Contributions à la commande optimale par programmation dynamique. Application à la gestion de l'énergie de l'automobile électrifiée

Willy Cottin

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE ENERGIE, MATERIAUX, SCIENCES DE LA TERRE ET DE L'UNIVERS

Laboratoire PRISME

THÈSE présentée par :

Willy COTTIN

Soutenue le : **5 avril 2024**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : Energétique

Contributions à la commande optimale par Programmation Dynamique

Application à la gestion de l'énergie de l'automobile électrifiée

THÈSE dirigée par :

COLIN Guillaume	Professeur, Université d'Orléans

RAPPORTEURS :

BOUSCAYROL Alain	Professeur, Université de Lille 1
DELPRAT Sébastien	Professeur, INSA Hauts-de-France

JURY :

CHARLET Alain	Maître de conférences, Université d'Orléans
CHRENKO Daniela	Maîtresse de conférences, Université Bourgogne Franche-Comté
TAUZIA Xavier	Professeur, Université de Nantes, **Président du jury**

INVITÉS :

HOUILLE Sébastien	Ingénieur, Stellantis

Remerciements

Je remercie Guillaume COLIN et Alain CHARLET de m'avoir donné l'envie de poursuivre mon parcours au travers de cette thèse, à l'issue de l'école d'ingénieur. Je tiens à vous remercier sincèrement, ainsi que Sébastien HOUILLE, pour la confiance que vous m'avez accordé, tous les trois, tout au long de la thèse. Vos enseignements sont précieux et m'accompagneront durant l'ensemble de ma carrière, avec certitiude.

Je remercie également l'équipe de Clément DUMAND chez Stellantis de m'avoir accueilli durant ces trois années de thèse. Ces mêmes remerciements vont bien évidemment à l'ensemble du personnel du laboratoire PRISME avec lequel j'ai apprécié chaque échange.

Je tiens à remercier Messieurs Sébastien DELPRAT, Professeur à l'INSA des Hauts-de-France et Alain BOUSCAYROL, Professeur à l'Université de Lille d'avoir accepté d'être rapporteurs de ce travail de thèse.

Je remercie particulièrement Wissam, David GAUDRIE, David FONTAINE, Stéphane, Antoine, Rossela et Pasquale pour m'avoir confié des projets et/ou accordé du temps pour bonifier les sujets de recherche.

Merci à l'ensemble des personnes qui ont participé à la vie des doctorants PRISME : Gaétan, Ernesto, Marin, Alfredo, Mathieu, Ahmed, Léo, Brian, Mylène, Khaled, Enzo, Amaury, Alex, Jack, Maxime, Aurélien, Flavie, Oliver, Abdel et Laïla. Un réel merci pour votre bonne humeur. J'ai finalement réussi à faire ma thèse sans boire de café, quel exploit !

Un merci également pour les élèves que j'ai pu encadrer à l'occasion de stages ou de projets : Mirado, Jules, Yuqi, Bastien et Marin (encore toi).

Une pensée à ma famille ainsi que les personnes qui n'ont pas été citées mais qui ont participé de près ou de loin à la réalisation de cette thèse.

Pour finir, je remercie Pauline qui m'a supporté, épaulé et soutenu tout au long de la thèse.

Table des matières

Nomenclature

Texte en italique pour les mots et acronymes anglais importants et utilisés dans le domaine. Le choix de conserver un certain nombre de termes anglais a été fait pour permettre au lecteur non-initié français d'apprendre les termes clés anglais pour poursuivre un travail bibliographique plus approfondi.

Acronymes	Description
ADP	*Approximate Dynamic Programming*
AI	*Artificial Interlligence* ou Intelligence Artificielle
BBN	Bilan Batterie Nul
BdS	*Boundary Surfaces*
BL	*Boundary Lines*
BW	*Backward*
CAFE	*Corporate Average Fuel Economy*
CD^3	*Column Diagonal Dominance Degree*
cf	*cold fluid* ou fluide froid
DDP	*Deterministic Dynamic Programming*
DP	*Dynamic Programming* ou Programmation Dynamique
ECU	*Engine Control Unit*
ETH	École Polytechnique Fédérale de Zurich
FIFO	*Full Inputs Full Outputs*
FW	*Forward*
GA	*Genetic Algorithm* ou Algorithme Génétique
GMP	Groupe Moto Propulseur
HEV	*Hybrid Electric Vehicle*
hf	*hot fluid* ou fluide chaud
HVAC	*Heat Ventilation Air Conditionning*
IDP	*Iterative Dynamic Programming*
IDW	*Inverse Distance Weighting*
iORC	*inversed Organic Rankine Cycle*
LdR	Liquide de refroidissement
LGE	Loi de Gestion de l'Energie
LMI	*Linear Matrix Inequalities*
LP	*Linear Programming*
MA	Maillage Adaptatif
MIMO	*Multi Inputs Multi Outputs*
MPC	*Model Predictive Control* ou Commande par Modèle Prédictif
OCP	*Optimal Control Problem* ou Problème de Contrôle Optimal
OCV	*Open Circuit Voltage*
ORC	*Organic Rankine Cycle*

Acronymes	Description
ORM	*Organic Rankine Machine*
PMP	Principe de Maximum (ou Minimum) de Pontryaguine
QP	*Quadratic Programming*
RDE	*Real Driving Emissions*
RGA	*Relative Gain Array*
RL	*Reinforcement Learning* ou Apprentissage par Renforcement
SDP	*Stochastic Dynamic Programming*
SISO	*Single Input Single Output*
SOC	*State Of Charge* ou état de charge
SOFC	*Solid Oxide Fuel Cell*
V2G	*Vehicle to Grid*
wf	*working fluid* ou fluide de travail
WHRS	*Waste Heat Recovery System*
WLTC	*Worldwide harmonized Light duty driving Test Cycle*
WRM	*Water Rankine Machine*
ZEV	*Zero Emission Vehicule*

Afin d'alléger les expressions, les variables en gras représentent des vecteurs, matrices ou tenseurs. Les variables dont l'unité varie en fonction du système n'ont auncune unité d'indiquée.

