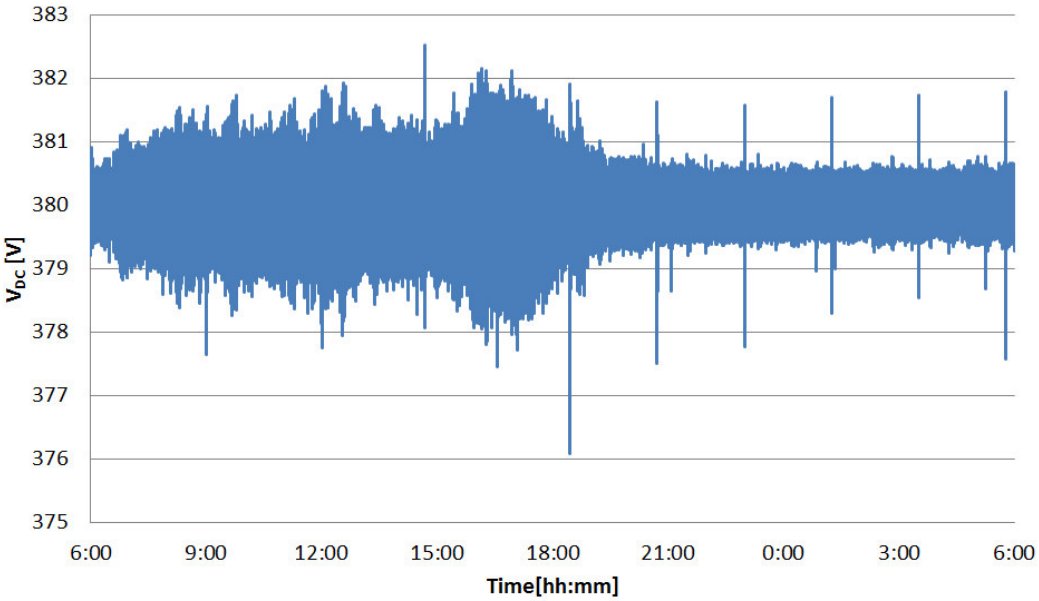

図 2.31 1 秒値シミュレーションにおける直流バス電圧

以上の結果から，PV 出力や負荷の変動が大きな時間帯でも電圧は維持されており，本論文で提案したアルゴリズムによる前日計画は，当日運用で対応できる誤差範囲での計画を策定できることが確認できた．

2.5　帯広市マイクログリッドにおけるエネルギーマネジメントシステムの実装

本論文で開発した EMS の各機能は，実際に帯広市 MG に実装され，その動作確認が行われた．実装された EMS のインターフェースを図 2.32 に示す．

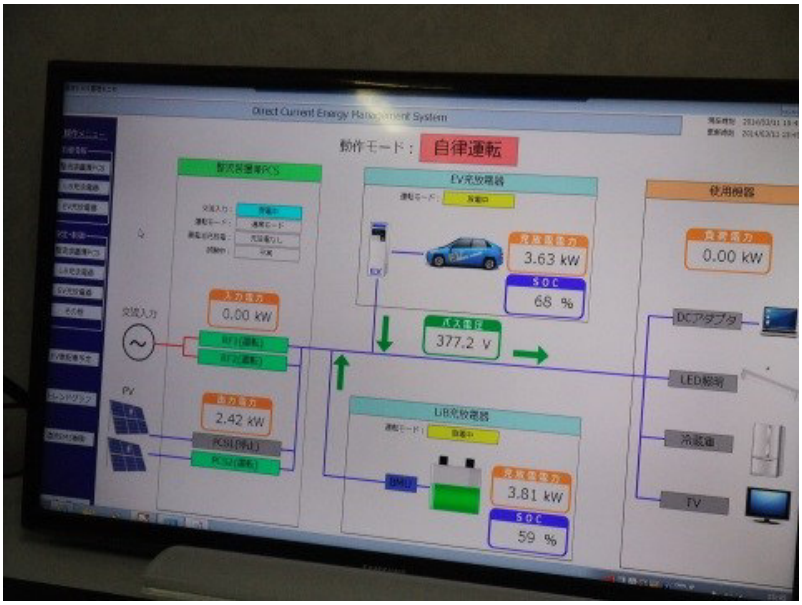

図 2.32　帯広市 MG に実装された EMS のインターフェース

実装された EMS においては，各設備の出力が常に監視され画面に出力されていた．また，本論文で開発した前日計画機能においては，天気予報データ，SOC データといった情報が予測開始時刻である午前 5 時にダウンロードされ，前日計画が行われた．さらに，当日運用機能についても実装され，電圧を維持するように各設備の出力が決定された．さらに短い時間帯幅におけるリアルタイム制御に関しては，各機器の電圧を当日運用において得られた電圧を適用し，電圧の高低から自動的に出力電流ならびに出力電力が決定されるような制御がなされた．ただし，当初実装を予定していた，当日運用における計画修正機能等は実装されなかった．

2.6　まとめ

本章では，MG の EMS において必要となる機能のうち，特に前日計画機能を開発し，このアルゴリズムについて述べた．本論文で提案する前日計画アルゴリズムは，MG から排出される CO_2 を最小化するために，区分線形近似を用いた線形計画法によって，MG の各設備の 30 分毎の出力

計画を行うものであった．より精度のよい計画を得るためには，翌日に発生しうる予測誤差に対しても対策を講じることが必要であるが，本論文では特に翌日の予測誤差の影響が大きいPV出力に対して，ある決まった出力カーブを得るのではなく，いくつかのカーブが確率によって発生するシナリオとして扱った．このことにより，得られた計画は，予測誤差がない場合には1本のカーブに集中した場合よりも劣るものの，予測誤差がある場合でもMGに期待されるパフォーマンスを低減させることなくMGを運用することが期待できる．期待される効果を検証するため，本論文では環境省のプロジェクトの一環として構築された帯広市直流MG実証設備で得られたデータを基に，EMSによるMG運用シミュレーションを行った．特に予測誤差の大きな日においては，単一のPV出力予測を利用した場合と比較して，シナリオを用いた本論文の手法の方が26.1%だけCO_2排出量を削減できることを確認した．また，年間を模擬した試算では，全データの平均で2.31%だけCO_2排出量削減効果が向上することを確認した．さらに，プロジェクト全体では本論文で開発した前日計画機能と当日運用機能を組み合わせた当日の1秒値試算を行った．1秒値試算は，本論文で得られた手法と同様の傾向を示したため，提案手法は短い時間帯幅の運用においても有効であることが示唆された．

第 3 章　マイクログリッドにおける最適設備容量設計

3.1　概要

第 2 章では, 実際に構築された MG を対象とした最適運用計画手法を提案した. しかしながら, 負荷データに対して PV 容量，蓄電池容量および EV 容量があらかじめ決められた試算条件においては，計画手法の改善だけでエネルギーの地産地消を実現することは困難なデータが散見された．すなわち，オフィスにおける休日のように負荷が極端に小さく，快晴の日で PV 出力が大きな場合には，ほぼすべての出力を蓄電池に吸収させることとなるが，帯広市 MG では蓄電池が最大充電電力 2 時間分の SOC しか用意されておらず, ほぼすべての PV 出力を抑制するような運用となってしまうことがわかった.

そこで, 本章では MG の最適運用を決定するに際して, その容量も含めた最適化計算を行った. これまでの試算では CO_2 排出量の最小化を目的としていたが，MG 導入に要するコストと MG を運用するためのコストの合計コストが最小となるような設備容量･運用計画を決定した. これは，MG を新たに導入する際に，そのユーザーが求めるのは環境性よりも低コストであろうという想定のためであった. しかしながら, 現在の PV, 蓄電池および EV は導入コストが依然として高く，運用コストを削減できたとしても導入コストの分を解消できない可能性が考えられた．そこで，本論文では，CO_2 排出量削減とコスト削減の折衷案として，「CO_2 排出量削減目標を達成しつつ，その制約下でコスト最小となる設備容量」を決定した．具体的には，我が国の政府が掲げている CO_2 排出量削減目標である 2030 年度までに 26%削減を達成することを考えた.

3.2　混合整数計画法を用いたマイクログリッドの最適容量設計

実際の MG の容量設計をするに当たっては，ユーザーは PV システムや蓄電池システム，EV システムを何もない状態から設計するのではなく，メーカーから発表・発売されているものを組み合わせて MG を構築することを考えるのが妥当であろう．したがって，本論文では各設備の容量を実数変数として扱うのではなく，ある定格を持ったユニットを何台導入するかという整数変数として扱う．したがって，最適容量を求めるためには混合整数計画法を用いる必要がある.

本論文では，どの種類の設備を導入し，どのような方式でそれらを設置するのかというような MG の骨格はあらかじめ与えられるものとして扱い，図 3.1 のような PV，蓄電池および EV から構成されるものを検討した．また，帯広市 MG にならい，負荷および給電方式は直流とした．ただし，EV についてはガソリン車を導入することで自動車としての性能を発揮することも考えられるため，図 3.2 のように PV および蓄電池だけを導入する場合についても試算を行った.

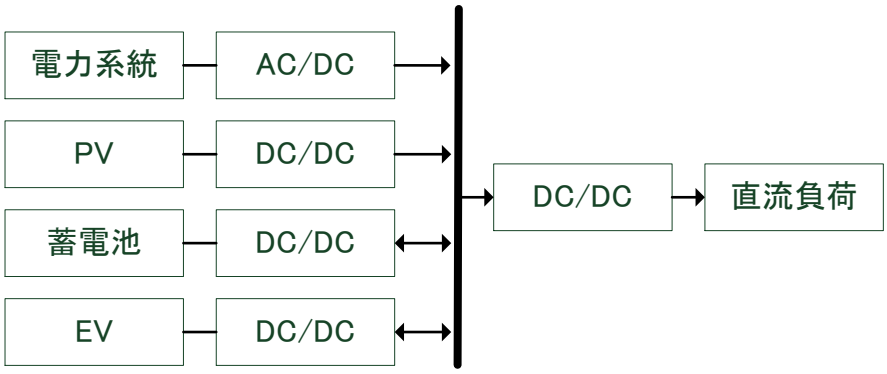

図 3.1 想定する直流 MG の設備構成(EV あり)

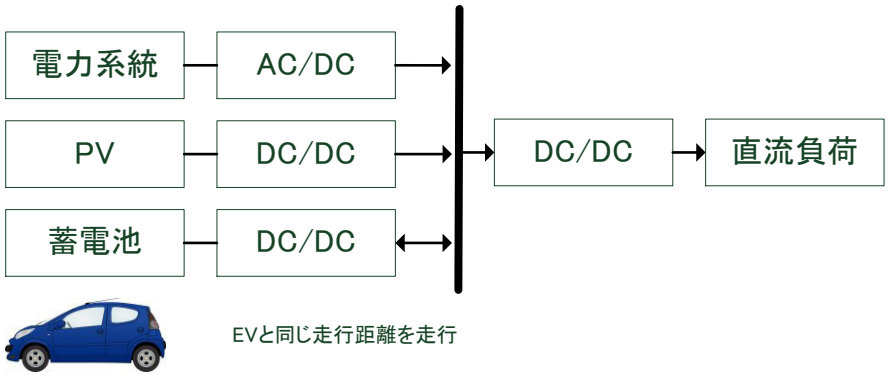

図 3.2 想定する直流 MG の設備構成(EV なし)

3.2.1 定式化

先述のとおり，本論文における最適化の目的関数はコストの最小化であり，以下のように記述できる．

$$min\ f = C_{PV} \cdot Cap_{PV} + C_{batt} \cdot Cap_{batt} + C_{rect} \cdot Cap_{rect} + OC \quad (3.1)$$

ここに，C_{PV}：単位容量あたりの PV 導入コスト[JPY/kW]，Cap_{PV}：PV の導入容量[kW]，C_{batt}：単位容量あたりの蓄電池導入コスト[JPY/kW]，Cap_{batt}：蓄電池の導入容量[kW]，C_{rect}：単位容量あたりの系統連系用コンバータ導入コスト[JPY/kW]，Cap_{rect}：系統連系用コンバータの導入容量[kW]，OC：マイクログリッドの運用コスト[JPY]である．各設備の導入コストに関しては，対象とするシミュレーション期間の幅も考慮して設定する必要がある．また，EV の導入に関しては導入するかしないかの 2 つしか解を取らないことから最適化問題そのものの決定変数としては含めず，有無の 2 回を計算する．

先述の通り，PV および蓄電池はある定格を持ったユニットを何台導入するかによって容量を決定するので，それぞれの容量は以下のように記述できる．

$$Cap_{PV} = Cap_{PV}^{unit} \cdot n_{PV} \tag{3.2}$$

$$Cap_{batt} = Cap_{batt}^{unit} \cdot n_{batt} \tag{3.3}$$

ここに，Cap_{PV}^{unit}：PV システムの 1 ユニットあたりの定格[kW:]，Cap_{batt}^{unit}：蓄電池システム 1 ユニットあたりの定格[kW]，n_{PV}：PV システムの導入数，n_{batt}：蓄電池システムの導入数である．系統との連系コンバータに関しても，本来ならば導入容量をユニット数で扱う必要があるが，本論文で想定する大容量のものは一般に普及が進んでおらず，ユニットの定格データを入手することが困難であった．そのため，本論文ではこれに関しては実数として扱う．

MG の運用コストに関しては，今回の想定では系統から購入する電力のみが関連する．したがって，以下のように記述できる．

$$OC = \sum_{t} TOU(t) \cdot P_{grid}(t) \cdot \Delta T \tag{3.4}$$

ここに t は時間帯，$TOU(t)$ は時間帯別電力料金[JPY/kWh]，$P_{grid}(t)$ は t における購入電力[kW]，ΔT は時間帯幅[h]である．

本論文では，制約条件として以下を考える．

a) 需給平衡制約

すべての時間帯において，MG 内における電力の需要と供給が一致しなければならない．

$$\begin{aligned} &\eta_{grid} \cdot P_{grid}(t) + \eta_{PV} \cdot Cap_{PV} \cdot P_{PV}(t) \\ &- \frac{P_{batt\_C}(t)}{\eta_{batt}} + \eta_{batt} \cdot P_{batt\_D}(t) \\ &- \frac{P_{EV\_C}(t)}{\eta_{EV}} + \eta_{EV} \cdot P_{EV\_D}(t) \\ &= \frac{P_L(t)}{\eta_L} \end{aligned} \tag{3.5}$$

ここに，η_{grid}：連系コンバータの効率，η_{PV}：PV システムにおける PCS の変換効率，$P_{PV}(t)$：時刻 t における PV 出力[kW]，η_{batt}：蓄電池システムにおける変換効率，$P_{batt\_C}(t)$：時刻 t における蓄電池の充電電力[kW]，$P_{batt\_D}(t)$：時刻 t における蓄電池の放電電力[kW]，η_{EV}：EV システムにおける変換効率，$P_{EV\_C}(t)$:時刻 t における EV の充電電力[kW]，$P_{EV\_D}(t)$：時刻 t における EV の放電電力[kW]，$P_L(t)$：時刻 t における MG の負荷[kW]，$\eta_L(t)$：負荷用コンバータの変換効率である．

b) CO_2 排出量削減目標達成制約

先述の通り，日本政府が公表している 26%の CO_2 排出量削減目標を，MG 導入によって期間中に達成する．

$$\frac{CO_{2\_nonMG} - CO_{2\_MG}}{CO_{2\_nonMG}} \geq 0.26 \tag{3.6}$$

ここに，CO_{2_nonMG} および CO_{2_MG} はそれぞれ MG 導入前の CO_2 排出量[kg-CO_2]と MG 導入後の CO_2 排出量[kg-CO_2]である．本論文では MG 導入前の住宅においては，系統から購入する電力によってその需要を全て賄うという想定のもと，CO_{2_nonMG} を以下の式で定義する．

$$CO_{2\_nonMG} = \sum_t \left[E_{UG}(t) \cdot \frac{P_L(t)}{\eta_{grid}} \cdot \Delta T \right] \quad (3.7)$$

ここに，$E_{UG}(t)$は系統からの購入電力における CO_2 排出原単位[kg-CO_2/kWh]である．

また，MG 導入後も CO_2 排出量に関連するのは系統からの購入電力だけあるため，CO_{2_MG} を以下の式で定義する．

$$CO_{2\_MG} = \sum_t E_{UG}(t) \cdot P_{grid}(t) \cdot \Delta T \quad (3.8)$$

c) 蓄電池および EV の SOC の時間推移

$$SOC_{batt}(t+1) = SOC_{batt}(t) + \left\{ \mu_{batt} \cdot P_{batt\_C}(t) - \frac{P_{batt\_D}(t)}{\mu_{batt}} \right\} \cdot \Delta T \quad (3.9)$$

$$SOC_{EV}(t+1) = SOC_{EV}(t) + \left\{ \mu_{EV} \cdot P_{EVt\_C}(t) - \frac{P_{EV\_D}(t)}{\mu_{EV}} \right\} \cdot \Delta T \quad (3.10)$$

ここに，$SOC_{batt}(t)$は時刻 t における蓄電池 SOC[kWh]，μ_{batt} は蓄電池における充放電効率，$SOC_{EV}(t)$は時刻 t における EVSOC[kWh]，μ_{EV} は EV における充放電効率である．基本的には第 2 章で提案したものと同様である．

d) 各設備の容量制約

$$0 \le P_{grid}(t) \le Cap_{rect} \quad (3.11)$$

$$0 \le P_{batt\_C}(t) \le Cap_{batt} \quad (3.12)$$

$$0 \le P_{batt\_D}(t) \le Cap_{batt} \quad (3.13)$$

$$0 \le P_{EV\_C}(t) \le P_{EV}^{max} \quad (3.14)$$

$$0 \le P_{EV\_D}(t) \le P_{EV}^{max} \quad (3.15)$$

第 2 章で示した帯広市 MG と同様に，本章における MG でも系統への逆潮流は禁止する．すなわち，購入電力は負値を取ることができない．

e) 蓄電池の同時充放電禁止制約

第 2 章の定式化においては，線路抵抗損失等の関係から蓄電池や EV が同じ時間帯に充電と放電行う解は得られなかった．しかしながら，本章における定式化では制約を設けることによって同時充放電を禁止する必要がある．すなわち，次式のような制約を設ける．

$$0 \le P_{batt\_C}(t) \le 100 \cdot Cap_{batt}^{unit} \cdot \{1 - u_{batt}(t)\} \quad (3.16)$$

$$0 \le P_{batt\_D}(t) \le 100 \cdot Cap_{batt}^{unit} \cdot u_{batt}(t) \quad (3.17)$$

$$0 \le P_{EV\_C}(t) \le P_{EV}^{max} \cdot \{1 - u_{EV}(t)\} \quad (3.18)$$

$$0 \le P_{EV\_D}(t) \le 100 \cdot Cap_{EV}^{max} \cdot u_{EV}(t) \quad (3.19)$$

ここに$u_{batt}(t)$：蓄電池の充放電方向を定義するバイナリ変数(0：充電，1：放電)，$u_{EV}(t)$：EV の充放電方向を定義するバイナリ変数(0：充電，1：放電)である．EV においては，$u_{EV}(t)$の値によって（3.14）式か（3.15）式のどちらかの制約だけが有効となる．蓄電池に関してもこのような定義を行い最大充放電電力の制約に帰着したいが，蓄電池の場合は Cap_{batt} が決定変数によって変化するため，EV と同様に記述してしまうと変数同士の乗算項が出てきてしまい，線

形計画問題として扱うことができない．そこで，本論文では実際的な導入台数の限界よりも十分に大きい 100 台という数値を与えた．このことにより，例えば $u_{batt}(t)$ が 0 の場合は蓄電池は充電となり，$Cap_{batt} < 100 \cdot Cap_{batt}^{unit}$ の関係から（3. 12）式だけが有効となる．

f) EV およびガソリン車の走行に関する制約

EV の運用に関しては，第 2 章における制約と同様のものを考慮する．ガソリン車を考慮する場合には，走行に用いたガソリンに関する CO_2 排出量も MG から排出したものとして計上する必要がある．すなわち，EV による走行距離 D [km]を用いて以下のように定義する．

$$CO_{2\_car} = E_{car} \cdot D \qquad (3.20)$$

ここに CO_{2_car} はガソリン車の走行に伴う CO_2 排出量[kg-CO_2]，E_{car} ：ガソリン車の CO_2 排出係数[kg-CO2/km]である．これによって計算された CO_{2_car} は，（3. 8）式に加算されるため，ガソリン車を導入した MG においては EV 導入の MG よりもさらに多くの CO_2 排出量削減を行う必要がある．また，コストに対してもガソリン車の燃費 η_{gas} [JPY/km]から以下のように定義されるガソリン料金 C_{gas} [JPY]を加算する必要がある．

$$C_{gas} = \eta_{gas} \cdot D \qquad (3.21)$$

3.2.2　電気自動車を導入するマイクログリッドにおける試算

ここでは NEDO が群馬県太田市において実施した実証研究において計測された PV 出力および負荷のデータを用いて MG におけるコスト試算を行った．オリジナルのデータには PV システムを設置した 535 軒の住宅需要家の PV および負荷それぞれの有効電力，無効電力の 1 秒値データが 1 年分含まれており，その有効電力平均値を住宅 1 軒のデータとして扱った．また，試算における時間帯幅は 30 分とし，1 秒値データの 30 分平均値を 1 時間帯のデータとして扱った．シミュレーション期間は 1 ヶ月とし，春季(4 月)，夏季(7 月)，秋季(10 月)，冬季(1 月)の各 1 ヶ月分のデータを用いた．夏季のある 1 日，ある 1 軒の住宅における PV および負荷，ならびにそれぞれの 30 分平均値を図 3.3 に示す．

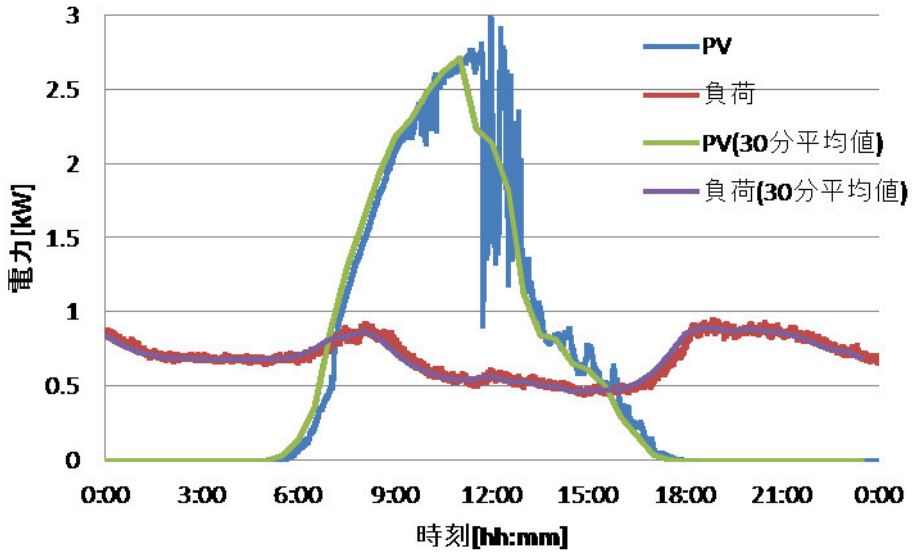

図 3.3 使用した太田市データの一例

第 2 章において試算の対象としたオフィス用負荷とは異なり，一日を通して負荷はほぼ一定であることがわかる．

次に，試算に用いた各パラメータを表 3.1 に示す．なお，特に文献の記載のないものは第 2 章における帯広市 MG と同じデータを利用した．

表 3.1 試算に用いたパラメータ

変数	意味	値
$TOU(t)$[28]	時間帯別電力料金	28.08 (AM 7:00~PM 11:00) [JPY/kWh]
		14.13 (PM11:00~AM6:00)[JPY/kWh]
C_{PV} [29]	1日におけるPVシステム1ユニットあたりのコスト	28.4 [JPY/kW]
C_{batt} [30]	日における蓄電池システム1ユニットあたりのコス	228.9 [JPY/kW]
C_{rect} [29]	1日における系統連系用コンバータの導入コスト	13.7 [JPY/kW]
C_{EV}[25]	1日におけるEVシステムの導入コスト	583 [JPY]
Cap_{PV}^{unit}	PVシステム1ユニットの定格	0.25 [kW]
Cap_{batt}^{unit}	蓄電池システム1ユニットの定格	3 [kW] (6.6 [kWh])
$Cap_{EV}^{\max}$	EVシステムの最大充放電電力	3 [kW] (12.8 [kWh])
$E_{UG}(t)$ [26][27]	購入電力に伴うCO_2排出量	0.681 (AM 6:00~PM 6:00) [kg-CO_2/kWh]
		0.485 (PM 6:00~AM 6:00) [kg-CO_2/kWh]
η_{PV}	PVシステムにおけるPCSの変換効率	0.965
η_{batt}	蓄電池システムの変換効率	0.94
η_{grid}	系統連系コンバータの変換効率	0.95
η_{EV}	EVシステムの変換効率	0.94
η_{L}	直流負荷用変換器の変換効率	0.9
μ_{batt}	蓄電池の充放電効率	0.95
μ_{EV}	EVの充放電効率	0.95
η_{run}	EVの燃費	0.11[kWh/km]
ΔT	試算における時間帯幅	0.5 [h]

なお，市販されている PV，蓄電池および EV と言った設備は交流用のものであるため，内部には DC/DC コンバータの他に AC/DC コンバータを内蔵している．そこで本論文では，住宅 PV 用 PCS が同容量の DC/DC コンバータと AC/DC コンバータの組み合せであることを考慮し，市販されている PCS 単体の価格の半額を各設備のコストから減ずることで，DC/DC コンバータのみを搭載した装置のコストとして扱った．また，系統連系コンバータの価格についても，この価格を利用した．

EV を 1 台導入した MG における試算結果を示す. 各季の CO_2 排出量および各種コストを図 3.4 および図 3.5 に示す.

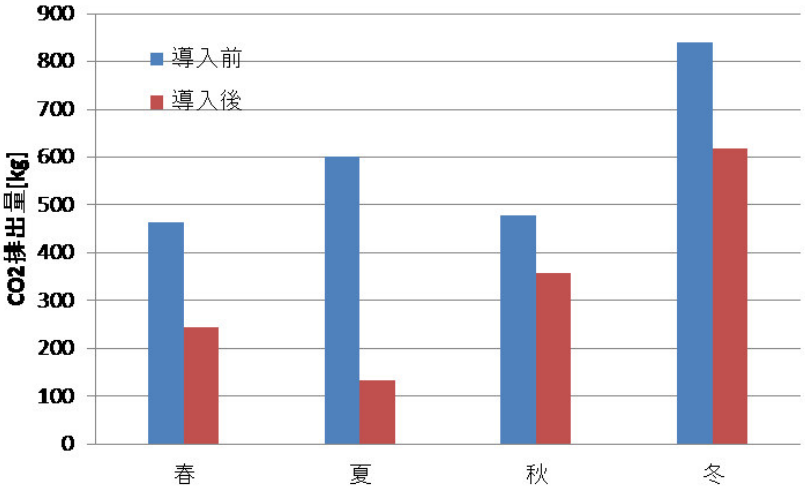

図 3.4 EV あり試算における各季の CO_2 排出量

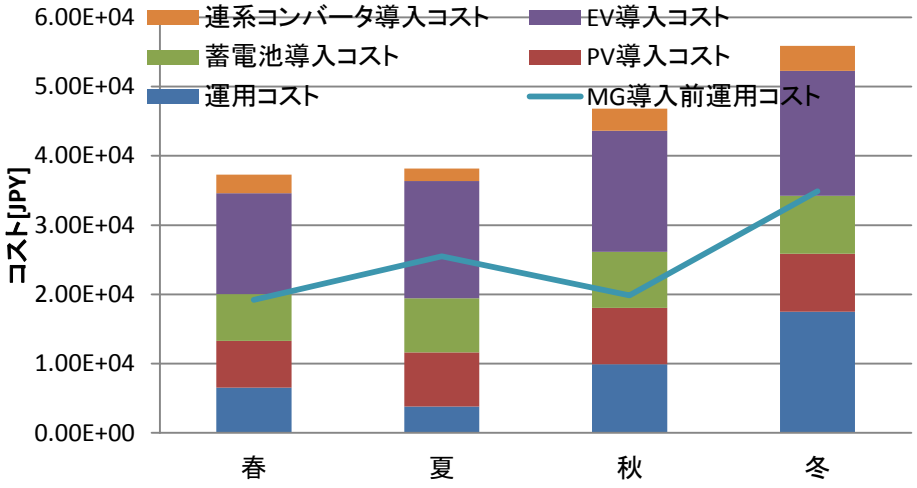

図 3.5 EV あり試算における各季のコスト

図 3.4 より，導入前の CO_2 は冬に最も多いことがわかる．試算の結果，どの季節においても目標量の 26%削減を達成できていることが確認できた．ただし，秋以外の結果では CO_2 排出量は目標を超えて大きく削減されていることが確認できた．図 3.5 よりどの季節でも MG 導入前と比較して運用コストを大幅に削減できていることが確認できた．しかしながら，設備の導入コストが全コスト中の 50%以上を占めており，合計コストでは導入後の方が高くなることが確認された．表 3.2 に試算における設備導入量を示す．

表 3.2 EV あり試算における各季の設備導入量

	PV[台]	蓄電池[台]	連系コンバータ [kW]
春	9	1	7.829
夏	12	1	4.570
秋	9	1	7.839
冬	12	1	8.465

どの季節でも蓄電池が 1 台導入されることが確認できた．これは，この試算では逆潮流ができないという制約がある上，EV が日中は走行して MG に接続されていないことが原因であると考えられる．PV の導入量に関しては，季節性があることが確認できた．表 3.3 に各季における PV 出力の合計および負荷の合計を示す．

表 3.3 各季の PV 出力合計および負荷合計

	PV出力合計[kWh]	負荷合計[kWh]
春	2.07E+02	4.11E+02
夏	3.24E+02	5.24E+02
秋	1.67E+02	4.20E+02
冬	2.63E+02	7.43E+02

表 3.3 より，PV 導入量に応じて PV 出力合計が大きく，また，同じ導入量となったものを比較すると，春と秋では春が，夏と冬では夏の方が大きいことがわかった．これは日射量に季節性があるためであると考えられる．負荷合計について比較すると，PV 導入量が同じであった春と秋では負荷合計がほぼ同じものとなった．夏と冬とを比較すると冬の方が大きいことがわかった．また，どちらも春や秋より大きいことがわかった．このことから，PV の導入量は日射量および PV 出力の多寡よりも負荷の大きさによる影響が大きいことがわかった．すなわち，余剰電力を一旦蓄電池に充電させるのではなく，負荷で直接 PV 出力を消費できるかが大きく影響していることが重要であると考えられる．連系コンバータの容量に関しては，春，秋における PV 導入量と夏における PV 導入量から， PV の導入量と概ね比例して少なくなることが示唆された．しかしながら，冬の試算においては PV が 12 枚導入されているのにも関わらず，春や秋の試算結果よりも多くのコンバータが導入されることがわかった. これは, 冬の負荷は四季の中で最も大きく，負荷の多くの部分を購入電力で賄う必要があるためであると考えられる．

図 3.6 から図 3.9 に各季における概ね晴れの天候であったと考えられる日の運用結果を示す. すべての結果において，PV 出力が負荷を上回る時間帯においては蓄電池に充電がなされ，電力料金および CO_2 排出原単位の高い時間帯に放電がなされていることがわかった．また，EV に関しては PV 出力の大きな時間帯に MG から切り離されているために，充電がなされないことがわかった．しかしながら，昼と夜の電力料金および CO_2 排出原単位の差を利用して，蓄電池同様に高い時間帯で放電，低い時間帯で充電がなされるように運用されていることがわかった．蓄電池および EV は，すべての時間帯で平均的に充電がなされるのではなく，ある 1 時間帯に集中して充電されることが多いことがわかった．コンバータ導入量に関しては平均的な充電の方がコストが安くなることが考えられるが，そのような結果にならなかった理由について考察した．理由としては，コンバータのコストは他の要素に対して安く設定されているため，目的関数に対する影響が小さいことと，ソルバーが解の変化が小さくなった際に計算を打ち切ることで計算時間の短縮をすることが影響し，最適解に至る前に計算が打ち切られたことが原因であると考えられる．計算機の制約上，最適解に至るのに十分な計算時間は確保できないことから，本論文では得られた準最適解を最適解として扱い掲載する．

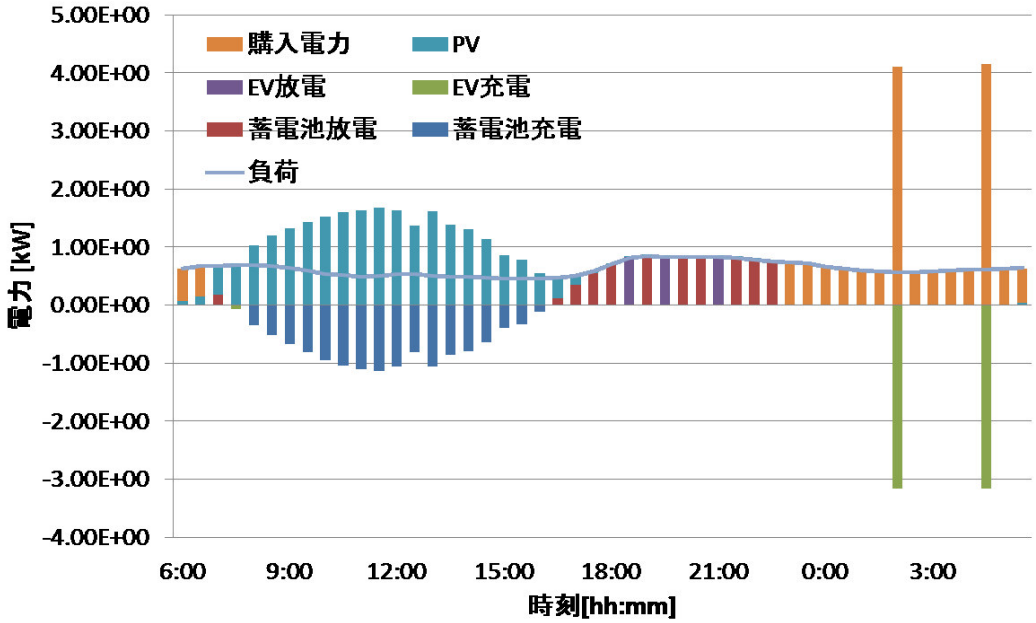

図 3.6 EV あり試算における運用例(春)

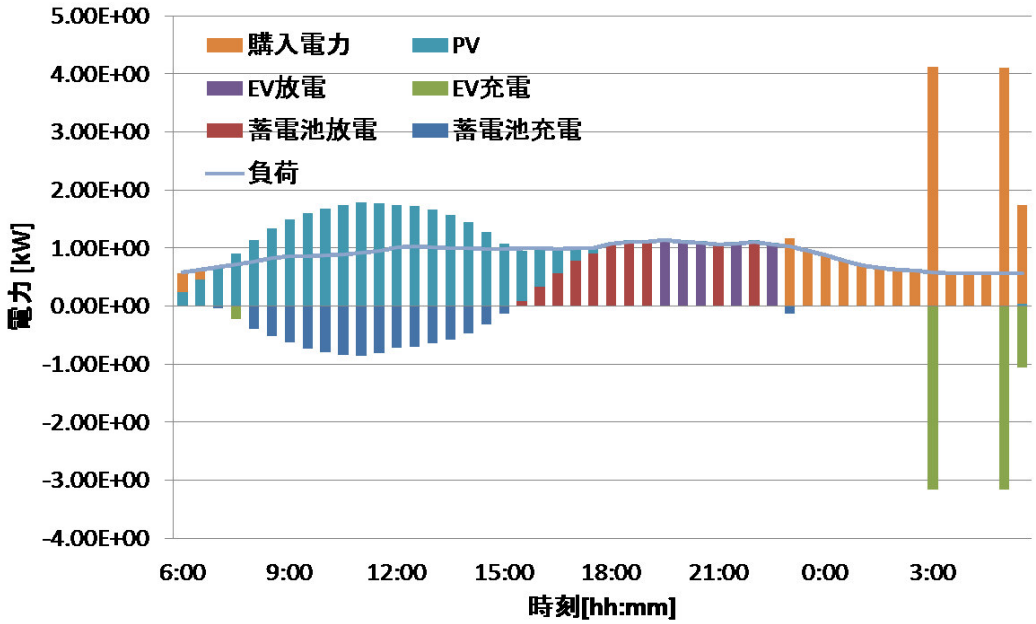

図 3.7 EV あり試算における運用例(夏)

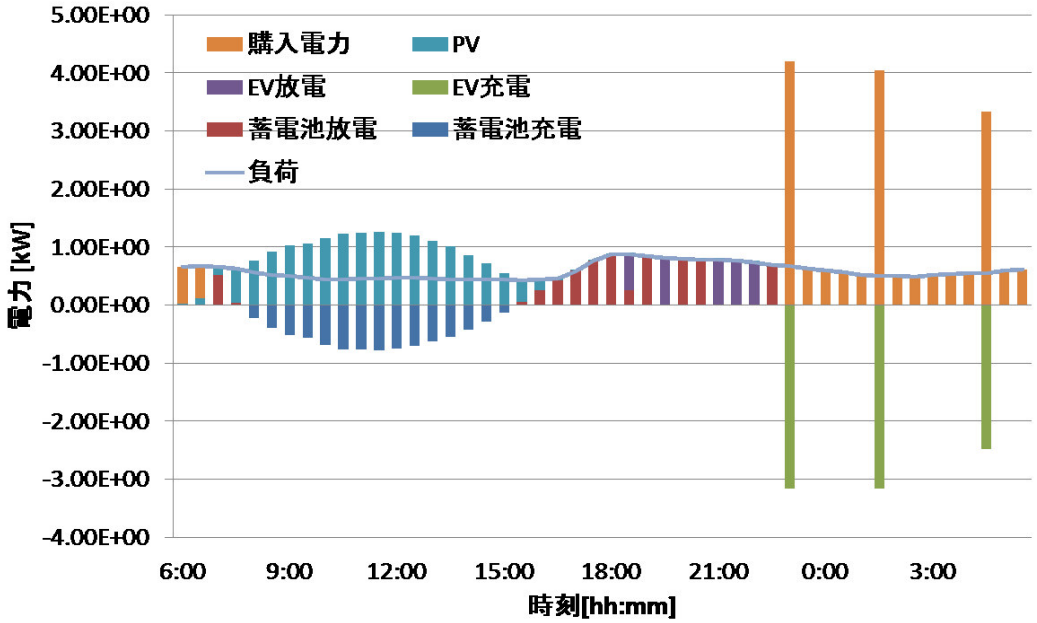

図 3.8 EV あり試算における運用例(秋)

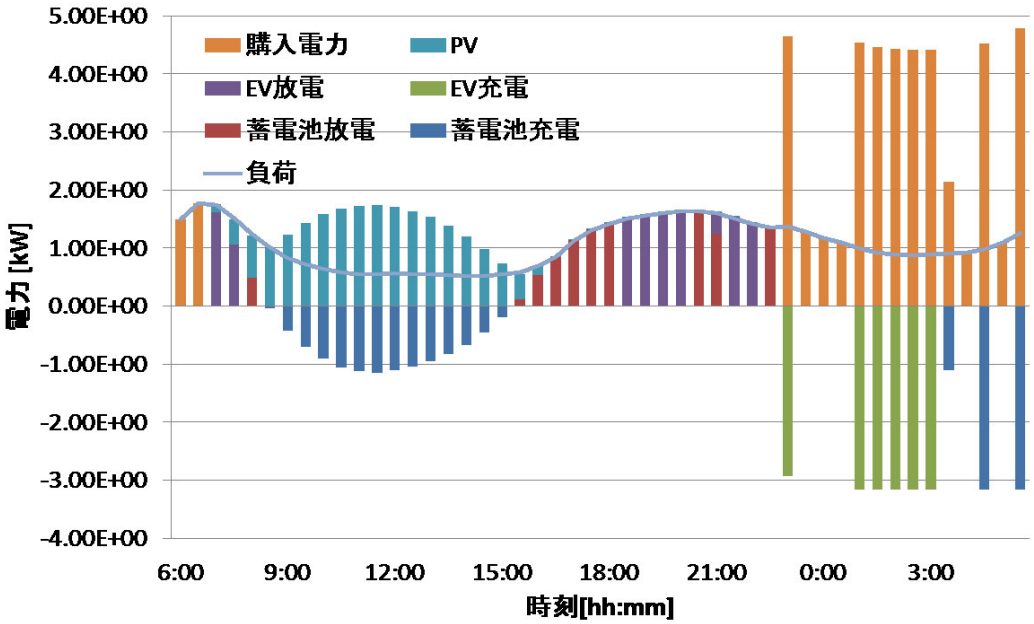

図 3.9 EV あり試算における運用例(冬)

3.2.3　ガソリン車を導入するマイクログリッドにおける試算

前項と同様の試算を EV ではなくガソリン車を導入する MG においても行った．表 3.4 は試算に用いたパラメータである．なお，ここに記載していないパラメータに関しては前項試算と同様のものを使用した．

表 3.4 EV あり試算における各季の設備導入量

変数	意味	値
C_{car}	1日におけるガソリン車の導入コスト	323.3 [JPY]
η_{gas}	ガソリン車の燃費	4.990 [JPY/km]
E_{car}	ガソリン車のCO_2排出係数	0.085 [kg-CO_2/km]