Variables	Unité	Description
$\boldsymbol{u}$	-	Vecteur des variables de commande
$\boldsymbol{x}$	-	Vecteur des variables d'état
$\dot{var}$	-	Dérivée temporelle
$\boldsymbol{w}$	-	Vecteur des variables de perturbations
f	-	Dynamique du modèle
t	-	Variable du temps
$J(.)$		Critère d'optimisation
t_0	s	Temps initial du problème
t_f	s	Temps final du problème
$L(.)$		Coût instantané
$\Phi(.)$		Pénalité finale
$\boldsymbol{X}(.)$		Ensemble des états possibles du système
$\boldsymbol{U}(.)$		Ensemble des commandes possibles du système
k	-	Indice temporel
N	-	Nombre de pas de temps
C	N.m	Couple
SOC	-	État de charge
ω	rad/s	vitesse de rotation ou pulsation
P	W	Puissance
v	m/s	vitesse linéaire
γ	-	rapport de démultiplication
ρ	kg/m^3	masse volumique

Variables	Unité	Decription
c	m	circonférence
η	-	rendement
A	C	Capacité
I	A	Intensité électrique
R	Ω	Résistance électrique
k	-	Indice temporel
r	N.m/A	Constante couple moteur
Δt	s	Pas de temps
$\underline{var}$		Varleur minimale
$\overline{var}$		Valeur maximale
p	J^{-2}	coefficient d'homogénéité
V		*Cost to go matrix* ou Matrice de coût optimal
O		Grand O
$\nu(.)$		Fonction d'interpolation
μ		Variable de pénalisation
ψ		Fonction de convergence
ϵ		Critère de convergence
σ		Restriction de l'espace de recherche
$\bar{J}(.)$		Opérateur Lagrangien
$\boldsymbol{\lambda}(.)$		Vecteur de coefficient de Lagrange
$H(.)$		Opérateur Hamiltonien
δ	-	Petites variations
h	J/kg	Enthalpie massique
s	J/(kg.K)	Entropie massique
F	N	Force
m	kg	Masse
a	m/s^2	Accélération
ζ	m	Rayon
$\dot{W}$	W	Travail mécanique
$\dot{m}$	kg/s	Débit massique
LHV	J/kg	Pouvoir calorifique inférieur
OCV	V	Tension à vide
T	°C	Température
Pr	Pa	Pression
$\dot{Q}$	W	Flux de chaleur
Cp	J/(kg.K)	Capacité thermique
Ws	J/kg	Travail spécifique
sf	-	*shrink factor*
ξ	-	complexité numérique
G	-	matrice de fonction de transfert
K		Gain statique
τ	s	Temps caractéristique

Avant-propos

Le domaine automobile est en perpétuelle évolution technologique afin de répondre aux attentes des clients mais également aux demandes sociétales. En effet, depuis la création de l'automobile (fin du XIX^{ème} siècle) jusqu'à nos jours, les attendus du domaine ne sont plus les mêmes. En effet, à ses début, la chaîne de traction du véhicule était composée d'une machine à vapeur ou d'une machine électrique et d'une batterie. Ces solutions, permettaient la mise en mouvement d'un véhicule mais l'autonomie ainsi que la puissance de ces derniers ont été dépassés par les capacités des véhicules équipés d'un moteur à combustion interne. Initialement utilisés comme engins militaire ou pour du transport publique, l'implantation de la voiture telle qu'elle est connue actuellement dans la société a nécessité beaucoup de temps et d'idées ingénieuses. Une des plus importantes étant celle d'Henry Ford, qui a permis une commercialisation d'un véhicule bon marché grâce à un véhicule simple et peu coûteux à construire. A partir du parc automobile et du nombre de foyer possédant une voiture générée par les véhicules basés le fordisme, des attentes autour du produit automobile sont survenues. En effet, la population s'est habituée à ces véhicules, les faisant ainsi entrer dans leur quotidien. L'utilisation par une partie grandissante de la population à généré ainsi une élévation de l'exigence aussi bien du confort (radio, sièges, climatisation) que de la sécurité des véhicules. Cependant, ces exigences contribuent à une augmentation de la consommation de carburant des véhicules car le nombre de systèmes non obligatoire pour le déplacement du véhicule augmente ainsi que la masse du véhicule.

C'est pourquoi, depuis une trentaine d'années maintenant, avec la prise de conscience du réchauffement climatique, l'enjeu est désormais à l'économie de carburant ainsi qu'à la réduction d'émissions de polluants tout en améliorant la sécurité du véhicule et en conservant le confort installé dans l'automobile. Pour répondre à ces enjeux sociétaux, l'optimisation du groupe motopropulseur a été un levier important. Pour finir, la mise en place de l'électrification des groupes motopropulseurs est, de nos jours, un levier important de la réduction de la consommation de carburant.

Il faut cependant noter ici que pour le domaine automobile, le CO_2 n'est pas considéré comme un polluant étant donné que les émissions de CO_2 sont le reflet de la consommation de carburant du véhicule. Parmi les polluants émis par un moteur thermique, certains diminuent avec la minimisation de la consommation comme le monoxyde de carbone CO par exemple mais d'autres augmentent comme les oxydes d'azote NOx. Étant donné que pour vendre un véhicule, il faut respecter les normes d'émissions de polluants imposées, le constructeur pourra quant à lui se démarquer sur la consommation de carburant pour être attractif auprès du client. L'évaluation de la consommation en carburant d'un véhicule ainsi que son émission de polluants est réalisée sur un cycle de conduite homologué dans des conditions spécifiques. En Europe, le cycle en vigueur actuellement est le cycle d'homologation des véhicules légers appelé *Worldwide harmonized Light vehicles Test Cycle* dont l'acronyme est *WLTC*. Ce cycle de vitesse dont la durée est de trente minutes et la distance parcourue de 23 km est illustré en figure 1. C'est sur ce cycle que la majorité des modèles véhicules utilisés au cours de la thèse

seront évalués. Ce cycle d'homologation est complété d'un cycle aléatoire appelé *Real Driving Emissions* pour s'assurer que les émissions de polluants déterminées lors du *WLTC* sont proches de celles obtenues lors d'un vrai roulage. Aux États-Unis, c'est la norme *Corporate Average Fuel Economy* (*CAFE*) qui est en vigueur depuis 1975.

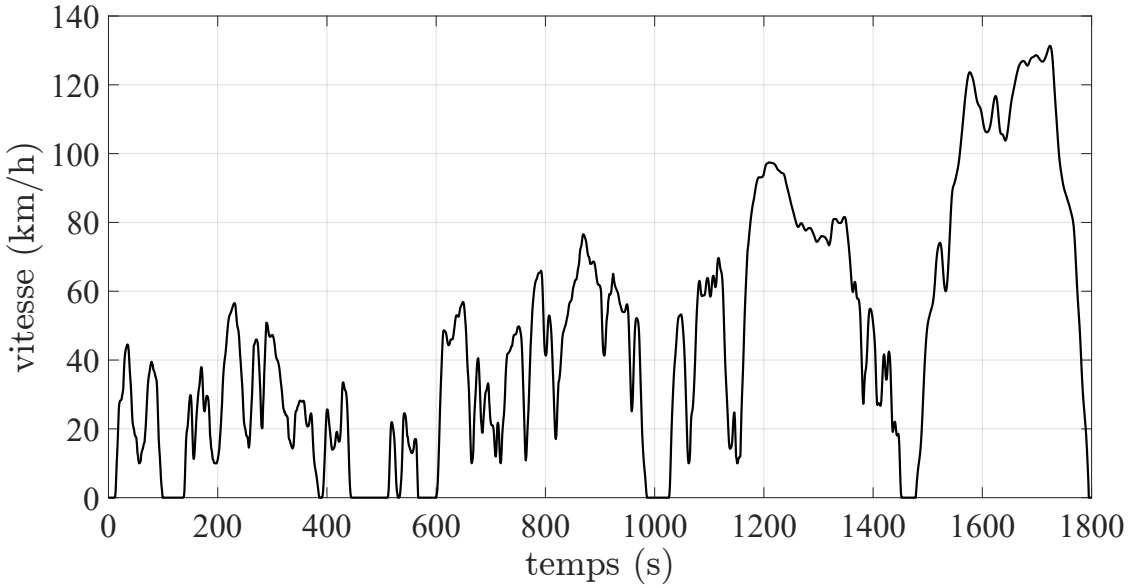

FIGURE 1 – Cycle d'homologation *WLTC*

Dès lors, l'hybridation des véhicules est une solution intéressante pour un constructeur afin d'atteindre ses objectifs. La définition d'un véhicule hybride est la suivante : un véhicule est considéré hybride s'il est composé d'au moins deux sources d'énergie qui participe à la traction du véhicule dont une est réversible. De nos jours, un véhicule hybride intègre un moteur thermique ou une pile à combustible pour transformer le carburant en puissance mécanique et une batterie ainsi qu'une ou plusieurs machines électriques. Différentes solutions sont possibles. Une légère hybridation comme le *start and stop* était une solution suffisante et innovante à son arrivée. Toutefois, cette solution n'est plus suffisante. C'est pourquoi un travail sur la répartition de puissance entre les différentes parties de la chaîne de traction est fait, pour réduire le plus possible les consommations des énergies fossiles.

Toutefois, les réglementations futures européennes portant sur les émissions de CO_2 et de polluants vont être drastiques pour ne plus autoriser la commercialisation de véhicules conventionnels (moteur thermique seul) et de véhicules hybrides avec un moteur thermique à carburant conventionnel. C'est pourquoi trois axes de développement sont très plébiscités actuellement : le véhicule électrique à batterie, le véhicule hybride à pile à combustible et le moteur à combustion d'e-fuel ou d'hydrogène. La gestion des différentes sources d'énergie un groupe motopropulseur nécessite un contrôleur pour définir quand doit fonctionner chacune des sources d'énergie. Parmi toutes les solutions possibles, l'utilisation de méthodes appartenant au domaine du contrôle optimal sont celles qui sont les plus intéressantes car celles-ci permettent une évaluation équitable des différentes propositions en déterminant la meilleure

suite de commande possible dans un contexte donné. Ces méthodes sont alors décrites comme génératrices de consignes, contrôleur de 'haut niveau' ou comme superviseurs car il est supposé que des régulateurs ou des contrôleurs 'bas niveau' permettront la bonne application des consignes.

La commande optimale, qui est un domaine affilié à l'automatique et aux mathématiques appliquées, est alors d'un intérêt tout particulier pour résoudre la problématique de la répartition de puissance entre les deux parties du groupe motopropulseur hybride. L'histoire de la commande optimale peut être rapporté à celle des calculs infinitésimaux qui sont apparus à la fin du XVII^{ème} siècle. De nombreux scientifiques ont continué à développer le domaine, comme Hamilton, Euler et Lagrange pour ne citer qu'eux. Toutefois, ce n'est que bien plus tard, au milieu du XX^{ème} siècle, que Bellman et Pontriaguine ont développé chacun de leur côté deux méthodologies totalement différentes révolutionnant le domaine. En effet, Richard Bellman a théorisé la programmation dynamique, une méthode numérique permettant de trouver la solution optimale à un problème en prenant en compte l'état du système. De ce fait, la solution est en boucle fermée ; c'est-à-dire qu'en fonction de l'état du système, il est possible que la commande change. Lev Pontriaguine a développé le principe de maximum de Pontriaguine donnant une solution du problème en boucle ouverte. Ces deux approches ont finalement le même socle commun. C'est pourquoi, il est possible de lier le principe de la Programmation dynamique de Bellman au Principe de Maximum de Pontriaguine. Finalement, l'utilisation des méthodes de contrôle optimal n'est pas exclusive à l'automobile. En effet, de nombreux domaines comme celui des transports avec l'aviation, le ferroviaire, le maritime utilisent des méthodes de commande optimale. Il est également possible de retrouver ces méthodes dans le domaine de l'économie ou des sciences de gestion par exemple.

Cette thèse réalisée en partenariat entre Stellantis, le laboratoire PRISME de l'université d'Orléans et l'Association Nationale de la Recherche et de la Technologie s'inscrit dans un contexte automobile en plein changement. En effet, l'électrification des véhicules se fait de plus en plus importante pour faire face aux enjeux sociétaux. Dans ce cadre de réduction de la consommation d'énergie des véhicules, l'optimisation énergétique des systèmes dans le véhicule est une partie de la solution. Une autre partie est la coordination des différents éléments composant la chaîne de traction. Afin d'aider les équipes lors de la préconception des véhicules, le développement d'outils et améliorations permettant l'application des méthodes de commande optimale sur des systèmes complexes a été investigué. Ainsi des modèles plus réalistes peuvent être considérés dès la phase de préconception, permettant une meilleure évaluation des composants de la chaîne de traction étudiée. La thèse est alors décomposée en cinq chapitres :

Le **chapitre 1** revient sur les méthodes ou groupes de méthodes de commande optimale existants et justifie les choix faits sur les différentes méthodes utilisées durant la thèse. Aussi, un approfondissement est fait sur la Programmation Dynamique, son fonctionnement et les différents outils qui existent dans la littérature pour améliorer son fonctionnement. Le Principe de Maximum de Pontriaguine est également développé pour illustrer le lien existant avec la première méthode.

Le **chapitre 2** développe un cas d'application de la Programmation Dynamique qui porte

sur l'intégration d'une structure de systèmes récupératifs de puissance dans un véhicule hybride électrique série. La modélisation du système et son contrôle obtenu par Programmation Dynamique sont détaillés.