ガソリン車に関しては，三菱自動車のミラージュ[31]という車種を導入するものとして試算した．本体価格は 1.180×10^6 [JPY]，燃費は 27.2 [km/l]である．この数値から，耐用年数を 10 年として 1 日あたりのガソリン車導入コストを算出した．また，経産省・環境省の省令[32]および札幌市消費者センターの調査結果[33]から，ガソリン 1 l あたりの CO_2 排出量を 2.322 [kg-CO_2/l]，ガソリン 1 l あたりの価格を 133 [JPY/l]として計算を行った．

ガソリン車を 1 台導入した MG における試算結果を示す．各季の CO_2 排出量および各種コストを図 3.10 および図 3.11 に示す．

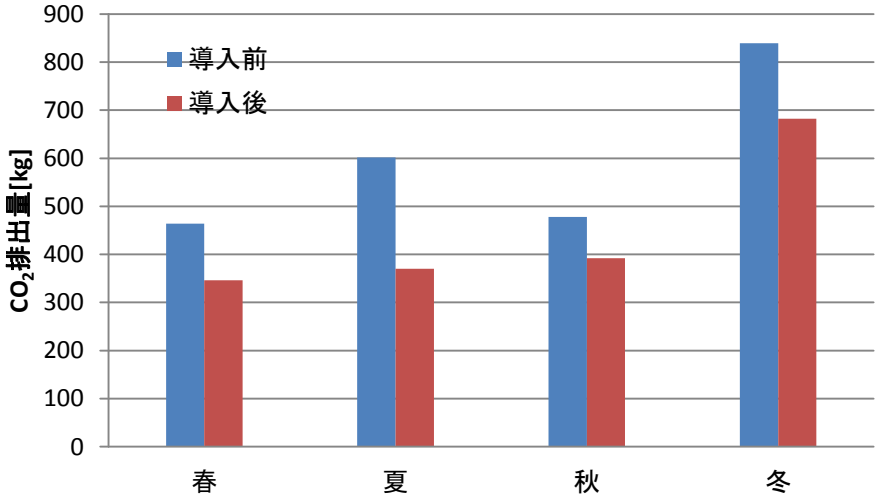

図 3.10 ガソリン車あり試算における各季の CO_2 排出量

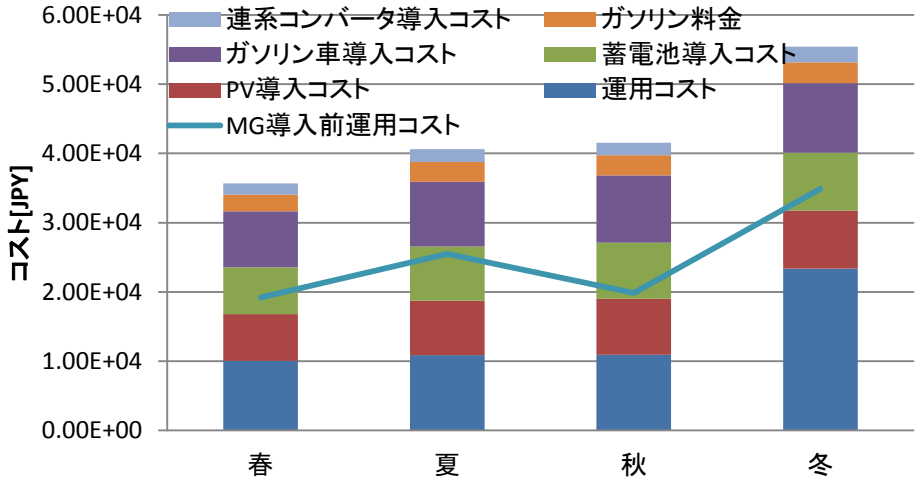

図 3.11 ガソリン車あり試算における各季のコスト

図 3.10 より，どの季節においても目標量の 26%削減を達成できていることが確認できた．ただし，EV 導入の場合と比較すると，過剰に CO_2 排出量を削減しているような季節は夏だけとなり，各設備の導入量は CO_2 排出量削減制約によって決定している事がわかった．図 3.11 よりどの季節でも MG 導入前と比較して運用コストを大幅に削減できていることが確認できた．ただし，運用コストのみについて EV 導入の場合と比較すると，ガソリン車導入の試算結果の方がややコストが高いという結果が得られた．また，EV 導入試算の場合と同様に合計コストは導入後の方が高くなることが確認された．表 3.5 に設備導入量を示す．

表 3.5 ガソリン車あり試算における各季の設備導入量

	PV[台]	蓄電池[台]	連系コンバータ [kW]
春	9	1	4.692
夏	12	1	4.733
秋	9	1	4.411
冬	12	1	5.294

どの季節でも蓄電池が 1 台導入されることが確認できた．これは EV あり試算の場合と同様に逆潮流禁止制約が原因であると考えられる．PV に関しても EV あり試算と同様に季節性があることが確認できた．

図 3.12 から図 3.15 に各季における概ね晴れの天候であったと考えられる日の運用結果を示す．EV がある場合の試算結果と同様に，PV 出力が大きい時間帯においては蓄電池に充電がなされていることが確認できた．また，図 3.13 の夏および図 3.14 の秋の結果において，EV あり試算の結果では午後 6 時以降は EV に重点的に充電がなされ，蓄電池には充電されていなかったのに対して，ガソリン車あり MG による運用では蓄電池に重点的に充電がなされている．これは，EV は出発前にその運行に利用するための SOC を確保しなければならない制約から，最優先で充電がなされることが原因であると考えられる．

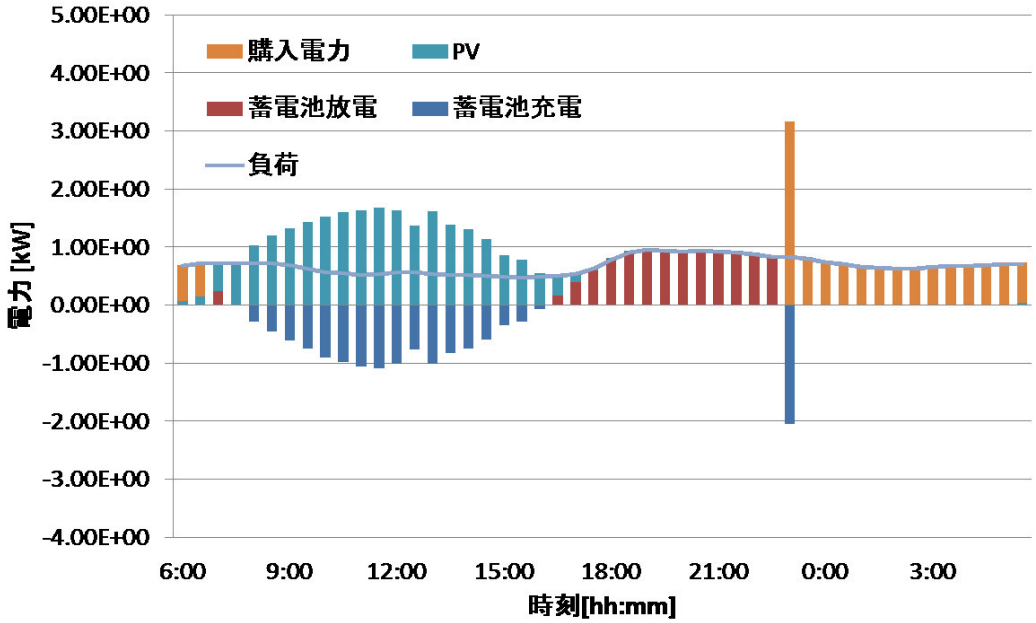

図 3.12 ガソリン車あり試算における運用例(春)

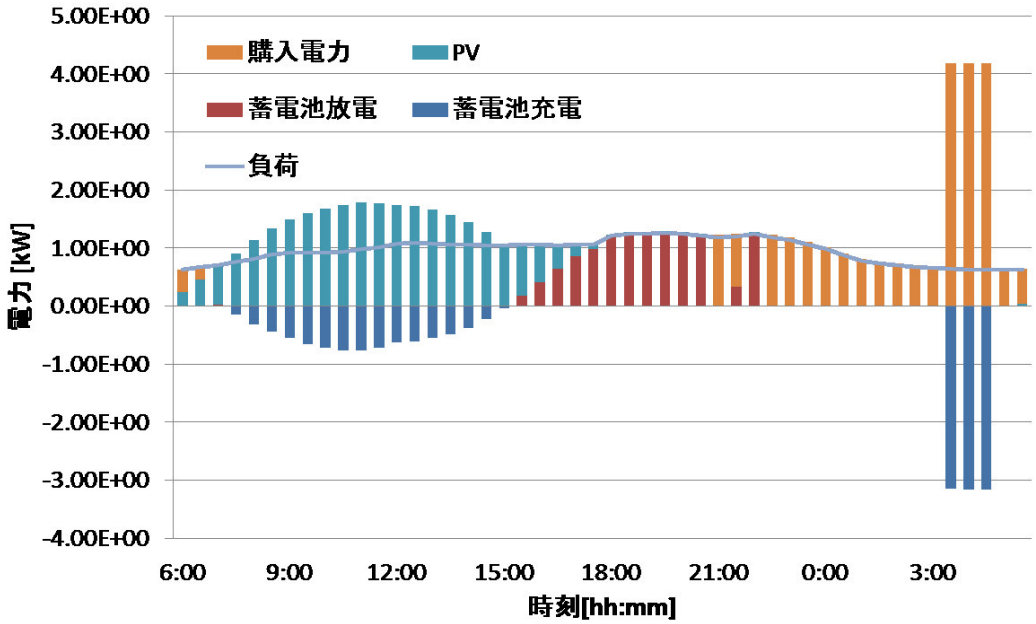

図 3.13 ガソリン車あり試算における運用例(夏)

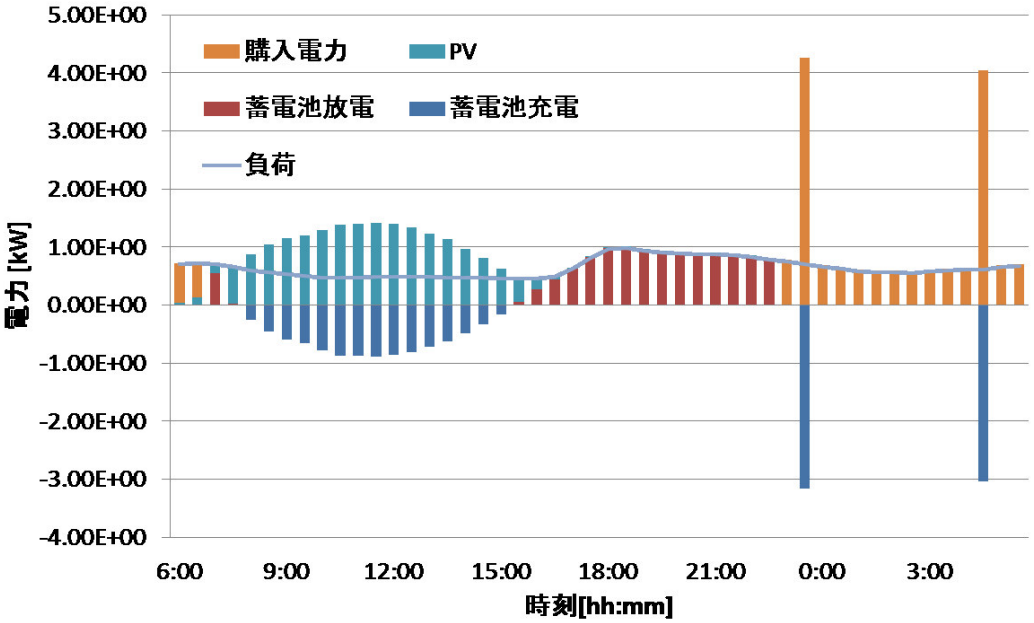

図 3.14 ガソリン車あり試算における運用例(秋)

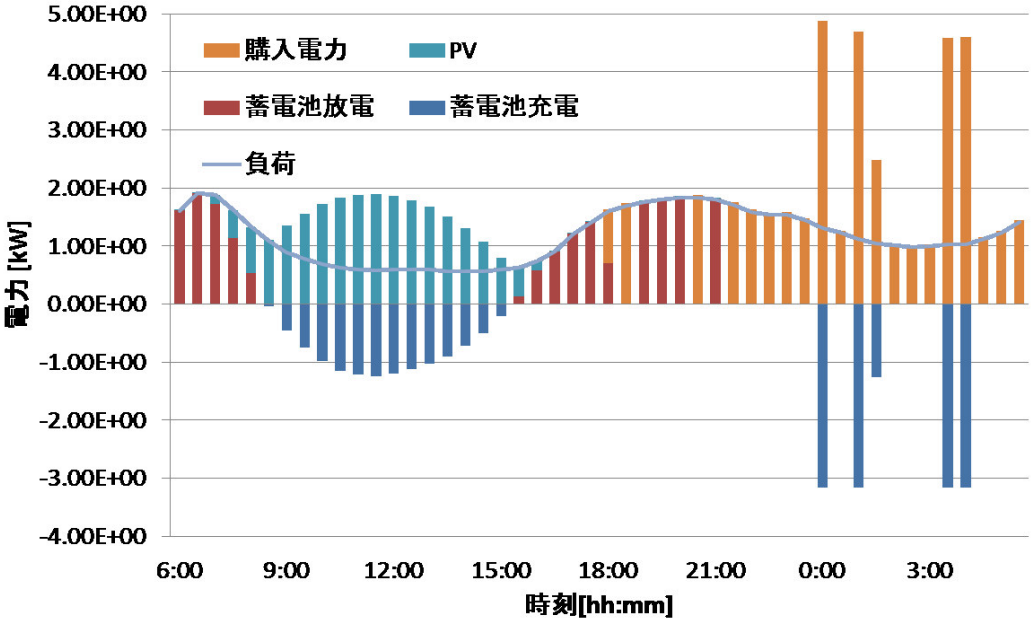

図 3.15 ガソリン車あり試算における運用(冬)

3.3　電気自動車の有無によるコスト比較

前節までのように，現在の設備導入コストで MG を新たに導入する際には，コストのメリットはないことが示唆された. ただし, CO_2 排出量削減に関してはメリットがあると考えられるため，環境負荷低減という明確な目的があるならば，MG は導入されるものと考えられる．今回の検討では，26%の CO_2 排出量削減という目標のもと，試算を行ったが EV ありの場合は過剰に削減されている様子が見受けられた．そこで，EV 導入の場合とガソリン車導入の場合とでコストを比較し，「目標達成」という部分にだけ着目した場合に，どちらの方がコストが安くなるのかを検討する必要がある．図 3.16 に四季の合計コストを示す.

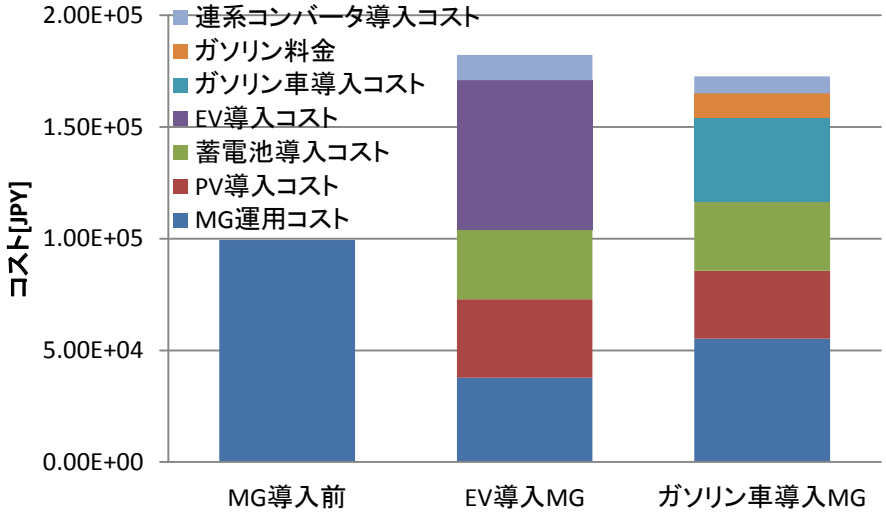

図 3.16 ガソリン車あり試算における各季のコスト

図 3.16 より，どちらの MG も合計コストが MG 導入前を上回ることが確認された．MG 導入後同士で比較すると，ガソリン車導入 MG の方が 3.44%コストが安くなることが確認できた．このことから，現在の価格設定では，CO_2 排出量削減目標達成という条件の下では，EV 導入によるメリットはなく，ガソリン車を導入する方がコストが安くなることが示唆された.

3.4　まとめ

本章では，住宅需要家を対象とした MG における各設備の最適導入量を決定した．各メーカーが発表している機器の組合せによって最適な導入量を決定することを想定し，混合整数計画問題によって各設備を何台導入するかを決定した．また，導入量を決定する設備として，PV，蓄電池および EV を想定した．EV に関しては，導入しない場合はガソリン車を導入するものとして試算を行った．試算の前に，MG のコストメリットはないことが想定されたため，本論文では CO_2 排出量を削減するという明確な目的を設定し，これを達成しながらコストが最小となる各設備の導入量を決定した．試算結果から，現在の価格によって MG を導入する際には，コストメリットがないことがわかった．特に EV が MG の導入コストを高める要因となり，ガソリン車導入型の MG の方がコストは安くなることがわかった.

第 4 章　交流マイクログリッドと直流マイクログリッドの経済性評価

4.1　概要

第 3 章では，直流負荷に対して直流給電を行う MG に対する最適容量設計を行った．しかしながら，住宅需要家においては依然として交流負荷が存在し，交流負荷が支配的であるならば交流による給電を行う方がコストが安い可能性がある．定性的にはこのように理解できるが，どの程度の負荷が交流あるいは直流である場合にどちらの給電方式が望ましいかに対する記述をしている論文はほとんどない．そこで本章では，住宅負荷に対してその利用形態ごとに交流，直流を設定し，どの種類の負荷が交流であれば，あるいは直流であればどちらの給電方式の MG を選択すればよいのかを定量的に評価した．

4.2　交流マイクログリッドと直流マイクログリッドの概要

本章における経済性評価に関しては，基本的に第 3 章における最適容量設計手法を踏襲する．すなわち，PV および蓄電池によって住宅負荷に対して電力の供給を行う際に，給電方式によってコストがどのように変化するのかを検証する．図 4.1 および図 4.2 に想定するそれぞれの MG の構成図を示す．なお，第 3 章での試算において，現在の導入コストでは EV の導入メリットがないことが示唆されたため，ここでは PV と蓄電池のみを有する MG を試算の対象とする．

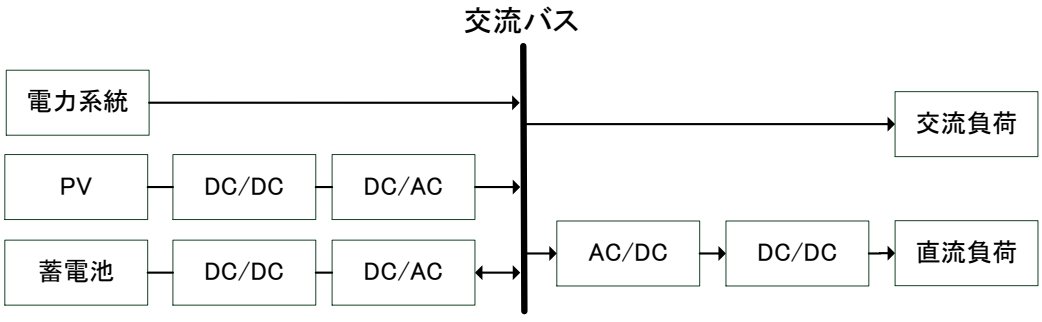

図 4.1 想定する交流 MG の設備構成

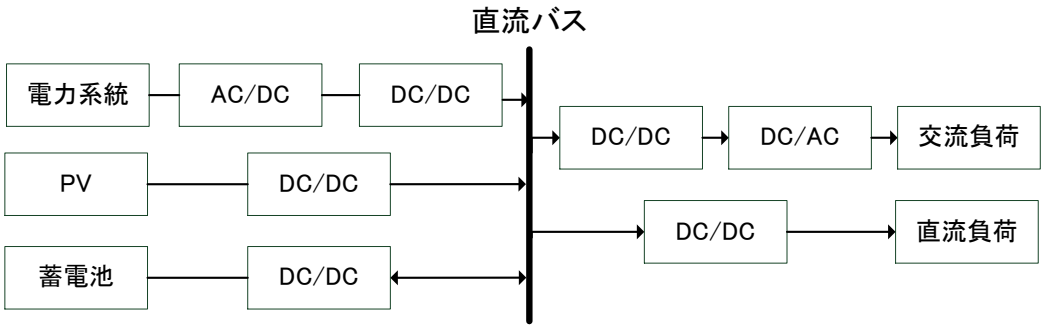

図 4.2 想定する直流 MG の設備構成

どちらの MG においても交流負荷および直流負荷に対して電力の供給が行われ，供給設備として PV および蓄電池を考慮している．また，各設備にはそれぞれの給電方式に適した変換器が接続されている．交流 MG においては，原理的には系統への電力逆潮流が可能であるが，本論文では直流 MG と公平に評価を行うため交流 MG においてもこれを禁止する．また，第 3 章と同様に各 MG には CO_2 排出量削減目標を達成するという制約を与える．

4.3 交流マイクログリッドと直流マイクログリッドの経済性評価試算

4.3.1 定式化

先述のとおり，ここでは第 3 章と同様にコストの最小化を目的関数としている．これはどちらの MG においても以下に示す式で定義できる．

$$min\ f = C_{PV} \cdot Cap_{PV} + C_{batt} \cdot Cap_{batt} + C_{rect} \cdot Cap_{rect} + OC \quad (4.1)$$

各記号の意味については 3 章と同様であるためここでは省略する．ただし，導入コストに関しては，交流用の設備と直流用の設備とで別の金額を採用する．理由としては，交流システムの方がコンバータ段数が多いため，その分だけコストが増加すると考えたためである．

本論文では，制約条件として以下を考える．なお，これらの制約条件に加えて，第 3 章で示した制約条件に関しても依然として考える必要がある．ただし，EV に関しては導入することを考慮しないため，その限りではない．

a) 需給平衡制約

需給平衡制約は第 3 章のものを修正する必要がある．すなわち，交流負荷と直流負荷のどちらも考える必要がある．まず，交流 MG においては以下の式で示される．

$$\begin{aligned} &P_{grid}(t)+\eta_{PV}\cdot Cap_{PV}\cdot P_{PV}(t) \\ &-\frac{P_{batt\_C}(t)}{\eta_{batt}}+\eta_{batt}\cdot P_{batt\_D}(t) \\ &=P_{ACL}(t)+\frac{P_{DCL}(t)}{\eta_{DCL}} \end{aligned} \tag{4.2}$$

ここに，$P_{ACL}(t)$：時刻 t における交流負荷[kW]，$P_{DCL}(t)$：：時刻 t における直流負荷[kW]
η_{DCL} ：直流負荷用コンバータの変換効率である．それ以外の記号に関しては第 3 章のものと同様である．

直流 MG の場合は以下のように示される．

$$\begin{aligned} &\eta_{grid}\cdot P_{grid}(t)+\eta_{PV}\cdot Cap_{PV}\cdot P_{PV}(t) \\ &-\frac{P_{batt\_C}(t)}{\eta_{batt}}+\eta_{batt}\cdot P_{batt\_D}(t) \\ &=\frac{P_{ACL}(t)}{\eta_{ACL}}+\frac{P_{DCL}(t)}{\eta_{DCL}} \end{aligned} \tag{4.3}$$

ここにη_{ACL} ：交流負荷用コンバータの変換効率である．

b) CO_2 排出量削減目標達成制約

第 3 章と同様に，日本政府が公表している 26%の CO_2 排出量削減目標を，MG 導入によって期間中に達成する．

c)　蓄電池 SOC の時間推移

第 3 章と同様に蓄電池 SOC は前時間帯の充放電電力より計算される．

d) 各設備の容量制約

第 3 章と同様に各設備の出力は最大電力を超えてはならない．

e) 各設備の容量制約

第 3 章と同様に蓄電池は充電と放電を同時に行えない．

4.3.2　試算条件

表 4.1 に交流 MG における各種パラメータを示す．

表 4.1 試算に用いたパラメータ

変数	意味	値
C_{PV} [28]	1日におけるPVシステム1ユニットあたりのコスト	31.9 [JPY/kW]
C_{batt} [29]	日における蓄電池システム1ユニットあたりのコス	270 [JPY/kW]
η_{PV}	PVシステムにおけるPCSの変換効率	0.93
η_{batt}	蓄電池システムの変換効率	0.86
η_{ACL}	交流負荷用変換器の変換効率	0.95
η_{DCL}	直流負荷用変換器の変換効率	0.86

前項で言及したように，交流システムにおいて PV や蓄電池を考慮する際には，AC/DC コンバータの分だけコストを上乗せする必要がある．そのため，直流システムよりも高いコストでの検討となる．第 3 章では直流用設備としてコンバータのコストを割引していたが，ここでは交流用

設備のコストを利用するため，メーカーの販売価格をそのまま適用した．また各種コンバータの変換効率に関しては，帯広市プロジェクトにおいて交流負荷にも供給する場合を想定した各種測定が行われており，そこで得られた値を利用した．

第 3 章での試算においては，負荷がすべて直流負荷であるものと仮定して最適化計算を行ってきた．しかしながら，需要家内には交流負荷も混在する．現在保持している NEDO の住宅需要家データにおいては，住宅で消費された有効電力および無効電力のみが記録されているため，その家庭における負荷の種類やその負荷の電力消費形態も知ることができない．そこで本論文では，資源エネルギー庁が発表した推計データ[34]を参考にシミュレーション用のデータを作成した．図 4.3 は資源エネルギーが発表した資料である．

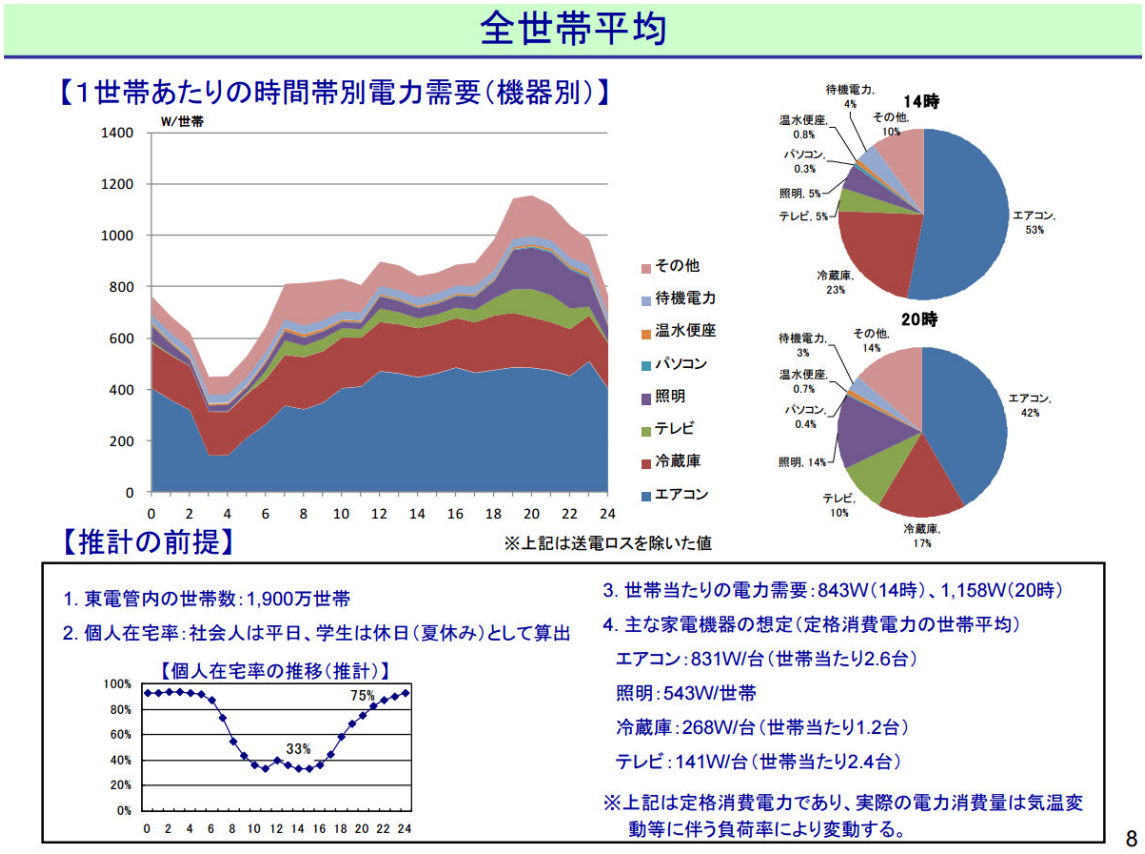

図 4.3 住宅需要家における負荷の種別(出典元：資源エネルギー庁 「夏期最大電力使用日の需要構造推計（東京電力管内）」)

この資料では，東京電力管内において，夏季の最大需要の際にどの程度まで需要が増えるかを推計したものである．参考にしたのは図中右上の円グラフである．この円グラフより，どの種類の負荷がどの程度含まれているかを知ることができる．本論文では，各負荷に対して交流負荷あるいは直流負荷として設定することで負荷の交流/直流比を想定する．なお，図 4.3 におけるすべての負荷を設定することは考えうる負荷の組合せが多すぎるうえ，元のデータもあくまで推計であることを踏まえて，本論文では 1)ベース負荷(冷蔵庫および待機電力)，2)照明，3)エアコン(以下空調と表記)および 4)その他の負荷と 4 種類に分けて扱うものとした．また，時間帯別の利用に

関しては，14 時の比率データを午前 6 時から午後 6 時に，20 時の比率データをそれ以外の時間帯に適用した．この 4 種類の負荷の比率をまとめると，表 4.2 のようになる．

表 4.2 時間帯毎の負荷の利用比率

	ベース[%]	照明[%]	空調[%]	その他[%]
午前6時～午後6時	27	5	53	15
それ以外	20	14	42	24

利用したデータが東京管内の夏季における最大需要データであるため，昼夜を通して空調負荷が多くの割合を占めていることがわかる．また，照明およびそれ以外負荷に関しては，昼ではなく夜において負荷に占める割合が増加している．ある 1 日の負荷データに対してこの比率を適用した日負荷曲線を図 4.4 に示す．

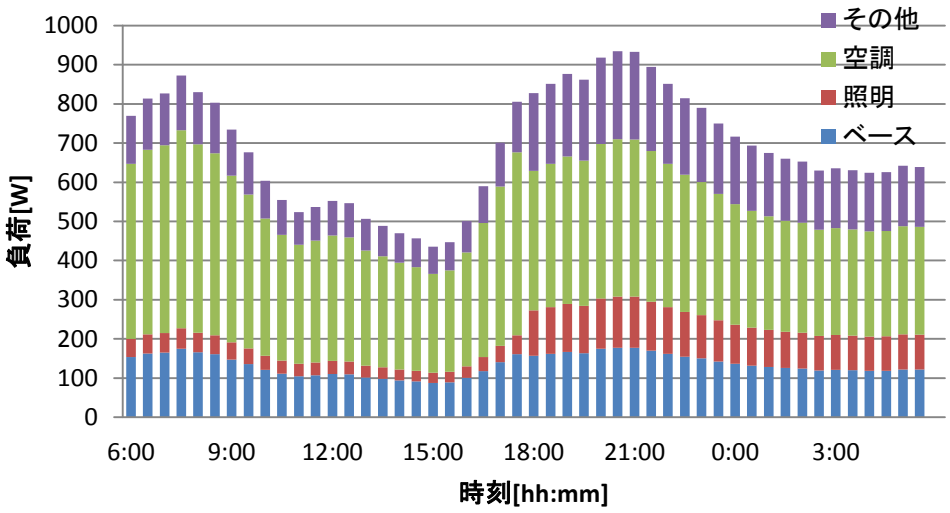

図 4.4 負荷利用比率を当てはめた 1 日の負荷データ

また, 4 つの負荷種別に対して交流および直流を想定すると, 2^4=16 通りを考慮する必要がある．しかしながら，すべての内容について掲載することは煩雑であると考え，本論文では表 4.3 に示す 5 通りの負荷組合せのみを考慮する．

表 4.3 試算において考慮する負荷の組合せ

	照明	その他	ベース	空調
(1)	交流	交流	交流	交流
(2)	直流	交流	交流	交流
(3)	直流	直流	交流	交流
(4)	直流	直流	直流	交流
(5)	直流	直流	直流	直流

このような負荷の組合せのみを考慮する理由として，各負荷の直流化傾向が挙げられる．すなわち，照明負荷およびその他負荷に関しては，すでに一般家庭において LED 照明が普及していること，その他負荷に含まれる PC やテレビ等の負荷は直流で駆動することから勘案し，将来的に直流化がなされやすいであろうということである．また，ベース負荷に含まれる冷蔵庫についても，各種家電メーカーが直流駆動の冷蔵庫を開発しており，第 2 章における帯広市プロジェクトにおいても冷蔵庫は直流負荷として扱われていた．このような事情からベース負荷が 3 番目に直流化がなされるであろうと考えた．最後に空調負荷であるが，こちらについてはまだ交流駆動のシステムが大半を占めており，直流化がなされるのは最後であろうと考えた．以上のような直流化の順番にしたがい，表 4.3 では上から下に向かって直流負荷が占める割合が増加する．以降の試算結果において参照するために，それぞれの組合せに対して便宜上(1)~(5)の番号を与える．

また，第 3 章では各季の試算によって 1 年間の試算を模擬していたが，本章では先述の通り夏季の電力需要データから負荷種別を推定したことから，夏季の 1 ヶ月間(7 月)のデータに対してのみ試算を行った．

4.3.3　試算結果

図 4.5 に各負荷比率における各 MG のコストを示す．図中の(1)~(5)は負荷の組合せに対応している．

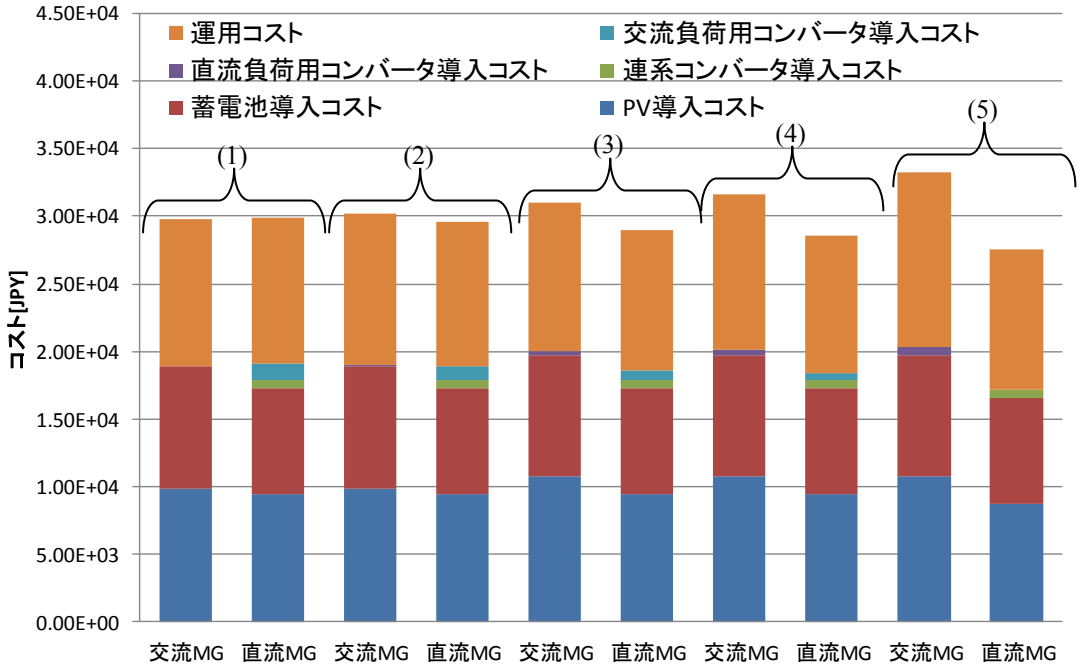

図 4.5 住宅需要家における各 MG のコスト

図 4.5 より，(1)負荷のようにすべての負荷が交流負荷の場合のみ，交流 MG の方がコスト安となることがわかった．その場合，直流 MG よりも 0.3%のコスト削減がなされることがわかった．MG 内の直流負荷比率が増加するに従い，直流 MG のコストは安く，交流 MG のコストは高くなることがわかった．(5)負荷のようにすべての負荷が直流である場合には，直流 MG は交流 MG よりも 17%だけコストが安くなることがわかった．

交流 MG と直流 MG の設備構成について比較すると，交流 MG および直流 MG は負荷比率によらず PV システム 13 台，蓄電池 1 台を導入する結果となった．今回の制約条件では，系統への逆潮流ができないため，必ず蓄電池を 1 台導入する解が選ばれたものと考えられる．また，蓄電池の kW 性能および kWh 性能が，負荷に対して十分な量であるために，蓄電池が 1 台導入されるような解では，PV の導入量も増えてしまう．そのため，住宅 1 軒に対する試算では解の選択の余地がなく，負荷の割合や給電方式によらず同じ導入量になったものと考えられる．

図 4.6 に各 MG における変換損失を示す．

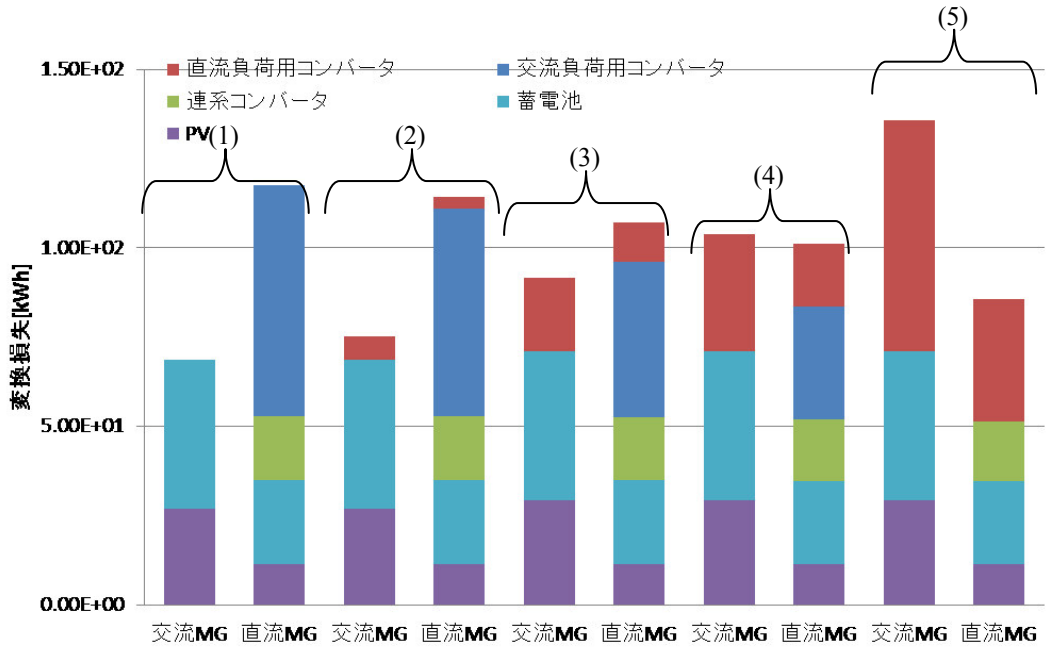

図 4.6 住宅需要家における各 MG の変換損失

図 4.6 より，負荷における交流/直流比率に応じて負荷用コンバータの変換損失は増減するものの，蓄電池や連系コンバータにおける変換損失は変化しないことがわかった．また，(1)負荷においては交流 MG と直流 MG とで変換損失の差は 41.4%あるが，これはコストと比較して明らかに大きな差である．それにも関わらず，コストにおいて直流 MG が有利となった原因は，この結果から導入コストにおける差であると考えられる．

4.4 住宅群を対象とした経済性評価

4.4.1 試算の対象とする住宅群用マイクログリッド

前節の試算では，蓄電池の性能に対して負荷が小さいためにどの負荷比率においても，同様の設備導入量が選択される可能性が示唆された．そこで，本節では MG 導入の対象とする設備を住宅 5 軒がグループとなった住宅群とし，MG における正味の負荷が十分に存在するデータに対して試算を行う．図 4.7 および図 4.8 に想定する住宅群に対する交流 MG および直流 MG の構成図を示す．