Le **chapitre 3** apporte, dans un premier temps, la généralisation des *boundary lines* qui est une des améliorations de la Programmation Dynamique. L'amélioration, disponible que pour des systèmes à une seule dynamique, permet d'améliorer significativement les résultats d'une Programmation Dynamique. Cependant, la construction de cette dernière ne permettait pas une application pour un système à deux dynamiques. Dans un second temps, un intérêt est porté aux maillage hétérogènes de la Programmation Dynamique. En répartissant au mieux les ressources de calcul, il est possible d'observer un compris à ne pas négliger entre le temps de calcul et la qualité du résultat obtenu.

Le **chapitre 4** revient sur le développement d'une méthodologie qui, en se basant sur une analyse fréquentielle du modèle, indique s'il est possible ou non de générer des sous-problèmes de commande optimale à partir du problème original. Cela a pour objectif de réduire la complexité de résolution des problèmes de commande optimale. Cette réduction de la complexité permet d'augmenter la rapidité de résolution du problème de commande optimale.

Le **chapitre 5** applique les différents outils et les méthodologies mises en place durant la thèse sur un cas d'application ayant pour objet la recharge rapide d'une batterie d'un véhicule électrique.

Chapitre 1

Commande optimale par Programmation Dynamique

Sommaire

1.1 Introduction

Le domaine de la commande optimale est inclus dans la science de l'automatique et à l'interface des mathématiques appliquées. L'automatique est caractérisé par la volonté d'influer sur un système dynamique dans l'objectif de contrôler ce système d'un état initial à un état final désiré. Cette transformation est sujette à diverses contraintes physiques. Une variable dite de contrôle, ou appelée commande, permet au système d'agir et de répondre aux perturbations de l'environnement. La question de la contrôlabilité du système est alors de rigueur. Le terme contrôlabilité est défini comme l'existence d'une suite de commande permettant au système étudié de varier d'un état initial vers l'état final voulu. Dans le reste du manuscrit, le vecteur des variables de commande sera dénoté $\boldsymbol{u}$ tandis que le vecteur des variables d'état sera identifié $\boldsymbol{x}$. Afin d'expliquer les contextes et enjeux du contrôle optimal, le reste de l'introduction permet d'évoquer les différentes notions importantes du domaine, ainsi qu'une description des méthodes utilisées.

Définition La problématique de commande optimale apparaît après la résolution du problème de contrôlabilité selon [1]. En effet, si le système est contrôlable et qu'un objectif particulier est à atteindre, alors les méthodologies de contrôle optimal sont à investiguer. A partir de cette définition, nous appliquerons des méthodologies de contrôle optimal à des modèles, qui simulent des systèmes, dans le but d'optimiser un objectif, appelé critère d'optimisation, tout en respectant les contraintes des systèmes.

Modèle Le modèle est défini par ses variables, commandes $\boldsymbol{u}$ et états $\boldsymbol{x}$, mais également par les variations des états dans le temps $\dot{\boldsymbol{x}}$, aussi appelées dynamiques.

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \tag{1.1}$$

(1.1) exprime alors que les dynamiques du modèle dépendent des états et des commandes de ce dernier mais également des perturbations $\boldsymbol{w}$ appliquées à l'instant t sur le modèle. Il est possible d'évaluer la valeur de toutes les grandeurs physiques à partir des équations du modèle, des états et des commandes.

Contraintes Le modèle est également amené à présenter certaines contraintes d'utilisation. Ces dernières limitent le modèle dans ses possibilités de contrôle car une commande est impossible dans la situation actuelle du système. Les contraintes physiques du modèle vont alors impacter la suite de commande, nommée *control policy* en anglais, à appliquer au système.

Critère Enfin, la suite de commande va également être affectée par le critère d'optimisation que le modèle doit respecter durant son fonctionnement. Le critère d'optimisation, nommé J est défini comme :

$$J(t_0, \boldsymbol{x}, \boldsymbol{u}) = \int_{t_0}^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t)).dt + \Phi(\boldsymbol{x}(t_f)) \tag{1.2}$$

Le critère est composé de l'intégrale des coûts instantanés L par rapport au temps à laquelle est ajoutée une pénalité finale Φ en fonction de l'état final du système $\boldsymbol{x}(t_f)$. Dans le domaine automobile, ce critère d'optimisation est généralement la consommation de carburant [2] [3] [4] [5] qui est à minimiser. Toutefois, ce critère pourrait très bien chercher à minimiser l'énergie totale consommée par le véhicule. Un autre objectif plébiscité dans le domaine est le temps de trajet [6]. En effet, [7] [8] montrent qu'un contrôle de la vitesse permet de minimiser la consommation de carburant avec une augmentation du temps de trajet très faible. Ce contrôle de vitesse est d'autant plus important pour les véhicules électriques [9] [10] [11]. Dans tous les cas, il est alors question de Loi de Gestion de l'Energie (LGE). Pour aller plus loin, il est également possible d'avoir des critères qui visent à faire des compromis entre différents objectifs. Ces critères multi-objectifs peuvent être illustrés par un compromis entre la consommation de carburant et les émissions de polluants du moteur thermique du véhicule hybride [12] [13] [14] ou consommation énergétique et temps de trajet [15] [16].

1.2 Problème de Commande Optimale

1.2.1 Formulation du Problème de Commande Optimale

Afin de déterminer des lois de contrôle optimales grâce aux méthodes citées précédemment, il faut résoudre un Problème de Commande Optimale (appelé *Optimal Control Problem*, soit *OCP* en anglais). L'écriture de ce problème décrit alors le système, ses dynamiques, le critère d'optimisation ainsi que les différentes contraintes que le système doit respecter dans le temps [17]. L'écriture générique d'un *OCP* sous sa forme primale dans le domaine temporel continu est définie dans l'équation (1.3).

$$\begin{cases} \min\limits_{\boldsymbol{u}} J(t_0, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) = \int_{t_0}^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t)).dt + \Phi(\boldsymbol{x}(t_f)) \\ \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \\ \boldsymbol{x}(t) \in \boldsymbol{X}, \forall t \geq t_0 \\ \boldsymbol{u}(t) \in \boldsymbol{U}, \forall t \geq t_0 \end{cases} \tag{1.3}$$

Le coût optimal $J(t_0, \boldsymbol{x}, \boldsymbol{u})$ est alors composé de l'intégrale des coûts instantanés $L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t))$ par rapport au temps à laquelle est ajoutée une pénalité finale Φ en fonction de l'état final du système $\boldsymbol{x}(t_f)$. Ces deux grandeurs dépendent des m états $\boldsymbol{x}$ et des n commandes $\boldsymbol{u}$ tandis que la dernière dépend uniquement des états finaux. Les dynamiques des états $\dot{\boldsymbol{x}}$ dépendent également des états et commandes mais également des perturbations $\boldsymbol{w}(t)$ extérieures appliquées au système. Les ensembles des états et des commandes sont définis comme suit :

$$\begin{aligned} \boldsymbol{U} &= \{\boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{u}_{\min}(t) \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max}(t)\} \\ \boldsymbol{X} &= \{\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{x}_{\min}(t) \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{\max}(t)\} \end{aligned} \tag{1.4}$$

Avec respectivement n et m le nombre de commandes et d'états impliqués dans le problème, l'équation (1.4) induit ici implicitement que les ensembles autorisés pour les commandes et les états varient dans le temps.

Dans le cas d'une résolution du problème de contrôle optimal avec une discrétisation temporelle, l'écriture du problème devient légèrement différente. La discrétisation temporelle implique alors que les dérivées temporelles des équations différentielles sont approximées par des schémas numériques. Ces derniers impliquent alors des relations de récurrence pour calculer les évolutions des variables dans le temps.

$$\begin{cases} \min\limits_{\boldsymbol{u}} J(k, \boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=k}^{N-1} L(i, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i) + \Phi\left(\boldsymbol{x}_N\right) \\ \boldsymbol{x}_{i+1} = f_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i) \\ \boldsymbol{x}_i \in \boldsymbol{X}_i, \forall i \geq 0 \\ \boldsymbol{u}_i \in \boldsymbol{U}_i, \forall i \geq 0 \end{cases} \tag{1.5}$$

L'écriture du problème discret (1.5) réutilise les mêmes variables que dans celle du problème continu (1.3). La variable k représente l'indice de temps du problème et l'intégrale est

devenue une somme. N est le nombre de pas de temps effectué pour atteindre le temps final t_f, soit $t_f = N.\Delta t + t_0$ et Δt est la durée temporelle d'un pas de temps. Les ensembles des états et des commandes à chaque pas de temps sont décrits par (1.6)

$$\begin{aligned} \boldsymbol{U}_k &= \{\boldsymbol{u}_k \in \mathbb{R}^n, \boldsymbol{u}_{\min,k} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max,k}\} \\ \boldsymbol{X}_k &= \{\boldsymbol{x}_k \in \mathbb{R}^m, \boldsymbol{x}_{\min,k} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{\max,k}\} \end{aligned} \tag{1.6}$$

Dans le reste du manuscrit, le schéma numérique utilisé pour approximer les dérivées temporelles des variables d'états est le schéma dit Euler explicite (1.7).

$$\dot{x} \approx \frac{x_{k+1} - x_k}{\Delta t} \tag{1.7}$$

D'autres schémas existent tels que les méthodes de Runge-Kutta par exemple. Nous avons alors les relations de récurrence suivantes :

$$\boldsymbol{x}_{k+1} = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \tag{1.8}$$

1.2.2 Méthodes de résolution du Problème de Commande Optimale

Un certain nombre de méthode permettent de traiter les problèmes de contrôle optimal. Ces méthodes ont des applications différentes en fonction du contexte d'utilisation. En effet, deux catégories de méthodes émergent : les méthodes dites hors-ligne et celles dites en-ligne, respectivement *offline* et *online* en anglais. Les premières, qui n'ont pas pour but d'être embarquées, permettent d'évaluer les résultats de référence du problème d'optimisation. D'un autre côté, les méthodes en-ligne sont celles embarquées dans le système. De ce fait, l'objectif des méthodes est de donner des résultats les plus proches possible des méthodes hors-ligne. Les principales méthodes ou groupes de méthodes sont décrits mais un intérêt particulier sera porté sur les deux méthodes les plus connues : la Programmation Dynamique (*DP*) en section 1.4 et le Principe de Maximum de Pontriaguine (PMP) en section 1.5.

Programmation Dynamique : Cette méthode basée sur le principe de R.Bellman dans les années 50, est une méthode numérique [18] [19] [20]. Cela veut dire que les variables de commandes et d'états ainsi que le temps sont discrétisés. La discrétisation des variables s'effectue entre les valeurs minimales et maximales possibles pour chaque variable tandis que le temps est discrétisé de l'instant initial t_0 à l'instant final t_f avec un pas de temps Δt. Cette discrétisation, également appelée maillage, implique l'inconvénient principal de la méthode qui est le temps de calcul. En effet, plus le maillage est fin (beaucoup de mailles), plus le temps de calcul augmente (fléau de la dimensionnalité ou *curse of dimensionality* en anglais). Cependant, malgré cet inconvénient, la programmation dynamique est une méthode régulièrement utilisée car elle à l'avantage de prendre en compte les non-linéarités du modèle ainsi que les contraintes imposées à ce dernier. De ce fait, l'optimalité de la solution est assurée. Il est important de notifier que la précision ou qualité de la solution obtenue dépend du maillage utilisé. Il est alors question de sous-optimalité de l'algorithme. La méthode est applicable à

des systèmes à plusieurs commandes et plusieurs états. L'augmentation du nombre de variable implique uniquement une augmentation du nombre de calculs à faire et donc du temps de calcul.

Principe de Maximum de Pontriaguine : Le principe de Maximum de Pontriaguine (PMP) [21] [22] [23] [24] est une méthode analytique qui peut être résolue numériquement, permettant de résoudre les problèmes de contrôle optimal. L'optimalité de la solution n'est cependant vérifiée que pour les problèmes convexes. Il est alors possible que les approximations faites pour que le modèle soit convexe mène à une sous-optimalité de la solution obtenue. Dans le domaine automobile, un moteur (thermique ou électrique) est souvent modélisé par une cartographie de rendement en fonction du couple et du régime de rotation du moteur. Pour utiliser le PMP avec cette modélisation des moteurs, il est nécessaire de déterminer des fonctions analytiques qui retranscrivent les informations contenues par la cartographie.

Méthodes Directes : Cette catégorie de méthode qui est composée principalement de *Linear Programming*, *Quadratic Programming*, *Linear Matrix Inequalities* [25][26][27] permet la résolution de problème d'optimisation statique. Malgré la considération de problèmes dynamiques dans notre cas, il est possible d'écrire ces derniers de telle manière à rendre une optimisation statique équivalente à une optimisation dynamique, notamment en résolvant le problème d'optimisation en une seule fois comme le montre [28]. L'intérêt de ces méthodes sont leur rapidité de résolution ainsi que les outils disponibles pour résoudre les différentes méthodes. En effet, un certain nombre d'algorithmes ont été mis en place afin de résoudre les différents problèmes. Par exemple, selon [29], l'algorithme du simplexe est le plus adéquat et le plus efficace pour résoudre les problèmes d'optimisation linéaires comme le *Linear Programming* par exemple. Les méthodes citées ici ne permettent que la résolution de problèmes linéaires ou quadratiques, ce qui est restrictif dans les modèles et approximations étudiés. Cependant, une recherche récente est portée sur des outils qui permettent la résolution de problèmes non-linéaires comme [30] CasADi par exemple.