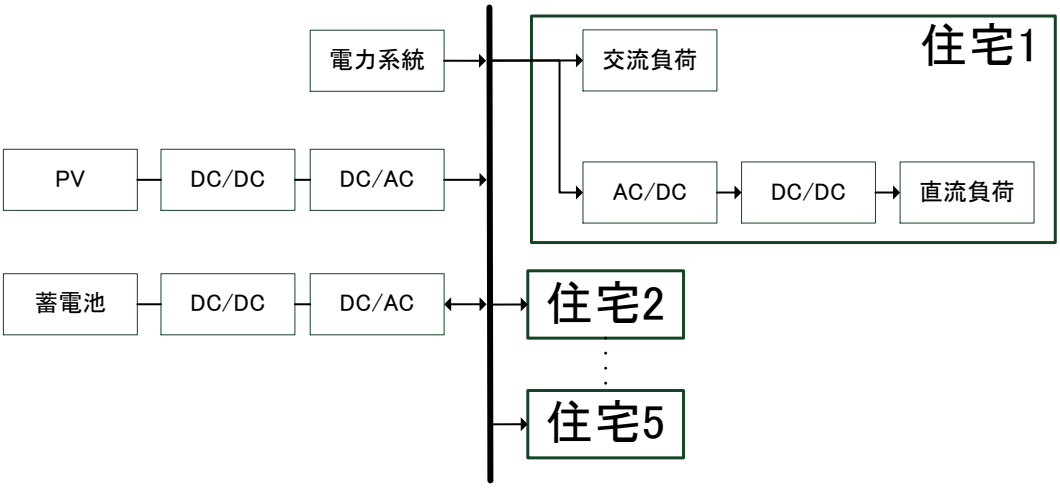

図 4.7 住宅群に対する交流 MG の構成図

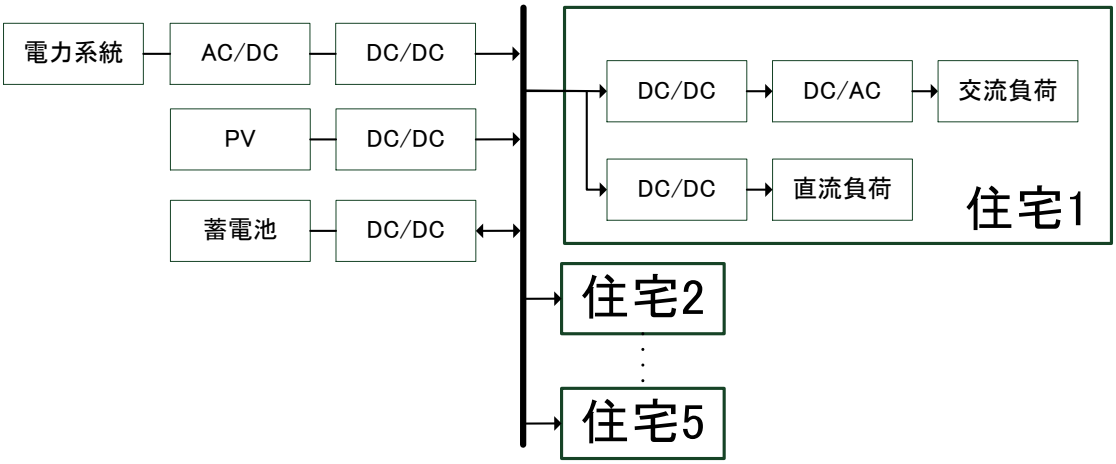

図 4.8 住宅群に対する直流 MG の構成図

MG は共通のバスによって給電がなされており，各住宅には違う大きさの交流負荷および直流負荷が存在する．本来ならば，各住宅における負荷の利用方法や交流/直流負荷比率も異なることが考えられるが，本論文ではこちらの比率においては同じものを適用した．また，PV および蓄電池は，配電線における損失が発生するため設置場所によって能力が変化するが，こちらについ

ては損失がないものとして考え，最適な容量のみを検討した．

4.4.2　試算結果

各住宅に関しては，535 軒の住宅負荷データを 5 等分し，それぞれの平均値を 1 軒の住宅における負荷データとして与えた．

図 4.9 に試算結果を示す．なお，グラフにおける(1)~(5)の数字の意味は，前節と同じく負荷の組合せである．

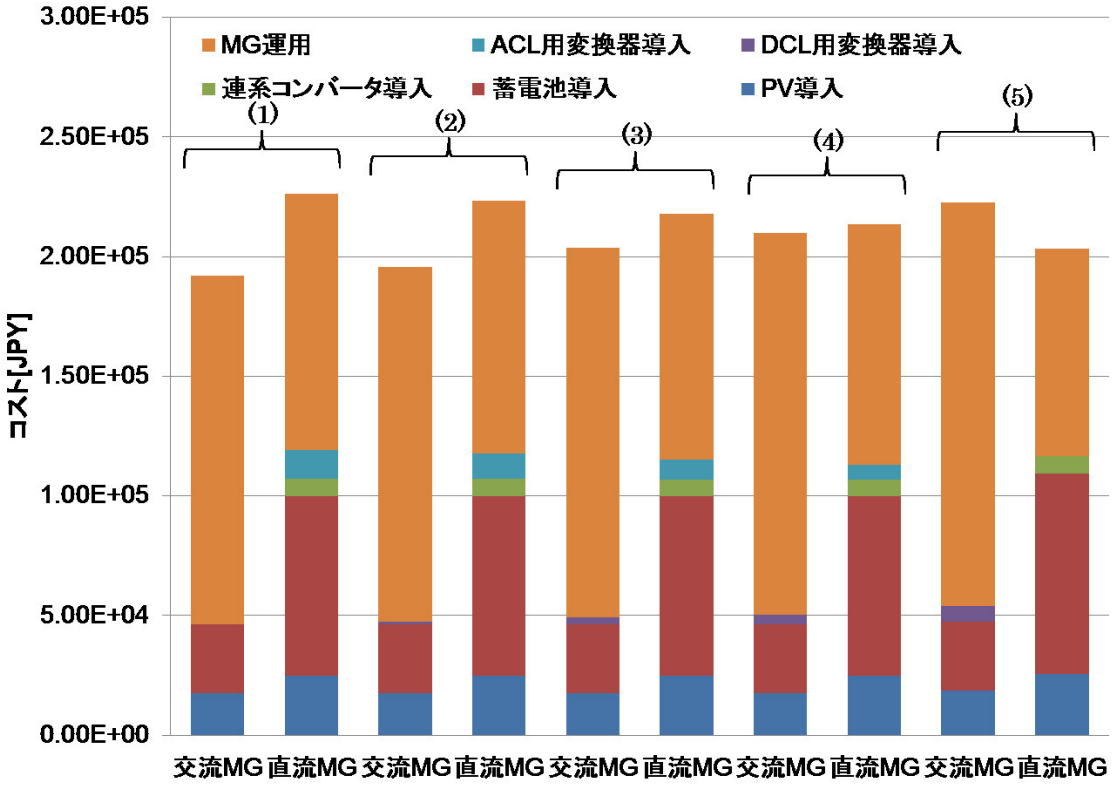

図 4.9 住宅需要家群における各 MG のコスト

図 4.9 より，交流 MG と直流 MG のコストが逆転する境界が(4)負荷と(5)負荷の間に移動したことがわかる．また，(1)負荷のようにすべての負荷が交流である場合には，住宅 1 軒の場合と比較して明らかに交流 MG が有利となり，15.2%コストが安くなることがわかった．(5)負荷のようにすべての負荷が直流である場合には，9.40%直流 MG が安くなることがわかった．また， PV および蓄電池の導入量に関しては，負荷における交流/直流比率よりも給電方式によって大きく差が現れ，交流 MG では PV システム 17 台，蓄電池システム 3 台，直流 MG では PV システム 26 台，蓄電池システム 10 台となった．これは，交流 MG では PV および蓄電池における変換損失が大きいこと，価格が高いことが原因であると考えられる．このことから，十分な負荷がある場合には，給電方式によって PV および蓄電池の導入量が変化することが確認できた．

図 4.10～図 4.13 に各 MG におけるある 1 日の運用試算結果を示す．なお，それぞれの MG に対

して(1)負荷のようにすべての負荷が交流の結果と(5)負荷のようにすべての負荷が直流の場合を示す．どの MG においても昼間は PV 出力が負荷を超えるため，その時間帯は蓄電池に充電がなされており，充電された電力は午後 6 時以降に放電されていることが確認できた．図 4.10 および図 4.12 の交流 MG ではこの時間帯に電力を購入しているものの，直流 MG，特に図 4.13 のすべて直流負荷の場合においては，この時間帯の負荷のほとんどを蓄電池からの放電電力で賄っていることが確認できた．図 4.9 において直流 MG の運用コストが交流 MG よりも少ない理由はこのためであると考えられる．

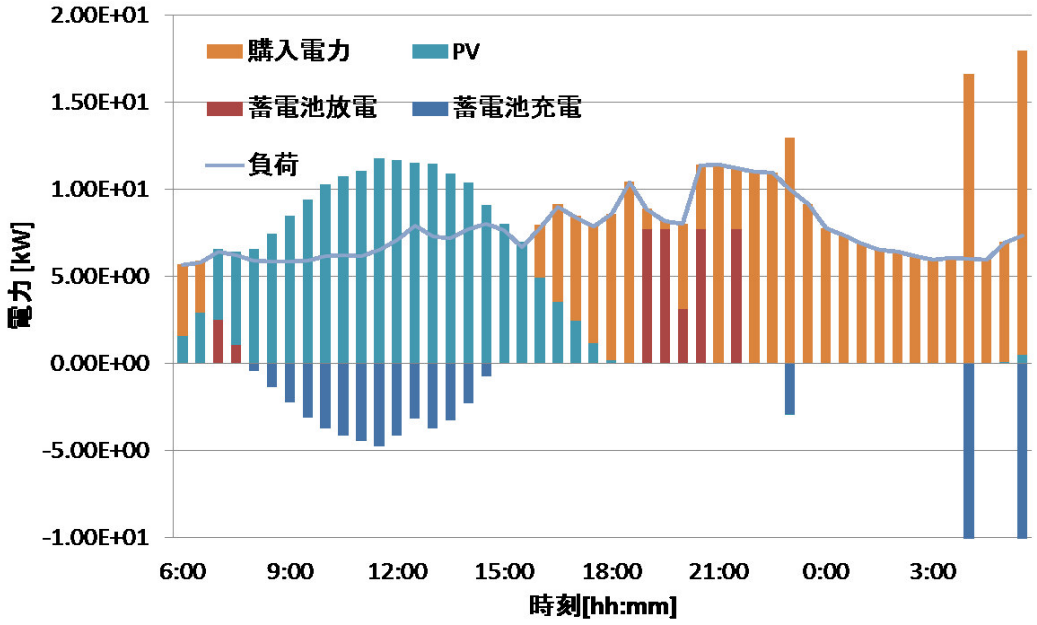

図 4.10 すべての負荷が交流の場合における交流 MG の運用試算結果

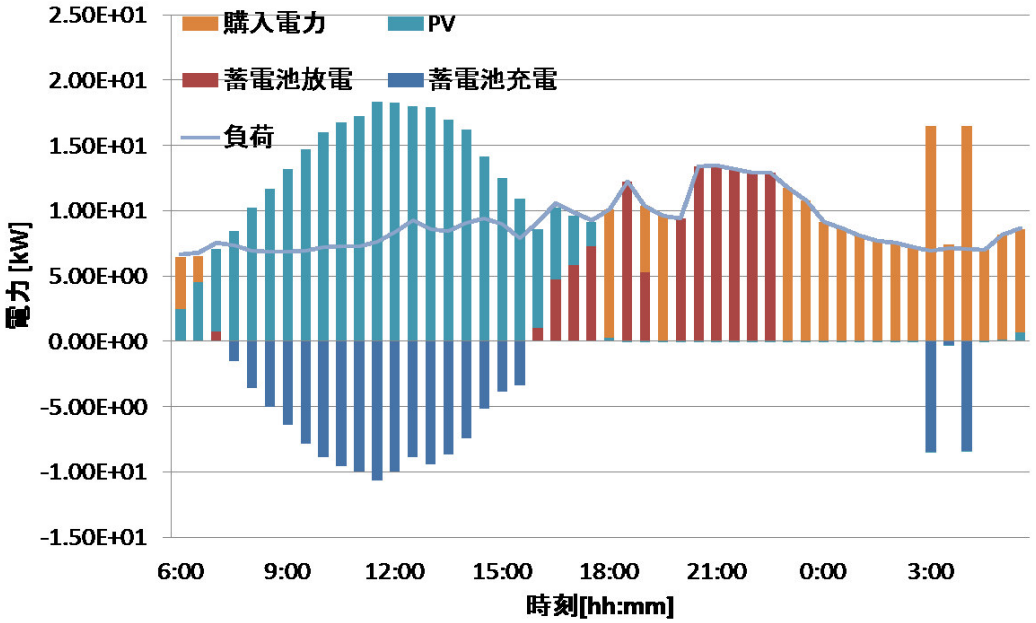

図 4.11 すべての負荷が交流の場合における直流 MG の運用試算結果

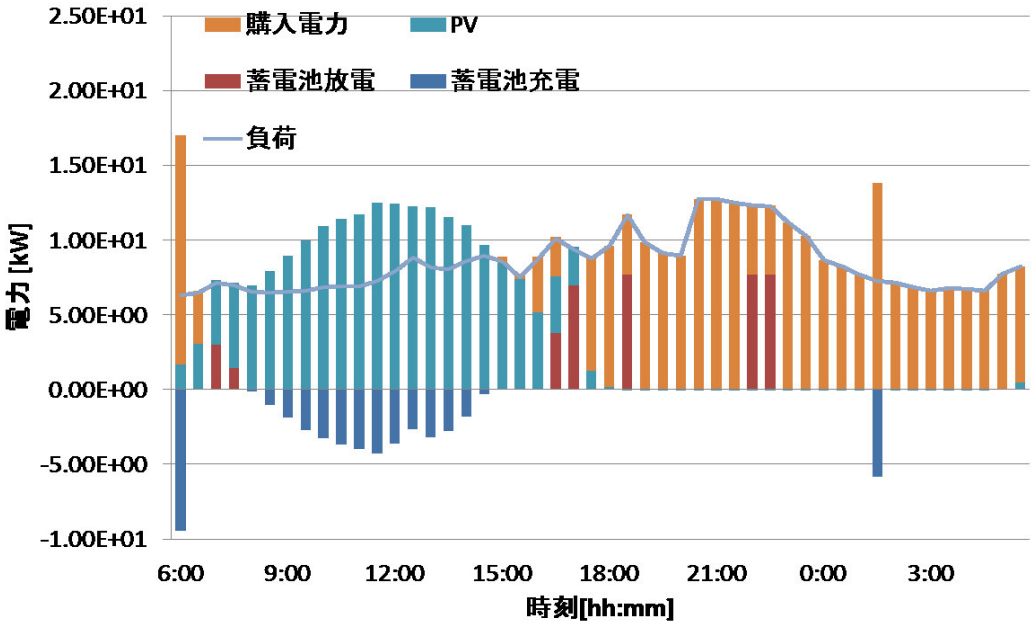

図 4.12 すべての負荷が直流の場合における交流 MG の運用試算結果

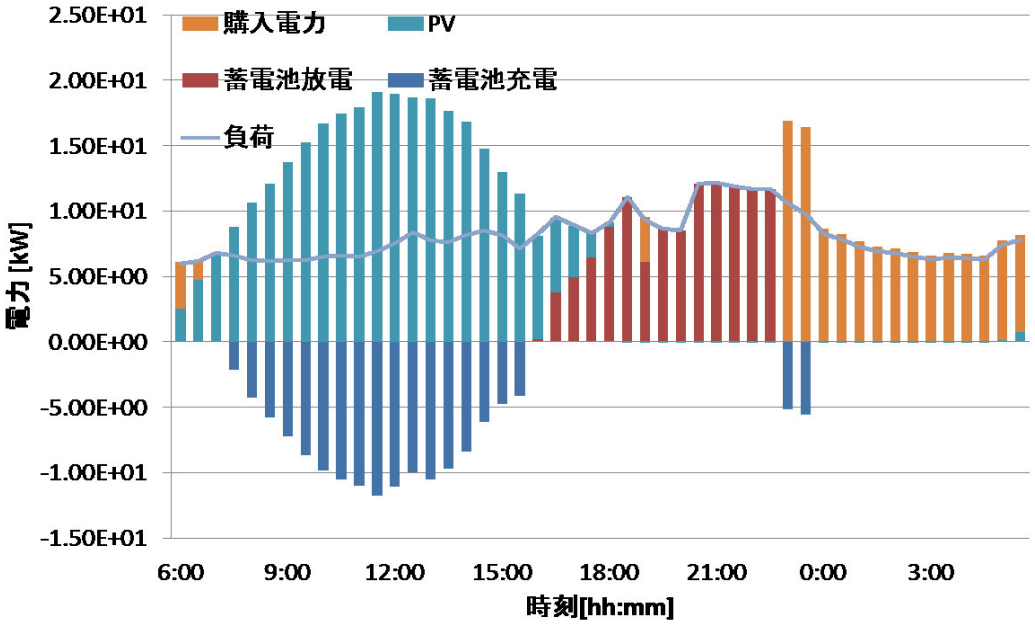

図 4.13 すべての負荷が直流の場合における直流 MG の運用試算結果

図 4.14 に各 MG における変換損失を示す．図 4.14 より，負荷における交流/直流比率に応じて負荷用コンバータの変換損失が大きく変化していることがわかる．また，住宅需要家を対象としているため，PV や蓄電池よりも負荷における変換損失の割合が大きいことがわかった．加えて，住宅 1 軒を対象とした試算では，常に直流 MG の方が損失が小さかったが，こちらの試算では PV および蓄電池の導入量が異なることから，交流 MG と直流 MG とでその合計値は変わらず，負荷による損失変化だけが影響を及ぼしていることが示唆された．負荷の損失にだけ着目すると，交

流 MG においては交流負荷に供給する際に変換損失が発生しないのに対し，直流 MG では直流負荷に供給する場合でも変換損失が発生する．このことから，直流 MG による変換損失低減のメリットは，自由に PV や蓄電池容量を決定できる場合のみ現れることが示唆された．

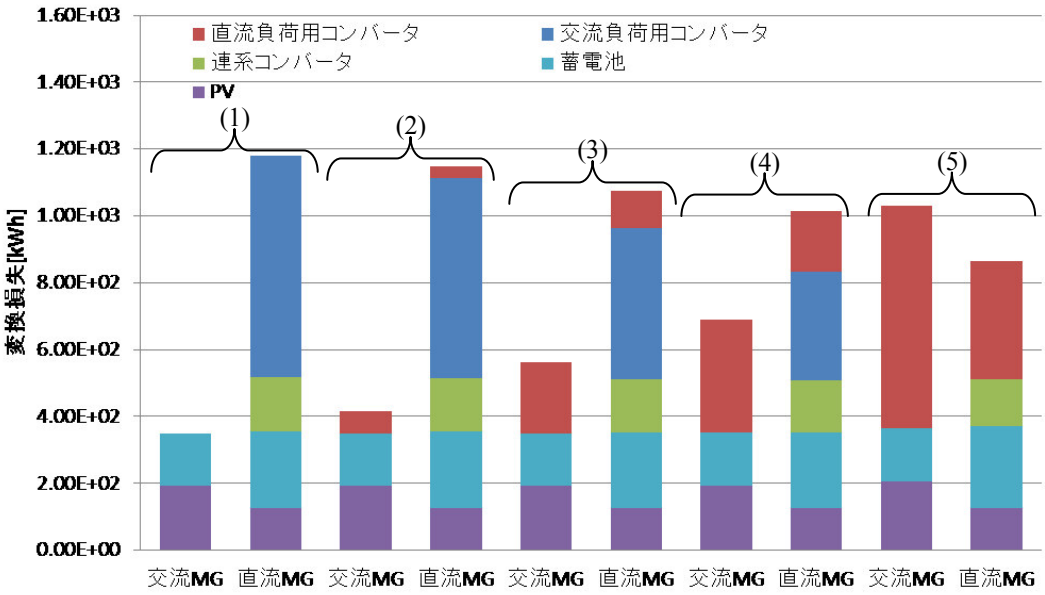

図 4.14 住宅需要家における各 MG の変換損失

4.4.3 安価な蓄電池コストに対する試算

ところで，今回の試算においては負荷の組合せを 5 種類だけ考慮した．これは，将来的に負荷が直流化されるものと想定し，直流化されやすい順番に直流負荷として扱うという方針のもと想定した．各 MG において最もコストが安くなる(1)負荷における交流 MG と(5)負荷における直流 MG とを比較すると，交流 MG の方がコストが安いという結果が得られた．すなわち，直流システムの方がコスト面で有利であるから直流化が進むという想定と逆の傾向が得られた．直流 MG のコストに注目すると，PV および蓄電池の導入コストが約半分を占めていることがわかる．また，経産省の調査[35]によると，蓄電池のコストは将来的に抑制されることが予測されている．そこで，本論文では蓄電池の価格を現在の価格の 80%として再試算し，将来における交流 MG と直流 MG とのコスト比較を行った．図 4.15 に試算結果を示す．

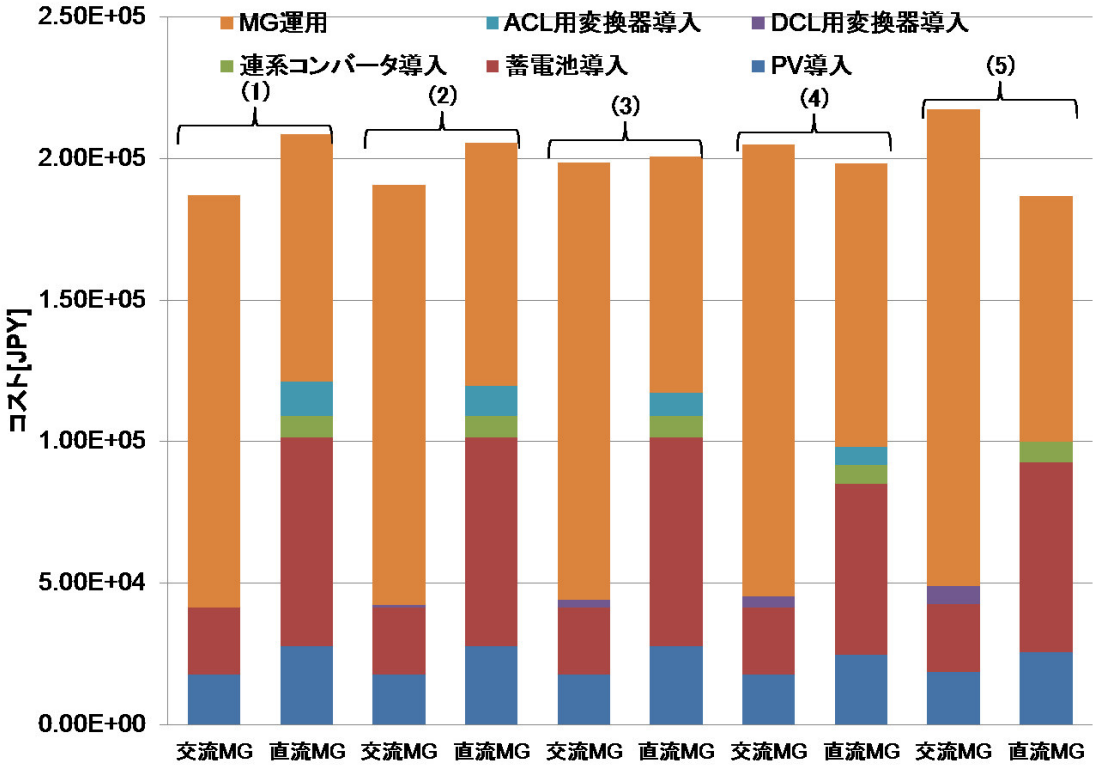

図 4.15 住宅需要家における各 MG のコスト(蓄電池価格 80%)

図 4.15 より，蓄電池のコストが下がったことに起因して，主に直流 MG における合計コストが減少したことが確認できた．交流 MG および直流 MG における PV システムおよび蓄電池システムの導入量は，どちらも図 4.9 の場合と同じものとなった．(1)負荷に対する交流 MG のコストと(5)負荷に対する直流 MG のコストを比較すると，0.14%とわずかではあるが直流 MG の方が有利となることがわかった．また，交流 MG と直流 MG のコストが逆転する境界は(3)負荷と(4)負荷の間となった．

4.4.4　集団設計と個別設計のコスト比較

ここまでの試算では，5 軒の住宅需要家に対して 1 つの MG を構築する際のコストを評価した．便宜上，これを集団設計とする．本項では同じ負荷データを持つ需要家に対して個別に MG を設計し，そのコストの合計を試算した．試算結果を図 4.16 に示す．図中における「集」は集団設計を，「個」は個別設計を表している．集団設計により，どちらの MG においても合計コストが減少することが確認できた．それぞれの給電方式に対して，集団設計によるコスト削減率を計算すると，交流 MG では平均 30.5%であったのに対して，直流 MG では 31.2%となった．このことから，集団設計によるコスト削減効果は直流 MG の方が大きいことが示唆された．

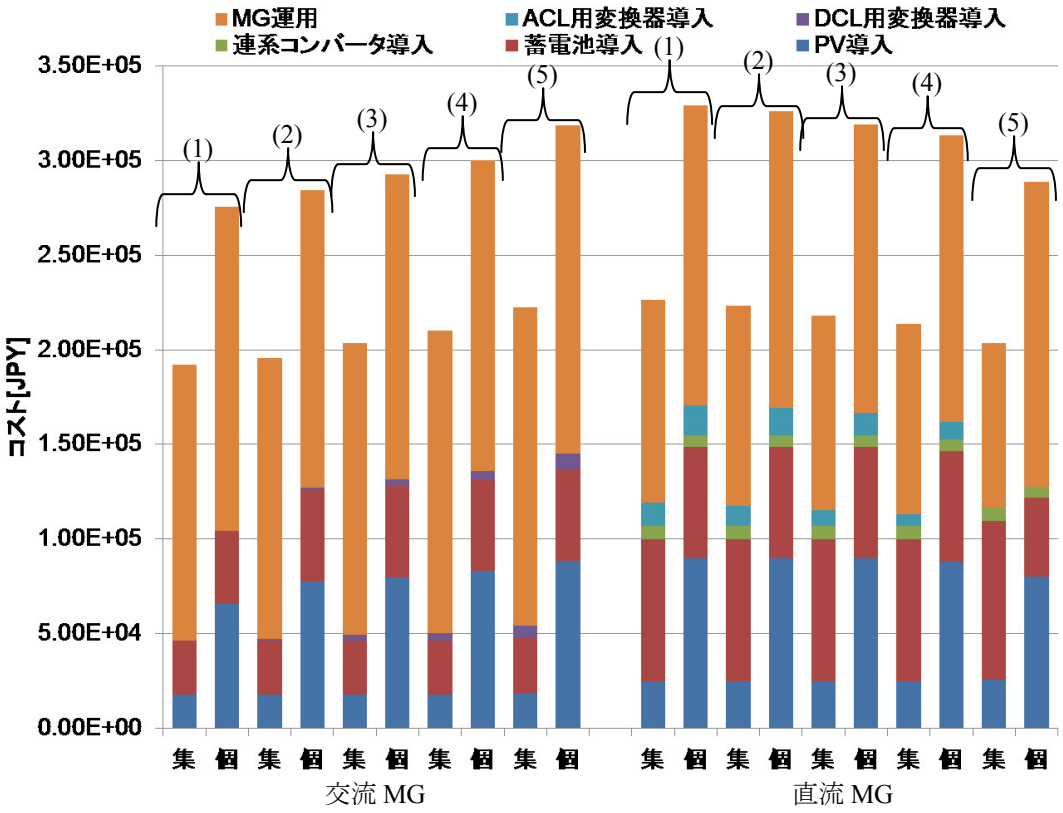

図 4.16 住宅需要家における各 MG のコスト(蓄電池価格 80%)

4.5 まとめ

本章では MG における給電方式に着目し，交流と直流のどちらの方式を採用した方がより経済性が高いかについて試算を行った．住宅需要家 1 軒の場合は，給電方式や需要内の交流/直流負荷比率に関わらず一定の割合で各設備が導入されることがわかった．これは，蓄電池の性能が住宅 1 軒に対しては十分であるためであると考え，住宅 5 軒を 1 つの負荷として扱い試算を行った．その結果，負荷比率ではなく給電方式によって導入される PV や蓄電池の容量は変化し，住宅内における負荷の比率によってコストが変化することが確認できた．しかしながら，PV および蓄電池の導入コストが原因で，直流 MG の有利がないことも同時に確認されたため，蓄電池のコストを低くして再試算を行った. その結果, 直流 MG の方が将来的に有利となることが示唆された.

第 5 章　結論

5.1　本論文のまとめ

本論文では，MG について着目し，その運用方法の提案，最適容量設計ならびに MG の給電方式の違いによる経済性について評価を行った．

第 1 章では現在の我が国におけるエネルギー事情について述べ，その中で RE の積極的な導入が検討されていること，RE を有効利用する技術として MG が注目されており，さまざまな研究がなされていることについて述べた．また，従来までの交流給電ではなく，直流給電によるメリットが再検討されており，給電範囲の狭い MG においてはその有効性が示唆されていることを示した．

第 2 章では MG の EMS において必要となる機能のうち，特に前日計画機能について具体的な計画アルゴリズムを開発した．提案する前日計画アルゴリズムは MG から排出される CO_2 を最小化するために，MG の各設備の 30 分毎の出力計画を決定するものであり，翌日の負荷予測や PV 出力予測を用いた．PV 出力予測には必然的に誤差が生じるため，本論文では特に翌日の PV 出力予測に関して単一の出力カーブを用いるのではなく，いくつかの出力カーブをシナリオとして考慮する手法を開発した．提案したアルゴリズムによる前日計画を検証するにあたり，本論文では環境省のプロジェクト（地球温暖化対策技術開発・実証研究事業「自立・分散型エネルギー社会の実現に向けた直流方式による地域間相互エネルギー融通システムの開発」）の一環として構築された帯広市直流 MG 実証設備で得られたデータを利用した. 特に予測誤差の大きな日においては，単一の PV 出力予測を利用した場合と比較して，シナリオを用いた本論文の手法の方が CO_2 排出量を削減できることを確認した．また，年間を模擬した試算においても CO_2 排出量削減効果が向上することを確認した．さらに，本論文で得られた前日計画を当日の 1 秒値シミュレーションに適用した結果，本論文で得られた手法と同様の傾向を示した．この試算においては，予測誤差の発生に対応して各設備が出力を変更し，MG バスにおける電圧の維持がなされていることを確認した．このことから，提案手法はより短い時間粒度の運用においても有効であることが示唆された．

第 3 章では帯広市プロジェクトにおいて各設備の容量が最適でないことが原因と思われるパフォーマンスの低下がいくつか散見されたことを受けて，MG における設備の最適容量設計を行った．この設計では，住宅需要家向け直流 MG における PV，蓄電池および EV の最適容量をそのコストが最小となるように設計した．この際，実際の設計においてはメーカーから発表されている機器の組合せによって最適な容量を実現する必要があることから，本論文では PV および蓄電池をある決まった容量を持ったユニットとして扱い，従来のように実数で導入容量を決定するのではなく，整数で容量を決定した．住宅を対象とした試算では，PV および蓄電池に関しては導入がなされるものの，EV に関しては導入を行わない方が経済的であることが確認できた．

第 4 章では，ここまで得られた MG の運用手法ならびに容量設計手法を用いて，交流で給電を行う交流 MG と直流 MG との経済性比較を行った. 本論文では住宅需要家内の負荷をベース負荷，照明負荷，空調負荷およびそれ以外の負荷 4 つの利用形態に分けて扱い，これらに対して交流負荷，直流負荷を設定することで MG の給電方式を定量的に決定した．住宅 1 軒を対象とした試算では，わずかでも直流負荷が需要にあれば直流 MG が有利となることが示された．一方で，住宅 5 軒をまとめた試算おいては，ベース負荷，照明負荷およびそれ以外負荷が直流となるまでは交流有利という結果が得られた．したがって，新たに MG を設計するに際しては，その需要内における交流負荷および直流負荷の比率を把握して給電方式を適切に設定することで，MG の経済性を向上させられることが示唆された.

5.2 今後の展望

本論文においては検討することができなかったが，以下の項目を今後の展望としたい.

- 帯広市 MG のように実証施設を用いた EMS の機能検証について，さらに長い時間での実運用データを計測すること.
- MG の給電方式による経済性比較において，住宅需要家における正確な負荷の交流/直流比率の推定．ならびに，住宅以外の負荷に対する経済性試算.

参考文献

[1] 経済産業省資源エネルギー庁 :「平成 25 年度　エネルギーに関する年次報告(エネルギー白書)」

[2] 高野富裕 :「自然エネルギー発電と電力貯蔵技術」，電学論 B，Vol.126，No.9，pp.857-860（2006）

[3] 廣瀬圭一 :「直流技術と応用の動向」，電学論 B，Vol.131，No.4，pp.358-361(2011)

[4] 横山明彦 :「スマートグリッド」，㈳日本電気協会新聞部（2010）

[5] 横山明彦，他 :「スマートグリッドの構成技術と標準化」，日本規格協会（2010）

[6] NEDO :「10　スマートコミュニティの構築に向けて(再生可能エネルギー技術白書)」p.634(2010)

[7] 欧州委員会 : ”Communication “Smart Grids: from innovation to development” [COM(2011)202]”(2011)

[8] NEDO :「2005 年日本国債博覧会・中部臨空都市における新エネルギー等地域集中実証実験に係る委託業務　実証研究報告書」(2008)

[9] NEDO :「新エネルギー等地域集中実証研究　京都エコエネルギープロジェクト」(2009)

[10] NEDO :「新エネルギー等地域集中実証研究　八戸市　水の流れを電気で返すプロジェクト」(2009)

[11] 清水建設 Web サイト :「ecoBCP 事例」，http://www.shimz.co.jp/theme/ecobcp/case.html (2015 年 12 月 1 日アクセス)

[12] 東京ガス ガ，スマート!Web サイト，http://home.tokyo-gas.co.jp/ga-smart/(2015 年 12 月 1 日アクセス)

[13] JAPAN SMART CITY PORTAL Web サイト，http://jscp.nepc.or.jp/index.shtml(2015 年 12 月 1 日アクセス)

[14] JAPAN SMART CITY PORTAL Web サイト :「横浜スマートシティプロジェクト(YSCP)」，http://jscp.nepc.or.jp/yokohama/index.shtml(2015 年 12 月 1 日アクセス)

[15] JAPAN SMART CITY PORTAL Web サイト :「豊田市低炭素社会システム実証プロジェクト(Smart Melit)」，http://jscp.nepc.or.jp/toyota/index.shtml(2015 年 12 月 1 日アクセス)

[16] JAPAN SMART CITY PORTAL Web サイト :「けいはんなエコシティ次世代エネルギー・社会システム実証プロジェクト」，http://jscp.nepc.or.jp/keihanna/index.shtml (2015 年 12 月 1 日アクセス)

[17] JAPAN SMART CITY PORTAL Web サイト :「北九州スマートコミュニティ創造事業」，http://jscp.nepc.or.jp/kitakyushu/index.shtml (2015 年 12 月 1 日アクセス)

[18] 省エネルギー技術戦略検討会 :「省エネルギー技術戦略」(2002)

[19] NTT ファシリティーズ :「IT 時代における電源システムの DC 化～提案の背景と課題～」http://www.dc-powers.com/

[20] A. Kwasinski: “Implication of smart-grids development for communication systems in normal

operation and during disasters” , Proc. IEEE Conf. on INTELEC, pp.1-8, Orland, FL(2010)
[21] D. Salomonsson, L. Soder, and A. Sannio: “An Adaptive Control System for a DC Microgrid for Data Centers”, IEEE Trans. Industr. Applic., Vol.44, No.6, pp.1910-1917(2008)
[22] 河本映，中島祥太：「太陽光発電の連系を想定した地域直流配電システムの損失評価」，電学論 B, Vol.131, No.4, pp.369-375(2011)
[23] 気象庁 Web サイト，http://www.jma.go.jp/jma/index.html (2015 年 12 月 1 日閲覧)
[24] Nicole Gröwe-Kuska, Holger Heitsch and Werner Römisch, “Scenario Reduction and Scenario Tree Construction for Power Management Problems”, Power Tech Conference Proceedings (2003)
[25] 三菱自動車株式会社 Web サイト：「i-MiEV | 軽自動車 | カーラインアップ | MITSUBISHI MOTORS JAPAN」，http://www.mitsubishi-motors.co.jp/i-miev/(2015 年 12 月 1 日閲覧)
[26] 北海道電力 Web サイト：「2011 年度の CO_2 排出実績について」，http://www.hepco.co.jp/info/info_event/1188484_824.html(2015 年 12 月 1 日閲覧)
[27] 北海道電力 Web サイト：「2013 年度の CO_2 排出実績について」，http://www.hepco.co.jp/info/info2014/1189637_1638.html (2015 年 12 月 1 日閲覧)
[28] 北海道電力 Web サイト：「時間帯別電灯や融雪用電力など選択できるご契約の単価（選択約款）」，http://www.hepco.co.jp/userate/price/unitprice/unitprice02.html (2015 年 12 月 1 日閲覧)
[29] 株式会社東芝 Web サイト：「住宅用太陽光発電システム」，http://www.toshiba.co.jp/sis/h-solar/inc/pdf/catalog_10.pdf (2015 年 12 月 1 日閲覧).
[30] 東芝ライテック株式会社 Web サイト:「基本機能 | フェミニティ対応蓄電システム エネグーン | ホーム IT システム　東芝 HEMS | 東芝ライテック(株)」，http://feminity.toshiba.co.jp/feminity/service/enegoon_basic4sub.html (2015 年 12 月 1 日閲覧)
[31] 三菱自動車株式会社 Web サイト：「ミラージュ | 乗用車 | カーラインアップ | MITSUBISHI MOTORS JAPAN」，http://www.mitsubishi-motors.co.jp/mirage/index.html (アクセス日 2015 年 12 月 1 日)
[32] 経産省・環境省「特定排出者の事業活動に伴う温室効果ガスの排出量の算定に関する省令」，http://law.e-gov.go.jp/htmldata/H18/H18F15002002003.html (2015 年)
[33] 札幌市消費者センターWeb サイト「物価情報石油製品」，http://www.shohi.sl-plaza.jp/bukka/sekiyu.html (2015 年 12 月 1 日閲覧)
[34] 資源エネルギー庁：「夏期最大電力使用日の需要構造推計(東京電力管内)」，http://www.meti.go.jp/setsuden/20110513taisaku/16.pdf (アクセス日 2015 年 12 月 1 日)
[35] 経産省蓄電池戦略プロジェクトチーム「蓄電池戦略」，http://www.enecho.meti.go.jp/committee/council/basic_problem_committee/028/pdf/28sankou2-2.pdf

本研究に関して著者が公表した論文

[1] 下町健太朗, 岩見俊幸, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用マイクログリッドにおけるEMS 開発」, 電気学会論文誌B 分冊,(2016 年136 巻4 号掲載決定)

本研究に関して著者が行った講演発表等

[1] 下町健太朗, 原亮一, 北裕幸:「北海道大学のスマートキャンパス化に向けた最適スケジューリング手法の開発」, 平成23 年度電気・情報関係学会北海道支部連合大会,No. 48(2011)

[2] 下町健太朗, 原亮一, 北裕幸:「北海道大学スマートキャンパス構想～供給設備の最適運用計画に関する基礎検討～」, 平成24 年電気学会全国大会,No. 6-120(2012)

[3] 下町健太朗, 原亮一, 北裕幸:「北海道大学スマートキャンパス構想～エリア間エネルギー融通を考慮した供給設備の最適運用計画」, 平成24 年電気学会電力・エネルギー部門大会,No.151(2012)

[4] 下町健太朗, 原亮一, 北裕幸:「北海道大学におけるスマートキャンパス導入効果の検討」, 平成24 年電気・情報関係学会北海道支部連合大会,No. 67(2012)

[5] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 廣瀬圭一, 星秀和:「オフィス用直流マイクログリッドのEMS に関する検討」, 平成25 年電気学会全国大会,No6-071(2013)

[6] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～最適化に基づく前日計画～」, 平成25 年電気学会電力・エネルギー部門大会,No. 281(2013)

[7] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～日射量予測とその誤差を考慮した前日計画～」, 平成25 年電気学会電力技術・電力系統技術合同研究会,PE-13-059, PSE-13-075(2013)

[8] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～日射量予測とその誤差を考慮した前日計画～」, 平成25 年度電気・情報関係学会北海道支部連合大会,No. 50(2013)

[9] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～一般的な天気予報情報に基づく前日計画～」, 平成26 年電気学会全国大会,Vol. 6,pp.15-16(2014)

[10] Kentaro Shimomachi, Ryoichi Hara, Hiroyuki Kita, Masatoshi Noritake, Hidekazu Hoshi, Keiichi Hirose:「Development of Energy Management System for DC Microgrid for Office Building」, Proc. of 18th Power Systems Computation Conference (PSCC2014), Wrocław, Poland, August , No. 310(6 pages) (2014)

[11] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討-実データに基づいた前日計画の評価-」, 平成26 年電気学会電力・エネルギー部門大会,No. 310(2014)

[12] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～日射量予測とその誤差を考慮した前日計画～」, 平成26 年電気学会電力技術・電力系統技術合同研究会,PE-14-067, PSE-14-067(2014)

[13] 下町健太朗, 原亮一, 北裕幸, 則竹政俊, 星秀和, 廣瀬圭一:「オフィス用直流マイクログリッドのEMS に関する検討～給電方法の違いがCO2 排出量削減効果に与える影響～」, 平成26 年度電気・情報関係学会北海道支部連合大会No. 55(2014)

[14] 下町健太朗, 原亮一, 北裕幸:「マイクログリッドの直流負荷比率を考慮した直流・交流給電方式の比較」, 平成27 年電気学会電力技術・電力系統技術合同研究

会,PE-015-088,PSE-15-110(2015)

[15] Kentaro Shimomachi, Ryoichi Hara, Hiroyuki Kita:「Comparison between DC and AC Microgrid Systems Considering Ratio of DC Load」, Proc. of IEEE PES Asia-Pacific Power and Energy Engineering Conference 2015 (IEEE PES APPEEC 2015) , Brisbane, Australia, November, PES-APPEEC 062(4 pages) (2015)

[16] 下町健太朗, 原亮一, 北裕幸:「住宅需要家群を対象とした直流・交流マイクログリッドの比較～直流負荷比率を考慮した最適容量設計～」平成27 年度電気・情報関係学会北海道支部連合大会,No. 63(2015)

謝辞

本研究の遂行および本論文の執筆にあたり，多くの方々から多大なるご指導・ご支援をいただきました．

北海道大学大学院情報科学研究科電力システム研究室　北　裕幸教授には著者が本研究を進めるにあたり，日頃の綿密な研究打ち合わせ，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，ご鞭撻を頂戴いたしました．ここに心から御礼申し上げます．

北海道大学大学院情報科学研究科電力システム研究室　原　亮一准教授には論文執筆，発表練習等の研究活動だけでなく，さまざまな場面で貴重な経験を与えて下さりました．ここに心から御礼申し上げます．

北海道大学大学院情報科学研究科電力システム研究室　田中　英一助教には，研究打ち合わせ，月例ゼミ等で多大なご指導，ご助言をいただきました．ここに心から御礼申し上げます．

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いただきました．深く御礼申し上げます．

株式会社 NTT ファシリティーズの則竹様，廣瀬様，星様には，打ち合わせ等でご助言いただくとともに，貴重なデータを提供していただきました．心より御礼申し上げます．

高橋　緑秘書，出村　澄世秘書には，本研究にあたりご支援いただいた他，研究以外の生活面でも多大なご支援をいただきました．心から御礼申し上げます．

日頃より公私共に多くのご支援をいただ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皆様には深く感謝いたします．

本研究の一部は環境省による平成 24 年度～平成 26 年度地球温暖化対策技術開発・実証研究事業「自立・分散型エネルギー社会の実現に向けた直流方式による地域間相互エネルギー融通システムの開発」にて検討したものであり，関係各位にはここに深く御礼申し上げます．

本研究の一部はパワーアカデミー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御礼申し上げます．

最後に，大学院進学を快諾し，生活上の支援をしていただいた両親に深く感謝いたします．

2016 年 3 月

電力システム研究室

下町健太朗

付録

プログラムリスト

本論文における試算は，すべて MATLAB によって実装した．また，第 3 章および第 4 章における混合整数線形計画問題の求解にはソルバーである CPLEX を用いた．以下に各プログラムのリストを示す.