Méthodes par apprentissage : Un autre groupe de méthodes porte sur de l'intelligence artificielle (IA) [31] [32] [33]. Un réseau de neurones est mis en place afin de résoudre le problème d'optimisation entre les entrées et les sorties de ce dernier. Les sorties du réseau de neurones sont les commandes du modèle tandis que les entrées sont les états du modèle. La complexité de ces méthodes réside dans le choix de la structure du réseau de neurones (nombre de couches intermédiaires, nombre de neurones utilisés, fonctions utilisées dans les neurones, etc...), ainsi que dans le choix de l'algorithme qui comprend le réglage des hyper-paramètres du réseau de neurones. Un exemple d'hyper-paramètre est le taux d'apprentissage [34]. Dans le cadre d'un stage en support de la thèse, le *Reinforcement Learning* (*RL*) a été étudié pour mettre en place une loi de commande basée sur un réseau de neurones. Cet algorithme, est composé d'un agent, d'un modèle et d'une fonction récompense. A chaque décision prise par l'agent, ce dernier est récompensé ou pénalisé grâce à la fonction récompense. C'est cette dernière qui va modifier le comportement de l'agent. Une séance d'entraînement est alors mise en place afin de permettre un bon comportement de l'agent vis-à-vis de l'objectif attendu. L'intérêt de la méthode est de pouvoir embarquer un agent entraîné hors-ligne dans le système souhaité, ce qui permet un contrôle performant et robuste. Cependant, le temps

et/ou la quantité de données nécessaires à l'apprentissage peuvent être particulièrement élevés. Cependant, aucun résultat probant n'a été obtenu pour le moment.

Méthodes d'algorithmes évolutionnaires : Cette catégorie de méthode (*EA*) repose sur des algorithmes reproduisant la théorie de l'évolution [35] [36] [37]. Les algorithmes génèrent un certain nombre d'individus, appelés population, avec des caractéristiques différentes. Ces dernières sont les commandes successives à appliquer au modèle dans notre cas. Ainsi, toute la population est évaluée sur la simulation entière, puis une transformation est effectuée au sein de cette population [38]. Différents opérateurs permettent ce changement : sélection, croisement et mutation selon [39]. Ces opérateurs visent à améliorer la performance des populations successives, également appelées générations. En fonction d'un nombre de générations donné, les paramètres utilisés dans les opérateurs varient pour explorer le mieux possible l'ensemble des solutions possibles. L'inconvénient de cette méthode est la non assurance de l'optimalité de la solution. En effet, étant donné que l'espace des solutions possibles est échantillonné, il est possible que le nombre d'essais ne soit pas suffisant pour trouver la solution optimale au problème donné. C'est pourquoi le temps de calcul de ces algorithmes est important si l'assurance de l'optimalité est recherchée.

La table 1.1 permet de remettre en contexte les différentes méthodes, et d'expliquer les choix opérés durant la thèse. Pour ce faire, les critères d'importance pour l'étude sont listés par ordre d'importance : assurance de l'optimalité de la solution, possibilité d'utiliser des modèles non linéaires, la prise en compte des contraintes, la faciliter à utiliser la méthode en augmentant le nombre de variables d'états et de commandes ainsi que le temps de calcul. L'évaluation des méthodes par rapport aux critères est effectuée en se basant sur les utilisations et limitations recensées des méthodes. L'évaluation peut varier en fonction de l'expertise possédée sur chacune des méthodes. En effet, un expert d'une méthode directe peut être capable de prendre en compte les contraintes plus facilement que ce qui est présenté dans la table par exemple.

Table 1.1 – Table récapitulative des méthodes ainsi que leurs avantages et inconvénients

Critère \ Méthode	*DP*	PMP	Méthodes Directes	AI	*EA*
Assurance de l'optimalité	++	++	++	-	-
Modèle non linéaire	++	+	+	++	++
Prise en compte des contraintes	++	+	+	++	++
Dimension du problème	++	+	++	++	++
Temps de calcul	- -	++	++	- -	- -

L'étude et l'utilisation de la programmation dynamique au cours de la thèse sont explicitées par la table. En effet, la méthode répond à toutes nos attentes mais le temps de calcul est le point bloquant de son utilisation. C'est pourquoi, au cours de la thèse, des outils permettant de réduire ce phénomène ont été développés.

Il existe une approche, le *Model Predictive Control* (*MPC*) [40] [41] [42] [43], qui se base comme son nom l'indique sur le modèle du système et sur son comportement. Grâce aux

dynamiques du système, il est alors possible de déterminer dans quel état sera le système à partir de ses conditions initiales. La méthode est discrète en temps (les calculs se font donc par pas de temps) mais la continuité ou la discrétisation des états et des commandes dépend de la méthode de résolution (*solver*) des problèmes de contrôle optimal sélectionnée. Ces méthodes de résolution sont celles présentées précédemment. L'approche *MPC* va imposer successivement la résolution de problèmes de contrôle optimal sur une période de temps plus ou moins longue de N_{hor} pas de temps. A partir des conditions initiales du problème global, une première optimisation est réalisée, dont la solution (commande optimale) du premier pas de temps uniquement est appliquée au modèle. L'état du modèle évolue ainsi dans le temps. Pour résoudre le problème dans son entièreté, un nouveau problème d'optimisation est reposé à partir des conditions actuelles du modèle comme conditions initiales. L'optimisation est alors ré-effectuée et le résultat du premier pas de temps du problème est appliqué au modèle. Le travail d'optimisation est alors réalisé jusqu'à atteindre la fin de problème global et ces N pas de temps. D'une manière générale, les modèles peuvent au plus être convexes pour ne pas avoir des solutions dans des extremums locaux. Il est toutefois possible d'avoir des méthodes de résolution non-linéaires et donc avoir un *MPC* non-linéaire.