・帯広市 PJ に関わる部分のプログラム

まず，大本のプログラムは”simulation_1sec_from_6_with_iwami_algo.m”というファイルである。内部で機能を呼び出しており，具体的には PV シナリオ作成に関わる部分が”scenario_2_fast.m”，区分線形近似を用いた線形計画法に関わる部分が”Piecewise_prob_from_6.m”である。

また，岩見作成部分のアルゴリズムにより，電圧・電流の 1 秒値制御がなされる。具体的には，”conv_regulate_2.m”，”batt_regulate_2.m”，”EV_regulate_2.m”，”PV_regulate_2.m”によりそれぞれの電源が電圧制御を維持する時の制御アルゴリズムに分類される。

最後に，本論文において比較対象として設計した，EMS のないタイマ式充放電アルゴリズムは”wo_ems.m”である。

・容量設計および交流 MG と直流 MG の比較に関連するプログラム

まず，大本のプログラムは”Opt_MG.m”というファイルである。この中において，EV_inst という変数は EV 導入の有無を表し，1 ならば 1 台導入，0 ならばガソリン車導入のシミュレーションが可能である。また，”Optimize_AC.m”および”Optimize_DC.m”というファイルはそれぞれ交流 MG による設計と直流 MG による設計を行うアルゴリズムである。

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ある期間分回すシミュレーション
% SOCは前の日の最終値を引き継ぎ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
%必要な諸データの読み込み
load('demand1.mat');
ans=demand1;
load('matlab_from_6.mat');
device_data;
count_flag=0;
V_DC_ref=380;
% I_conv_min=0.1;
% I_batt_min=0.1;
% I_EV_min=0.1;
% V_conv_init=380;
% R_line=0.1;
% X_batt=eff_batt;
% X_EV=eff_EV;
% ene_batt_init=SOC_batt_init;
% ene_EV_init=SOC_EV_init;
% ene_batt_max=SOC_batt_max;
% ene_EV_max=SOC_EV_max;
% ene_batt_min=SOC_batt_min;
% ene_EV_min=SOC_EV_min;
EV=csvread('EV.csv');
%削減後のシナリオ数
n=10;
%%日付と予報の読み込み（天気予報のあるもののみ）
date_weather=xlsread('Weather.xlsx');
%負荷データの読み込み(とりあえず1週間分)
% load_buff(:,:,1)=xlsread('需要データ.xlsx','10_29');
% load_buff(:,:,2)=xlsread('需要データ.xlsx','10_30');
% load_buff(:,:,3)=xlsread('需要データ.xlsx','10_31');
% load_buff(:,:,4)=xlsread('需要データ.xlsx','11_01');
% load_buff(:,:,5)=xlsread('需要データ.xlsx','11_02');
% load_buff(:,:,6)=xlsread('需要データ.xlsx','11_03');
% load_buff(:,:,7)=xlsread('需要データ.xlsx','11_04');
% i=1:7;
% load=load_buff(:,1,i)+load_buff(:,2,i);
% load=reshape(load,48,7);
%
% load=horzcat(load,load);
% load=horzcat(load,load);
% load=horzcat(load,load);
% load=horzcat(load,load);
% %平日の負荷予測値データ
% load_week=(load(:,1)+load(:,2)+load(:,3)+load(:,4)+load(:,5))/5;
% %休日の負荷予測値データ
% load_hol=(load(:,6)+load(:,7))/2;

SOC_batt_max_hardware=20000;
SOC_EV_max_hardware=12800;

```



```
%% データベース読み込み
%天気データを読み込む
weather = xlsread('日射量データベース.xlsx');
weather(:,1)=weather(:,1)+datenum('30-Dec-1899');
%日射量データを読み込む
radiation(:,1)=weather(:,1);
for num=1:24
    radiation(:,num+1)=xlsread('日射量データベース.xlsx',num+1,'B:B');
end
%ブランクのある日を削除
num=1;
day_blank=zeros(1,1);
while num<=numel(radiation)/25
    if any(isnan(radiation(num,:)))==1
        radiation(num,:)=[];
        weather(num,:)=[];
        day_blank=vertcat(day_blank,radiation(num,1)-1);
        num=num-1;
    end
    num=num+1;
end

day=1;
while day<=numel(date_weather)/5
    num=1;
    while num<=numel(day_blank)
        if date_weather(day,1)+datenum('30-Dec-1899')==day_blank(num,1);
            date_weather(day,:)=[];
            num=num-1;
        end
    num=num+1;
    end
    day=day+1;
end


%シミュレーション期間の決定, データのある14日間とする
%期間：2013年6月19日から7月17日まで
day=1;
while day<=numel(date_weather)/5
    if date_weather(day,1)+datenum('30-Dec-1899')<datenum('19-Jun-2013') || date_weather(day,1) ↙
+datenum('30-Dec-1899')>datenum('17-Jul-2013')
            date_weather(day,:)=[];
            day=day-1;
    end
    day=day+1;
end

%負荷3秒値の読み込み
demand_100V_3sec=xlsread('load_3sec.xlsx');
demand_200V_3sec=xlsread('load_3sec.xlsx','200V');
demand_3sec=demand_100V_3sec+demand_200V_3sec;
demand_3sec(1,:)=demand_3sec(1,:)/2;
%コピー
num=1;
```

```
while num<numel(demand_3sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1)
    demand(num+2*(num-1),:)=demand_3sec(num,:);
    demand(num+2*num-1,:)=demand_3sec(num,:);
    demand(num+2*num,:)=demand_3sec(num,:);
    num=num+1;
end
%最後の1分間はデータがないので，最後のデータをコピー
for t=1:63
    demand(86340+t,:)=demand(86340,:);
end

dem_size=(numel(demand)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1));
num=1;
while num<=dem_size
    if num>1
        %20%の誤差を載せる
        demand(num,:)=demand(num,:)*(1+0.4*(-0.5+rand));
    end
    num=num+1;
end
demand(2,:)=[];
demand(2,:)=[];


%PVデータの作成
%日射量10秒データの読み込み
radiation_10sec=xlsread('rad_10sec.xlsx');
%コピー
num=1;
while num<numel(radiation_10sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1)
    radiation_1sec(1+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(2+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(3+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(4+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(5+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(6+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(7+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(8+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(9+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(10+(num-1)*10,:)=radiation_10sec(num,:);
    num=num+1;
end
%最後の10秒は前のデータを利用
radiation_1sec(1+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(2+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(3+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(4+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(5+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(6+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(7+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(8+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(9+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(10+(num-1)*10,:)=radiation_10sec(num-1,:);

%20%の誤差を載せる
```



```
rad_size=(numel(radiation_1sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1));
num=1;
while num<=rad_size
    if num>1
        %20%の誤差を載せる
        radiation_1sec(num,:)=radiation_1sec(num,:)*(1+0.4*(-0.5+rand));
    end
    num=num+1;
end
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];

%PV出力に変換
%130.9:20kWパネルの面積(m2)，0.15:パネルの変換効率
PV_1sec=130.9*PCS*0.15*radiation_1sec;
PV_1sec(1,:)=PV_1sec(1,:)/(130.9*PCS*0.15);
PV_act=PV_1sec;
PV_act(1,:)=[];
%データを修正 負の値および定格を超える値を除去
for day=1:datenum('17-Jul-2013')-datenum('19-Jun-2013')
    for time=1:(24*60*60)
        if PV_act(time,day)<0
            PV_act(time,day)=0;
        elseif PV_act(time,day)>20000
            PV_act(time,day)=20000;
        end
    end
end

%シミュレーション期間外のデータを排除
num=1;
while num<=numel(PV_1sec)/((24*60*60)+1)
    flag=0;
    day=1;
    while day<=numel(date_weather)/5
        if PV_1sec(1,num)==date_weather(day,1)
            flag=1;
        end
        day=day+1;
    end
    if flag==0
        demand(:,num)=[];
        PV_1sec(:,num)=[];
        PV_act(:,num)=[];
        num=num-1;
    end
    num=num+1;
end
```

```
220
221 %%0時から始まっているデータを午前6時にずらす
222 %負荷に関しては，最初の日の0:00~6:00のデータを最後の日の0:00~6:00のデータとして扱う
223 %PVに関しては，夜間は出力がないので，0のままつなげる
224 demand_copy=demand;
225 PV_1sec_copy=PV_1sec;
226 PV_act_copy=PV_act;
227 %負荷およびPV1secに関しては，最初のデータを削除(日付)
228 demand_copy(1,:)=[];
229 PV_1sec_copy(1,:)=[];
230 demand_copy_size=size(demand_copy);
231 PV_1sec_copy_size=size(PV_1sec_copy);
232 PV_act_copy_size=size(PV_act_copy);
233 %ベクトル化
234 demand_copy=reshape(demand_copy,[],1);
235 PV_1sec_copy=reshape(PV_1sec_copy,[],1);
236 PV_act_copy=reshape(PV_act_copy,[],1);
237 %負荷の最初の６時間を最後の６時間に接続
238 time=1:6*60*60;
239 demand_6h=demand_copy(time,1);
240 demand_copy=vertcat(demand_copy,demand_6h);
241 time=1:6*60*60;
242 demand_copy(time)=[];
243 %PVに関しても同様の操作
244 time=1:6*60*60;
245 PV_1sec_6h=PV_1sec_copy(time,1);
246 PV_1sec_copy=vertcat(PV_1sec_copy,PV_1sec_6h);
247 time=1:6*60*60;
248 PV_1sec_copy(time)=[];
249 time=1:6*60*60;
250 PV_act_6h=PV_act_copy(time,1);
251 PV_act_copy=vertcat(PV_act_copy,PV_act_6h);
252 time=1:6*60*60;
253 PV_act_copy(time)=[];
254
255 %ベクトルから日付別に戻す
256 demand_copy=reshape(demand_copy,demand_copy_size);
257 PV_1sec_copy=reshape(PV_1sec_copy,PV_1sec_copy_size);
258 PV_act_copy=reshape(PV_act_copy,PV_act_copy_size);
259
260 time=2:24*60*60+1;
261 demand(time,:)=[];
262 time=2:24*60*60+1;
263 PV_1sec(time,:)=[];
264
265 demand=vertcat(demand,demand_copy);
266 PV_1sec=vertcat(PV_1sec,PV_1sec_copy);
267 PV_act=PV_act_copy;
268
269 %EVの運行予約に関して
270 EV_30min=EV;
271 time=1:12;
272 EV_30min_6h=EV(time,:);
273 time=1:12;
274 EV_30min(time,:)=[];
```



```
275 EV_30min=vertcat(EV_30min,EV_30min_6h);
276
277
278 %負荷30分平均値の作成
279 %平日と休日に分ける
280 day=1;
281 day_week=0;
282 day_hol=0;
283 demand_week=zeros(24*60*60+1,1);
284 demand_hol=zeros(24*60*60+1,1);
285 while day<numel(demand)/(24*60*60+1)
286     if weekday(demand(1,day)+datenum('30-Dec-1899'))==1 || weekday(demand(1,day)+datenum('30-Dec-1899'))==7
287         demand_hol=demand_hol+demand(:,day);
288         day_hol=day_hol+1;
289     else
290         demand_week=demand_week+demand(:,day);
291         day_week=day_week+1;
292     end
293     day=day+1;
294 end
295 demand_week=demand_week/day_week;
296 demand_hol=demand_hol/day_hol;
297
298 aaa=demand_week(2:86401);
299 buff=reshape(aaa,1800,48);
300 load_week=sum(buff)'/1800;
301
302 aaa=demand_hol(2:86401);
303 buff=reshape(aaa,1800,48);
304 load_hol=sum(buff)'/1800;
305
306 %%ここからシミュレーション開始
307 PV_rjc_cnt=0;
308 PV_rjc=0;
309 day=1;
310 while day<=numel(date_weather)/5
311     %1日目のみSOCが50%からスタート
312     %後は最終値を引き継ぎ
313     if day==1
314 %         ene_batt_act(1,1)=ene_batt_init;
315 %         ene_EV_act(1,1)=ene_EV_init;
316     else
317         ene_batt_init=ene_batt_act(24*60*60,day-1);
318         ene_EV_init=ene_EV_act(24*60*60,day-1);
319     end
320
321     %EV運用に関して
322     %EVの運行予約に関して
323     EV_30min=EV;
324     time=1:12;
325     EV_30min_6h=EV(time,:);
326     time=1:12;
327     EV_30min(time,:)=[];
328     EV_30min=vertcat(EV_30min,EV_30min_6h);
```

```
329     %休日は常時接続
330     if weekday(date_weather(day,1)+datenum('30-Dec-1899'))==1 || weekday(date_weather(day,1) ↙
+datenum('30-Dec-1899'))==7
331             EV_30min(:,1)=1;
332             EV_30min(:,2)=0;
333     end
334     num=1;
335     while num<=numel(EV)/2
336         EV_30min_buff=EV(num,:);
337         if weekday(date_weather(day,1)+datenum('30-Dec-1899'))==1 || weekday(date_weather(day,1) ↙
+datenum('30-Dec-1899'))==7
338             EV_30min_buff(:,1)=1;
339             EV_30min_buff(:,2)=0;
340         end
341         EV_1sec_buff=EV_30min_buff;
342         for t=1:1799
343             EV_1sec_buff=vertcat(EV_1sec_buff,EV_30min_buff);
344         end
345         if num==1
346             EV_1sec=EV_1sec_buff;
347         else
348             EV_1sec=vertcat(EV_1sec,EV_1sec_buff);
349         end
350         num=num+1;
351     end
352
353     num=1;
354     while num<numel(EV_1sec)/2-1
355         if (EV_1sec(num,1)~=1) || (EV_1sec(num+1,1)~=0)
356             EV_1sec(num+1,2)=0;
357 %         if (rem(num,11)~=0)
358 %             EV_1sec(num,2)=0;
359 %         end
360         end
361         num=num+1;
362     end
363
364     P_demand=demand(:,day);
365     P_demand(1)=[];
366     if weekday(date_weather(day,1)+datenum('30-Dec-1899'))==1 || weekday(date_weather(day,1) ↙
+datenum('30-Dec-1899'))==7
367         PL=load_hol;
368     else
369         PL=load_week;
370     end
371     date_act=date_weather(day,1)+datenum('30-Dec-1899');
372     weather_num=date_weather(day,2:5);
373     %翌日の天気予報(シミュレーションのための仮データ)
374     weather_num_tmrw=date_weather(day,2);
375     %計画
376     %蓄電池およびEVの計画出力を決める(30分値)
377     %PVシナリオの作成
378     scenario_2_fast;
379     %出てきたシナリオを６時間ずらす
380     time=1:12;
```



```
381     PV_6h=PV(time,:);
382     time=1:12;
383     PV(time,:)=[];
384     PV=vertcat(PV,PV_6h);
385     weather_pre_day(:,:)=weather_pre;
386     Piecewise_prob_from_6;
387     CO2_est(day,1)=fval;
388     batt_C_sim(:,day)=batt_C;
389     batt_D_sim(:,day)=batt_D;
390     EV_C_sim(:,day)=EV_C;
391     EV_D_sim(:,day)=EV_D;
392     %1秒値の指令値に変換
393     buff_batt_C=reshape(batt_C_sim(:,day),1,T);
394     buff_batt_D=reshape(batt_D_sim(:,day),1,T);
395     buff_EV_C=reshape(EV_C_sim(:,day),1,T);
396     buff_EV_D=reshape(EV_D_sim(:,day),1,T);
397     for t=1:(60*30)-1
398         buff_batt_C=vertcat(buff_batt_C,reshape(batt_C_sim(:,day),1,T));
399         buff_batt_D=vertcat(buff_batt_D,reshape(batt_D_sim(:,day),1,T));
400         buff_EV_C=vertcat(buff_EV_C,reshape(EV_C_sim(:,day),1,T));
401         buff_EV_D=vertcat(buff_EV_D,reshape(EV_D_sim(:,day),1,T));
402     end
403     %コピー
404     P_PV=PV_act(:,day);
405     P_batt_C_opt(:,day)=reshape(buff_batt_C,T*30*60,1);
406     P_batt_D_opt(:,day)=reshape(buff_batt_D,T*30*60,1);
407     P_EV_C_opt(:,day)=reshape(buff_EV_C,T*30*60,1);
408     P_EV_D_opt(:,day)=reshape(buff_EV_D,T*30*60,1);
409     P_batt_opt=P_batt_D_opt-P_batt_C_opt;
410     P_EV_opt=P_EV_D_opt-P_EV_C_opt;
411     %実際のPVデータを用いたシミュレーション
412     L_L_act=((V_DC_ref*V_DC_ref/R_L+2*P_demand/DC_DC) ...
413         -sqrt(...
414         (V_DC_ref*V_DC_ref/R_L+2*P_demand/DC_DC).*(V_DC_ref*V_DC_ref/R_L+2*P_demand/DC_DC) ...
415         -4*(P_demand.*P_demand/(DC_DC*DC_DC) ...
416         )))/2;
417 %     L_L_act=((V_DC_ref*V_DC_ref/R_L-2*P_demand/DC_DC)...
418 %         -sqrt(...
419 %         (V_DC_ref*V_DC_ref/R_L-2*P_demand/DC_DC).*(V_DC_ref*V_DC_ref/R_L-2*P_demand/DC_DC)...
420 %         -4*(P_demand.*P_demand/(DC_DC*DC_DC)...
421 %         )))/2;
422     P_demand_net=P_demand/DC_DC+L_L_act;
423     L_PV_act=((2*PV_act+V_DC_ref*V_DC_ref/R_PV)-sqrt((2*PV_act+V_DC_ref*V_DC_ref/R_PV).* ↙
(2*PV_act+V_DC_ref*V_DC_ref/R_PV)-4*PV_act))/2;
424     P_load(:,day)=P_demand_net;
425     %%岩見作成部分
426     EV_now=EV_1sec;
427     for t=1:24*60*60
428         if P_batt_opt(t)>=0
429             P_plan_batt(t)=P_batt_opt(t)*eff_batt;
430         else
431             P_plan_batt(t)=P_batt_opt(t)/eff_batt;
432         end
433         if P_EV_opt(t)>=0
434             P_plan_EV(t)=P_EV_opt(t)*eff_EV;
```

```
435         else
436             P_plan_EV(t)=P_EV_opt(t)/eff_EV;
437         end
438     end
439     P_plan_batt_record=P_plan_batt;
440     P_plan_EV_record=P_plan_EV;
441
442     new_P_plan_batt=P_plan_batt;
443     new_P_plan_EV=P_plan_EV;
444     P_conv_buff=zeros(T,1);
445
446     P_batt=zeros(T,1);
447     P_EV=zeros(T,1);
448     P_batt=P_plan_batt;
449     P_EV=P_plan_EV;
450     I_PV=P_PV/380;
451     I_batt=P_plan_batt/380;
452     I_EV=P_plan_EV/380;   %最適化データの読み込み、作成
453     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
454
455     %%%%%%%%%%%%%%%%%%%%%%%%各機器の運転状態の宣言%%%%%%%%%%%%%%%%%%%%%%%%
456     %%%%%%%%%%%%%%%%%1:CONV 2:蓄電池 3:EV 4:PV%%%%%%%%%%%%%%%%%%%%%%%%%%%%
457     %%%%%%%1:電圧制御 2:LOCK, MPPT(PVのみ)  3:計画運転 4:走行 (EVのみ) %%%%%%%%
458     mode=zeros(T,4);
459
460     %初期状態の決定
461     mode(1,:)=[1 3 3 2];
462
463     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
464
465     counttei=0;
466     overcount=0;
467
468     oldPpv=P_PV;
469     oldIpv=I_PV;
470
471     %直流バス電圧目標値
472     V_DC_ref=380;
473     %CONV出力電力を宣言
474     P_conv=zeros(24*60*60,1);
475
476     P_PV_rest=zeros(24*60*60,1);
477
478     %蓄電池およびEVのロックフラグの宣言
479     lock_batt_dch=0; lock_batt_ch=0;
480     lock_EV_dch=0; lock_EV_ch=0;
481     PCM_flag=0; VDC_flag=0;
482
483     for cnt=1:24*60*60
484         %%%%%%%%%%%%%%%%%状態に応じた制御の実行%%%%%%%%%%%%
485         %%まず現在の値から指令値を与える
486         I_batt_buff=I_batt(cnt);
487         I_EV_buff=I_EV(cnt);
488
489         I_conv(cnt)=(mode(cnt,1)==2)*I_conv_min;
```



```
490         I_batt(cnt)=(mode(cnt,2)==2 & lock_batt_dch==1)*I_batt_max+(mode(cnt,2)==2 & ↙
lock_batt_ch==1)*I_batt_min...
491             +(mode(cnt,2)==3)*I_batt(cnt);
492         I_EV(cnt)=(mode(cnt,3)==2 & lock_EV_dch==1)*I_EV_max+(mode(cnt,3)==2 & lock_EV_ch==1) ↙
*I_batt_min...
493             +(mode(cnt,3)==3)*I_EV(cnt)+(mode(cnt,3)==4)*0;
494         if count_flag==0
495         %計画運転モードからの遷移の場合、運用計画修正開始
496             if cnt>=2 && mode(cnt,1)-mode(cnt-1,1)==1
497                 Modification_flag=1;
498                 count=0; count_flag=1;
499             end
500
501 %     if update_flag==0
502 %         %午前11時の発表がでた時点（すなわち、運用開始より6時間後）になった場合も修正開始
503 %         if cnt>18000
504 %             Modification_flag=1;
505 %             count=0; count_flag=1; update_flag=1;
506 %             disp('午前11時。天気予報更新');
507 %         end
508 %     end
509         end
510         switch find(mode(cnt,:)==1)
511             case 1
512                 conv_regulate_2;
513             case 2
514                 batt_regulate_2;
515             case 3
516                 EV_regulate_2;
517             case 4
518                 PV_regulate_2;
519         end
520
521         for cnt_cnt=1:4
522             %%ハードウェアの制約を超える場合
523             %%蓄電池放電レート逸脱
524             if lock_batt_dch==1 && lock_EV_dch==0
525                 %%モード変更して再計算
526                 mode(cnt,2)=2;
527                 mode(cnt,1)=1;
528                 conv_regulate_2;
529             %%蓄電池充電レート逸脱
530             elseif lock_batt_ch==1 && lock_EV_ch==0
531                 %%モード変更して再計算
532                 mode(cnt,2)=2;
533                 if mode(cnt,3)==2
534                     PCM_flag=1;
535                     mode(cnt,1)=1;
536                     mode(cnt,4)=2;
537                     PV_regulate_2;
538                 else
539                     mode(cnt,3)=1;
540                     EV_regulate_2;
541                 end
542             %%EV放電レート逸脱
```

```
543             elseif lock_EV_dch==1 && lock_batt_dch==0
544                 %%モード変更して再計算
545                 mode(cnt,3)=2;
546                 mode(cnt,1)=1;
547                 conv_regulate_2;
548             %%EV充電レート逸脱
549             elseif lock_EV_ch==1 && lock_batt_ch==0
550                 mode(cnt,3)=2;
551                 %%モード変更して再計算
552                 if mode(cnt,2)==2
553                     PCM_flag=1;
554                     mode(cnt,1)=1;
555                     mode(cnt,4)=2;
556                     PV_regulate_2;
557                 else
558                     mode(cnt,2)=1;
559                     batt_regulate_2;
560                 end
561             %%PVレート逸脱
562             elseif lock_batt_ch==1 && lock_EV_ch==1 && PCM_flag==1 && VDC_flag==1
563 %             elseif PCM_flag==1 && VDC_flag==1
564                 %%蓄電池とEV停止
565                 I_batt(cnt)=0.0;
566                 I_EV(cnt)=0.0;
567                 %%モード変更して再計算
568                 mode(cnt,:)=[1 2 2 2];
569                 PV_rest_ctrl;
570 %                 mode(cnt,4)=2;
571 %                 PV_regulate_2
572             end
573         end
574
575         P_grid(cnt)=P_conv(cnt)/AC_DC;
576         SOC_calculate2; %この時間帯の消費電力量、SOCを計算
577
578 %         if cnt~=24*60*60
579 %             switch  find(mode(cnt,:)==1)
580 %
581 %         %%%%%%%%%%%%%%%%%%%%次時間帯の状態を決定%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
582 %
583 %                 case 1 %SOM滞在時
584 %                     %逆潮流発生の可能性
585 %                     if P_conv(cnt)<=P_conv_min
586 %                         %SOCをチェックして、どちらもMAXであればPCMへ遷移
587 %                        if SOC_batt(cnt)>=SOC_batt_max && (SOC_EV_act(cnt)>=SOC_EV_max || EV_now ↙
(cnt)==0)
588 %                             mode(cnt,:)=[2 2 2 1];
589 %                             mode_before=mode(cnt,:); %PCM遷移前の状態を保存
590 %                         else
591 %                             %電圧制御機器の決定
592 %                             if mode(cnt,3)~=4
593 %                                 %充電可能電力を比較
594 %                                 if (P_batt_max+P_batt(cnt))-(P_EV_max+P_EV(cnt))>=0;
595 %                                     mode(cnt,:)=[2 1 3 2];
596 %                                 else
```



```
597 %                                     mode(cnt,:)=[2 3 1 2];
598 %                                 end
599 %
600 %                             else
601 %                                 mode(cnt,:)=[2 1 4 2];
602 %                             end
603 %                         end
604 % %                     else
605 % %                         %条件に抵触していない場合
606 % %                         mode(cnt+1,:)=mode(cnt,:);
607 %                     end
608 %         %-----------------------------------------------------------------------
609 %                 case 2 %CLM1滞在時
610 %
611 %                     %コンバータロックモードから計画運転モードへの遷移判定
612 %                     %計画修正にかかった時間を超えていれば判定開始
613 %                     if count_flag==1 && count>=time_Piecewise && ...
614 %                             img(I_conv(cnt-1), new_P_plan_batt(cnt)/380,I_batt(cnt-1), (EV_now ↙
(cnt)==1)*(new_P_plan_EV(cnt)/380),I_EV(cnt-1), I_PV(cnt), P_load(cnt), SOC_batt(cnt), SOC_EV(cnt)) ↙
>=500;
615 %                       %閾値を超えている場合、計画運転モードへ復帰
616 %                         mode(cnt,:)=[1 3 3 2];
617 %                         count_flag=0;
618 %                         count=0;
619 %                         %充電レートフラグのリセット
620 %                         lock_batt_ch=0; lock_EV_ch=0;
621 %                     else
622 %                         %超えていない場合
623 %                         %蓄電池のSOCを確認、MAXの場合
624 %                         if SOC_batt(cnt)>=SOC_batt_max
625 %                             %EVのSOCを確認、MAXの場合PCMへ遷移
626 %                             if SOC_EV(cnt)>=SOC_EV_max || mode(cnt,3)==4
627 %                                 mode(cnt,:)=[2 2 2 1];
628 %                                 mode_before=mode(cnt,:);
629 %                             else
630 %                                 %余裕がある場合、CLM2へ遷移
631 %                                 mode(cnt,:)=[2 2 1 2];
632 %                             end
633 %                         elseif lock_batt_dch==1
634 %                             %放電レート抵触の場合はSOMへ戻る
635 %                             mode(cnt,:)=[1 3 3 2];
636 %                             lock_batt_dch=0;
637 %                         elseif lock_batt_ch==1
638 %                             %充電レートの場合は、EVがロックされていなければCLM2へ遷移
639 %                             if  mode(cnt,3)==3
640 %                                 mode(cnt,:)=[2 2 1 2];
641 %                             else
642 %                                 mode(cnt,:)=[2 2 2 1];
643 %                             end
644 % %                         else
645 % %                             %CLM1を維持
646 % %                             mode(cnt+1,:)=mode(cnt,:);
647 %                         end
648 %                     end
649 %         %-----------------------------------------------------------------------
```

```
650 %                 case 3 %CLM2滞在時
651 %
652 %                     %コンバータロックモードから計画運転モードへの遷移判定
653 %                     %計画修正にかかった時間を超えていれば判定開始
654 %                     if count_flag==1 && count>=time_Piecewise && ...
655 %                             img(I_conv(cnt-1), new_P_plan_batt(cnt)/380,I_batt(cnt-1), (EV_now ↙
(cnt)==1)*(new_P_plan_EV(cnt)/380),I_EV(cnt-1), I_PV(cnt), P_load(cnt), SOC_batt(cnt), SOC_EV(cnt)) ↙
>=500;
656 %                       %閾値を超えていれば計画運転モードへ復帰
657 %                         mode(cnt,:)=[1 3 3 2];
658 %                         count_flag=0;
659 %                         count=0;
660 %                         lock_batt_ch=0; lock_EV_ch=0;
661 %                     else
662 %                         %CLM2の最中にEVが走行してしまったら
663 %                         if mode(cnt,3)==4
664 %                             %蓄電池のSOCに余裕がある場合はCLM1に遷移
665 %                             if SOC_batt(cnt)<SOC_batt_max;
666 %                                 mode(cnt,:)=[2 1 4 2];
667 %                             else
668 %                                 %余裕がない場合はPCMへ遷移
669 %                                 mode(cnt,:)=[2 2 4 1];
670 %                                 mode_before=mode(cnt,:);
671 %                             end
672 %                         elseif SOC_EV(cnt)>=SOC_EV_max
673 %                             %EVのSOCがMAXになった場合
674 %                             %蓄電池のSOCを確認、MAXの場合はPCMへ遷移
675 %                             if SOC_batt(cnt)>=SOC_batt_max
676 %                                 mode(cnt,:)=[2 2 2 1];
677 %                                 mode_before=mode(cnt,:);
678 %                             else
679 %                                 %余裕がある場合は、CLM1へ遷移
680 %                                 mode(cnt,:)=[2 1 2 2];
681 %                             end
682 %                         elseif lock_EV_dch==1
683 %                             %放電レート抵触の場合はSOMへ戻る
684 %                             mode(cnt,:)=[1 3 3 2];
685 %                         elseif lock_EV_ch==1
686 %                              %充電レートの場合は、蓄電池がロックされていなければCLM1へ遷移
687 %                             if mode(cnt,2)~=2
688 %                                 mode(cnt,:)=[2 1 2 2];
689 %                             else
690 %                                 mode(cnt,:)=[2 2 2 1];
691 %                             end
692 % %                         else
693 % %                             %何もない場合はそのまま滞在
694 % %                             mode(cnt+1,:)=mode(cnt,:);
695 %                         end
696 %                     end
697 %         %-------------------------------------------------------------------------
698 %                 case 4 %PCM滞在時
699 %
700 %                     %PCMから他モードへの遷移判定
701 %                     if SOC_batt(cnt)>SOC_batt_max && lock_batt_ch==1
702 %                         lock_batt_ch=0;
```



```
703 %                     end
704 %                     if SOC_EV(cnt)>SOC_EV_max && lock_EV_ch==1
705 %                         lock_EV_ch=0;
706 %                     end
707 %                     if PCM_flag==1 %供給力不足となった場合
708 %                         %PCMに遷移する前に滞在していたモードに復帰
709 %
710 %                         mode(cnt,:)=[1 3 3 2];
711 %                         lock_batt_ch=0; lock_EV_ch=0;
712 %                     else
713 %                         %そのままPCMを継続
714 %                         mode(cnt,:)=mode(cnt,:);
715 %                     end
716 %
717 %
718 %
719 %         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
720 %
721 %             end
722 %         end
723         count=(count_flag==1)*(count+1);
724         %%ここまで現在のモードを決めるやつ
725
726
727         %%ここから次の時間帯のモードを決める
728 %         I_batt_buff=I_batt(cnt);
729 %         I_EV_buff=I_EV(cnt);
730 %
731 %         I_conv(cnt)=(mode(cnt,1)==2)*I_conv_min;
732 %         I_batt(cnt)=(mode(cnt,2)==2 & lock_batt_dch==1)*I_batt_max+(mode(cnt,2)==2 & ↙
lock_batt_ch==1)*I_batt_min...
733 %             +(mode(cnt,2)==3)*I_batt(cnt);
734 %         I_EV(cnt)=(mode(cnt,3)==2 & lock_EV_dch==1)*I_EV_max+(mode(cnt,3)==2 & lock_EV_ch==1) ↙
*I_batt_min...
735 %             +(mode(cnt,3)==3)*I_EV(cnt)+(mode(cnt,3)==4)*0;
736 %
737 %         switch find(mode(cnt,:)==1)
738 %             case 1
739 %                 conv_regulate;
740 %             case 2
741 %                 batt_regulate;
742 %             case 3
743 %                 EV_regulate;
744 %             case 4
745 %                 PV_regulate;
746 %         end
747 %
748 %         P_grid(cnt)=P_conv(cnt)/AC_DC;
749 %         SOC_calculate2; %この時間帯の消費電力量、SOCを計算
750         if cnt~=24*60*60
751             switch  find(mode(cnt,:)==1)
752
753             %%%%%%%%%%%%%%%%%%%%次時間帯の状態を決定%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
754
755             case 1 %SOM滞在時
```

```
                %逆潮流発生の可能性
                if P_conv(cnt)<=P_conv_min
                    %SOCをチェックして、どちらもMAXであればPCMへ遷移
                    if SOC_batt(cnt)>=SOC_batt_max && (SOC_EV(cnt)>=SOC_EV_max || EV_now(cnt)==0)
                        mode(cnt+1,:)=[2 2 2 1];
                        mode_before=mode(cnt,:);  %PCM遷移前の状態を保存
                    else
                        %電圧制御機器の決定
                        if mode(cnt,3)~=4
                            %充電可能電力を比較
                            if (P_batt_max+P_batt(cnt))-(P_EV_max+P_EV(cnt))>=0;
                                mode(cnt+1,:)=[2 1 3 2];
                            else
                                mode(cnt+1,:)=[2 3 1 2];
                            end

                        else
                            mode(cnt+1,:)=[2 1 4 2];
                        end
                    end
                else
                    %条件に抵触していない場合
                    mode(cnt+1,:)=mode(cnt,:);
                end
    %-------------------------------------------------------------------
            case 2 %CLM1滞在時

                %コンバータロックモードから計画運転モードへの遷移判定
                %計画修正にかかった時間を超えていれば判定開始
                if count_flag==1 && count>=time_Piecewise && ...
                        img(I_conv(cnt-1), new_P_plan_batt(cnt)/380,I_batt(cnt-1), (EV_now(cnt) ↙
==1)*(new_P_plan_EV(cnt)/380),I_EV(cnt-1), I_PV(cnt), P_load(cnt), SOC_batt(cnt), SOC_EV(cnt))>=500;
                  %閾値を超えている場合、計画運転モードへ復帰
                    mode(cnt+1,:)=[1 3 3 2];
                    count_flag=0;
                    count=0;
                    %充電レートフラグのリセット
                    lock_batt_ch=0; lock_EV_ch=0;
                else
                    %超えていない場合
                    %蓄電池のSOCを確認、MAXの場合
                    if SOC_batt(cnt)>=SOC_batt_max
                        %EVのSOCを確認、MAXの場合PCMへ遷移
                        if SOC_EV(cnt)>=SOC_EV_max || mode(cnt,3)==4
                            mode(cnt+1,:)=[2 2 2 1];
                            mode_before=mode(cnt,:);
                        else
                            %余裕がある場合、CLM2へ遷移
                            mode(cnt+1,:)=[2 2 1 2];
                        end
                    elseif lock_batt_dch==1
                        %放電レート抵触の場合はSOMへ戻る
                        mode(cnt+1,:)=[1 3 3 2];
                        lock_batt_dch=0;
                    elseif lock_batt_ch==1
```



```
                        %充電レートの場合は、EVがロックされていなければCLM2へ遷移
                        if mode(cnt,3)==3
                            mode(cnt+1,:)=[2 2 1 2];
                        else
                            mode(cnt+1,:)=[2 2 2 1];
                        end
                    else
                        %CLM1を維持
                        mode(cnt+1,:)=mode(cnt,:);
                    end
                end
    %-------------------------------------------------------------------
            case 3 %CLM2滞在時

                %コンバータロックモードから計画運転モードへの遷移判定
                %計画修正にかかった時間を超えていれば判定開始
                if count_flag==1 && count>=time_Piecewise && ...
                        img(I_conv(cnt-1), new_P_plan_batt(cnt)/380,I_batt(cnt-1), (EV_now(cnt) ↙
==1)*(new_P_plan_EV(cnt)/380),I_EV(cnt-1), I_PV(cnt), P_load(cnt), SOC_batt(cnt), SOC_EV(cnt))>=500;
                  %閾値を超えていれば計画運転モードへ復帰
                    mode(cnt+1,:)=[1 3 3 2];
                    count_flag=0;
                    count=0;
                    lock_batt_ch=0; lock_EV_ch=0;
                else
                    %CLM2の最中にEVが走行してしまったら
                    if mode(cnt,3)==4
                        %蓄電池のSOCに余裕がある場合はCLM1に遷移
                        if SOC_batt(cnt)<SOC_batt_max;
                            mode(cnt+1,:)=[2 1 4 2];
                        else
                            %余裕がない場合はPCMへ遷移
                            mode(cnt+1,:)=[2 2 4 1];
                            mode_before=mode(cnt,:);
                        end
                    elseif SOC_EV(cnt)>=SOC_EV_max
                        %EVのSOCがMAXになった場合
                        %蓄電池のSOCを確認、MAXの場合はPCMへ遷移
                        if SOC_batt(cnt)>=SOC_batt_max
                            mode(cnt+1,:)=[2 2 2 1];
                            mode_before=mode(cnt,:);
                        else
                            %余裕がある場合は、CLM1へ遷移
                            mode(cnt+1,:)=[2 1 2 2];
                        end
                    elseif lock_EV_dch==1
                        %放電レート抵触の場合はSOMへ戻る
                        mode(cnt+1,:)=[1 3 3 2];
                    elseif lock_EV_ch==1
                        %充電レートの場合は、蓄電池がロックされていなければCLM1へ遷移
                        if mode(cnt,2)~=2
                            mode(cnt+1,:)=[2 1 2 2];
                        else
                            mode(cnt+1,:)=[2 2 2 1];
                        end
```

```
                    else
                        %何もない場合はそのまま滞在
                        mode(cnt+1,:)=mode(cnt,:);
                    end
                end
    %-------------------------------------------------------------------------
            case 4 %PCM滞在時

                %PCMから他モードへの遷移判定
                if SOC_batt(cnt)>SOC_batt_max && lock_batt_ch==1
                    lock_batt_ch=0;
                end
                if SOC_EV(cnt)>SOC_EV_max && lock_EV_ch==1
                    lock_EV_ch=0;
                end
                if PCM_flag==1 %供給力不足となった場合
                    %PCMに遷移する前に滞在していたモードに復帰

                    mode(cnt+1,:)=[1 3 3 2];
                    lock_batt_ch=0; lock_EV_ch=0;
                else
                    %そのままPCMを継続
                    mode(cnt+1,:)=mode(cnt,:);
                end



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            end
        end
        count=(count_flag==1)*(count+1);
    end


%%岩見作成部分, ここまで

%     %1秒シミュレーション
%     %挙動は30分と同様
%     for time=1:24*60*60
%         %PVの余剰電力あり
%         if PV_act(time,day)-L_PV_act(time,day)>P_demand_net(time)
%             %蓄電池SOCの余裕計算
%             SOC_batt_act_buff=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)+(1/ ↙
(60*60))*(PV_act(time,day)-L_PV_act(time,day)-P_demand_net(time));
%             %蓄電池SOC余裕なしのとき
%             if SOC_batt_act_buff>SOC_batt_max_hardware
%                 %蓄電池SOCのギリギリまで充電
%                 if (60*60)*(SOC_batt_max_hardware-SOC_batt_act(time,day))<10000
%                     batt_C_act(time,day)=(60*60)*(SOC_batt_max_hardware-SOC_batt_act(time,day));
%                 else
%                     batt_C_act(time,day)=10000;
%                 end
%                 %EVの放電を停止
% %                 EV_C_act(time)=0.0;
%                 EV_D_act(time,day)=0.0;
```