1.3 Exemple introductif : Modèle véhicule hybride électrique parallèle

Modèle Le modèle utilisé pour présenter et illustrer les différentes méthodes durant le chapitre est visible en figure 1.1. Ce système est un véhicule hybride électrique parallèle dont l'objectif est d'effectuer un cycle *WLTC*. Le contrôleur *ECU* du véhicule émet des consignes de répartition de puissance entre les deux moteurs de la chaîne de traction. Le contrôle "haut niveau" du système est effectué en pilotant le couple de la machine électrique $\boldsymbol{u} = C_{Mel}$. Cette décision se base sur la quantité d'énergie, appelée *State Of Charge* en anglais et est désignée par l'acronyme *SOC*, contenue dans la batterie $\boldsymbol{x} = SOC_{Bat}$.

La puissance mécanique de la machine électrique est définie simplement par le produit de son couple C_{Mel} en N.m et son régime de rotation ω_{Mel} en rad/s :

$$P_{Mel} = C_{Mel}.\omega_{Mel} \tag{1.9}$$

Le régime de rotation est soumis à une perturbation $\boldsymbol{w}(t)$ qui varie dans le temps. Cette perturbation est ici la vitesse de déplacement du véhicule $v(t)$. Le rapport de réduction $\gamma_r = 32/5$ ainsi que la circonférence des roues $c_W = 2$ m permettent de lier directement la vitesse de rotation de la machine électrique à la vitesse du véhicule :

$$\omega_{Mel} = \frac{2.\pi.\gamma_r.v(t)}{c_W} \tag{1.10}$$

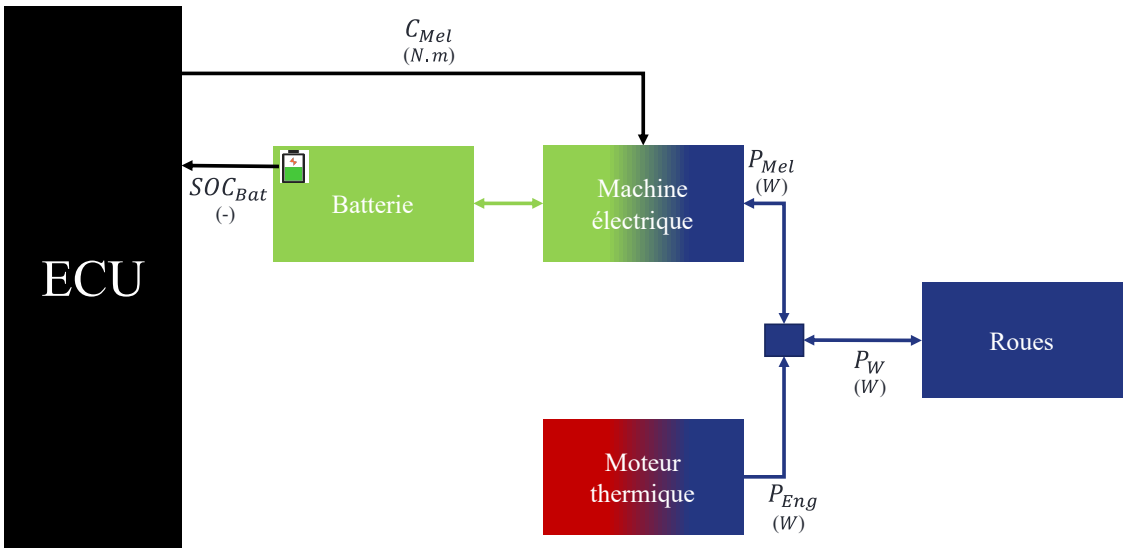

FIGURE 1.1 – Schéma bloc du modèle véhicule utilisé. Les blocs et flèches en bleu marine représentent les éléments mécaniques et en vert clair sont représentés les éléments ou flux électrique. L'*ECU* et les flux associés sont représentés en noir.

A tout instant, la relation de conservation de puissance (1.11) doit être vérifiée.

$$P_{Mel} + P_{Eng} = P_W \tag{1.11}$$

La puissance nécessaire à fournir aux roues P_W pour propulser le véhicule à la vitesse souhaitée est égale à la somme des puissances de la machine électrique P_{Mel} et du moteur thermique P_{Eng}. La puissance requise aux roues P_W dépend de la masse volumique de l'air $\rho = 1,2$ kg/m^3, du produit de la surface frontale S et du coefficient de trainée Cx tel que $S.Cx = 0,625$ m^2, de la masse $m_{veh} = 1000$ kg du véhicule ainsi que de la vitesse et de l'accélération désirées. Cette puissance est une conséquence de la perturbation appliquée au modèle :

$$P_W = \frac{1}{2}.\rho.S.Cx.v^3(t) + m_{veh}.a(t).v(t) \tag{1.12}$$

Il n'est alors pas nécessaire de conserver la variable P_{Eng} comme état ou commande car il est possible de retrouver cette valeur à tout moment grâce à la commande C_{Mel} ainsi qu'à la perturbation $v(t)$.

$$\begin{aligned} P_{Eng}(t) =& P_W(t) - P_{Mel}(t) \\ P_{Eng}(t) =& \frac{1}{2}.\rho.S.Cx.v^3(t) + m_{veh}.a(t).v(t) - C_{Mel}(t).\omega_{Mel}(t) \end{aligned} \tag{1.13}$$

Pour fournir la puissance P_{Eng}, le moteur thermique converti une puissance chimique contenue par le carburant P_F en puissance mécanique. Cette conversion est caractérisée par le rendement du moteur $\eta_{Eng} = 0,3$ pour ce modèle. Ainsi l'équation suivante relie ces deux

puissances et ce rendement :

$$P_F = \frac{P_{Eng}}{\eta_{Eng}} \tag{1.14}$$

Afin de prendre ces décisions, l'unité de contrôle a besoin de l'information de l'état de charge de la batterie SOC_{Bat}. Cette variable, qui évolue entre 0 et 1, représente le rapport de capacité entre celle actuellement contenue dans la batterie A_{act} et celle maximale admissible A_{max}.

$$SOC_{Bat}(t) = \frac{A_{act}(t)}{A_{max}} \tag{1.15}$$

La variation de cette dernière est directement liée à la quantité de charge électrique entrant ou sortant de la batterie :

$$SOC_{Bat}(t) = \int_{t_0}^{t} \frac{-I_{Bat}(\tau)}{A_{max}}.d\tau + SOC_{Bat}(t_0) \tag{1.16}$$