```
%                 %蓄電池およびEVの線路損失を計算
%     %                 L_batt_act(time,day)=(2*batt_C_act(time,day)/(eff_batt/AC_DC)+ ↙
(V_DC*V_DC/R_batt))-sqrt((2*batt_C_act(time,day)/(eff_batt/AC_DC)+(V_DC*V_DC/R_batt))*((2*batt_C_act ↙
(time,day)/(eff_batt/AC_DC)+(V_DC*V_DC/R_batt))-4*(AC_DC/eff_batt)*(AC_DC/eff_batt)*batt_C_act(time, ↙
day)*batt_C_act(time,day)))/2;
%                 L_batt_act(time,day)=((...
%                                         2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt)...
%                                        )-...
%                                        sqrt(...
%                                        (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))*...
%                                        (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))-...
%                                        4*(1/eff_batt)*(1/eff_batt)*batt_C_act(time,day) ↙
*batt_C_act(time,day))...
%                                        )/2;
%                 L_EV_act(time,day)=((...
%                                       2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))-...
%                                       sqrt(...
%                                       (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))*...
%                                       (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))-...
%                                       4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)))...
%                                       )/2;
%                 %PV出力を抑制する
%                 PV_rjc_cnt=PV_rjc_cnt+1;
%                 PV_rjc=PV_rjc+PV_act(time,day)-(P_demand_net(time)+L_PV_act(time,day)+ ↙
(batt_C_act(time,day)+L_batt_act(time,day))/(eff_batt)+(EV_C_act(time,day)+L_EV_act(time,day))/ ↙
(eff_EV));
%                 PV_act(time,day)=P_demand_net(time)+L_PV_act(time,day)+(batt_C_act(time,day) ↙
+L_batt_act(time,day))/(eff_batt)+(EV_C_act(time,day)+L_EV_act(time,day))/(eff_EV);
% %                 PV_rjc=PV_rjc+PV_act(time,day)-(P_demand_net(time)+L_PV_act(time,day)+ ↙
(batt_C_act(time,day)+L_batt_act(time,day))/(eff_batt/AC_DC));
% %                 PV_act(time,day)=P_demand_net(time)+L_PV_act(time,day)+(batt_C_act(time,day) ↙
+L_batt_act(time,day))/(eff_batt/AC_DC);
%                 P_UG(time,day)=0.0;
%             %蓄電池SOC余裕ありの場合
%             else
%                 batt_C_act(time,day)=(eff_batt)*(PV_act(time,day)-L_PV_act(time,day)- ↙
P_demand_net(time));
%                 L_batt_act(time,day)=((...
%                                         2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt)...
%                                        )-...
%                                        sqrt(...
%                                        (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))*...
%                                        (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))-...
%                                        4*(1/eff_batt)*(1/eff_batt)*batt_C_act(time,day) ↙
*batt_C_act(time,day))...
%                                        )/2;
%                 batt_D_act(time,day)=0.0;
%                 L_EV_act(time,day)=((...
%                                       2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))-...
%                                       sqrt(...
```

```
%                                   (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))*...
%                                   (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))+(V_DC*V_DC/R_EV))-...
%                                   4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)* ↙
(eff_EV))*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)))...
%                                   )/2;
%                   P_UG_act(time,day)=0.0;
%               end
%           %余剰電力なしの場合は、購入電力および蓄電池で調整
%           else
%               %逆潮流が起きそうなときは蓄電池放電をやめる
%               if EV_D_act(time,day)*(eff_EV)+batt_D_act(time,day)*(eff_EV)+PV_act(time,day) ↙
>P_demand_net(time)+L_PV_act(time)+EV_C_act(time,day)/(eff_EV)+batt_C_act(time,day)/(eff_EV)
%                   P_UG_act(time,day)=0.0;
%       %               L_L_act(time,day)=0.0;
%                   batt_D_act(time,day)=(P_demand_net(time)-PV_act(time,day)-L_PV_act(time,day)- ↙
EV_D_act(time,day)*(eff_EV))/(eff_batt);
%                   %まだ逆潮流が発生する場合にはEVからの放電を減らす
%                   if batt_D_act(time,day)<0
%                       EV_D_act(time,day)=EV_D_act(time,day)*(eff_EV)+(DC_DC)*batt_D_act(time,day);
%                       batt_D_act(time,day)=0.0;
%                   end
%                   L_batt_act(time,day)=((...
%                         2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt)...
%                       )-...
%                       sqrt(...
%                       (2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))*...
%                       (2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))-...
%                       4*(eff_batt)*(eff_batt)*batt_D_act(time,day)*batt_D_act(time,day))...
%                       )/2;
%                   L_EV_act(time,day)=((...
%                       2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
%                       sqrt(...
%                       (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))*...
%                       (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
%                       4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))*abs(EV_C_act ↙
(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)))...
%                       )/2;
%               else
%                   %SOCの下限に抵触しているかどうか
%                   SOC_batt_act_buff=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)-(1/ ↙
(60*60))*batt_D_act(time,day);
%                   if SOC_batt_act_buff<0
%                       batt_D_act(time,day)=0.0;
%                   end
%                   %SOCの上限に抵触しているかどうか
%                   if SOC_batt_act_buff>SOC_batt_max_hardware
%                       batt_C_act(time,day)=0.0;
%                   end
%                   SOC_EV_act_buff=SOC_EV_act(time,day)+(1/(60*60))*EV_C_act(time,day)-(1/(60*60)) ↙
*EV_D_act(time,day);
```



```
%                   %SOCの下限に抵触しているかどうか
%                   if SOC_EV_act_buff<0
%                       EV_D_act(time,day)=0.0;
%                   end
%                   %SOCの上限に抵触しているかどうか
%                   if SOC_EV_act_buff>SOC_EV_max_hardware
%                       EV_C_act(time,day)=0.0;
%                   end
%                   L_batt_act(time,day)=((...
%                      2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt)...
%                     )-...
%                     sqrt(...
%                     (2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt))*...
%                     (2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt))-...
%                     4*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))*abs ↙
(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt)))...
%                     )/2;
%                   L_EV_act(time,day)=((...
%                      2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
%                      sqrt(...
%                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))*...
%                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
%                      4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))*abs(EV_C_act ↙
(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)))...
%                      )/2;
%                  P_UG_act(time,day)=P_demand_net(time)+(batt_C_act(time,day)/(eff_batt)- ↙
batt_D_act(time,day)*(eff_batt))+(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))-PV_act(time, ↙
day)+L_PV_act(time,day);
%              end
%          end
%          SOC_batt_act(time+1,day)=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)-(1/ ↙
(60*60))*batt_D_act(time,day);
%          SOC_EV_act(time+1,day)=SOC_EV_act(time,day)+(1/(60*60))*EV_C_act(time,day)-(1/(60*60)) ↙
*EV_D_act(time,day)-110*EV_1sec(time,2);
%      end
%      SOC_batt_act(1,day+1)=SOC_batt_act(24*60*60+1,day);
%      SOC_EV_act(1,day+1)=SOC_EV_act(24*60*60+1,day);
    P_grid_act(:,day)=P_grid.';
    P_PV_act(:,day)=P_PV;
    P_batt_act(:,day)=P_batt.';
    P_EV_act(:,day)=P_EV.';
    P_PV_rest_act(:,day)=P_PV_rest;
    ene_batt_act(:,day)=ene_batt.';
    ene_EV_act(:,day)=ene_EV.';
    mode_act(:,:,day)=mode;
    day=day+1;
end

%期間でのCO2排出量の計算
```

```
day=1;
while day<=numel(date_weather)/5
    for time=1:24*60*60
        if time<60*60*12
            CO2(time,day)=0.69*P_grid_act(time,day)/((1000*60*60));
        else
            CO2(time,day)=0.469*P_grid_act(time,day)/((1000*60*60));
        end
    end
    CO2_day(1,day)=sum(CO2(:,day));
    day=day+1;
% for day=1:1
%     for time=1:24*60*60
%         if time<60*60*6
%             CO2(time,day)=0.469*P_UG_act(time,day)/((1000*60*60)*(AC_DC));
%         elseif time>=60*60*6||time<60*60*18
%             CO2(time,day)=0.69*P_UG_act(time,day)/((1000*60*60)*(AC_DC));
%         else
%             CO2(time,day)=0.469*P_UG_act(time,day)/((1000*60*60)*(AC_DC));
%         end
%     end
%     CO2_day(1,day)=sum(CO2(:,day));
%     day=day+1;
end
CO2_total=sum(CO2);
day=1;
while day<=numel(date_weather)/5
    for time=1:24*60*60
        L_L_act(time,day)=((V_DC_ref*V_DC_ref/R_L+2*demand(time+1,day)/(DC_DC))-sqrt ↙
((V_DC_ref*V_DC_ref/R_L+2*demand(time+1,day)/(DC_DC))*(V_DC_ref*V_DC_ref/R_L+2*demand(time+1,day)/ ↙
(DC_DC))-4*(demand(time+1,day)*demand(time+1,day)/((DC_DC)*(DC_DC)))))/2;
        L_UG_act(time,day)=((V_DC_ref*V_DC_ref/R_conv+2*P_grid_act(time,day)/(AC_DC))-sqrt ↙
((V_DC_ref*V_DC_ref/R_conv+2*P_grid_act(time,day)/(AC_DC))*(V_DC_ref*V_DC_ref/R_conv+2*P_grid_act(time, ↙
day)/(AC_DC))-4*(P_grid_act(time,day)*P_grid_act(time,day)/((AC_DC)*(AC_DC)))))/2;
    end
    day=day+1;
end
save('1sec_14days_from_6_iwami_algo');
% save('1sec_14days_from_6');
```

```
%%%%%%%%%%%%%%%%
% 区分線形近似により、線形計画法にする。
% その後、解く
% PV出力を期待値として扱う
%%%%%%%%%%%%%%%%

% %デバッグ用
% clear all
% load('matlab.mat')
% P_demand=load_hol;
% n=10;
% EV=zeros(T,2);
% time=1:T;
% EV(time,1)=1;
% time=1:T;
% EV(time,2)=0;

%weather dataを読み込み、PV出力の予測値を決定

%時間帯の数(1日を何時間帯に分けるか)
T=48;
%逆潮流期待値の上限
% P_rev=1000;


%メモリ不足対策のため、必要なデータ以外消去
clear ~.(PV,prob_pre_scenario)
clear Aeq beq A b ub lb x
clear Aeq_buff beq_buff A_buff b_buff ub_buff lb_buff
%決定変数の数
vr_num=T*9*n;



%% 線形計画問題の作成

% UG=1,UG_loss=2,PV抑制量=3,batt_C=4,batt_D=5,batt_loss=6,EV_C=7,EV_D=8,EV_loss=9

%下準備
%PVにおける線路損失
PV_loss=((2*PV+V_DC_ref*V_DC_ref/R_PV)-sqrt((2*PV+V_DC_ref*V_DC_ref/R_PV).* ↙
(2*PV+V_DC_ref*V_DC_ref/R_PV)-4*PV))/2;
%正味のPV出力
PV_net=PV-PV_loss;

%負荷における損失
PL_loss=((V_DC_ref*V_DC_ref)*(DC_DC*DC_DC)/R_L-2*PL-sqrt(((V_DC_ref*V_DC_ref)*(DC_DC*DC_DC)/R_L- ↙
2*PL).*((V_DC_ref*V_DC_ref)*(DC_DC*DC_DC)/R_L-2*PL)-4*PL.*PL))/2;
% PL_loss=((V_DC_ref*V_DC_ref/R_L-2*PL/DC_DC)-sqrt((V_DC_ref*V_DC_ref/R_L-2*PL/DC_DC).* ↙
(V_DC_ref*V_DC_ref/R_L-2*PL/DC_DC)-4*(PL.*PL/(DC_DC*DC_DC))))/2;
%正味の負荷
PL_net=PL/DC_DC+PL_loss;
PL_net=repmat(PL_net,1,n);

%目的関数
```



```
%昼間(午前6時~午後6時)の排出原単位 0.690kg/kWh、夜間の排出原単位 0.469kg/kWh
f=zeros(1,vr_num);
f_buff=zeros(1,T);
% f_buff=zeros(1,vr_num/(3^T_weather));
%系統からの購入電力によるCO2期待値の最小化
vr=1:T;
%6時からの計画に伴い変更
f_buff(1,vr)=((vr<13)*0.69+(vr>=13)*0.469)/(1000);
f=repmat(f_buff,1,n);
f=reshape(f,[T,n]).';
f=f.*repmat(prob_pre_scenario.',1,T);
f=reshape(f.',[T*n,1]).';
%系統購入電力以外の目的関数は零
%UG_lossについて
f=horzcat(f,zeros(1,T*n));
%PV抑制量について
f=horzcat(f,zeros(1,T*n));
%それ以外について
f=horzcat(f,zeros(1,T*6));

%%等式制約
%電力の需給一致制約
%UGからの購入電力
[time,vr]=meshgrid(1:T,1:T);
Aeq(time,vr)=1;
Aeq=repmat(Aeq,n,n);
Aeq=Aeq.*eye(T*n,T*n);

%UGの損失
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1.0;
Aeq_buff=repmat(Aeq_buff,n,n);
Aeq_buff=Aeq_buff.*eye(T*n,T*n);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;
%PV抑制量
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1.0;
Aeq_buff=repmat(Aeq_buff,n,n);
Aeq_buff=Aeq_buff.*eye(T*n,T*n);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%蓄電池充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1/(eff_batt);
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%蓄電池放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=(eff_batt);
Aeq_buff=Aeq_buff.*eye(T,T);
```

```
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%蓄電池による損失
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1;
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%EVの充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1/(eff_EV);
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%EVの放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=(eff_EV);
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%EVによる損失
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=-1;
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=repmat(Aeq_buff,n,1);
Aeq=horzcat(Aeq,Aeq_buff);
clear Aeq_buff;

%右辺
beq=reshape((PL_net-PV_net),T*n,1);

% %蓄電池SOCの初期値と最終値が一致
% clear Aeq_buff;
% clear beq_buff;
% Aeq_buff=0.5*ones(1,2*T);
% time=T+1:2*T;
% Aeq_buff(time)=-1*Aeq_buff(time);
% Aeq_buff=horzcat(zeros(1,n*T*3),Aeq_buff);
% Aeq_buff=horzcat(Aeq_buff,zeros(1,T*4));
% beq_buff=0.0;
% Aeq=vertcat(Aeq,Aeq_buff);
% beq=vertcat(beq,beq_buff);
% clear Aeq_buff;
% clear beq_buff;

% % % %蓄電池SOCの最終値SOCに関する制約
% % % %翌々日の天気予報が晴なら30%, 曇なら50%, 雨なら80%
```



```
% % % clear Aeq_buff;
% % % clear beq_buff;
% % % Aeq_buff=0.5*ones(1,2*T);
% % % time=T+1:2*T;
% % % Aeq_buff(time)=-1*Aeq_buff(time);
% % % Aeq_buff=horzcat(zeros(1,n*T*3),Aeq_buff);
% % % Aeq_buff=horzcat(Aeq_buff,zeros(1,T*4));
% % % % beq_buff=0.5*SOC_batt_max_hardware-SOC_batt_init;
% % % beq_buff=((weather_num_tmrw==1)*0.3+(weather_num_tmrw==2)*0.5+(weather_num_tmrw==3)*0.8) ↙
*SOC_batt_max-SOC_batt_init;
% % % Aeq=vertcat(Aeq,Aeq_buff);
% % % beq=vertcat(beq,beq_buff);
% % % clear Aeq_buff;
% % % clear beq_buff;

%EVのSOCの初期値と最終値が一致
% Aeq_buff=0.5*ones(1,2*T);
% time=T+1:2*T;
% Aeq_buff(time)=-1*Aeq_buff(time);
% Aeq_buff=horzcat(zeros(1,n*T*3+T*3),Aeq_buff);
% Aeq_buff=horzcat(Aeq_buff,zeros(1,T));
% beq_buff=110*sum(EV(:,2));
% Aeq=vertcat(Aeq,Aeq_buff);
% beq=vertcat(beq,beq_buff);
% clear Aeq_buff;
% clear beq_buff;

%EVのSOCの最終値に関する制約
%とりあえず, 満充電-3h分にするように制約
%翌日9時から運転するなら, ３時間しか充電できないため
% Aeq_buff=0.5*ones(1,2*T);
% time=T+1:2*T;
% Aeq_buff(time)=-1*Aeq_buff(time);
% Aeq_buff=horzcat(zeros(1,n*T*3+T*3),Aeq_buff);
% Aeq_buff=horzcat(Aeq_buff,zeros(1,T));
% beq_buff=((weather_num_tmrw==1)*0.3+(weather_num_tmrw==2)*0.5+(weather_num_tmrw==3)*0.8) ↙
*SOC_EV_max-SOC_EV_init+110*sum(EV_30min(:,2));
% % beq_buff=0.5*SOC_EV_max-SOC_EV_init+110*sum(EV(:,2));
% % beq_buff=0.5*SOC_EV_max_hardware-SOC_EV_init+110*sum(EV(:,2));
% % beq_buff=0.8*SOC_EV_max-SOC_EV_init+110*sum(EV(:,2));
% Aeq=vertcat(Aeq,Aeq_buff);
% beq=vertcat(beq,beq_buff);
% clear Aeq_buff;
% clear beq_buff;

%EV走行中の充電なし
clear Aeq_buff;
clear beq_buff;
Aeq_buff=reshape((EV_30min(:,1)==0),1,T);
Aeq_buff=repmat(Aeq_buff,T,1);
Aeq_buff=Aeq_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,-Aeq_buff);
Aeq_buff=horzcat(zeros(T,n*T*3+T*3),Aeq_buff);
Aeq_buff=horzcat(Aeq_buff,zeros(T,T));
beq_buff=zeros(T,1);
```

```
% Aeq_buff=blkdiag(Aeq_buff,Aeq_buff);
% Aeq_buff=horzcat(zeros(2*T,n*T*3+T*3),Aeq_buff);
% Aeq_buff=horzcat(Aeq_buff,zeros(2*T,T));
% beq_buff=zeros(2*T,1);
time=1:T;
beq_buff(EV_30min(time,1)==1)=[];
time=1:T;
Aeq_buff(EV_30min(time,1)==1,:)=[];
Aeq=vertcat(Aeq,Aeq_buff);
beq=vertcat(beq,beq_buff);


%不等式制約
%SOCの最大値制約
A_buff=0.5*ones(1,2*T);
time=1:T;
A_buff(time)=-A_buff(time);
A_buff=repmat(A_buff,T,1);
aaa=eye(T,T)+tril(ones(T,T),-1);
A_buff=A_buff.*repmat(aaa,1,2);
A_buff=horzcat(zeros(T,n*T*3),A_buff);
A_buff=horzcat(A_buff,zeros(T,T*4));
A_buff=-1*A_buff;
time=1:T;
b_buff(time,1)=SOC_batt_max-SOC_batt_init;
% b_buff(time,1)=SOC_batt_max-SOC_batt_init;

A=A_buff;
b=b_buff;
clear A_buff;
clear b_buff;

%SOCの最小値制約
A_buff=0.5*ones(1,2*T);
time=1:T;
A_buff(time)=-1*A_buff(time);
A_buff=repmat(A_buff,T,1);
A_buff=A_buff.*repmat(aaa,1,2);
A_buff=horzcat(zeros(T,n*T*3),A_buff);
A_buff=horzcat(A_buff,zeros(T,T*4));
time=1:T;
b_buff(time,1)=SOC_batt_init-SOC_batt_min;
% b_buff(time,1)=SOC_batt_init-SOC_batt_min;

A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

%EVのSOC最大値制約
% A_buff=0.5*ones(1,2*T);
% time=1:T;
% A_buff(time)=-1*A_buff(time);
% A_buff=repmat(A_buff,T,1);
% A_buff=A_buff.*repmat(aaa,1,2);
```



```
% A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
% A_buff=horzcat(A_buff,zeros(T,T));
% A_buff=-1*A_buff;
% time=1:T;
% b_buff(time,1)=SOC_EV_max-SOC_EV_init;
% A=vertcat(A,A_buff);
% b=vertcat(b,b_buff);
% clear A_buff;
% clear b_buff;

A_buff=0.5*ones(1,2*T);
time=1:T;
A_buff(time)=-1*A_buff(time);
A_buff=repmat(A_buff,T,1);
A_buff=A_buff.*repmat(aaa,1,2);
A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
A_buff=horzcat(A_buff,zeros(T,T));
A_buff=-1*A_buff;
EV_run_buff=repmat(EV_30min(:,2),1,T);
EV_run_buff=EV_run_buff.*triu(ones(T,T),0);
time=1:T;
b_buff(time,1)=SOC_EV_max-SOC_EV_init+110*sum(EV_run_buff(:,time));
% b_buff(time,1)=SOC_EV_max-SOC_EV_init+110*sum(EV_run_buff(:,time));
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

%EVのSOC最小値制約
% A_buff=0.5*ones(1,2*T);
% time=1:T;
% A_buff(time)=-1*A_buff(time);
% A_buff=repmat(A_buff,T,1);
% A_buff=A_buff.*repmat(aaa,1,2);
% A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
% A_buff=horzcat(A_buff,zeros(T,T));
% time=1:T;
% b_buff(time,1)=SOC_EV_init-SOC_EV_min;
%
% A=vertcat(A,A_buff);
% b=vertcat(b,b_buff);
% clear A_buff;
% clear b_buff;

A_buff=0.5*ones(1,2*T);
time=1:T;
A_buff(time)=-1*A_buff(time);
A_buff=repmat(A_buff,T,1);
A_buff=A_buff.*repmat(aaa,1,2);
A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
A_buff=horzcat(A_buff,zeros(T,T));
EV_run_buff=repmat(EV(:,2),1,T);
EV_run_buff=EV_run_buff.*triu(ones(T,T),0);
time=1:T;
b_buff(time,1)=SOC_EV_init-SOC_EV_min-110*sum(EV_run_buff(:,time));
```

```
% b_buff(time,1)=SOC_EV_init-SOC_EV_min-110*sum(EV_run_buff(:,time));

A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

%出発前にEVが満充電
aaa=eye(T,T)+tril(ones(T,T),-1);
A_buff=0.5*ones(1,T);
time=2:T;
flag=xor(EV_30min(time,1),EV_30min(time-1,1));
flag(T,1)=0;
time=1:T;
flag(time)=(flag(time)&EV_30min(time,1));
A_buff=repmat(A_buff,1,2);
time=1:T;
A_buff(time)=-1*A_buff(time);
A_buff=repmat(A_buff,T,1);
A_buff=A_buff.*repmat(aaa,1,2);
for time=1:T
    if flag(time)~=1
        A_buff(time,:)=0.0;
    end
end
A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
A_buff=horzcat(A_buff,zeros(T,T));

time=2:T;
b_buff(time,1)=(EV_30min(time,1)==0&EV_30min(time-1,1)==1).*(SOC_EV_init-0.9*SOC_EV_max);
% b_buff(time,1)=(EV_30min(time,1)==0&EV_30min(time-1,1)==1).*(SOC_EV_init-0.9*SOC_EV_max);
time=1:T;
b_buff(b_buff(time)==0)=[];
hoge=any(A_buff,2);
time=1:T;
A_buff(hoge(time)==0,:)=[];

A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

%蓄電池SOCの最終値SOCに関する制約
%翌々日の天気予報が晴なら30%, 曇なら50%, 雨なら80%
A_buff=-0.5*ones(1,2*T);
time=T+1:2*T;
A_buff(time)=-1*A_buff(time);
A_buff=horzcat(zeros(1,n*T*3),A_buff);
A_buff=horzcat(A_buff,zeros(1,T*4));
% beq_buff=0.5*SOC_batt_max_hardware-SOC_batt_init;
b_buff=((weather_num_tmrw==1)*-0.3+(weather_num_tmrw==2)*-0.5+(weather_num_tmrw==3)*-0.8) ↙
*SOC_batt_max+SOC_batt_init;
% b_buff=((weather_num_tmrw==1)*-0.3+(weather_num_tmrw==2)*-0.5+(weather_num_tmrw==3)*-0.8) ↙
*SOC_batt_max+SOC_batt_init;
A=vertcat(A,A_buff);
```



```
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

% % % %EVのSOCの最終値に関する制約
% 満充電-3h分以上になる（９時出発なら６時から３時間しか充電できないため）
A_buff=-0.5*ones(1,2*T);
time=T+1:2*T;
A_buff(time)=-1*A_buff(time);
A_buff=horzcat(zeros(1,n*T*3+T*3),A_buff);
A_buff=horzcat(A_buff,zeros(1,T));
b_buff=-SOC_EV_max+3*3000;
% b_buff=-SOC_EV_max+3*3000;
% b_buff=((weather_num_tmrw==1)*-0.4+(weather_num_tmrw==2)*-0.5+(weather_num_tmrw==3)*-0.8) ↙
*SOC_EV_max+SOC_EV_init+110*sum(EV_30min(:,2));
% b_buff=((weather_num_tmrw==1)*-0.3+(weather_num_tmrw==2)*-0.5+(weather_num_tmrw==3)*-0.8) ↙
*SOC_EV_max+SOC_EV_init+110*sum(EV_30min(:,2));
% beq_buff=0.5*SOC_EV_max-SOC_EV_init+110*sum(EV(:,2));
% beq_buff=0.5*SOC_EV_max_hardware-SOC_EV_init+110*sum(EV(:,2));
% beq_buff=0.8*SOC_EV_max-SOC_EV_init+110*sum(EV(:,2));
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff;
clear b_buff;

%EVの充電制約(出発の直前に余裕のある充電)
% A_buff=0.5*ones(1,2*T);
% time=1:T;
% A_buff(time)=-1*A_buff(time);
% A_buff=repmat(A_buff,T,1);
% A_buff=A_buff.*repmat(aaa,1,2);
% A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
% A_buff=horzcat(A_buff,zeros(T,T));
% time=2:T;
% b_buff(time,1)=(EV(time,1)==0&EV(time-1,1)==1).*(SOC_EV_init-EV(time,2)*110);
%
% A=vertcat(A,A_buff);
% b=vertcat(b,b_buff);
% clear A_buff;
% clear b_buff;
% A_buff=0.5*ones(1,T);
% time=2:T;
% flag=xor(EV_30min(time,1),EV_30min(time-1,1));
% flag(T,1)=0;
% time=1:T;
% flag(time)=(flag(time)&EV_30min(time,1));
% % A_buff=A_buff.*flag.';
% A_buff=repmat(A_buff,1,2);
% time=1:T;
% A_buff(time)=-1*A_buff(time);
% A_buff=repmat(A_buff,T,1);
% A_buff=A_buff.*repmat(aaa,1,2);
% for time=1:T
%     if flag(time)~=1
%         A_buff(time,:)=0.0;
```

```
%     end
% end
% A_buff=horzcat(zeros(T,n*T*3+T*3),A_buff);
% A_buff=horzcat(A_buff,zeros(T,T));
%
% time=2:T;
% %出発前満充電
% b_buff(time,1)=(EV_30min(time,1)==0&EV_30min(time-1,1)==1).*(SOC_EV_init-SOC_EV_max);
%
% A=vertcat(A,A_buff);
% b=vertcat(b,b_buff);
% clear A_buff;
% clear b_buff;

%区分線形近似した式より大きい
%コンバータの電力に関して
csvread('L_UG.csv');
for scenario_num=1:n
    for piece=1:101
        clear A_buff;
        clear b_buff;
        [time,vr]=meshgrid(1:T,1:T);
        A_buff(time,vr)=ans(piece,1);
        A_buff=A_buff.*eye(T,T);
        if scenario_num==1
            A_buff=horzcat(A_buff,zeros(T,T*(n-1)));
        else
            A_buff=horzcat(zeros(T,T*(scenario_num-1)),A_buff);
            A_buff=horzcat(A_buff,zeros(T,T*(n-scenario_num+(scenario_num-1))));
        end
        A_buff=horzcat(A_buff,-1*eye(T,T));
        A_buff=horzcat(A_buff,zeros(T,T*(n-scenario_num+n)));
        A_buff=horzcat(A_buff,zeros(T,T*6));
        time=1:T;
        b_buff(time)=-ans(piece,2);
        b_buff=reshape(b_buff,T,1);
        A=vertcat(A,A_buff);
        b=vertcat(b,b_buff);
    end
end

%蓄電池に関して
csvread('L_batt_C.csv');
for piece=1:101
    clear A_buff;
    clear b_buff;
    [time,vr]=meshgrid(1:T,1:T);
    A_buff(time,vr)=ans(piece,1);
    A_buff=A_buff.*eye(T,T);
    A_buff=horzcat(zeros(T,(n+n+n)*T),A_buff);
    A_buff=horzcat(A_buff,zeros(T,T));
    A_buff=horzcat(A_buff,-1*eye(T,T));
    A_buff=horzcat(A_buff,zeros(T,T*3));
    b_buff(time)=-ans(piece,2);
    b_buff=reshape(b_buff,T,1);
```



```
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
end

csvread('L_batt_D.csv');
for piece=1:101
    clear A_buff;
    clear b_buff;
    [time,vr]=meshgrid(1:T,1:T);
    A_buff(time,vr)=ans(piece,1);
    A_buff=A_buff.*eye(T,T);
    A_buff=horzcat(zeros(T,(n+n+n+1)*T),A_buff);
    A_buff=horzcat(A_buff,-1*eye(T,T));
    A_buff=horzcat(A_buff,zeros(T,T*3));
    b_buff(time)=-ans(piece,2);
    b_buff=reshape(b_buff,T,1);
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
end

%EVに関して
csvread('L_EV_C.csv');
for piece=1:101
    clear A_buff;
    clear b_buff;
    [time,vr]=meshgrid(1:T,1:T);
    A_buff(time,vr)=ans(piece,1);
    A_buff=A_buff.*eye(T,T);
    A_buff=horzcat(zeros(T,(n+n+n+3)*T),A_buff);
    A_buff=horzcat(A_buff,zeros(T,T));
    A_buff=horzcat(A_buff,-1*eye(T,T));
    b_buff(time)=-ans(piece,2);
    b_buff=reshape(b_buff,T,1);
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
end

csvread('L_EV_D.csv');
for piece=1:101
    clear A_buff;
    clear b_buff;
    [time,vr]=meshgrid(1:T,1:T);
    A_buff(time,vr)=ans(piece,1);
    A_buff=A_buff.*eye(T,T);
    A_buff=horzcat(zeros(T,(n+n+n+4)*T),A_buff);
    A_buff=horzcat(A_buff,-1*eye(T,T));
    b_buff(time)=-ans(piece,2);
    b_buff=reshape(b_buff,T,1);
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
end

%購入電力が正になる制約
% for scenario_num=1:n
%     clear A_buff;
```

```
%     clear b_buff;
%     A_buff=-eye(T,T);
%     %コンバータ分
%     if scenario_num==1
%         A_buff=horzcat(A_buff,zeros(T,T*(n-1)));
%     elseif scenario_num==n
%         A_buff=horzcat(zeros(T,T*(n-1)),A_buff);
%     else
%          A_buff=horzcat(zeros(T,T*(scenario_num-1)),A_buff);
%          A_buff=horzcat(A_buff,zeros(T,T*(n-scenario_num)));
%     end
%     %損失分
%     A_buff=horzcat(A_buff,zeros(T,T*n));
%     %逆潮流変数
%     if scenario_num==1
%         A_buff=horzcat(A_buff,-eye(T,T));
%         A_buff=horzcat(A_buff,zeros(T,T*(n-1)));
%     elseif scenario_num==n
%         A_buff=horzcat(zeros(T,T*(n-1)),A_buff);
%         A_buff=horzcat(A_buff,-eye(T,T));
%     else
%          A_buff=horzcat(zeros(T,T*(scenario_num-1)),A_buff);
%          A_buff=horzcat(A_buff,-eye(T,T));
%          A_buff=horzcat(A_buff,zeros(T,T*(n-scenario_num)));
%     end
%     %その他の変数
%     A_buff=horzcat(A_buff,zeros(T,T*6));
%     time=1:T;
%     b_buff(time)=0;
%     b_buff=reshape(b_buff,T,1);
%     A=vertcat(A,A_buff);
%     b=vertcat(b,b_buff);
% end

% %逆潮流の期待値が指令値以下になる制約
% clear A_buff;
% clear b_buff;
% A_buff=prob_pre_scenario(1,1)*ones(1,T);
% for scenario_num=2:n
%     A_buff=horzcat(A_buff,prob_pre_scenario(1,scenario_num)*ones(1,T));
% end
% A_buff=horzcat(zeros(1,2*n*T),A_buff);
% A_buff=horzcat(A_buff,zeros(1,6*T));
% b_buff=P_rev;
% A=vertcat(A,A_buff);
% b=vertcat(b,b_buff);

%%上下限制約
%上限
%コンバータ、損失
vr=1:2*n*T;
ub(vr,1)=inf;
%PV抑制量について
ub_buff=zeros(T*n,1)+inf;
ub=vertcat(ub,ub_buff);
```



```
clear ub_buff;
% ub_buff=1000*ones(T*n,1);
% ub=vertcat(ub,ub_buff);
% clear ub_buff;
%蓄電池について
ub_buff=10000*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%蓄電池損失について
ub_buff=zeros(T,1)+inf;
ub=vertcat(ub,ub_buff);
clear ub_buff;
%EVについて
ub_buff=3000*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%EV損失について
ub_buff=zeros(T,1)+inf;
ub=vertcat(ub,ub_buff);
clear ub_buff;

%下限
%コンバータ
% vr=1:n*T;
% lb(vr,1)=-inf;
vr=1:n*T;
lb(vr,1)=0;
%損失
lb_buff=zeros(n*T,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%PV抑制量
lb_buff=zeros(n*T,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%それ以降
lb_buff=zeros(6*T,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
% lb=zeros(vr_num,1);
% vr=1:T;
% lb(vr,1)=0;
% vr=T+1:2*T;
% lb(vr,1)=0;
% vr=2*T+1:4*T;
% lb(vr,1)=0;
% vr=4*T+1:5*T;
% lb(vr,1)=0;
% vr=5*T+1:7*T;
% lb(vr,1)=0;
% vr=7*T+1:8*T;
% lb(vr,1)=0;

%%求解
options=optimoptions(@linprog,'MaxIter',3000);
```

```
652 [x,fval,exitflag,lambda,output] =linprog(f,A,b,Aeq,beq,lb,ub,[],options);
653 % [x,fval,exitflag,lambda,output] =linprog(f,A,b,Aeq,beq,lb,ub);
654 % x = linprog(f,A,b,Aeq,beq,lb,ub);
655 % vr=1:T;
656 % UG=x(vr);
657 % vr=2*T+1:3*T;
658 % batt_C=x(vr);
659 % vr=3*T+1:4*T;
660 % batt_D=x(vr);
661 % vr=5*T+1:6*T;
662 % EV_C=x(vr);
663 % vr=6*T+1:7*T;
664 % EV_D=x(vr);
665
666 % xを分割
667 % vr=1:T;
668 % UG(vr,1)=x(vr);
669 % vr=T+1:2*T;
670 % L_UG(vr-T,1)=x(vr);
671 x=reshape(x,T,3*n+6);
672 time=1:T;
673 batt_C(time,:)=x(time,3*n+1);
674 time=1:T;
675 batt_D(time,:)=x(time,3*n+2);
676 time=1:T;
677 L_batt(time,:)=x(time,3*n+3);
678 time=1:T;
679 EV_C(time,:)=x(time,3*n+4);
680 time=1:T;
681 EV_D(time,:)=x(time,3*n+5);
682 time=1:T;
683 L_EV(time,:)=x(time,3*n+6);
```

```
%% 気象データから、シナリオ群を作成
%   過去のデータから、日射量の1時間値を作成
%   日射量は前1時間値(最初の要素は0:00~1:00の合計)
%  データベース 2008年5月2日から2013年6月2日まで
% 快晴:1, 晴:2, 薄曇:3, 曇:4, 雨:5
%  2012年はうるう年
% 今のところ、入力するのは午前6時から午後6時までの12時間を4時間帯で
% clear all;
% %
% % % %ここは手打ち
% date_act=datenum(2013,6,26);  %%日付
% weather_act=['R' 'R' 'C' 'C'];
weather_act=date_weather(day,2:5);  %天候

% weather_num=1*(weather_act=='F')+2*(weather_act=='C')+3*(weather_act=='R')+4*(weather_act=='S');
weather_num=weather_act;
% %
%
% %% データベース読み込み(単体で動かすときはここの%を消す)
% %天気データを読み込む
% weather = xlsread('日射量データベース.xlsx');
% weather(:,1)=weather(:,1)+datenum('30-Dec-1899');
% %日射量データを読み込む
% radiation(:,1)=weather(:,1);
% for num=1:24
%     radiation(:,num+1)=xlsread('日射量データベース.xlsx',num+1,'B:B');
% end
% シナリオ数
n=10;


%天気予報の精度をパラメータ化 変更の余地あり
%prob_〇_△ :〇が予報,△が実際
% prob_f_f=0.8;
% prob_f_c=0.15;
% prob_f_r=0.05;
% prob_f_s=0.05;
% prob_c_f=0.1;
% prob_c_c=0.8;
% prob_c_r=0.1;
% prob_c_s=0.1;
% prob_r_f=0.05;
% prob_r_c=0.15;
% prob_r_r=0.8;
% prob_r_s=0.0;
% prob_s_f=0.05;
% prob_s_c=0.15;
% prob_s_r=0.0;
% prob_s_s=0.8;
%
prob_f_cl=0.3605;
prob_f_f=0.2517;
prob_f_sc=0.1769;
prob_f_c=0.1769;
prob_f_r=0.0340;
```



```

prob_c_cl=0.0519;
prob_c_f=0.1556;
prob_c_sc=0.0370;
prob_c_c=0.5778;
prob_c_r=0.1778;

prob_r_cl=0.0556;
prob_r_f=0.1481;
prob_r_sc=0.055;
prob_r_c=0.2407;
prob_r_r=0.5000;

%%シナリオの発生
%組み合わせは3^4(天候が3種類の日射のある12時間)
T_weather=4;
weather_scenario=zeros(5^T_weather,T_weather);

for i=1:5^T_weather
    weather_scenario(i,1)=mod(i-1,5)+1;
    weather_scenario(i,2)=floor(mod(i-1,5^2)/5)+1;
    weather_scenario(i,3)=floor(floor(mod(i-1,5^3)/5)/5)+1;
    weather_scenario(i,4)=floor(floor(floor(mod(i-1,5^4)/5)/5)/5)+1;
%     weather_scenario(i,5)=floor(floor(floor(floor(mod(i-1,3^5)/3)/3)/3)/3)+1;
%     weather_scenario(i,6)=floor(floor(floor(floor(floor(mod(i-1,3^6)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,7)=floor(floor(floor(floor(floor(floor(mod(i-1,3^7)/3)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,8)=floor(floor(floor(floor(floor(floor(floor(mod(i-1,3^8)/3)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,9)=floor(floor(floor(floor(floor(floor(floor(floor(mod(i-1,3^9)/3)/3)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,10)=floor(floor(floor(floor(floor(floor(floor(floor(floor(mod(i-1,3^10)/3)/3)/3)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,11)=floor(floor(floor(floor(floor(floor(floor(floor(floor(floor(mod(i-1,3^11)/3)/3)/3)/3)/3)/3)/3)/3)/3)+1;
%     weather_scenario(i,12)=floor(floor(floor(floor(floor(floor(floor(floor(floor(floor(floor(mod(i-1,3^12)/3)/3)/3)/3)/3)/3)/3)/3)/3)/3)+1;
end

%各シナリオの発生確率の計算
prob_scenario=zeros(5^T_weather,1)+1;
for scenario_num=1:5^T_weather
    for time=1:T_weather
        if weather_num(time)==1
            if weather_scenario(scenario_num,time)==1
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_f_cl;
            elseif weather_scenario(scenario_num,time)==2
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_f_f;
            elseif weather_scenario(scenario_num,time)==3
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_f_sc;
            elseif weather_scenario(scenario_num,time)==4
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_f_c;
            elseif weather_scenario(scenario_num,time)==5
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_f_r;
            end
        elseif weather_num(time)==2
```

```
            if weather_scenario(scenario_num,time)==1
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_c_cl;
            elseif weather_scenario(scenario_num,time)==2
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_c_f;
            elseif weather_scenario(scenario_num,time)==3
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_c_sc;
            elseif weather_scenario(scenario_num,time)==4
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_c_c;
            elseif weather_scenario(scenario_num,time)==5
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_c_r;
            end
        elseif weather_num(time)==3
            if weather_scenario(scenario_num,time)==1
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_r_cl;
            elseif weather_scenario(scenario_num,time)==2
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_r_f;
            elseif weather_scenario(scenario_num,time)==3
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_r_sc;
            elseif weather_scenario(scenario_num,time)==4
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_r_c;
            elseif weather_scenario(scenario_num,time)==5
                prob_scenario(scenario_num)=prob_scenario(scenario_num)*prob_r_r;
            end
        end
    end
end

%データベースから、適合するデータを探す
radiation_clear=zeros(1,25);
radiation_fine=zeros(1,25);
radiation_scloudy=zeros(1,25);
radiation_cloudy=zeros(1,25);
radiation_rainy=zeros(1,25);
% radiation_snowy=zeros(1,25);
day_num_clear=zeros(1,4);
day_num_fine=zeros(1,4);
day_num_scloudy=zeros(1,4);
day_num_cloudy=zeros(1,4);
day_num_rainy=zeros(1,4);
% day_num_snowy=0;

%データベースの範囲より大きい場合
if date_act>datenum(2012,1,1)
    %予測日の1年前から5年前まで
    for year=1:5
        for date=date_act-14-datenum(2008,5,1)-365*year:date_act+16-datenum(2008,5,1)-365*year
            for time=1:4
                if(weather(date,time+1)==1)
                    radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_clear(1,time)=day_num_clear(1,time)+1;
                end
                if(weather(date,time+1)==2)
                    radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
```



```
                    day_num_fine(1,time)=day_num_fine(1,time)+1;
                end
                if(weather(date,time+1)==3)
                    radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_scloudy(1,time)=day_num_scloudy(1,time)+1;
                end
                if(weather(date,time+1)==4)
                    radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_cloudy(1,time)=day_num_cloudy(1,time)+1;
                end
                if(weather(date,time+1)==5)
                    radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_rainy(1,time)=day_num_rainy(1,time)+1;
                end
% 
            if(weather(date,2)==4)
                radiation_snowy=radiation_snowy+radiation(date,:);
                day_num_snowy=day_num_snowy+1;
            end
        end
    end
else
    %予測日前15日まで
    for date=date_act-15-datenum(2008,5,1):date_act-1-datenum(2008,5,1)
        for time=1:4
                if(weather(date,time+1)==1)
                    radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_clear(1,time)=day_num_clear(1,time)+1;
                end
                if(weather(date,time+1)==2)
                    radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_fine(1,time)=day_num_fine(1,time)+1;
                end
                if(weather(date,time+1)==3)
                    radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_scloudy(1,time)=day_num_scloudy(1,time)+1;
                end
                if(weather(date,time+1)==4)
                    radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_cloudy(1,time)=day_num_cloudy(1,time)+1;
                end
                if(weather(date,time+1)==5)
                    radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
                    day_num_rainy(1,time)=day_num_rainy(1,time)+1;
                end
            if(weather(date,2)==4)
                radiation_snowy=radiation_snowy+radiation(date,:);
```

```
206 %                 day_num_snowy=day_num_snowy+1;
207 %             end
208         end
209     end
210     %予測日の1年前から5年前まで
211     for year=1:5
212         for date=date_act-14-datenum(2008,5,1)-365*year:date_act+16-datenum(2008,5,1)-365*year
213             for time=1:4
214                 if(weather(date,time+1)==1)
215                     radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_clear(1,6+(time-1) ↙
*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
216                     day_num_clear(1,time)=day_num_clear(1,time)+1;
217                 end
218                 if(weather(date,time+1)==2)
219                     radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_fine(1,6+(time-1)*3: ↙
6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
220                     day_num_fine(1,time)=day_num_fine(1,time)+1;
221                 end
222                 if(weather(date,time+1)==3)
223                     radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_scloudy(1,6+ ↙
(time-1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
224                     day_num_scloudy(1,time)=day_num_scloudy(1,time)+1;
225                 end
226                 if(weather(date,time+1)==4)
227                     radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_cloudy(1,6+(time- ↙
1)*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
228                     day_num_cloudy(1,time)=day_num_cloudy(1,time)+1;
229                 end
230                 if(weather(date,time+1)==5)
231                     radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_rainy(1,6+(time-1) ↙
*3:6+(time)*2+time-1)+radiation(date,6+(time-1)*3+1:6+(time)*2+time);
232                     day_num_rainy(1,time)=day_num_rainy(1,time)+1;
233                 end
234 %             if(weather(date,2)==4)
235 %                 radiation_snowy=radiation_snowy+radiation(date,:);
236 %                 day_num_snowy=day_num_snowy+1;
237 %             end
238             end
239         end
240     end
241 end
242
243 %ゼロ除算の回避
244 if day_num_clear==0
245     day_num_clear=1;
246 end
247 if day_num_fine==0
248     day_num_fine=1;
249 end
250 if day_num_scloudy==0
251     day_num_scloudy=1;
252 end
253 if day_num_cloudy==0
254     day_num_cloudy=1;
255 end
```



```
256 if day_num_rainy==0
257     day_num_rainy=1;
258 end
259 % if day_num_snowy==0
260 %     day_num_snowy=1;
261 % end
262
263 %データベースの日数で平均化
264 for time=1:4;
265     radiation_clear(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_clear(1,6+(time-1)*3:6+(time) ↙
*2+time-1)/day_num_clear(1,time);
266     radiation_fine(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_fine(1,6+(time-1)*3:6+(time)*2+time- ↙
1)/day_num_fine(1,time);
267     radiation_scloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_scloudy(1,6+(time-1)*3:6+(time) ↙
*2+time-1)./day_num_scloudy(1,time);
268     radiation_cloudy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_cloudy(1,6+(time-1)*3:6+(time) ↙
*2+time-1)./day_num_cloudy(1,time);
269     radiation_rainy(1,6+(time-1)*3:6+(time)*2+time-1)=radiation_rainy(1,6+(time-1)*3:6+(time) ↙
*2+time-1)./day_num_rainy(1,time);
270 end
271     % radiation_snowy=radiation_snowy/day_num_snowy;
272
273 %シナリオ毎の日射量パターン作成
274 radiation_scenario=zeros(5^T_weather,24);
275 for scenario_num = 1:5^T_weather
276     for time=1:24
277         if time >=7 && time<10
278 %         if time >=6 && time<9
279             if weather_scenario(scenario_num,1)==1
280                 radiation_scenario(scenario_num,time)=radiation_clear(1,time+1);
281             elseif weather_scenario(scenario_num,1)==2
282                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
283             elseif weather_scenario(scenario_num,1)==3
284                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time+1);
285             elseif weather_scenario(scenario_num,1)==4
286                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
287             elseif weather_scenario(scenario_num,1)==5
288                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
289             end
290 %             if weather_scenario(scenario_num,1)==1
291 %                 radiation_scenario(scenario_num,time)=radiation_clear(1,time-1);
292 %             elseif weather_scenario(scenario_num,1)==2
293 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time-1);
294 %             elseif weather_scenario(scenario_num,1)==3
295 %                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time-1);
296 %             elseif weather_scenario(scenario_num,1)==4
297 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time-1);
298 %             elseif weather_scenario(scenario_num,1)==5
299 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time-1);
300 %             end
301 %             if weather_scenario(scenario_num,1)==1
302 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
303 %             elseif weather_scenario(scenario_num,1)==2
304 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
305 %             elseif weather_scenario(scenario_num,1)==3
```

```
306 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
307 %             end
308         elseif time>=10 && time<13
309 %         elseif time>=9 && time<12
310             if weather_scenario(scenario_num,2)==1
311                 radiation_scenario(scenario_num,time)=radiation_clear(1,time+1);
312             elseif weather_scenario(scenario_num,2)==2
313                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
314             elseif weather_scenario(scenario_num,2)==3
315                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time+1);
316             elseif weather_scenario(scenario_num,2)==4
317                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
318             elseif weather_scenario(scenario_num,2)==5
319                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
320             end
321 %             if weather_scenario(scenario_num,2)==1
322 %                 radiation_scenario(scenario_num,time)=radiation_clear(1,time-1);
323 %             elseif weather_scenario(scenario_num,2)==2
324 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time-1);
325 %             elseif weather_scenario(scenario_num,2)==3
326 %                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time-1);
327 %             elseif weather_scenario(scenario_num,2)==4
328 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time-1);
329 %             elseif weather_scenario(scenario_num,2)==5
330 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time-1);
331 %             end
332 %             if weather_scenario(scenario_num,2)==1
333 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
334 %             elseif weather_scenario(scenario_num,2)==2
335 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
336 %             elseif weather_scenario(scenario_num,2)==3
337 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
338 %             end
339         elseif time>=13 && time<16
340 %         elseif time>=12 && time<15
341             if weather_scenario(scenario_num,3)==1
342                 radiation_scenario(scenario_num,time)=radiation_clear(1,time+1);
343             elseif weather_scenario(scenario_num,3)==2
344                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
345             elseif weather_scenario(scenario_num,3)==3
346                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time+1);
347             elseif weather_scenario(scenario_num,3)==4
348                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
349             elseif weather_scenario(scenario_num,3)==5
350                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
351             end
352 %             if weather_scenario(scenario_num,3)==1
353 %                 radiation_scenario(scenario_num,time)=radiation_clear(1,time-1);
354 %             elseif weather_scenario(scenario_num,3)==2
355 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time-1);
356 %             elseif weather_scenario(scenario_num,3)==3
357 %                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time-1);
358 %             elseif weather_scenario(scenario_num,3)==4
359 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time-1);
360 %             elseif weather_scenario(scenario_num,3)==5
```



```
361 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time-1);
362 %             end
363 %             if weather_scenario(scenario_num,3)==1
364 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
365 %             elseif weather_scenario(scenario_num,3)==2
366 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
367 %             elseif weather_scenario(scenario_num,3)==3
368 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
369 %             end
370 %         elseif time>=15 && time<18
371         elseif time>=16 && time<19
372             if weather_scenario(scenario_num,4)==1
373                 radiation_scenario(scenario_num,time)=radiation_clear(1,time+1);
374             elseif weather_scenario(scenario_num,4)==2
375                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
376             elseif weather_scenario(scenario_num,4)==3
377                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time+1);
378             elseif weather_scenario(scenario_num,4)==4
379                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
380             elseif weather_scenario(scenario_num,4)==5
381                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
382             end
383 %             if weather_scenario(scenario_num,4)==1
384 %                 radiation_scenario(scenario_num,time)=radiation_clear(1,time-1);
385 %             elseif weather_scenario(scenario_num,4)==2
386 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time-1);
387 %             elseif weather_scenario(scenario_num,4)==3
388 %                 radiation_scenario(scenario_num,time)=radiation_scloudy(1,time-1);
389 %             elseif weather_scenario(scenario_num,4)==4
390 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time-1);
391 %             elseif weather_scenario(scenario_num,4)==5
392 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time-1);
393 %             end
394 %             if weather_scenario(scenario_num,4)==1
395 %                 radiation_scenario(scenario_num,time)=radiation_fine(1,time+1);
396 %             elseif weather_scenario(scenario_num,4)==2
397 %                 radiation_scenario(scenario_num,time)=radiation_cloudy(1,time+1);
398 %             elseif weather_scenario(scenario_num,4)==3
399 %                 radiation_scenario(scenario_num,time)=radiation_rainy(1,time+1);
400 %             end
401         end
402     end
403 end
404
405
406
407 %%シナリオ削減アルゴリズム
408 %ステップ1
409 % シナリオ間距離の作成
410 distance_scenario=zeros(5^T_weather,1);
411 [j k]=meshgrid(1:5^T_weather, 1:5^T_weather);
412 distance_scenario=max(abs(radiation_scenario(j,:)-radiation_scenario(k,:)),[],2);
413 distance_scenario=reshape(distance_scenario,5^T_weather,5^T_weather);
414 % distance_scenario=max(abs(radiation_scenario(j,:)-radiation_scenario(k,:)));
415 % %     distance_scenario(k,j)=radiation_scenario(j,:)-radiation_scenario(k,:);
```

```
% distance_scenario(k,j)=max(abs(radiation_scenario(j,:)-radiation_scenario(k,:)));

%シナリオ間距離が最少となるものを選択
distance_scenario_original=distance_scenario;
[l j]=meshgrid(1:5^T_weather, 1:5^T_weather);
distance_scenario(l==j)=999999;
distance_scenario_min=min(distance_scenario);

%確率距離を計算
distance_stochastics=distance_scenario_min.*transpose(prob_scenario);

%確率距離が最少となるものを除外
distance_stochastics_min=min(distance_stochastics);
reduction_num=find(distance_stochastics_min==distance_stochastics);

%
% for k=1:5^T_weather
%     for j=1:5^T_weather
%         distance_scenario(k,j)=max(abs(radiation_scenario(j,:)-radiation_scenario(k,:)));
%     end
% end
% %シナリオ間距離が最少となるものを選択
% distance_scenario_min=(zeros(5^T_weather,5^T_weather)+1)*9999999999999;
% for l=1:5^T_weather
%     for j=1:5^T_weather
%         if l~=j
%             if distance_scenario_min(l,l)>distance_scenario(l,j)
%                 distance_scenario_min(l,l)=distance_scenario(l,j);
%             end
%         end
%     end
% end
% %確率距離を計算
% for l=1:5^T_weather
%     distance_stochastics(l,1)=distance_scenario_min(l,l)*prob_scenario(l,1);
% end
% %確率距離が最少となるものを除外
% reduction_num=0;
% for l=1:5^T_weather
%     if l==1
%         distance_stochastics_min=distance_stochastics(1,1);
%         reduction_num=1;
%     else
%         if distance_stochastics_min>distance_stochastics(l,1);
%             distance_stochastics_min=distance_stochastics(l,1);
%             reduction_num=l;
%         end
%     end
% end

reduction_num_list=reduction_num;
%ステップi
for i=1:5^T_weather-n-1
    %シナリオ間距離が最少となるもの、かつリストアップされていないものを選択
    [l j]=meshgrid(1:5^T_weather, 1:5^T_weather);
```



```
    distance_scenario(l==j | l==reduction_num(1,1))=999999;
    distance_scenario_min=min(distance_scenario);

    %確率距離を計算
    distance_stochastics=distance_scenario_min.*transpose(prob_scenario);

    %確率距離が最少となるものを除外
    distance_stochastics_min=min(distance_stochastics);
    reduction_num=find(distance_stochastics_min==distance_stochastics);
    reduction_num_list=horzcat(reduction_num_list,reduction_num(1,1));
end
% for i=1:5^T_weather-n-1
%     %シナリオ間距離が最少となるものを選択
%     distance_scenario_min=(zeros(5^T_weather,5^T_weather)+1)*9999999999999;
%     distance_stochastics=zeros(5^T_weather,1);
%     for l=1:5^T_weather
%         for k=1:5^T_weather
%             if l~=k
%                 if k~=reduction_num
%                     if distance_scenario_min(k,l)>distance_scenario(k,l)
%                         distance_scenario_min(k,l)=distance_scenario(k,l);
%                     end
%                 end
%             end
%         end
%     end
%     for l=1:5^T_weather
%         for k=1:5^T_weather
%             if distance_scenario_min(k,l)==9999999999999
%                 distance_scenario_min(k,l)=0;
%             end
%         end
%     end
%     %確率距離を計算
%     for l=1:5^T_weather
%         if l~=reduction_num
%             for k=1:5^T_weather
%                 distance_stochastics(l,1)=distance_stochastics(l,1)+prob_scenario(l,1) ↙
*distance_scenario_min(k,l);
% %                 distance_stochastics(l,1)=sum(prob_scenario(k,1)*distance_scenario_min(k,l));
%             end
%         end
%     end
%
%     %確率距離が最少となるものを除外
%     reduction_num_buff=0;
%     distance_stochastics_min=999999999999999999;
%     for l=1:5^T_weather
%         if l~=reduction_num
%             if distance_stochastics_min>distance_stochastics(l,1);
%                 distance_stochastics_min=distance_stochastics(l,1);
%                 reduction_num_buff=l;
%             end
%         end
%     end
```

```
%     reduction_num=vertcat(reduction_num,reduction_num_buff);
% end
%生き残ったシナリオを整理
reduction_num_buff=sort(reduction_num_list);
list_buff=zeros(1,5^T_weather);
time=1;
step=1;
while time<=numel(reduction_num_buff)
    if time==reduction_num_buff(1,step) && time==reduction_num_buff(1,step)
        list_buff(1,time)=reduction_num_buff(1,step);
        step=step+1;
    else
        list_buff(1,time)=0;
    end
    time=time+1;
end
pre_scenario_num_buff=find(list_buff==0);
pre_scenario_num=pre_scenario_num_buff(1:n);

prob_pre_scenario=zeros(1,n);
scenario_num=1:n;
prob_pre_scenario=transpose(prob_scenario(pre_scenario_num(scenario_num),1));

% sort(reduction_num);
% pre_scenario_num=zeros(1,n);
% flag=1;
% for scenario_num=1:5^T_weather
%     if reduction_num~=scenario_num;
%         pre_scenario_num(1,flag)=scenario_num;
%         flag=flag+1;
%     end
% end
%
% prob_pre_scenario=zeros(1,n);
% flag=1;
% for scenario_num=1:5^T_weather
%     if reduction_num~=scenario_num
%         prob_pre_scenario(1,flag)=prob_scenario(scenario_num,1);
%         flag=flag+1;
%     end
% end

%%シナリオの再分配
q=zeros(1,n);
%削減されたシナリオと生き残ったシナリオとの距離を比較し、最少のものを選択

for i=1:5^T_weather
    min_distance=9999999999;
    flag=0;
    for j=1:5^T_weather
        if j~=reduction_num_list
            if i~=pre_scenario_num
                if (i~=j)
                    if min_distance>distance_scenario_original(i,j)
                        min_distance=distance_scenario_original(i,j);
```



```
                    %                 index_q=j;
                    flag=flag+1;
                end
            end
        end
    end
    if i~=pre_scenario_num
        prob_pre_scenario(1,flag)=prob_pre_scenario(1,flag)+prob_scenario(i,1);
    end
end
% for i=1:5^T_weather
%     min=9999999999;
%     flag=0;
%     for j=1:5^T_weather
%         if j~=reduction_num
%             if i~=pre_scenario_num
%                 if (i~=j)
%                     if min>distance_scenario(i,j)
%                         min=distance_scenario(i,j);
%         %                 index_q=j;
%                         flag=flag+1;
%                     end
%                 end
%             end
%         end
%     end
%     if i~=pre_scenario_num
%         prob_pre_scenario(1,flag)=prob_pre_scenario(1,flag)+prob_scenario(i,1);
%     end
% end
%goukei=sum(prob_pre_scenario);
% %PVへの日射量の期待値算出
% for time=1:24
%     if weather_num(time)==1
%         Rad(time)=prob_f_f*radiation(1,time+1)+prob_f_c*radiation_cloudy(1,time+1) ↙
+prob_f_r*radiation_rainy(1,time+1)+prob_f_s*radiation_snowy(1,time+1);
%     elseif weather_num(time)==2
%         Rad(time)=prob_c_f*radiation(1,time+1)+prob_c_c*radiation_cloudy(1,time+1) ↙
+prob_c_r*radiation_rainy(1,time+1)+prob_c_s*radiation_snowy(1,time+1);
%     elseif weather_num(time)==3
%         Rad(time)=prob_r_f*radiation(1,time+1)+prob_r_c*radiation_cloudy(1,time+1) ↙
+prob_r_r*radiation_rainy(1,time+1)+prob_r_s*radiation_snowy(1,time+1);
%     elseif weather_num(time)==4
%         Rad(time)=prob_s_f*radiation(1,time+1)+prob_s_c*radiation_cloudy(1,time+1) ↙
+prob_s_r*radiation_rainy(1,time+1)+prob_s_s*radiation_snowy(1,time+1);
%     end
% end
%

%単位をWにし、30分データにする
PV=zeros(48,n);
flag=1;
for scenario_num=1:5^T_weather
    for time=1:24
```

```
631         if scenario_num~=reduction_num_list
632 %             PV(time+time-1,flag)=1000*radiation_scenario(scenario_num,time)*0.15*200/(3.6*2);
633 %             PV(time+time,flag)=1000*radiation_scenario(scenario_num,time)*0.15*200/(3.6*2);
634                 PV(time+time-1,flag)=PCS*(radiation_scenario(scenario_num,time)*1000/3.6)*0.15*130.
9;
635                 PV(time+time,flag)=PCS*(radiation_scenario(scenario_num,time)*1000/3.6)*0.15*130.9;
636                 %130.9：パネル面積, 1000/3.6：MJとWhの換算, 0.15：パネル変換効率
637             if time==24
638                 flag=flag+1;
639             end
640         end
641     end
642 end
643 flag=1;
644 for scenario_num=1:5^T_weather
645     if scenario_num~=reduction_num_list
646         weather_pre(flag,:)=weather_scenario(scenario_num,:);
647         flag=flag+1;
648     end
649 end
650
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%計画運転モード%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%計画運転モードへ戻った段階で、蓄電池及びEVのロックフラグを解除
lock_batt=0; lock_batt_ch=0; lock_batt_dch=0;
lock_EV=0; lock_EV_ch=0; lock_EV_dch=0;

if cnt>=2
%%%%%%%%%%%%%%%%%%%%%%%%%蓄電池・EV出力変化制約%%%%%%%%%%%%%%%%%%%%%%%

    if I_batt(cnt-1)-I_batt(cnt)>I_batt_lim
        I_batt(cnt)=I_batt(cnt-1)-I_batt_lim;
    elseif I_batt(cnt)-I_batt(cnt-1)>I_batt_lim
        I_batt(cnt)=I_batt(cnt-1)+I_batt_lim;
    end
    if I_EV(cnt-1)-I_EV(cnt)>I_EV_lim
        I_EV(cnt)=I_EV(cnt-1)-I_EV_lim;
    elseif I_EV(cnt)-I_EV(cnt-1)>I_EV_lim
        I_EV(cnt)=I_EV(cnt-1)+I_EV_lim;
    end

%%%%%%%%%%%%%%%%SOCに関する制約（max→放電不可　min→充電不可）%%%%%%%

if mode(cnt,2)==2
    if I_batt_buff>0
        I_batt(cnt)=I_batt_buff;
        mode(cnt,2)=3;
    else
        I_batt(cnt)=0;
    end

else

    if SOC_batt(cnt-1)>=SOC_batt_max && I_batt(cnt)<0
        I_batt(cnt)=0;
        mode(cnt,2)=2;
    end

end

if mode(cnt,3)==2
    if I_EV_buff>0
        I_EV(cnt)=I_EV_buff;
        mode(cnt,3)=3;
    else
        I_EV(cnt)=0;

    end
else

    if (SOC_EV(cnt-1)>=SOC_EV_max && I_EV(cnt)<0) && mode(cnt,3)~=4
        I_EV(cnt)=0;
        mode(cnt,3)=2;
    end

end
```



```

    if SOC_batt(cnt-1)<=(SOC_batt_min+0.05) && I_batt(cnt)>0
        I_batt(cnt)=0;
    end
    if (SOC_EV(cnt-1)<=(SOC_EV_min+0.05) && I_EV(cnt)>0) && mode(cnt,3)~=4
        I_EV(cnt)=0;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%EVプラグアウト時は出力0
I_EV(cnt)=(mode(cnt,3)==4)*0+(mode(cnt,3)~=4)*I_EV(cnt);

%CONV出力電圧の決定
if cnt==1
    V_conv(cnt)=V_conv_init;
else
    V_conv(cnt)=V_DC_ref+R_conv*I_conv(cnt-1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%計算したV_convをもとにI_convを潮流計算
%I_conv以外の電流をI_etcとまとめる
I_etc(cnt)=I_PV(cnt)+I_batt(cnt)+I_EV(cnt);
c_I_conv=[R_line+R_conv 2*R_line*I_etc(cnt)+R_conv*I_etc(cnt)-V_conv(cnt) R_line*I_etc(cnt)^2+P_load(cnt,day)-V_conv(cnt)*I_etc(cnt)];
ans_I_conv=roots(c_I_conv);
%2次方程式の解は2つあるが、値は小さい方を選択（大きい方は線路抵抗に高電圧がかかる）
I_conv(cnt)=ans_I_conv(2);

%得られた電流を保存
I_conv_temp=I_conv(cnt);

%逆潮流が発生していた場合は、CONV出力を0にする
if I_conv(cnt)<0
    I_conv(cnt)=0;
    flag_VDC=1;
else
    flag_VDC=0;
end

I_load=I_etc(cnt)+I_conv(cnt);

if flag_VDC==1
    %CONV出力が負になった場合は、制限された
    %CONV出力電流を用いて、改めて出力電圧を求める
    %電流並行時の負荷を求めると、
    R_load=P_load(cnt,day)/(I_etc(cnt)+I_conv_temp)^2;
    V_DC(cnt)=(R_line+R_load)*I_load;
    %改めてV_EV、並びに負荷にかかる電圧も求める
    V_conv(cnt)=V_DC(cnt)+R_conv*I_conv(cnt);
    V_load(cnt)=R_load*I_load;
    flag_VDC=0;
else
```

```
110     %直流バス電圧V_DCを求める
111     V_DC(cnt)=V_conv(cnt)-R_conv*I_conv(cnt);
112     V_load(cnt)=P_load(cnt,day)/I_load;
113 end
114
115
116 %負荷にかかる電圧を計算、負荷前線路抵抗での電圧降下、V_batt、V_EVの算出
117 V_drop(cnt)=R_line*I_load;
118 V_batt(cnt)=V_DC(cnt)+R_batt*I_batt(cnt);
119 V_EV(cnt)=V_DC(cnt)+R_EV*I_EV(cnt);
120 V_PV(cnt)=V_DC(cnt)+R_PV*I_PV(cnt);
121
122 %商用系統、蓄電池、EVの出力電圧PbatとPevを計算
123 P_conv(cnt)=I_conv(cnt)*V_conv(cnt);
124 P_batt(cnt)=I_batt(cnt)*V_batt(cnt);
125 P_EV(cnt)=I_EV(cnt)*V_EV(cnt);
126 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127
128
```

```
%%%%%%%%%%%%%%%%%%%%コンバータロックモード1%%%%%%%%%%%%%%%%%%%%%%%%
if cnt>=2 && (lock_EV_dch==1 || lock_EV_ch==1)
%%%%%%%%%%%%%%%%%%%%%%%%%EV出力変化制約%%%%%%%%%%%%%%%%%%%%%%%%
    if I_EV(cnt-1)-I_EV(cnt)>I_EV_lim
        I_EV(cnt)=I_EV(cnt-1)-I_EV_lim;
    elseif I_EV(cnt)-I_EV(cnt-1)>I_EV_lim
        I_EV(cnt)=I_EV(cnt-1)+I_EV_lim;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%batt出力電圧の決定
V_batt(cnt)=V_DC_ref+R_batt*I_batt(cnt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%計算したV_battをもとにI_battを潮流計算
%I_batt以外の電流をI_etcとまとめる
I_etc(cnt)=I_PV(cnt)+I_conv(cnt)+I_EV(cnt);
c_I_batt=[R_line+R_batt 2*R_line*I_etc(cnt)+R_batt*I_etc(cnt)-V_batt(cnt) R_line*I_etc(cnt)^2+P_load ↙
(cnt,day)-V_batt(cnt)*I_etc(cnt)];
ans_I_batt=roots(c_I_batt);
%2次方程式の解は2つあるが、値は小さい方を選択（大きい方は線路抵抗に高電圧がかかる）
I_batt(cnt)=ans_I_batt(2);

%得られた電流を保存
I_batt_temp=I_batt(cnt);

%充放電レート評価
if I_batt(cnt)>=I_batt_max
    I_batt(cnt)=I_batt_max;
    fprintf('%d秒、蓄電池放電レート逸脱¥n',cnt);
    flag_VDC=1;
    lock_batt_dch=1;
elseif I_batt(cnt)<=I_batt_min
    I_batt(cnt)=I_batt_min;
    fprintf('%d秒、蓄電池充電レート逸脱¥n',cnt);
    flag_VDC=1;
    lock_batt_ch=1;
else
    flag_VDC=0;
end

I_load=I_etc(cnt)+I_batt(cnt);

if flag_VDC==1
    %充放電レートに抵触していた場合は、制限された
    %蓄電池出力電流を用いて、改めて出力電圧を求める
    %電流並行時の負荷を求めると、
    R_load=P_load(cnt,day)/(I_etc(cnt)+I_batt_temp)^2;
    V_DC(cnt)=(R_line+R_load)*I_load;
    %改めてV_EV、並びに負荷にかかる電圧も求める
    V_batt(cnt)=V_DC(cnt)+R_batt*I_batt(cnt);
    V_load(cnt)=R_load*I_load;
```



```
    flag_VDC=0;
else
    V_load(cnt)=P_load(cnt,day)/I_load;
    %直流バス電圧V_DCを求める
    V_DC(cnt)=V_batt(cnt)-R_batt*I_batt(cnt);
end

%負荷にかかる電圧を計算、負荷前線路抵抗での電圧降下、V_batt、V_EVの算出
V_drop(cnt)=R_line*I_load;
V_conv(cnt)=V_DC(cnt)+R_conv*I_conv(cnt);
V_EV(cnt)=V_DC(cnt)+R_EV*I_EV(cnt);
V_PV(cnt)=V_DC(cnt)+R_PV*I_PV(cnt);

%商用系統、蓄電池、EVの出力電圧PbatとPevを計算
P_conv(cnt)=I_conv(cnt)*V_conv(cnt);
P_batt(cnt)=I_batt(cnt)*V_batt(cnt);
P_EV(cnt)=I_EV(cnt)*V_EV(cnt);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%コンバータロックモード2%%%%%%%%%%%%%%%%%%%%%%%%
if cnt>=2 && (lock_batt_dch==1 || lock_batt_ch==1)
%%%%%%%%%%%%%%%%%%%%%%%%%蓄電池・EV出力変化制約%%%%%%%%%%%%%%%%%%%%%%
    if I_batt(cnt-1)-I_batt(cnt)>I_batt_lim
        I_batt(cnt)=I_batt(cnt-1)-I_batt_lim;
    elseif I_batt(cnt)-I_batt(cnt-1)>I_batt_lim
        I_batt(cnt)=I_batt(cnt-1)+I_batt_lim;
    end

% %%%%%%%%%%%%%%%%SOCに関する制約（max→充電不可　min→放電不可）%%%%%%
%
%     if SOC_batt(cnt-1)>=0.95 && I_batt(cnt)<0
%         I_batt(cnt)=0;
%         mode(cnt,2)=2;
%     elseif SOC_batt(cnt-1)<=0 && I_batt(cnt)>0
%         I_batt(cnt)=0;
%     end
%
%     if lock_batt==1;
%         I_batt(cnt)=0;
%     end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%EV出力電圧の決定
V_EV(cnt)=V_DC_ref+R_EV*I_EV(cnt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%計算したV_EVをもとにI_EVを潮流計算
%I_EV以外の電流をI_etcとまとめる
I_etc(cnt)=I_PV(cnt)+I_conv(cnt)+I_batt(cnt);
c_I_EV=[R_line+R_EV 2*R_line*I_etc(cnt)+R_EV*I_etc(cnt)-V_EV(cnt) R_line*I_etc(cnt)^2+P_load(cnt, ↙
day)-V_EV(cnt)*I_etc(cnt)];
ans_I_EV=roots(c_I_EV);
%2次方程式の解は2つあるが、値は小さい方を選択（大きい方は線路抵抗に高電圧がかかる）
I_EV(cnt)=ans_I_EV(2);

%得られた電流を保存
I_EV_temp=I_EV(cnt);

%EVプラグアウト時は出力0
I_EV(cnt)=(mode(cnt,3)==4)*0+(mode(cnt,3)~=4)*I_EV(cnt);


%充放電レート評価
if I_EV(cnt)>=I_EV_max
    I_EV(cnt)=I_EV_max;
    fprintf('%d秒、EV放電レート逸脱¥n',cnt);
    flag_VDC=1;
    lock_EV_dch=1;
elseif I_EV(cnt)<=I_EV_min
    I_EV(cnt)=I_EV_min;
    fprintf('%d秒、EV充電レート逸脱¥n',cnt);
    flag_VDC=1;
```



```
    lock_EV_ch=1;
else
    flag_VDC=0;
end

I_load=I_etc(cnt)+I_EV(cnt);

if flag_VDC==1
    %充放電レートに抵触していた場合は、制限された
    %EV出力電流を用いて、改めて出力電圧を求める
    %電流並行時の負荷を求めると、
    R_load=P_load(cnt,day)/(I_etc(cnt)+I_EV_temp)^2;
    V_DC(cnt)=(R_line+R_load)*I_load;
    %改めてV_EV、並びに負荷にかかる電圧も求める
    V_EV(cnt)=V_DC(cnt)+R_EV*I_EV(cnt);
    V_load(cnt)=R_load*I_load;
    flag_VDC=0;
else
    V_DC(cnt)=V_EV(cnt)-R_EV*I_EV(cnt);
    V_load(cnt)=P_load(cnt,day)/I_load;
end

%負荷にかかる電圧を計算、負荷前線路抵抗での電圧降下、V_batt、V_EVの算出
V_drop(cnt)=R_line*I_load;
V_conv(cnt)=V_DC(cnt)+R_conv*I_conv(cnt);
V_batt(cnt)=V_DC(cnt)+R_batt*I_batt(cnt);
V_PV(cnt)=V_DC(cnt)+R_PV*I_PV(cnt);

%商用系統、蓄電池、EVの出力電圧PbatとPevを計算
P_conv(cnt)=I_conv(cnt)*V_conv(cnt);
P_batt(cnt)=I_batt(cnt)*V_batt(cnt);
P_EV(cnt)=I_EV(cnt)*V_EV(cnt);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%PV抑制モード%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PCM_flag=0; %モード遷移フラグのリセット
%%%%%%%%%%%%%%%%%%%%%%%%蓄電池およびEVは充電も放電もしない%%%%%%%%%%%%%%%%%%%%
I_batt(cnt)=0; I_EV(cnt)=0;
mode(cnt,2)=2;
mode(cnt,3)=(EV_now(cnt)==1)*2+(EV_now(cnt)==0)*4;

%CONV出力電流の決定（ロック状態）
I_conv(cnt)=I_conv_min;

%PV出力電圧の決定
V_PV(cnt)=V_DC_ref+R_PV*I_PV(cnt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%計算したV_battをもとにI_battを潮流計算
%I_batt以外の電流をI_etcとまとめる
I_etc(cnt)=I_batt(cnt)+I_conv(cnt)+I_EV(cnt);
c_I_PV=[R_line+R_PV 2*R_line*I_etc(cnt)+R_PV*I_PV(cnt)-V_PV(cnt) R_line*I_etc(cnt)^2+P_load(cnt,day) ↙
-V_PV(cnt)*I_etc(cnt)];
ans_I_PV=roots(c_I_PV);
%2次方程式の解は2つあるが、値は小さい方を選択（大きい方は線路抵抗に高電圧がかかる）
I_PV(cnt)=ans_I_PV(2);
P_PV_rest(cnt)=I_PV(cnt)*V_PV(cnt);

%得られた電流を保存
I_PV_temp=I_PV(cnt);

%抑制結果と本来のPV出力の比較
if P_PV_rest(cnt)>P_PV(cnt)
    fprintf('%d秒、抑制後出力が抑制後出力よりも大きいです。供給不足です¥n',cnt);
    I_PV(cnt)=P_PV(cnt)/V_PV(cnt);
    PCM_flag=1;
    flag_VDC=1;
else
    P_PV(cnt)=P_PV_rest(cnt);
    flag_VDC=0;
end

I_load=I_etc(cnt)+I_PV(cnt);

if flag_VDC==1;
    %充放電レートに抵触していた場合は、制限された
    %EV出力電流を用いて、改めて出力電圧を求める
    %電流並行時の負荷を求めると、
    R_load=P_load(cnt,day)/(I_etc(cnt)+I_PV_temp)^2;
    V_DC(cnt)=(R_line+R_load)*I_load;
    %改めてV_PV、並びに負荷にかかる電圧も求める
    V_PV(cnt)=V_DC(cnt)+R_PV*I_PV(cnt);
    V_load(cnt)=R_load*I_load;
    flag_VDC=0;
else
    %直流バス電圧V_DCを求める
    V_DC(cnt)=V_PV(cnt)-R_PV*I_PV(cnt);
```



```
    V_load(cnt)=P_load(cnt,day)/I_load;
end

%負荷にかかる電圧を計算、負荷前線路抵抗での電圧降下、V_batt、V_EVの算出
V_drop(cnt)=R_line*I_load;
V_conv(cnt)=V_DC(cnt)+R_conv*I_conv(cnt);
V_batt(cnt)=V_DC(cnt)+R_batt*I_batt(cnt);
V_EV(cnt)=V_DC(cnt)+R_EV*I_EV(cnt);

%商用系統、蓄電池、EVの出力電圧PbatとPevを計算
P_conv(cnt)=I_conv(cnt)*V_conv(cnt);
P_batt(cnt)=I_batt(cnt)*V_batt(cnt);
P_EV(cnt)=I_EV(cnt)*V_EV(cnt);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%蓄電池・EVのSOCを計算

%理想出力に戻す
if P_batt(cnt)>0
    P_batt_as(cnt)=P_batt(cnt)/X_batt;
else
    P_batt_as(cnt)=P_batt(cnt)*X_batt;
end
if P_EV(cnt)>0
    P_EV_as(cnt)=P_EV(cnt)/X_EV;
else
    P_EV_as(cnt)=P_EV(cnt)*X_EV;
end

%Whに変換
P_batt_wh(cnt)=P_batt_as(cnt)/3600;
P_EV_wh(cnt)=P_EV_as(cnt)/3600;

%%%%%%%%%%%%%%%%%%%%%%%%残存電力量を計算%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------蓄電池について-------------------
%充放電判断　充電の場合
if P_batt(cnt)<=0
    %cntが1のとき
    if cnt==1
        ene_batt(cnt)=ene_batt_init-P_batt_wh(cnt);
    else
        ene_batt(cnt)=ene_batt(cnt-1)-P_batt_wh(cnt);
    end

%放電の場合
elseif P_batt(cnt)>0
    %cntが1のとき
    if cnt==1
        ene_batt(cnt)=ene_batt_init-P_batt_wh(cnt);
    else
        ene_batt(cnt)=ene_batt(cnt-1)-P_batt_wh(cnt);
    end
end

%------------------------EVについて----------------------------------
%充放電判断　充電の場合
if P_EV(cnt)<=0
    %cntが1のとき
    if cnt==1
        ene_EV(cnt)=ene_EV_init-P_EV_wh(cnt);
    else
        ene_EV(cnt)=ene_EV(cnt-1)-P_EV_wh(cnt);
    end
    %放電の場合
elseif P_EV(cnt)>0
    %cntが1のとき
    if cnt==1
        ene_EV(cnt)=ene_EV_init-P_EV_wh(cnt);
    else
        ene_EV(cnt)=ene_EV(cnt-1)-P_EV_wh(cnt);
```



```
    end
end


%EV走行終了後
if cnt~=1
    if EV_now(cnt)-EV_now(cnt-1)==-1
%                       fprintf('%d秒、EV走行開始¥n',cnt);
    elseif EV_now(cnt)-EV_now(cnt-1)==1
        ene_EV(cnt)=ene_EV(cnt)-110*40;
%                       fprintf('%d秒、EV走行終了。4400Wh消費¥n',cnt);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%SOC計算
SOC_batt(cnt)=ene_batt(cnt)/ene_batt_mmax;
SOC_EV(cnt)=ene_EV(cnt)/ene_EV_max;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%PV、負荷に関する設定%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PCS_PV=0.96;
DC_DC_load=0.94;
AC_DC=0.95;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%蓄電池、EVに関する設定%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X_batt=0.95;
X_EV=0.95; %充放電効率の設定

ene_batt_mmax=25000*0.8;
% ene_batt_mmin=0;
ene_batt_max=ene_batt_mmax;
ene_batt_min=ene_batt_mmax*0.3;
ene_EV_max=16000*0.8;  %蓄電池、EVの最大貯蔵電力量設定[Wh]
ene_EV_min=16000*0.3;
% ene_EV_max=16000;  %蓄電池、EVの最大貯蔵電力量設定[Wh]
% ene_EV_min=0;
ene_batt_init=SOC_batt_init;
ene_EV_init=SOC_EV_init;
SOC_batt_min=ene_batt_min/ene_batt_mmax;
SOC_batt_max=ene_batt_max/ene_batt_mmax;

SOC_batt_init=ene_batt_init/ene_batt_mmax;
SOC_EV_init=ene_EV_init/ene_EV_max; %蓄電池、EVのシミュレーション開始時SOC

SOC_EV_max=0.8;
SOC_EV_min=0.3;

SOC_batt=zeros(T,1); SOC_EV=zeros(T,1); %初期化
SOC_batt(1)=SOC_batt_init; SOC_EV(1)=SOC_EV_init;

%充放電レート制約
P_batt_max=10000*X_batt;
I_batt_max=P_batt_max/380;
I_batt_min=-P_batt_max/380;

P_EV_max=3000*X_EV;
I_EV_max=P_EV_max/380;
I_EV_min=-P_EV_max/380;

%EVが存在しているか否か。1ならEVいる
EV_now=zeros(T,1);

%出力変化上限値
I_batt_lim=100/380;
I_EV_lim=100/380;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

V_conv_init=381; %Vconvの初期値
V_batt_init_=380;
P_conv_max=30000; %マージンをとったコンバータ最大容量
I_conv_max=P_conv_max/380;
P_conv_min=500; %マージンをとったコンバータ最小容量
I_conv_min=P_conv_min/380;

```



```
P_PV=zeros(T,1);

violation_batt=zeros(T,1);
violation_EV=zeros(T,1);

R_line=0.1; %線路インピーダンス
R_conv=R_line; R_batt=R_line; R_EV=R_line; R_PV=R_line;
Vconv=[]; Iconv=[]; Vdc=[]; Vload=[];
outcnt=0; %コンバータ容量逸脱回数を記録

SOCflag=1;
Modification_flag=0; %計画修正フラグ初期化
time_Piecewise=0; %計画修正経過時間修正
count=0;
modi_count=0; %計画修正回数をカウント


```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ある期間分回すシミュレーション
% SOCは前の日の最終値を引き継ぎ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
%必要な諸データの読み込み
load('matlab_revised.mat');
EV=csvread('EV.csv');
%削減後のシナリオ数
n=10;
%%日付と予報の読み込み（天気予報のあるもののみ）
date_weather=xlsread('Weather.xlsx');
%負荷データの読み込み(とりあえず1週間分)
load_buff(:,:,1)=xlsread('需要データ.xlsx','10_29');
load_buff(:,:,2)=xlsread('需要データ.xlsx','10_30');
load_buff(:,:,3)=xlsread('需要データ.xlsx','10_31');
load_buff(:,:,4)=xlsread('需要データ.xlsx','11_01');
load_buff(:,:,5)=xlsread('需要データ.xlsx','11_02');
load_buff(:,:,6)=xlsread('需要データ.xlsx','11_03');
load_buff(:,:,7)=xlsread('需要データ.xlsx','11_04');
i=1:7;
load=load_buff(:,1,i)+load_buff(:,2,i);
load=reshape(load,48,7);

load=horzcat(load,load);
load=horzcat(load,load);
load=horzcat(load,load);
load=horzcat(load,load);
%平日の負荷予測値データ
load_week=(load(:,1)+load(:,2)+load(:,3)+load(:,4)+load(:,5))/5;
%休日の負荷予測値データ
load_hol=(load(:,6)+load(:,7))/2;

SOC_batt_max_hardware=25000;
SOC_EV_max_hardware=16000;

%% データベース読み込み
%天気データを読み込む
weather = xlsread('日射量データベース.xlsx');
weather(:,1)=weather(:,1)+datenum('30-Dec-1899');
%日射量データを読み込む
radiation(:,1)=weather(:,1);
for num=1:24
    radiation(:,num+1)=xlsread('日射量データベース.xlsx',num+1,'B:B');
end
%ブランクのある日を削除
num=1;
day_blank=zeros(1,1);
while num<=numel(radiation)/25
    if any(isnan(radiation(num,:)))==1
        radiation(num,:)=[];
        weather(num,:)=[];
        day_blank=vertcat(day_blank,radiation(num,1)-1);
        num=num-1;
    end
    num=num+1;
end

day=1;
while day<=numel(date_weather)/5
    num=1;
    while num<=numel(day_blank)
        if date_weather(day,1)+datenum('30-Dec-1899')==day_blank(num,1);
            date_weather(day,:)=[];
            num=num-1;
        end
    num=num+1;
    end
    day=day+1;
end


%シミュレーション期間の決定，データのある14日間とする
%期間：2013年6月19日から7月17日まで
day=1;
while day<=numel(date_weather)/5
    if date_weather(day,1)+datenum('30-Dec-1899')<datenum('19-Jun-2013') || date_weather(day,1) ↙
+datenum('30-Dec-1899')>datenum('17-Jul-2013')
            date_weather(day,:)=[];
            day=day-1;
    end
    day=day+1;
end

%負荷3秒値の読み込み
demand_100V_3sec=xlsread('load_3sec.xlsx');
demand_200V_3sec=xlsread('load_3sec.xlsx','200V');
demand_3sec=demand_100V_3sec+demand_200V_3sec;
demand_3sec(1,:)=demand_3sec(1,:)/2;
%コピー
num=1;
while num<numel(demand_3sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1)
    demand(num+2*(num-1),:)=demand_3sec(num,:);
    demand(num+2*num-1,:)=demand_3sec(num,:);
    demand(num+2*num,:)=demand_3sec(num,:);
    num=num+1;
end
%最後の1分間はデータがないので，最後のデータをコピー
for t=1:63
    demand(86340+t,:)=demand(86340,:);
end

size=(numel(demand)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1));
num=1;
while num<=size
    if num>1
        %20%の誤差を載せる
        demand(num,:)=demand(num,:)*(1+0.4*(-0.5+rand));
    end
```

```
    num=num+1;
end
demand(2,:)=[];
demand(2,:)=[];

%PVデータの作成
%日射量10秒データの読み込み
radiation_10sec=xlsread('rad_10sec.xlsx');
%コピー
num=1;
while num<numel(radiation_10sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1)
    radiation_1sec(1+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(2+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(3+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(4+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(5+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(6+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(7+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(8+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(9+(num-1)*10,:)=radiation_10sec(num,:);
    radiation_1sec(10+(num-1)*10,:)=radiation_10sec(num,:);
    num=num+1;
end
%最後の10秒は前のデータを利用
radiation_1sec(1+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(2+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(3+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(4+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(5+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(6+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(7+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(8+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(9+(num-1)*10,:)=radiation_10sec(num-1,:);
radiation_1sec(10+(num-1)*10,:)=radiation_10sec(num-1,:);

%20%の誤差を載せる
size=(numel(radiation_1sec)/(datenum('17-Jul-2013')-datenum('19-Jun-2013')+1));
num=1;
while num<=size
    if num>1
        %20%の誤差を載せる
        radiation_1sec(num,:)=radiation_1sec(num,:)*(1+0.4*(-0.5+rand));
    end
    num=num+1;
end
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];
radiation_1sec(2,:)=[];

```



```
%PV出力に変換
%130.9:20kWパネルの面積(m2), 0.15:パネルの変換効率
PV_1sec=130.9*PCS*0.15*radiation_1sec;
PV_1sec(1,:)=PV_1sec(1,:)/(130.9*PCS*0.15);
PV_act=PV_1sec;
PV_act(1,:)=[];
%データを修正 負の値および定格を超える値を除去
for day=1:datenum('17-Jul-2013')-datenum('19-Jun-2013')
    for time=1:(24*60*60)
        if PV_act(time,day)<0
            PV_act(time,day)=0;
        elseif PV_act(time,day)>20000
            PV_act(time,day)=20000;
        end
    end
end

%シミュレーション期間外のデータを排除
num=1;
while num<=numel(PV_1sec)/((24*60*60)+1)
    flag=0;
    day=1;
    while day<=numel(date_weather)/5
        if PV_1sec(1,num)==date_weather(day,1)
            flag=1;
        end
        day=day+1;
    end
    if flag==0
        demand(:,num)=[];
        PV_1sec(:,num)=[];
        PV_act(:,num)=[];
        num=num-1;
    end
    num=num+1;
end

%%ここからシミュレーション開始
%0時→6時で充電, 6時→18時で放電, 18時→0時で充電
PV_rjc_cnt=0;
PV_rjc=0;
day=1;
while day<=numel(date_weather)/5
    %1日目のみSOCが50%からスタート
    %後は最終値を引き継ぎ
    if day==1
        SOC_batt_act(1,1)=SOC_batt_init;
        SOC_EV_act(1,1)=SOC_EV_init;
    else
        SOC_batt_init=SOC_batt_act(1,day);
        SOC_EV_init=SOC_EV_act(1,day);
    end

    %EV運用に関して
    %コピー
```

```
    num=1;
    while num<=numel(EV)/2
        EV_30min=EV(num,:);
        EV_1sec_buff=EV_30min;
        for t=1:1799
            EV_1sec_buff=vertcat(EV_1sec_buff,EV_30min);
        end
        if num==1
            EV_1sec=EV_1sec_buff;
        else
            EV_1sec=vertcat(EV_1sec,EV_1sec_buff);
        end
        num=num+1;
    end

    num=1;
    while num<numel(EV_1sec)/2-1
        if (EV_1sec(num,1)~=1) || (EV_1sec(num+1,1)~=0)
            EV_1sec(num+1,2)=0;
        end
        num=num+1;
    end

    %休日は常時接続
    if weekday(date_weather(day,1)+datenum('30-Dec-1899'))==1 || weekday(date_weather(day,1) ↙
+datenum('30-Dec-1899'))==7
        EV_1sec(:,1)=1;
        EV_1sec(:,2)=0;
    end

    P_demand=demand(:,day);
    P_demand(1)=[];
    PL=load_week;
    date_act=date_weather(day,1)+datenum('30-Dec-1899');
    weather_num=date_weather(day,2:5);

    %計画
    %蓄電池およびEVの計画出力を決める(30分値)
    %現在SOCが満充電になるまで充電
    %scenario_2;
    %weather_pre_day(:,:)=weather_pre;
    %Piecewise_prob;
    SOC_batt_buff=SOC_batt_max-SOC_batt_act(1,day);
    SOC_EV_buff=SOC_EV_max-SOC_EV_act(1,day);
    %SOCが通常運転のSOCを超えない場合
    if SOC_batt_buff>0
        for t=1:12
            batt_C_sim(t,day)=SOC_batt_buff/6;
            batt_D_sim(t,day)=0;
            if EV(t,1)==1
                if SOC_EV_buff>0
                    EV_C_sim(t,day)=(SOC_EV_buff-110*EV(t,2))/6;
                    EV_D_sim(t,day)=0;
                else
                    EV_C_sim(t,day)=0;
```



```
                    EV_D_sim(t,day)=0;
                end
            else
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=0;
            end
        end
        for t=13:36
            batt_C_sim(t,day)=0;
            batt_D_sim(t,day)=SOC_batt_max/12;
            if EV(t,1)==1
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=(SOC_EV_max-110*EV(t,2))/3;
            else
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=0;
            end
        end
        for t=37:48
            batt_C_sim(t,day)=0.5*SOC_batt_max/6;
            batt_D_sim(t,day)=0;
            if EV(t,1)==1
                EV_C_sim(t,day)=(0.5*SOC_EV_max)/6;
                EV_D_sim(t,day)=0;
            else
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=0;
            end
        end
    else
       for t=1:12
            batt_C_sim(t,day)=0;
            batt_D_sim(t,day)=0;
            if EV(t,1)==1
                if SOC_EV_buff>0
                    EV_C_sim(t,day)=(SOC_EV_buff-110*EV(t,2))/6;
                    EV_D_sim(t,day)=0;
                else
                    EV_C_sim(t,day)=0;
                    EV_D_sim(t,day)=0;
                end
            else
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=0;
            end
        end
        for t=13:36
            batt_C_sim(t,day)=0;
            batt_D_sim(t,day)=SOC_batt_act(1,day)/12;
            if EV(t,1)==1
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=SOC_EV_max/12;
            else
                EV_C_sim(t,day)=0;
                EV_D_sim(t,day)=0;
```

```
329             end
330         end
331         for t=37:48
332             batt_C_sim(t,day)=0.5*SOC_batt_max_hardware/6;
333             batt_D_sim(t,day)=0;
334             if EV(t,1)==1
335                 EV_C_sim(t,day)=(0.5*SOC_EV_max_hardware)/6;
336                 EV_D_sim(t,day)=0;
337             else
338                 EV_C_sim(t,day)=0;
339                 EV_D_sim(t,day)=0;
340             end
341         end
342     end
343
344     %1秒値の指令値に変換
345     buff_batt_C=reshape(batt_C_sim(:,day),1,T);
346     buff_batt_D=reshape(batt_D_sim(:,day),1,T);
347     buff_EV_C=reshape(EV_C_sim(:,day),1,T);
348     buff_EV_D=reshape(EV_D_sim(:,day),1,T);
349     for t=1:(60*30)-1
350         buff_batt_C=vertcat(buff_batt_C,reshape(batt_C_sim(:,day),1,T));
351         buff_batt_D=vertcat(buff_batt_D,reshape(batt_D_sim(:,day),1,T));
352         buff_EV_C=vertcat(buff_EV_C,reshape(EV_C_sim(:,day),1,T));
353         buff_EV_D=vertcat(buff_EV_D,reshape(EV_D_sim(:,day),1,T));
354     end
355     %コピー
356     batt_C_act(:,day)=reshape(buff_batt_C,T*30*60,1);
357     batt_D_act(:,day)=reshape(buff_batt_D,T*30*60,1);
358     EV_C_act(:,day)=reshape(buff_EV_C,T*30*60,1);
359     EV_D_act(:,day)=reshape(buff_EV_D,T*30*60,1);
360
361     %実際のPVデータを用いたシミュレーション
362     L_L_act=((V_DC*V_DC/R_L-2*P_demand/DC_DC)-sqrt((V_DC*V_DC/R_L-2*P_demand/DC_DC).* ↙
(V_DC*V_DC/R_L-2*P_demand/DC_DC)-4*(P_demand.*P_demand/(DC_DC*DC_DC))))/2;
363     P_demand_net=P_demand/DC_DC+L_L_act;
364     L_PV_act=((2*PV_act+V_DC*V_DC/R_PV)-sqrt((2*PV_act+V_DC*V_DC/R_PV).*(2*PV_act+V_DC*V_DC/R_PV)-4 ↙
*PV_act))/2;
365
366     %1秒シミュレーション
367     %挙動は30分と同様
368     for time=1:24*60*60
369         %PVの余剰電力あり
370         if PV_act(time,day)-L_PV_act(time,day)>P_demand_net(time)
371             %蓄電池SOCの余裕計算
372             SOC_batt_act_buff=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)+(1/(60*60))* ↙
(PV_act(time,day)-L_PV_act(time,day)-P_demand_net(time));
373             %蓄電池SOC余裕なしのとき
374             if SOC_batt_act_buff>SOC_batt_max_hardware
375                 %蓄電池SOCのギリギリまで充電
376                 batt_C_act(time,day)=min((60*60)*(SOC_batt_max_hardware-SOC_batt_act(time,day)), ↙
10000);
377                 batt_D_act(time,day)=0.0;
378                 %EVの充放電を停止
379 %                 EV_C_act(time)=0.0;
```



```
380                 EV_D_act(time)=0.0;
381                 %蓄電池およびEVの線路損失を計算
382     %                 L_batt_act(time,day)=(2*batt_C_act(time,day)/(eff_batt/AC_DC)+ ↙
(V_DC*V_DC/R_batt))-sqrt((2*batt_C_act(time,day)/(eff_batt/AC_DC)+(V_DC*V_DC/R_batt))*((2*batt_C_act ↙
(time,day)/(eff_batt/AC_DC)+(V_DC*V_DC/R_batt))-4*(AC_DC/eff_batt)*(AC_DC/eff_batt)*batt_C_act(time, ↙
day)*batt_C_act(time,day)))/2;
383                 L_batt_act(time,day)=((...
384                                        2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt) ...
385                                       )-...
386                                       sqrt(...
387                                       (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))* ...
388                                       (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))- ...
389                                       4*(1/eff_batt)*(1/eff_batt)*batt_C_act(time,day)*batt_C_act ↙
(time,day))...
390                                       )/2;
391                 L_EV_act(time,day)=((...
392                                      2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
393                                      sqrt(...
394                                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
+(V_DC*V_DC/R_EV))*...
395                                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
+(V_DC*V_DC/R_EV))-...
396                                      4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))) ...
397                                      )/2;
398                 %PV出力を抑制する
399                 PV_rjc_cnt=PV_rjc_cnt+1;
400                 PV_rjc=PV_rjc+PV_act(time,day)-(P_demand_net(time)+L_PV_act(time,day)+(batt_C_act ↙
(time,day)+L_batt_act(time,day))/(eff_batt)+(EV_C_act(time,day)+L_EV_act(time,day))/(eff_EV));
401                 PV_act(time,day)=P_demand_net(time)+L_PV_act(time,day)+(batt_C_act(time,day) ↙
+L_batt_act(time,day))/(eff_batt)+(EV_C_act(time,day)+L_EV_act(time,day))/(eff_EV);
402                 P_UG(time,day)=0.0;
403             %蓄電池SOC余裕ありの場合
404             else
405                 batt_C_act(time,day)=(eff_batt)*(PV_act(time,day)-L_PV_act(time,day)-P_demand_net ↙
(time));
406                 L_batt_act(time,day)=((...
407                                        2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt) ...
408                                       )-...
409                                       sqrt(...
410                                       (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))* ...
411                                       (2*batt_C_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))- ...
412                                       4*(1/eff_batt)*(1/eff_batt)*batt_C_act(time,day)*batt_C_act ↙
(time,day))...
413                                       )/2;
414                 batt_D_act(time,day)=0.0;
415                 L_EV_act(time,day)=((...
416                                      2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
417                                      sqrt(...
418                                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
+(V_DC*V_DC/R_EV))*...
419                                      (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
+(V_DC*V_DC/R_EV))-...
```

```
                                    4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV)) ↙
*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))) ...
                                    )/2;
                P_UG_act(time,day)=0.0;
            end
        %余剰電力なしの場合は、購入電力および蓄電池で調整
        else
            %逆潮流が起きそうなときは蓄電池放電をやめる
            if EV_D_act(time,day)*(eff_EV)+batt_D_act(time,day)*(eff_batt)+PV_act(time,day) ↙
>P_demand_net(time)+L_PV_act(time)+EV_C_act(time,day)/(eff_EV)+batt_C_act(time,day)/(eff_batt)
                P_UG_act(time,day)=0.0;
    %               L_L_act(time,day)=0.0;
                batt_D_act(time,day)=(P_demand_net(time)-PV_act(time,day)-L_PV_act(time,day)- ↙
EV_D_act(time,day)*(eff_EV))/(eff_batt);
                %まだ逆潮流が発生する場合にはEVからの放電を減らす
                if batt_D_act(time,day)<0
                    EV_D_act(time,day)=EV_D_act(time,day)*(eff_EV)+(eff_batt)*batt_D_act(time,day);
                    batt_D_act(time,day)=0.0;
                end
                L_batt_act(time,day)=((...
                       2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt) ...
                      )-...
                      sqrt(...
                      (2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))* ...
                      (2*batt_D_act(time,day)/(eff_batt)+(V_DC*V_DC/R_batt))- ...
                      4*(eff_batt)*(eff_batt)*batt_D_act(time,day)*batt_D_act(time,day)) ...
                      )/2;
                L_EV_act(time,day)=((...
                    2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
                    sqrt(...
                    (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))*...
                    (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
                    4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))*abs(EV_C_act ↙
(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))) ...
                    )/2;
            else
                %SOCの下限に抵触しているかどうか
                SOC_batt_act_buff=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)-(1/ ↙
(60*60))*batt_D_act(time,day);
                if SOC_batt_act_buff<0
                    batt_D_act(time,day)=0.0;
                end
                %SOCの上限に抵触しているかどうか
                if SOC_batt_act_buff>SOC_batt_max_hardware
                    batt_C_act(time,day)=0.0;
                end
                SOC_EV_act_buff=SOC_EV_act(time,day)+(1/(60*60))*EV_C_act(time,day)-(1/(60*60)) ↙
*EV_D_act(time,day);
                %SOCの下限に抵触しているかどうか
                if SOC_EV_act_buff<0
                    EV_D_act(time,day)=0.0;
                end
```



```
                %SOCの上限に抵触しているかどうか
                if SOC_EV_act_buff>SOC_EV_max_hardware
                    EV_C_act(time,day)=0.0;
                end
                L_batt_act(time,day)=((...
                   2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt)...
                  )-...
                  sqrt(...
                  (2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt))*...
                  (2*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))+ ↙
(V_DC*V_DC/R_batt))-...
                  4*abs(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))*abs ↙
(batt_C_act(time,day)/(eff_batt)-batt_D_act(time,day)*(eff_batt))) ...
                  )/2;
                L_EV_act(time,day)=((...
                    2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
                    sqrt(...
                    (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))*...
                    (2*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))+ ↙
(V_DC*V_DC/R_EV))-...
                    4*abs(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))*abs(EV_C_act ↙
(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))) ...
                    )/2;
                P_UG_act(time,day)=P_demand_net(time)+(batt_C_act(time,day)/(eff_batt)-batt_D_act ↙
(time,day)*(eff_batt))+(EV_C_act(time,day)/(eff_EV)-EV_D_act(time,day)*(eff_EV))-PV_act(time,day) ↙
+L_PV_act(time,day);
            end
        end
        SOC_batt_act(time+1,day)=SOC_batt_act(time,day)+(1/(60*60))*batt_C_act(time,day)-(1/ ↙
(60*60))*batt_D_act(time,day);
        SOC_EV_act(time+1,day)=SOC_EV_act(time,day)+(1/(60*60))*EV_C_act(time,day)-(1/(60*60)) ↙
*EV_D_act(time,day)-110*EV_1sec(time,2);
    end
    SOC_batt_act(1,day+1)=SOC_batt_act(24*60*60+1,day);
    SOC_EV_act(1,day+1)=SOC_EV_act(24*60*60+1,day);
    day=day+1;
end

%期間でのCO2排出量の計算
day=1;
while day<=numel(date_weather)/5
% for day=1:1
    for time=1:24*60*60
        if time<60*60*6
            CO2(time,day)=0.469*P_UG_act(time,day)/(1000*60*60)/(AC_DC);
        elseif time>=60*60*6||time<60*60*18
            CO2(time,day)=0.69*P_UG_act(time,day)/(1000*60*60)/(AC_DC);
        else
            CO2(time,day)=0.469*P_UG_act(time,day)/(1000*60*60)/(AC_DC);
        end
    end
```

```
509      CO2_day(1,day)=sum(CO2(:,day));
510      day=day+1
511 end
512 CO2_total=sum(CO2);
```

```
%%CO2排出量25%削減に必要なMGの構成を決定する
clear all
%%ACMGかDCMGか  AC=1, DC=2
% Sup=1;
%%EVを何台導入するか
EV_inst=0;
for Sup=1:2
    for compo=1:16    %%どの季節についてシミュレーションするか 春=1、夏=2、秋=3、冬=4
    %     for season=1:4
        if compo~=1 && compo~=5 && compo~=6 && compo~=14 && compo~=16
            continue;
        end
        season=2;
        clearvars -except season EV_inst Sup compo
        %%データ読み込み
        demand_component=dec2bin(compo-1,4);
        data_import;

        %%CPLEXによる混合整数計画
        if Sup==1
            Optimize_AC;
        elseif Sup==2
            Optimize_DC;
        end

        %%各種コストに分解
        PV_inst_cost=c_inst_PV*sim_size*PV_install;
        batt_inst_cost=c_inst_batt*sim_size*battery_install;
         if Sup==1
            DCL_cost=sim_size*max(max(demand_DC))*c_conv/1000;
            ACL_cost=0;
            operation_cost=fval-PV_inst_cost-batt_inst_cost;
        else
            DCL_cost=0.0;
            ACL_cost=sim_size*max(max(demand_AC))*c_conv*2/1000;
            conv_inst_cost=sim_size*c_conv*conv_install;
            operation_cost=fval-PV_inst_cost-batt_inst_cost-conv_inst_cost;
        end

        if Sup==1
            save(strcat(num2str(compo),strcat(num2str(season), '_result_noEV_AC_sp_conv_ACDC_40. ↙
mat')));
        elseif Sup==2
            save(strcat(num2str(compo),strcat(num2str(season), '_result_noEV_DC_sp_conv_40.mat')));
        end
        % save(strcat(num2str(season),'_resultnoEV.mat'));
        %%データ出力
    %     end
    end
end
```

```
%%混合整数計画を解いて、ACMGにおける最適容量を得る
clear f_buff
clear A_buff b_buff Aeq_buff beq_buff ub_buff lb_buff
clear A_buff_buff b_buff_buff Aeq_buff_buff beq_buff_buff
clear big_eyes ub lb
clear ctype

%まず、負荷をAC計算用に割り戻す
clear demand
if bin2dec(demand_component(1,1))==0
    demand_base=demand_base*(AC_DC*DC_DC);
else
    demand_base=demand_base/(AC_DC);
end
if bin2dec(demand_component(1,2))==0
    demand_light=demand_light*(AC_DC*DC_DC);
else
    demand_light=demand_light/(AC_DC);
end
if bin2dec(demand_component(1,3))==0
    demand_air=demand_air*(AC_DC*DC_DC);
else
    demand_air=demand_air/(AC_DC);
end
if bin2dec(demand_component(1,4))==0
    demand_other=demand_other*(AC_DC*DC_DC);
else
    demand_other=demand_other/(AC_DC);
end
demand=demand_base+demand_light+demand_air+demand_other;

%ACMGでは、AC/DCコンバータの分だけ、コストが値上がりする
%蓄電池1台あたりの価格(1日)
c_inst_batt=270+3*c_DG_AC_DC/2;
%PVパネル1枚あたりの価格(1日)
c_inst_PV=25+(c_DG_AC_DC/8);

%%変数の種類を指定(実数、バイナリ変数、整数)
%実数変数
%UG_buy=1,batt_C=2,batt_D=3,EV_C=4,EV_D=5
%バイナリ変数
%batt_bin=6,EV_bin=7
%binが1の時、MG内の需要が増える方向、すなわち充電
%整数変数
%PV導入枚数=8、蓄電池導入台数=9
%(EVは有無を考えて2回シミュレーションをやる)
%%シミュレーション期間の決定

%デバッグ時は5日でやる
sim_size=numel(PV)/T;
% sim_size=30;
num=1:numel(PV)/T;
PV(:,num>sim_size)=[];
num=1:numel(demand)/T;
demand(:,num>sim_size)=[];
```



```
%sim_size=numel(PV)/T

% %導入前CO2排出量の計算
% CO2=zeros(T,sim_size);
% time=1:T;
% CO2(time<=24,:)=(0.69/1000);
% time=1:T;
% CO2(time>24,:)=(0.469/1000);
% CO2=CO2.*(demand);
% CO2_nonMG=sum(sum(CO2));
%整数変数の指定
% 'B','I','C','S','N'
%それぞれバイナリ、整数、実数、半実数、半整数

%イメージ　30分の電力やり取り×日数、そのあと導入台数
%購入電力、充放電を定義
vr=1:5*T*sim_size;
ctype(vr)='C';
%充放電方向変数
vr=5*T*sim_size+1:(5*T*sim_size)+2*T*sim_size;
ctype(vr)='B';
%導入台数
vr=(5*T*sim_size)+2*T*sim_size+1:(5*T*sim_size)+2*T*sim_size+2;
ctype(vr)='I';

%%目的関数
%運用コスト+導入コストの最小化
%電気料金は昼間(午前7時~午後11時)の電力料金28.08円/kWh、それ以外の時間帯の料金14.13円/kWh
%起点は午前6時

%まず1日分の電気料金を作る(実数部分)
vr=1:T;
f_buff(1,vr)=((vr>=3 & vr<35)*28.08+(vr<3 | vr>=35)*14.13)/(1000);
%それ以外の変数は電気料金に関係ない
f_buff=horzcat(f_buff,zeros(1,4*T));

%シミュレーション期間分だけコピー
f_buff=repmat(f_buff,1,sim_size);
%充放電方向変数も関係ない
f_buff=horzcat(f_buff,zeros(1,2*T*sim_size));
%導入料金
f_buff=horzcat(f_buff,c_inst_PV*sim_size);
f=horzcat(f_buff,c_inst_batt*sim_size);
clear f_buff
%%等式制約
%電力の需給一致制約
%まず1日分を作る
%UGからの購入電力
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=1;
Aeq_buff=Aeq_buff.*eye(T,T);
%蓄電池充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=-1/(eff_batt*DC_DC*AC_DC);
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
```

```
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%蓄電池放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=eff_batt*DC_DC*AC_DC;
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%EVの充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=-1/(eff_EV*DC_DC*AC_DC);
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%EVの放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=eff_EV*DC_DC*AC_DC;
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;

%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
Aeq_buff=repmat(Aeq_buff,sim_size,sim_size);
Aeq_buff=Aeq_buff.*big_eyes;
clear big_eyes

%バイナリ変数
Aeq_buff_buff=zeros(sim_size*T,2*T);
Aeq_buff_buff=repmat(Aeq_buff_buff,1,sim_size);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%導入台数
%PVの出力は台数に依存
Aeq_buff_buff=ones(sim_size*T,1);
Aeq_buff_buff=reshape(PV*1000,[],1).*Aeq_buff_buff;
Aeq_buff_buff=horzcat(Aeq_buff_buff,zeros(sim_size*T,1));
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
Aeq=PCS*AC_DC*Aeq_buff;
clear Aeq_buff Aeq_buff_buff;

%右辺
beq_buff=demand(:,1:sim_size);
beq=reshape(beq_buff,1,[]);
clear Aeq_buff;
clear beq_buff;

%EV走行中の充電なし
if EV_inst~=0
```



```
    %EVの充放電に関する部分
    Aeq_buff=not(EV_30min(:,1));
    Aeq_buff=repmat(Aeq_buff,1,T);
    Aeq_buff=Aeq_buff.*repmat(eye(T,T),sim_size,1);
    Aeq_buff=horzcat(Aeq_buff,-Aeq_buff);
    Aeq_buff=repmat(Aeq_buff,1,sim_size);
    %日数分に増やす
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    Aeq_buff=Aeq_buff.*big_eyes;
    %EV以外の部分を連結する
    for num=1:sim_size
        %Aeq_buff_buff(:,:,num)=horzcat(zeros(T,3*T),Aeq_buff((num-1)*2*T+1,num*2*T));
        Aeq_buff_buff(:,:,num)=horzcat(zeros(T*sim_size,3*T),Aeq_buff(:,(num-1)*2*T+1:num*2*T));
%         Aeq_buff_buff(:,:,num)=horzcat(zeros(T*sim_size,3*T),Aeq_buff_buff(:,:,num));
    end
    clear Aeq_buff

    Aeq_buff=reshape(Aeq_buff_buff,sim_size*T,sim_size*T*5);
    clear big_eyes Aeq_buff_buff

    %バイナリ変数
    Aeq_buff_buff=zeros(T,2*T);
    Aeq_buff_buff=repmat(Aeq_buff_buff,sim_size,sim_size);
    Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
    clear Aeq_buff_buff;
    %導入台数
    Aeq_buff_buff=zeros(T*sim_size,2);
    Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
    Aeq=vertcat(Aeq,Aeq_buff);
    beq_buff=zeros(1,size(Aeq_buff,1));
    clear Aeq_buff Aeq_buff_buff;

    %右辺
    beq=horzcat(beq,beq_buff);
    clear beq_buff;
end


% %EV走行中の充電なし
% if EV_inst~=0
%     %1日分をまず作る
%     %EVの充放電に関する部分
%     Aeq_buff=reshape((EV_30min(:,1)==0),1,T);
%     Aeq_buff=repmat(Aeq_buff,T,1);
%     Aeq_buff=Aeq_buff.*eye(T,T);
%     Aeq_buff=horzcat(Aeq_buff,-Aeq_buff);
%     %EV以外の部分を連結する
%     Aeq_buff=horzcat(zeros(T,3*T),Aeq_buff);
```

```
%     %日数分に増やす
%     %シミュレーション日数分に拡大
%     %単位行列の大きなものを作る
%     big_eyes=zeros(T*sim_size,T*5*sim_size);
%     for num=1:sim_size
%         [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
%         big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
%     end
%     %単位行列とかけて、関係ない日を削除
%     Aeq_buff=repmat(Aeq_buff,sim_size,sim_size);
%     Aeq_buff=Aeq_buff.*big_eyes;
%     clear big_eyes
%
%     %バイナリ変数
%     Aeq_buff_buff=zeros(T,2*T);
%     Aeq_buff_buff=repmat(Aeq_buff_buff,sim_size,sim_size);
%     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
%     clear Aeq_buff_buff;
%     %導入台数
%     Aeq_buff_buff=zeros(T*sim_size,2);
%     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
%     Aeq=vertcat(Aeq,Aeq_buff);
%     beq_buff=zeros(1,size(Aeq_buff,1));
%     clear Aeq_buff Aeq_buff_buff;
%
%     %右辺
%     beq=horzcat(beq,beq_buff);
%     clear beq_buff;
% end


%%不等式制約
%SOCの最大値制約
%イメージ　蓄電池充放電の部分を作り、その後その他を挿入していく
A_buff_buff=-ones(1,2*T);
time=1:T;
A_buff_buff(time)=-A_buff_buff(time);
A_buff_buff=repmat(A_buff_buff,T,1);
%大きな三角行列を作る
aaa=eye(T,T)+tril(ones(T,T),-1);
%一日分の充放電
A_buff_buff=A_buff_buff.*repmat(aaa,1,2);

%日数分に増やす
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear big_eyes
```



```
%前日までの充放電(全時間帯1,-1)を加える
clear A_buff_buff
A_buff_buff=-ones(1,2*T);
time=1:T;
A_buff_buff(time)=-A_buff_buff(time);
A_buff_buff=repmat(A_buff_buff,T*sim_size,sim_size);
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i<=(num-1)*T*2 & j>(num-1)*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=A_buff+A_buff_buff;
clear big_eyes A_buff_buff
%蓄電池以外の変数を連結
for num=1:sim_size
    clear A_buff_buff_buff
    A_buff_buff_buff=A_buff(:,(num-1)*T*2+1:num*T*2);
    A_buff_buff_buff=horzcat(A_buff_buff_buff,zeros(size(A_buff_buff_buff,1),3*T));
    if num==1
        A_buff_buff=A_buff_buff_buff;
    else
        A_buff_buff=horzcat(A_buff_buff,A_buff_buff_buff);
    end
end
A_buff_buff=horzcat(zeros(T*sim_size,T),A_buff_buff);
clear A_buff
A_buff=A_buff_buff;
A_buff(:,5*sim_size*T+1:5*sim_size*T+T)=[];
clear A_buff_buff
%バイナリ変数を接続
A_buff=horzcat(A_buff,zeros(T*sim_size,2*T*sim_size));
%30分単位の計画なので、すべての変数に0.5時間をかける
A_buff=0.5*A_buff;
%整数変数を接続
%SOC最大値やSOC初期値は導入台数に依存
A_buff=horzcat(A_buff,zeros(T*sim_size,1));
A_buff=horzcat(A_buff,-1000*kWh_batt_unit*ones(T*sim_size,1));
time=1:sim_size*T;
b_buff(time,1)=0;

A=A_buff;
b=b_buff;
clear b_buff

%SOCの最小値制約
%最大値と方向を逆にする
A_buff=-A_buff;
%導入台数と最小値の関係だけは変化
A_buff(:,size(A_buff,2))=0;
% A_buff=horzcat(A_buff,zeros(T*sim_size,1));
time=1:sim_size*T;
```

```
b_buff(time,1)=0;
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff

%EVがあるときのみ考えるEVの制約
if EV_inst~=0
    %EVのSOC最大値制約 EV走行中でないとき
    %イメージ　EV充放電の部分を作り、その後その他を挿入していく
    A_buff_buff=-ones(1,2*T);
    time=1:T;
    A_buff_buff(time)=-A_buff_buff(time);
    A_buff=repmat(A_buff_buff,sim_size,sim_size);
    A_buff_buff=repmat(A_buff_buff,T,1);
    %大きな三角行列を作る
    aaa=eye(T,T)+tril(ones(T,T),-1);
    %一日分の充放電
    A_buff_buff=A_buff_buff.*repmat(aaa,1,2);

    %日数分に増やす
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    A_buff=repmat(A_buff_buff,sim_size,sim_size);
    A_buff=A_buff.*big_eyes;
    clear big_eyes
    %前日までの充放電(全時間帯1,-1)を加える
    clear A_buff_buff
    A_buff_buff=ones(T,T);
    A_buff_buff=horzcat(A_buff_buff,-ones(T,T));
%     A_buff_buff=A_buff_buff+repmat(EV_30min(:,1),1,T).'-ones(T,T);
%     A_buff_buff=horzcat(A_buff_buff,-A_buff_buff);
    A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i<=(num-1)*T*2 & j>(num-1)*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    A_buff_buff=A_buff_buff.*big_eyes;
    A_buff=A_buff+A_buff_buff;
    clear big_eyes A_buff_buff
    %EV以外の変数を連結
    for num=1:sim_size
        clear A_buff_buff_buff
        A_buff_buff_buff=A_buff(:,(num-1)*T*2+1:num*T*2);
        A_buff_buff_buff=horzcat(zeros(size(A_buff_buff_buff,1),3*T),A_buff_buff_buff);
        if num==1
```



```
            A_buff_buff=A_buff_buff_buff;
        else
            A_buff_buff=horzcat(A_buff_buff,A_buff_buff_buff);
        end
    end
    clear A_buff
    A_buff=A_buff_buff;
    %バイナリ変数を接続
    A_buff=horzcat(A_buff,zeros(T*sim_size,2*T*sim_size));
    %30分単位の計画なので、すべての変数に0.5時間をかける
    A_buff=0.5*A_buff;
    %整数変数を接続
    %SOC最大値やSOC初期値は導入台数に依存
    A_buff=horzcat(A_buff,zeros(T*sim_size,2));
    %走行に使ったSOCを引く
%     EV_run_buff=repmat(EV_30min(:,2),1,T);
%     EV_run_buff=EV_run_buff.*triu(ones(T,T),0);
    %%ここは後で改良
    clear EV_run_buff
%     EV_run_buff=repmat(EV_30min(:,2),sim_size,1);
    EV_run_buff=EV_30min(:,2);
    distance=cumsum(EV_run_buff.');
%     distance=sum(EV_run_buff(:,time));
%     distance=repmat(distance,1,sim_size);
    time=1:sim_size*T;
    b_buff(time,1)=1000*(SOC_EV_max-SOC_EV_init+0.11*distance(1,time));
    %走行中は削除
%     del_flag=zeros(T,sim_size);
%     time=1:T;
%     del_flag(EV_30min(time,1)==0,:)=1;
%     del_flag=reshape(del_flag,[],1);
%     time=1:sim_size*T;
%     b_buff(del_flag(time,1)==1)=[];
    A_buff_buff_buff_buff=-A_buff;
%     time=1:sim_size*T;
%     A_buff(del_flag(time,1)==1,:)=[];
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
    clear b_buff del_flag;

    %EVのSOC最小値制約 走行中でないとき
    %Aはコピーしたものを反転
    A_buff=-A_buff;
    time=1:sim_size*T;
    b_buff(time,1)=1000*(-SOC_EV_min+SOC_EV_init-0.11*distance(1,time));
    %走行中は削除
%     del_flag=zeros(T,sim_size);
%     time=1:T;
%     del_flag(EV_30min(time,1)==0,:)=1;
%     del_flag=reshape(del_flag,[],1);
%     time=1:sim_size*T;
%     b_buff(del_flag(time,1)==1)=[];
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
    clear b_buff A_buff_buff;
```

```
441
442     %出発前にEVが満充電
443 %     if sum(EV_30min(:,1))~=T
444         b_buff=zeros(T*sim_size,1);
445         time=1:sim_size*T;
446         b_buff(EV_30min(time,2)~=0,1)=(1000*(SOC_EV_init-0.9*SOC_EV_max));
447         %1時間帯ずらす
448         b_buff=vertcat(b_buff,b_buff(1,1));
449         b_buff(1)=[];
450         del_flag=zeros(T*sim_size,1);
451         time=1:T*sim_size-1;
452         del_flag(EV_30min(time+1,2)~=0,1)=1;
453 %         b_buff=repmat(b_buff,sim_size,1);
454         b_buff=b_buff-1000*0.11*distance.';
455         del_flag=repmat(del_flag,sim_size,1);
456         %走行直前以外を削除
457         num=1:T*sim_size;
458         A_buff_buff_buff_buff(del_flag(num,1)==0,:)=[];
459         num=1:T*sim_size;
460         b_buff(del_flag(num,1)==0)=[];
461         A=vertcat(A,A_buff_buff_buff_buff);
462         b=vertcat(b,b_buff);
463         clear A_buff;
464         clear b_buff;
465         clear A_buff_buff_buff_buff del_flag
466 %     end
467
468 end
469
470
471 %バイナリ変数と各変数との関係
472 %例 P_batt_C < P_batt_max*u
473 %蓄電池充電
474 %実数部分の1日分を作る
475 A_buff_buff=eye(T,T);
476 A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
477 A_buff_buff=horzcat(A_buff_buff,zeros(T,3*T));
478 %シミュレーション日数分に拡大
479 %単位行列の大きなものを作る
480 big_eyes=zeros(T*sim_size,T*5*sim_size);
481 for num=1:sim_size
482     [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
483     big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
484 end
485 %単位行列とかけて、関係ない日を削除
486 A_buff=repmat(A_buff_buff,sim_size,sim_size);
487 A_buff=A_buff.*big_eyes;
488 clear A_buff_buff big_eyes
489 %バイナリ変数の部分を加える
490 %適当に大きな数字をつける
491 A_buff_buff=-1000*10000*eye(T,T);
492 A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
493 A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
494 %単位行列の大きなものを作る
495 big_eyes=zeros(T*sim_size,T*2*sim_size);
```



```
496 for num=1:sim_size
497     [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
498     big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
499 end
500 A_buff_buff=A_buff_buff.*big_eyes;
501 A_buff=horzcat(A_buff,A_buff_buff);
502 %整数変数を加える
503 A_buff=horzcat(A_buff,zeros(sim_size*T,2));
504 b_buff=zeros(sim_size*T,1);
505 A=vertcat(A,A_buff);
506 b=vertcat(b,b_buff);
507 clear A_buff b_buff;
508 clear A_buff_buff big_eyes
509
510 %蓄電池放電
511 %実数部分の1日分を作る
512 A_buff_buff=eye(T,T);
513 A_buff_buff=horzcat(zeros(T,2*T),A_buff_buff);
514 A_buff_buff=horzcat(A_buff_buff,zeros(T,2*T));
515 %シミュレーション日数分に拡大
516 %単位行列の大きなものを作る
517 big_eyes=zeros(T*sim_size,T*5*sim_size);
518 for num=1:sim_size
519     [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
520     big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
521 end
522 %単位行列とかけて、関係ない日を削除
523 A_buff=repmat(A_buff_buff,sim_size,sim_size);
524 A_buff=A_buff.*big_eyes;
525 clear A_buff_buff big_eyes
526 %バイナリ変数の部分を加える
527 %適当に大きな数字をつける
528 A_buff_buff=1000*10000*eye(T,T);
529 A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
530 A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
531 %単位行列の大きなものを作る
532 big_eyes=zeros(T*sim_size,T*2*sim_size);
533 for num=1:sim_size
534     [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
535     big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
536 end
537 A_buff_buff=A_buff_buff.*big_eyes;
538 A_buff=horzcat(A_buff,A_buff_buff);
539 %整数変数を加える
540 A_buff=horzcat(A_buff,zeros(sim_size*T,2));
541 b_buff=1000*10000*ones(sim_size*T,1);
542 A=vertcat(A,A_buff);
543 b=vertcat(b,b_buff);
544 clear A_buff b_buff;
545 clear A_buff_buff big_eyes
546
547 %EV充電
548 %実数部分の1日分を作る
549 A_buff_buff=eye(T,T);
550 A_buff_buff=horzcat(zeros(T,3*T),A_buff_buff);
```

```
A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
A_buff_buff=-1000*EV_max*eye(T,T);
A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=horzcat(A_buff,A_buff_buff);
%実数変数を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2));
b_buff=zeros(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff big_eyes

%EV放電
%実数部分の1日分を作る
A_buff_buff=eye(T,T);
A_buff_buff=horzcat(zeros(T,4*T),A_buff_buff);
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
A_buff_buff=1000*EV_max*eye(T,T);
A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
```



```
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=horzcat(A_buff,A_buff_buff);
%実数変数を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2));
b_buff=1000*EV_max*ones(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff big_eyes

%蓄電池の最大値制約
%蓄電池充電
%実数部分の1日分を作る
A_buff_buff=eye(T,T);
A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
A_buff_buff=horzcat(A_buff_buff,zeros(T,3*T));
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2*sim_size*T));
%実数変数を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,1));
A_buff=horzcat(A_buff,-kW_batt_unit*1000*ones(sim_size*T,1));
b_buff=zeros(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff

%蓄電池放電
A_buff_buff=eye(T,T);
A_buff_buff=horzcat(zeros(T,2*T),A_buff_buff);
A_buff_buff=horzcat(A_buff_buff,zeros(T,2*T));
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
```

```
%バイナリ変数の部分を加える
%バイナリ変数の部分を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2*sim_size*T));
%実数変数を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,1));
A_buff=horzcat(A_buff,-1000*kW_batt_unit*ones(sim_size*T,1));
b_buff=1000*kW_batt_unit*ones(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff


%CO2を25%削減する制約
%まず1日分を作る
%UGからの購入電力
%午前6時～午後6時0.69kg/kWh、それ以外0.469kg/kWh
vr=1:T;
A_buff(1,vr)=((vr<=24)*0.69+(vr>24)*0.469)/(1000);
%その他の変数はCO2に寄与しない
A_buff=horzcat(A_buff,zeros(1,4*T));
%シミュレーション日数分に拡大
A_buff=repmat(A_buff,1,sim_size);
%バイナリ変数および実数変数
A_buff=horzcat(A_buff,zeros(1,sim_size*T*2+2));
%右辺
%EVがないときはガソリン消費分を計上
% if EV_inst~=0
    b_buff=0.75*CO2_nonMG;
% else
%     b_buff=0.75*CO2_nonMG+20*sim_size*(2.322/27.2);
% end
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff

%%上下限制約
%上限
%1日分
%コンバータ
vr=1:T;
ub(vr,1)=1000*kW_conv;
%蓄電池について
ub_buff=Inf*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%EVについて
ub_buff=1000*EV_max*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%日数分に拡大
ub=repmat(ub,sim_size,1);

%バイナリ変数
```



```
ub=vertcat(ub,ones(2*T*sim_size,1));
% 実数変数
ub=vertcat(ub,30);
ub=vertcat(ub,3);
% ub=vertcat(ub,0);
% ub=vertcat(ub,Inf);

%下限
%1日分
%コンバータ
vr=1:T;
lb(vr,1)=0;
%蓄電池について
lb_buff=zeros(T*2,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%EVについて
lb_buff=zeros(T*2,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%日数分に拡大
lb=repmat(lb,sim_size,1);

%バイナリ変数
lb=vertcat(lb,zeros(2*T*sim_size,1));
%実数変数
lb=vertcat(lb,0);
lb=vertcat(lb,0);

options = cplexoptimset;
options.Display = 'off';

[x, fval, exitflag, output] = cplexmilp (f, A, b, Aeq, beq.', ...
  [ ], [ ], [ ], lb, ub, ctype ,[ ], options);
clear f Aeq beq A b ub lb
clear Aeq_buff beq_buff A_buff b_buff ub_buff lb_buff

%解を解析
%導入台数
PV_install=x((5*T*sim_size)+2*T*sim_size+1);
battery_install=x((5*T*sim_size)+2*T*sim_size+2);
vr=1:numel(x);
x((5*T*sim_size)+2*T*sim_size+1:(5*T*sim_size)+2*T*sim_size+2)=[];
operation=x(1:T*5*sim_size,1);
operation=reshape(operation,[],sim_size);
P_UG=operation(1:T,:);
P_batt_C=operation(T+1:2*T,:);
P_batt_D=operation(2*T+1:3*T,:);
P_EV_C=operation(3*T+1:4*T,:);
P_EV_D=operation(4*T+1:5*T,:);
PV_Act=1000*PV*PV_install;
clear CO2
CO2=zeros(T,sim_size);
time=1:T;
CO2(time<=24,:)=(0.69/1000);
```

```
771 time=1:T;
772 CO2(time>24,:)=(0.469/1000);
773 CO2=CO2.*P_UG;
774 total_CO2=sum(sum(CO2));
```

```
%%混合整数計画を解いて、DCMGにおける最適容量を得る
clear f_buff
clear A_buff b_buff Aeq_buff beq_buff ub_buff lb_buff
clear A_buff_buff b_buff_buff Aeq_buff_buff beq_buff_buff
clear big_eyes ub lb
clear ctype

%%変数の種類を指定(実数、バイナリ変数、整数)
%実数変数
%UG_buy=1,batt_C=2,batt_D=3,EV_C=4,EV_D=5
%バイナリ変数
%batt_bin=6,EV_bin=7
%binが1の時、MG内の需要が増える方向、すなわち充電
%整数変数
%PV導入枚数=8、蓄電池導入台数=9、コンバータ導入量=10
%(EVは有無を考えて2回シミュレーションをやる)
%%シミュレーション期間の決定

%デバッグ時は5日でやる
sim_size=numel(PV)/T;
% sim_size=30;
num=1:numel(PV)/T;
PV(:,num>sim_size)=[];
num=1:numel(demand)/T;
demand(:,num>sim_size)=[];
%sim_size=numel(PV)/T


%整数変数の指定
% 'B','I','C','S','N'
%それぞれバイナリ、整数、実数、半実数、半整数

%イメージ　30分の電力やり取り×日数、そのあと導入台数
%購入電力、充放電を定義
vr=1:5*T*sim_size;
ctype(vr)='C';
%充放電方向変数
vr=5*T*sim_size+1:(5*T*sim_size)+2*T*sim_size;
ctype(vr)='B';
%導入台数
vr=(5*T*sim_size)+2*T*sim_size+1:(5*T*sim_size)+2*T*sim_size+2;
ctype(vr)='I';
% %連系用コンバータの容量 実数
vr=(5*T*sim_size)+2*T*sim_size+3:(5*T*sim_size)+2*T*sim_size+3;
ctype(vr)='C';

%%目的関数
%運用コスト+導入コストの最小化
%電気料金は昼間(午前7時~午後11時)の電力料金28.08円/kWh、それ以外の時間帯の料金14.13円/kWh
%起点は午前6時

%まず1日分の電気料金を作る(実数部分)
vr=1:T;
f_buff(1,vr)=((vr>=3 & vr<35)*28.08+(vr<3 | vr>=35)*14.13)/(1000);
%それ以外の変数は電気料金に関係ない
```



```
f_buff=horzcat(f_buff,zeros(1,4*T));

%シミュレーション期間分だけコピー
f_buff=repmat(f_buff,1,sim_size);
%充放電方向変数も関係ない
f_buff=horzcat(f_buff,zeros(1,2*T*sim_size));
%導入料金
f_buff=horzcat(f_buff,c_inst_PV*sim_size);
f_buff=horzcat(f_buff,c_inst_batt*sim_size);
f=horzcat(f_buff,c_conv*sim_size);
clear f_buff

%%等式制約
%電力の需給一致制約
%まず1日分を作る
%UGからの購入電力
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff(time,vr)=AC_DC;
Aeq_buff=Aeq_buff.*eye(T,T);
%蓄電池充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=-1/(eff_batt*DC_DC);
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%蓄電池放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=eff_batt*DC_DC;
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%EVの充電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=-1/(eff_EV*DC_DC);
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;
%EVの放電
[time,vr]=meshgrid(1:T,1:T);
Aeq_buff_buff(time,vr)=eff_EV*DC_DC;
Aeq_buff_buff=Aeq_buff_buff.*eye(T,T);
Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
clear Aeq_buff_buff;

%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
Aeq_buff=repmat(Aeq_buff,sim_size,sim_size);
Aeq_buff=Aeq_buff.*big_eyes;
clear big_eyes
```

```
111
112 %バイナリ変数
113 Aeq_buff_buff=zeros(sim_size*T,2*T);
114 Aeq_buff_buff=repmat(Aeq_buff_buff,1,sim_size);
115 Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
116 clear Aeq_buff_buff;
117 %導入台数
118 %PVの出力は台数に依存
119 Aeq_buff_buff=PCS*ones(sim_size*T,1);
120 Aeq_buff_buff=reshape(PV*1000,[],1).*Aeq_buff_buff;
121 Aeq_buff_buff=horzcat(Aeq_buff_buff,zeros(sim_size*T,2));
122 Aeq=horzcat(Aeq_buff,Aeq_buff_buff);
123 % %コンバータ導入量
124 % Aeq=horzcat(Aeq_buff,zeros(sim_size*T,1));
125 %右辺
126 beq_buff=demand(:,1:sim_size);
127 beq=reshape(beq_buff,1,[]);
128 clear Aeq_buff;
129 clear beq_buff;
130
131 %EV走行中の充電なし
132 if EV_inst~=0
133     %EVの充放電に関する部分
134     Aeq_buff=not(EV_30min(:,1));
135     Aeq_buff=repmat(Aeq_buff,1,T);
136     Aeq_buff=Aeq_buff.*repmat(eye(T,T),sim_size,1);
137     Aeq_buff=horzcat(Aeq_buff,-Aeq_buff);
138     Aeq_buff=repmat(Aeq_buff,1,sim_size);
139     %日数分に増やす
140     %シミュレーション日数分に拡大
141     %単位行列の大きなものを作る
142     big_eyes=zeros(T*sim_size,T*2*sim_size);
143     for num=1:sim_size
144         [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
145         big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
146     end
147     %単位行列とかけて、関係ない日を削除
148     Aeq_buff=Aeq_buff.*big_eyes;
149     %EV以外の部分を連結する
150     for num=1:sim_size
151         %Aeq_buff_buff(:,:,num)=horzcat(zeros(T,3*T),Aeq_buff((num-1)*2*T+1,num*2*T));
152         Aeq_buff_buff(:,:,num)=horzcat(zeros(T*sim_size,3*T),Aeq_buff(:,(num-1)*2*T+1:num*2*T));
153 %         Aeq_buff_buff(:,:,num)=horzcat(zeros(T*sim_size,3*T),Aeq_buff_buff(:,:,num));
154     end
155     clear Aeq_buff
156
157     Aeq_buff=reshape(Aeq_buff_buff,sim_size*T,sim_size*T*5);
158     clear big_eyes Aeq_buff_buff
159
160     %バイナリ変数
161     Aeq_buff_buff=zeros(T,2*T);
162     Aeq_buff_buff=repmat(Aeq_buff_buff,sim_size,sim_size);
163     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
164     clear Aeq_buff_buff;
165     %導入台数+コンバータ導入量
```



```
166     Aeq_buff_buff=zeros(T*sim_size,3);
167     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
168     Aeq=vertcat(Aeq,Aeq_buff);
169     beq_buff=zeros(1,size(Aeq_buff,1));
170     clear Aeq_buff Aeq_buff_buff;
171
172     %右辺
173     beq=horzcat(beq,beq_buff);
174     clear beq_buff;
175 end
176
177
178 % %EV走行中の充電なし
179 % if EV_inst~=0
180 %     %1日分をまず作る
181 %     %EVの充放電に関する部分
182 %     Aeq_buff=reshape((EV_30min(:,1)==0),1,T);
183 %     Aeq_buff=repmat(Aeq_buff,T,1);
184 %     Aeq_buff=Aeq_buff.*eye(T,T);
185 %     Aeq_buff=horzcat(Aeq_buff,-Aeq_buff);
186 %     %EV以外の部分を連結する
187 %     Aeq_buff=horzcat(zeros(T,3*T),Aeq_buff);
188 %     %日数分に増やす
189 %     %シミュレーション日数分に拡大
190 %     %単位行列の大きなものを作る
191 %     big_eyes=zeros(T*sim_size,T*5*sim_size);
192 %     for num=1:sim_size
193 %         [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
194 %         big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
195 %     end
196 %     %単位行列とかけて、関係ない日を削除
197 %     Aeq_buff=repmat(Aeq_buff,sim_size,sim_size);
198 %     Aeq_buff=Aeq_buff.*big_eyes;
199 %     clear big_eyes
200 %
201 %     %バイナリ変数
202 %     Aeq_buff_buff=zeros(T,2*T);
203 %     Aeq_buff_buff=repmat(Aeq_buff_buff,sim_size,sim_size);
204 %     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
205 %     clear Aeq_buff_buff;
206 %     %導入台数
207 %     Aeq_buff_buff=zeros(T*sim_size,2);
208 %     Aeq_buff=horzcat(Aeq_buff,Aeq_buff_buff);
209 %     Aeq=vertcat(Aeq,Aeq_buff);
210 %     beq_buff=zeros(1,size(Aeq_buff,1));
211 %     clear Aeq_buff Aeq_buff_buff;
212 %
213 %     %右辺
214 %     beq=horzcat(beq,beq_buff);
215 %     clear beq_buff;
216 % end
217
218
219 %%不等式制約
220 %SOCの最大値制約
```

```
%イメージ　蓄電池充放電の部分を作り、その後その他を挿入していく
A_buff_buff=-ones(1,2*T);
time=1:T;
A_buff_buff(time)=-A_buff_buff(time);
A_buff_buff=repmat(A_buff_buff,T,1);
%大きな三角行列を作る
aaa=eye(T,T)+tril(ones(T,T),-1);
%一日分の充放電
A_buff_buff=A_buff_buff.*repmat(aaa,1,2);

%日数分に増やす
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear big_eyes
%前日までの充放電(全時間帯1,-1)を加える
clear A_buff_buff
A_buff_buff=-ones(1,2*T);
time=1:T;
A_buff_buff(time)=-A_buff_buff(time);
A_buff_buff=repmat(A_buff_buff,T*sim_size,sim_size);
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i<=(num-1)*T*2 & j>(num-1)*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=A_buff+A_buff_buff;
clear big_eyes A_buff_buff
%蓄電池以外の変数を連結
for num=1:sim_size
    clear A_buff_buff_buff
    A_buff_buff_buff=A_buff(:,(num-1)*T*2+1:num*T*2);
    A_buff_buff_buff=horzcat(A_buff_buff_buff,zeros(size(A_buff_buff_buff,1),3*T));
    if num==1
        A_buff_buff=A_buff_buff_buff;
    else
        A_buff_buff=horzcat(A_buff_buff,A_buff_buff_buff);
    end
end
A_buff_buff=horzcat(zeros(T*sim_size,T),A_buff_buff);
clear A_buff
A_buff=A_buff_buff;
A_buff(:,5*sim_size*T+1:5*sim_size*T+T)=[];
clear A_buff_buff
```



```
%バイナリ変数を接続
A_buff=horzcat(A_buff,zeros(T*sim_size,2*T*sim_size));
%30分単位の計画なので、すべての変数に0.5時間をかける
A_buff=0.5*A_buff;
%整数変数を接続
%SOC最大値やSOC初期値は導入台数に依存
A_buff=horzcat(A_buff,zeros(T*sim_size,1));
A_buff=horzcat(A_buff,-1000*kWh_batt_unit*ones(T*sim_size,1));
%コンバータ導入量
A_buff=horzcat(A_buff,zeros(T*sim_size,1));
time=1:sim_size*T;
b_buff(time,1)=0;

A=A_buff;
b=b_buff;
clear b_buff

%SOCの最小値制約
%最大値と方向を逆にする
A_buff=-A_buff;
%導入台数と最小値の関係だけは変化
A_buff(:,size(A_buff,2)-1)=0;
% A_buff=horzcat(A_buff,zeros(T*sim_size,1));
time=1:sim_size*T;
b_buff(time,1)=0;
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff

%EVがあるときのみ考えるEVの制約
if EV_inst~=0
    %EVのSOC最大値制約 EV走行中でないとき
    %イメージ　EV充放電の部分を作り、その後その他を挿入していく
    A_buff_buff=-ones(1,2*T);
    time=1:T;
    A_buff_buff(time)=-A_buff_buff(time);
    A_buff=repmat(A_buff_buff,sim_size,sim_size);
    A_buff_buff=repmat(A_buff_buff,T,1);
    %大きな三角行列を作る
    aaa=eye(T,T)+tril(ones(T,T),-1);
    %一日分の充放電
    A_buff_buff=A_buff_buff.*repmat(aaa,1,2);

    %日数分に増やす
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    A_buff=repmat(A_buff_buff,sim_size,sim_size);
    A_buff=A_buff.*big_eyes;
    clear big_eyes
```

```
    %前日までの充放電(全時間帯1,-1)を加える
    clear A_buff_buff
    A_buff_buff=ones(T,T);
    A_buff_buff=horzcat(A_buff_buff,-ones(T,T));
%     A_buff_buff=A_buff_buff+repmat(EV_30min(:,1),1,T).'-ones(T,T);
%     A_buff_buff=horzcat(A_buff_buff,-A_buff_buff);
    A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i<=(num-1)*T*2 & j>(num-1)*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    A_buff_buff=A_buff_buff.*big_eyes;
    A_buff=A_buff+A_buff_buff;
    clear big_eyes A_buff_buff
    %EV以外の変数を連結
    for num=1:sim_size
        clear A_buff_buff_buff
        A_buff_buff_buff=A_buff(:,(num-1)*T*2+1:num*T*2);
        A_buff_buff_buff=horzcat(zeros(size(A_buff_buff_buff,1),3*T),A_buff_buff_buff);
        if num==1
            A_buff_buff=A_buff_buff_buff;
        else
            A_buff_buff=horzcat(A_buff_buff,A_buff_buff_buff);
        end
    end
    clear A_buff
    A_buff=A_buff_buff;
    %バイナリ変数を接続
    A_buff=horzcat(A_buff,zeros(T*sim_size,2*T*sim_size));
    %30分単位の計画なので、すべての変数に0.5時間をかける
    A_buff=0.5*A_buff;
    %整数変数+コンバータ導入量を接続
    A_buff=horzcat(A_buff,zeros(T*sim_size,3));
    %走行に使ったSOCを引く
%     EV_run_buff=repmat(EV_30min(:,2),1,T);
%     EV_run_buff=EV_run_buff.*triu(ones(T,T),0);
    %%ここは後で改良
    clear EV_run_buff
%     EV_run_buff=repmat(EV_30min(:,2),sim_size,1);
    EV_run_buff=EV_30min(:,2);
    distance=cumsum(EV_run_buff.');
%     distance=sum(EV_run_buff(:,time));
%     distance=repmat(distance,1,sim_size);
    time=1:sim_size*T;
    b_buff(time,1)=1000*(SOC_EV_max-SOC_EV_init+0.11*distance(1,time));
    %走行中は削除
%     del_flag=zeros(T,sim_size);
%     time=1:T;
%     del_flag(EV_30min(time,1)==0,:)=1;
%     del_flag=reshape(del_flag,[],1);
%     time=1:sim_size*T;
```



```
%     b_buff(del_flag(time,1)==1)=[];
    A_buff_buff_buff_buff=-A_buff;
%     time=1:sim_size*T;
%     A_buff(del_flag(time,1)==1,:)=[];
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
    clear b_buff del_flag;

    %EVのSOC最小値制約 走行中でないとき
    %AIはコピーしたものを反転
    A_buff=-A_buff;
    time=1:sim_size*T;
    b_buff(time,1)=1000*(-SOC_EV_min+SOC_EV_init-0.11*distance(1,time));
    %走行中は削除
%     del_flag=zeros(T,sim_size);
%     time=1:T;
%     del_flag(EV_30min(time,1)==0,:)=1;
%     del_flag=reshape(del_flag,[],1);
%     time=1:sim_size*T;
%     b_buff(del_flag(time,1)==1)=[];
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
    clear b_buff A_buff_buff;

    %出発前にEVが満充電
%     if sum(EV_30min(:,1))~=T
        b_buff=zeros(T*sim_size,1);
        time=1:sim_size*T;
        b_buff(EV_30min(time,2)~=0,1)=(1000*(SOC_EV_init-0.9*SOC_EV_max));
        %1時間帯ずらす
        b_buff=vertcat(b_buff,b_buff(1,1));
        b_buff(1)=[];
        del_flag=zeros(T*sim_size,1);
        time=1:T*sim_size-1;
        del_flag(EV_30min(time+1,2)~=0,1)=1;
%         b_buff=repmat(b_buff,sim_size,1);
        b_buff=b_buff-1000*0.11*distance.';
        del_flag=repmat(del_flag,sim_size,1);
        %走行直前以外を削除
        num=1:T*sim_size;
        A_buff_buff_buff_buff(del_flag(num,1)==0,:)=[];
        num=1:T*sim_size;
        b_buff(del_flag(num,1)==0)=[];
        A=vertcat(A,A_buff_buff_buff_buff);
        b=vertcat(b,b_buff);
        clear A_buff;
        clear b_buff;
        clear A_buff_buff_buff_buff del_flag
%     end

end


%バイナリ変数と各変数との関係
%例 P_batt_C < P_batt_max*u
```

```
%蓄電池充電
%実数部分の1日分を作る
A_buff_buff=eye(T,T);
A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
A_buff_buff=horzcat(A_buff_buff,zeros(T,3*T));
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
%適当に大きな数字をつける
A_buff_buff=-1000*10000*eye(T,T);
A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=horzcat(A_buff,A_buff_buff);
%整数変数＋コンバータ導入量を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,3));
b_buff=zeros(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff big_eyes

%蓄電池放電
%実数部分の1日分を作る
A_buff_buff=eye(T,T);
A_buff_buff=horzcat(zeros(T,2*T),A_buff_buff);
A_buff_buff=horzcat(A_buff_buff,zeros(T,2*T));
%シミュレーション日数分に拡大
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*5*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
end
%単位行列とかけて、関係ない日を削除
A_buff=repmat(A_buff_buff,sim_size,sim_size);
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
%適当に大きな数字をつける
```



```
A_buff_buff=1000*10000*eye(T,T);
A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
%単位行列の大きなものを作る
big_eyes=zeros(T*sim_size,T*2*sim_size);
for num=1:sim_size
    [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
    big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
end
A_buff_buff=A_buff_buff.*big_eyes;
A_buff=horzcat(A_buff,A_buff_buff);
%整数変数+コンバータ導入量を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,3));
b_buff=1000*10000*ones(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff big_eyes

if EV_inst~=0
    %EV充電
    %実数部分の1日分を作る
    A_buff_buff=eye(T,T);
    A_buff_buff=horzcat(zeros(T,3*T),A_buff_buff);
    A_buff_buff=horzcat(A_buff_buff,zeros(T,T));
    %シミュレーション日数分に拡大
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*5*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
        big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
    end
    %単位行列とかけて、関係ない日を削除
    A_buff=repmat(A_buff_buff,sim_size,sim_size);
    A_buff=A_buff.*big_eyes;
    clear A_buff_buff big_eyes
    %バイナリ変数の部分を加える
    A_buff_buff=-1000*EV_max*eye(T,T);
    A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
    A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
    %単位行列の大きなものを作る
    big_eyes=zeros(T*sim_size,T*2*sim_size);
    for num=1:sim_size
        [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
        big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
    end
    A_buff_buff=A_buff_buff.*big_eyes;
    A_buff=horzcat(A_buff,A_buff_buff);
    %整数変数＋コンバータ導入量を加える
    A_buff=horzcat(A_buff,zeros(sim_size*T,3));
    b_buff=zeros(sim_size*T,1);
    A=vertcat(A,A_buff);
    b=vertcat(b,b_buff);
    clear A_buff b_buff;
    clear A_buff_buff big_eyes
```

```
551
552     %EV放電
553     %実数部分の1日分を作る
554     A_buff_buff=eye(T,T);
555     A_buff_buff=horzcat(zeros(T,4*T),A_buff_buff);
556     %シミュレーション日数分に拡大
557     %単位行列の大きなものを作る
558     big_eyes=zeros(T*sim_size,T*5*sim_size);
559     for num=1:sim_size
560         [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
561         big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
562     end
563     %単位行列とかけて、関係ない日を削除
564     A_buff=repmat(A_buff_buff,sim_size,sim_size);
565     A_buff=A_buff.*big_eyes;
566     clear A_buff_buff big_eyes
567     %バイナリ変数の部分を加える
568     A_buff_buff=1000*EV_max*eye(T,T);
569     A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
570     A_buff_buff=repmat(A_buff_buff,sim_size,sim_size);
571     %単位行列の大きなものを作る
572     big_eyes=zeros(T*sim_size,T*2*sim_size);
573     for num=1:sim_size
574         [i,j]=meshgrid(1:T*2*sim_size,1:T*sim_size);
575         big_eyes(i>=(num-1)*T*2+1 & i<= num*T*2 & j>=(num-1)*T+1 & j<=num*T)=1;
576     end
577     A_buff_buff=A_buff_buff.*big_eyes;
578     A_buff=horzcat(A_buff,A_buff_buff);
579     %整数変数＋コンバータ導入量を加える
580     A_buff=horzcat(A_buff,zeros(sim_size*T,3));
581     b_buff=1000*EV_max*ones(sim_size*T,1);
582     A=vertcat(A,A_buff);
583     b=vertcat(b,b_buff);
584     clear A_buff b_buff;
585     clear A_buff_buff big_eyes
586 end
587
588 %蓄電池の最大値制約
589 %蓄電池充電
590 %実数部分の1日分を作る
591 A_buff_buff=eye(T,T);
592 A_buff_buff=horzcat(zeros(T,T),A_buff_buff);
593 A_buff_buff=horzcat(A_buff_buff,zeros(T,3*T));
594 %シミュレーション日数分に拡大
595 %単位行列の大きなものを作る
596 big_eyes=zeros(T*sim_size,T*5*sim_size);
597 for num=1:sim_size
598     [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
599     big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
600 end
601 %単位行列とかけて、関係ない日を削除
602 A_buff=repmat(A_buff_buff,sim_size,sim_size);
603 A_buff=A_buff.*big_eyes;
604 clear A_buff_buff big_eyes
605 %バイナリ変数の部分を加える
```



```
606 A_buff=horzcat(A_buff,zeros(sim_size*T,2*sim_size*T));
607 %整数変数を加える
608 A_buff=horzcat(A_buff,zeros(sim_size*T,1));
609 A_buff=horzcat(A_buff,-kW_batt_unit*1000*ones(sim_size*T,1));
610 %コンバータ導入量
611 A_buff=horzcat(A_buff,zeros(sim_size*T,1));
612 b_buff=zeros(sim_size*T,1);
613 A=vertcat(A,A_buff);
614 b=vertcat(b,b_buff);
615 clear A_buff b_buff;
616 clear A_buff_buff
617
618 %蓄電池放電
619 A_buff_buff=eye(T,T);
620 A_buff_buff=horzcat(zeros(T,2*T),A_buff_buff);
621 A_buff_buff=horzcat(A_buff_buff,zeros(T,2*T));
622 %シミュレーション日数分に拡大
623 %単位行列の大きなものを作る
624 big_eyes=zeros(T*sim_size,T*5*sim_size);
625 for num=1:sim_size
626     [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
627     big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
628 end
629 %単位行列とかけて、関係ない日を削除
630 A_buff=repmat(A_buff_buff,sim_size,sim_size);
631 A_buff=A_buff.*big_eyes;
632 clear A_buff_buff big_eyes
633 %バイナリ変数の部分を加える
634 %バイナリ変数の部分を加える
635 A_buff=horzcat(A_buff,zeros(sim_size*T,2*sim_size*T));
636 %整数変数を加える
637 A_buff=horzcat(A_buff,zeros(sim_size*T,1));
638 A_buff=horzcat(A_buff,-1000*kW_batt_unit*ones(sim_size*T,1));
639 %コンバータ導入量
640 A_buff=horzcat(A_buff,zeros(sim_size*T,1));
641 b_buff=zeros(sim_size*T,1);
642 % b_buff=1000*kW_batt_unit*ones(sim_size*T,1);
643 A=vertcat(A,A_buff);
644 b=vertcat(b,b_buff);
645 clear A_buff b_buff;
646 clear A_buff_buff
647
648 %コンバータの最大値制約
649 %実数部分の1日分を作る
650 A_buff_buff=eye(T,T);
651 A_buff_buff=horzcat(A_buff_buff,zeros(T,4*T));
652 %シミュレーション日数分に拡大
653 %単位行列の大きなものを作る
654 big_eyes=zeros(T*sim_size,T*5*sim_size);
655 for num=1:sim_size
656     [i,j]=meshgrid(1:T*5*sim_size,1:T*sim_size);
657     big_eyes(i>=(num-1)*T*5+1 & i<= num*T*5 & j>=(num-1)*T+1 & j<=num*T)=1;
658 end
659 %単位行列とかけて、関係ない日を削除
660 A_buff=repmat(A_buff_buff,sim_size,sim_size);
```

```
A_buff=A_buff.*big_eyes;
clear A_buff_buff big_eyes
%バイナリ変数の部分を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2*sim_size*T));
%整数変数を加える
A_buff=horzcat(A_buff,zeros(sim_size*T,2));
%コンバータ導入量
A_buff=horzcat(A_buff,-1000*ones(sim_size*T,1));
b_buff=zeros(sim_size*T,1);
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff

%CO2を25%削減する制約
%まず1日分を作る
%UGからの購入電力
%午前6時～午後6時0.69kg/kWh、それ以外0.469kg/kWh
vr=1:T;
A_buff(1,vr)=((vr<=24)*0.69+(vr>24)*0.469)/(1000);
%その他の変数はCO2に寄与しない
A_buff=horzcat(A_buff,zeros(1,4*T));
%シミュレーション日数分に拡大
A_buff=repmat(A_buff,1,sim_size);
%バイナリ変数および実数変数
A_buff=horzcat(A_buff,zeros(1,sim_size*T*2+3));
%右辺
%EVがないときはガソリン消費分を計上
if EV_inst~=0
    b_buff=0.74*CO2_nonMG;
else
    b_buff=0.74*CO2_nonMG-20*sim_size*(2.322/27.2);
end
A=vertcat(A,A_buff);
b=vertcat(b,b_buff);
clear A_buff b_buff;
clear A_buff_buff

%%上下限制約
%上限
%1日分
%コンバータ
vr=1:T;
ub(vr,1)=Inf;
% ub(vr,1)=1000*kW_conv;
%蓄電池について
ub_buff=Inf*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%EVについて
ub_buff=1000*EV_max*ones(T*2,1);
ub=vertcat(ub,ub_buff);
clear ub_buff;
%日数分に拡大
ub=repmat(ub,sim_size,1);
```



```
%バイナリ変数
ub=vertcat(ub,ones(2*T*sim_size,1));
% 実数変数
ub=vertcat(ub,30);
ub=vertcat(ub,30);
ub=vertcat(ub,Inf);
% ub=vertcat(ub,0);
% ub=vertcat(ub,Inf);

%下限
%1日分
%コンバータ
vr=1:T;
lb(vr,1)=0;
%蓄電池について
lb_buff=zeros(T*2,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%EVについて
lb_buff=zeros(T*2,1);
lb=vertcat(lb,lb_buff);
clear lb_buff;
%日数分に拡大
lb=repmat(lb,sim_size,1);

%バイナリ変数
lb=vertcat(lb,zeros(2*T*sim_size,1));
%実数変数
lb=vertcat(lb,0);
lb=vertcat(lb,0);
lb=vertcat(lb,0);

options = cplexoptimset;
options.Display = 'on';

[x, fval, exitflag, output] = cplexmilp (f, A, b, Aeq, beq.', ...
  [ ], [ ], [ ], lb, ub, ctype ,[ ], options);
clear f Aeq beq A b ub lb
clear Aeq_buff beq_buff A_buff b_buff ub_buff lb_buff

%解を解析
%導入台数
PV_install=x((5*T*sim_size)+2*T*sim_size+1);
battery_install=x((5*T*sim_size)+2*T*sim_size+2);
conv_install=x((5*T*sim_size)+2*T*sim_size+3);
vr=1:numel(x);
x((5*T*sim_size)+2*T*sim_size+1:(5*T*sim_size)+2*T*sim_size+2)=[];
operation=x(1:T*5*sim_size,1);
operation=reshape(operation,[],sim_size);
P_UG=operation(1:T,:);
P_batt_C=operation(T+1:2*T,:);
P_batt_D=operation(2*T+1:3*T,:);
P_EV_C=operation(3*T+1:4*T,:);
P_EV_D=operation(4*T+1:5*T,:);
PV_Act=PV*PV_install;
```

```
771 clear CO2
772 CO2=zeros(T,sim_size);
773 time=1:T;
774 CO2(time<=24,:)=(0.69/1000);
775 time=1:T;
776 CO2(time>24,:)=(0.469/1000);
777 CO2=CO2.*P_UG;
778 total_CO2=sum(sum(CO2));
```

```
%必要な諸データの読み込み
load('MG_data.mat');

%%導入予定設備1台あたりのパラメータ
%PVパネル1枚あたりの定格[kW]
PV_unit=0.25;
%蓄電池1台あたりの出力[kW]
kW_batt_unit=3;
%蓄電池1台あたりの容量[kWh]
kWh_batt_unit=6.6;
%EV1台あたりの出力[kW]
kW_EV_unit=3;
%EV1台あたりの容量[kWh]
kWh_EV_unit=16;
%EVの運行予定を読み込む
if EV_inst~=0
    EV=csvread('EV.csv');
else
    EV=csvread('noEV.csv');
end
%DCバスの定格電圧
V_DC_ref=380;
%コンバータ容量[kW]
kW_conv=20;
%連系コンバータ1kWあたりの値段
c_conv=0.4*27.4/2;
%DG用のAC/DCの価格
c_DG_AC_DC=0.4*27.4;



%時間帯数
T=48;

%蓄電池1台あたりの価格(1日)
c_inst_batt=270;
%PVパネル1枚あたりの価格(1日)
c_inst_PV=25;
%EV1台の導入コスト
c_inst_EV=583;

%EVに関して
SOC_EV_max=EV_inst*kWh_EV_unit;
SOC_EV_init=0.9*SOC_EV_max;
SOC_EV_min=0.0;

EV_max=EV_inst*kW_EV_unit;

% for season=1:4
    clear Ohta_demand Ohta_PV
    clear PV_buff demand_buff del_num
    clear PV_copy demand_copy
    %太田市のデータをデマンドを読み込む
    load(strcat(num2str(season),'.mat'));
    %負荷およびPVの30分平均値の作成
```



```
    demand_buff=Ohta_demand;
    PV_buff=Ohta_PV;
    %Ohta_PVのうちNANの含まれる日を除外
    del_num=isnan(sum(PV_buff));
    PV_buff(:,del_num)=[];
    clear del_num
    %Ohta_demandのうちNANの含まれる日を除外
    del_num=isnan(sum(demand_buff));
    demand_buff(:,del_num)=[];

    %1秒値データを30分平均値に加工
    PV_buff=reshape(PV_buff,60*30,numel(PV_buff)/(60*30));
    PV_buff=mean(PV_buff);
    PV=transpose(PV_buff);
    demand_buff=reshape(demand_buff,60*30,numel(demand_buff)/(60*30));
    demand_buff=mean(demand_buff);
    demand=transpose(demand_buff);
    %%日ごとにならべる
    PV=reshape(PV,48,numel(PV)/48);
    demand=reshape(demand,48,numel(demand)/48);

    %%0時から始まっているデータを午前6時にずらす
    %負荷に関しては，最初の日の0:00~6:00のデータを最後の日の0:00~6:00のデータとして扱う
    %PVに関しては，夜間は出力がないので，0のままつなげる
    demand_copy=demand;
    PV_copy=PV;
    demand_copy_size=size(demand_copy);
    PV_copy_size=size(PV_copy);

    %ベクトル化
    demand_copy=reshape(demand_copy,[],1);
    PV_copy=reshape(PV_copy,[],1);
    %負荷の最初の６時間を最後の６時間に接続
    time=1:12;
    demand_6h=demand_copy(time,1);
    demand_copy=vertcat(demand_copy,demand_6h);
    time=1:12;
    demand_copy(time)=[];
    %PVに関しても同様の操作
    time=1:12;
    PV_6h=PV_copy(time,1);
    PV_copy=vertcat(PV_copy,PV_6h);
    time=1:12;
    PV_copy(time)=[];

    %ベクトルから日付別に戻す
    demand_copy=reshape(demand_copy,demand_copy_size);
    PV_copy=reshape(PV_copy,PV_copy_size);


    time=1:48;
    demand(time,:)=[];
    time=1:48;
    PV(time,:)=[];
    demand_buff_buff=1000*demand_copy;
```

```
    PV_buff_buff=PV_copy;
% % % %
% % % %     if season==1
% % % %         demand_buff_buff=1000*demand_copy;
% % % %         PV_buff_buff=PV_copy;
% % % %     else
% % % %         demand_buff_buff=horzcat(demand_buff_buff,1000*demand_copy);
% % % %         PV_buff_buff=horzcat(PV_buff_buff,PV_copy);
% % % %     end
% end

demand=demand_buff_buff;
PV=PV_buff_buff;
%EVの運行予約に関して
%30分のものに変更
EV_30min=EV;
time=1:12;
EV_30min_6h=EV(time,:);
time=1:12;
EV_30min(time,:)=[];
EV_30min=vertcat(EV_30min,EV_30min_6h);

%土日は走行なしのスケジュール
EV_30min_con=repmat(EV_30min(:,1),1,numel(PV)/T);
EV_30min_dis=repmat(EV_30min(:,2),1,numel(PV)/T);
for day=1:numel(PV)/T
    if rem(day,7)==0 || rem(day,7)==6
        EV_30min_con(:,day)=1;
        EV_30min_dis(:,day)=0;
    end
end
EV_30min=horzcat(reshape(EV_30min_con,[],1),reshape(EV_30min_dis,[],1));

%パネル1枚あたりに出力に変換
%太田市のデータは4 kWのものなので、4/0.25で割る
PV=PV/(4/0.25);

%%負荷を4つに分別、ベース、証明、空調、その他
%時間帯によって比率が違うものとして考える
%昼  0.2  0.06 0.58 0.16
%夜  0.19 0.14 0.43 0.29
for time=1:T;
    if time<=24
        demand_base(time,1)=0.2;
        demand_light(time,1)=0.06;
        demand_air(time,1)=0.58;
        demand_other(time,1)=0.16;
    else
        demand_base(time,1)=0.19;
        demand_light(time,1)=0.14;
        demand_air(time,1)=0.43;
        demand_other(time,1)=0.29;
    end
end
%日数だけ拡張
```



```
demand_base=repmat(demand_base,1,numel(demand)/48);
demand_light=repmat(demand_light,1,numel(demand)/48);
demand_air=repmat(demand_air,1,numel(demand)/48);
demand_other=repmat(demand_other,1,numel(demand)/48);

%もとのデマンドデータをかける
demand_base=demand_base.*demand;
demand_light=demand_light.*demand;
demand_air=demand_air.*demand;
demand_other=demand_other.*demand;

%導入前CO2排出量の計算
sim_size=numel(demand)/T;
CO2=zeros(T,sim_size);
time=1:T;
CO2(time<=24,:)=(0.69/1000);
time=1:T;
CO2(time>24,:)=(0.469/1000);
CO2=CO2.*(demand);
CO2_nonMG=sum(sum(CO2));
clear demand

%与えられたAC/DC比によって分ける
%DCMGをベースに考えるので、AC負荷をAC_DC*DC_DCで割る
demand_AC=0;
demand_DC=0;
if bin2dec(demand_component(1,1))==0
    demand_AC=demand_AC+demand_base;
    demand_base=demand_base/(AC_DC*DC_DC);
else
    demand_base=demand_base/(DC_DC);
    demand_DC=demand_DC+demand_base;
end
if bin2dec(demand_component(1,2))==0
    demand_AC=demand_AC+demand_light;
    demand_light=demand_light/(AC_DC*DC_DC);
else
    demand_light=demand_light/(DC_DC);
    demand_DC=demand_DC+demand_light;
end
if bin2dec(demand_component(1,3))==0
    demand_AC=demand_AC+demand_air;
    demand_air=demand_air/(AC_DC*DC_DC);
else
    demand_air=demand_air/(DC_DC);
    demand_DC=demand_DC+demand_air;
end
if bin2dec(demand_component(1,4))==0
    demand_AC=demand_AC+demand_other;
    demand_other=demand_other/(AC_DC*DC_DC);
else
    demand_other=demand_other/(DC_DC);
    demand_DC=demand_DC+demand_other;
end
demand=demand_base+demand_light+demand_air+demand_other;
```