L'équation (1.17) décrit ainsi la variation d'état de charge de la batterie dans le temps. Diviser par la tension de la batterie V_{Bat} permet de convertir la puissance électrique en courant et diviser ce courant par la capacité maximale A_{max} permet de calculer la variation d'état de charge de la batterie. La résistance de la machine électrique $R = 48,47$ mΩ ainsi que la constance de couple moteur $r = 0,6143$ N.m/A permettent de calculer les pertes par effet Joule de la machine élextique

$$\dot{SOC}_{Bat}(t) = \frac{-I_{Bat}(t)}{A_{max}} = -\frac{R.(\frac{C_{Mel}(t)}{r})^2 + C_{Mel}(t).\omega_{Mel}(t)}{A_{max}.V_{Bat}} \tag{1.17}$$

L'expression dans le domaine discrète de (1.17) est l'expression suivante, avec Δt le pas de temps de la discrétisation :

$$SOC_{k+1} = f_k(SOC_{Bat_k}, C_{Mel_k}) = SOC_k - \frac{R.(\frac{C_{Mel_k}}{r})^2 + C_{Mel_k}.\omega_{Mel_k}}{A_{max}.V_{Bat}}.\Delta t \tag{1.18}$$

Contraintes Six inéquations permettent d'exprimer les contraintes physiques que doit respecter le modèle. Pour commencer, l'état de charge de la batterie est nécessairement compris entre $\underline{SOC_{Bat}} = 0$ et $\overline{SOC_{Bat}} = 1$. il peut alors être observé que si $SOC_{Bat}(t) = 1$, alors il n'est pas possible de recharger la batterie. Alors, selon la convention de signe appliquée ici, un couple de machine électrique négatif $C_{Mel}(t) < 0$ est inenvisageable pour le bon respect de toutes les contraintes imposées au modèle.

Le couple de machine électrique doit être compris entre $\underline{C_{Mel}} = -86$ N.m et $\overline{C_{Mel}} = 86$ N.m. La puissance du moteur thermique quant à elle, doit être supérieure ou égale à la valeur minimale possible $\underline{P_{Eng}} = 0$ W mais également inférieure ou égale à la puissance maximale délivrable par ce dernier $\overline{P_{Eng}} = 15$ kW.

$$\begin{cases} \underline{SOC_{Bat}} \leq SOC_{Bat}(t) \leq \overline{SOC_{Bat}} \\ \underline{C_{Mel}} \leq C_{Mel}(t) \leq \overline{C_{Mel}} \\ \underline{P_{Eng}} \leq P_{Eng}(t) \leq \overline{P_{Eng}} \end{cases} \tag{1.19}$$

Critère L'objectif dans cet exemple est de réduire la consommation d'énergie du véhicule. Cet objectif est traduit par le critère de minimisation suivant :

$$J(t_0, \boldsymbol{x}, \boldsymbol{u}) = \int_{t_0}^{t_f} p. \left(P_F^2(t) + P_{Bat}^2(t)\right) .dt + \Phi(\boldsymbol{x}(t_f)) \tag{1.20}$$

Avec P_F la puissance de carburant nécessaire au moteur thermique pour développer la puissance P_{Eng} et P_{Bat} la puissance totale délivrée ou perçue par la batterie. Le terme $p = 10^{-11}$ J^{-2} permet d'avoir le critère d'optimisation J homogène à une énergie.

1.4 Programmation Dynamique

Richard Bellman a énoncé le principe suivant dans [18] :« *Une suite de décisions est optimale si, quel que soit l'état et l'instant considérés sur la trajectoire qui lui est associée, les décisions ultérieures constituent une suite optimale de décisions pour le sous problème dynamique ayant cet état et cet instant comme conditions initiales.* ». Par définition, la variable V représente la solution optimale au problème posé en fonction du temps t ou de l'indice temporel k en fonction de la nature continue ou discrète du problème et de l'état du système. Le principe de la programmation dynamique, exprimé en discret [44], est le suivant :

$$V(k, \boldsymbol{x}_k) = \min_{\boldsymbol{u}}(J(k, \boldsymbol{x}_k, \boldsymbol{u}_k)) \tag{1.21}$$

En réécrivant la première première ligne de (1.5) en décomposant la somme en deux parties, il peut être écrit :

$$J(k, \boldsymbol{x}_k, \boldsymbol{u}_k) = L(k, \boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k).\Delta t + \sum_{i=k+1}^{N-1} L(i, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i).\Delta t + \Phi\left(\boldsymbol{x}_N\right) \tag{1.22}$$

Il peut être vu que la seconde somme de (1.22) peut être récrite comme (1.23) en utilisant le principe de la Programmation Dynamique (1.21) :

$$\sum_{i=k+1}^{N-1} L(i, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i).\Delta t + \Phi\left(\boldsymbol{x}_N\right) = V(k+1, \boldsymbol{x}_{k+1}) \tag{1.23}$$

Avec $\boldsymbol{x}_{k+1}$ défini par (1.7). Il est ainsi possible d'écrire la relation suivante :

$$J(k, \boldsymbol{x}_k, \boldsymbol{u}_k) = L(k, \boldsymbol{x}_k, \boldsymbol{u}_k) + V(k+1, \boldsymbol{x}_{k+1}) \tag{1.24}$$

L'équation de la programmation dynamique utilisée dans un algorithme homonyme est alors la suivante :

$$V(k, \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}(L(k, \boldsymbol{x}_k, \boldsymbol{u}_k) + V(k+1, \boldsymbol{x}_{k+1})) \tag{1.25}$$

La variable $V(k, \boldsymbol{x}_k)$ est appelée coût jusqu'à la fin, ou *cost to go* en anglais. On retrouve le coût instantané $L(k, \boldsymbol{x}_k, \boldsymbol{u}_k)$ mais on voit également une récursivité dans l'expression. En effet, afin de calculer V à un instant k, il est nécessaire de connaître V à l'instant $k+1$. On rappelle que $\boldsymbol{x}_{k+1}$ est défini dans l'équation (1.5). Un constat alors évident se fait avec cette équation : pour appliquer le principe de Bellman, l'information doit se transporter de la fin temporelle du problème vers le début. A chaque pas de temps remonté, on pourra alors estimer l'ensemble des commandes $\boldsymbol{u}_k$ qui minimisent $V(k, \boldsymbol{x}_k)$.

Il n'est pas forcément simple de voir la différence entre la *DP* et la méthode de recherche exhaustive, aussi appelée recherche par force brute ou *brute-force search* en anglais. Toutefois, l'ordre et la récursivité des calculs créent toute la différence entre les deux méthodes. D'un côté, la méthode *brut force* essaie toutes les commandes possibles à tous les états possibles à tous les instants, ce qui revient à tester toutes les propositions possibles. D'un autre côté, la *DP* essaie toutes les commandes possibles à tous les états possibles à un instant donné, mais conserve la meilleure solution pour chaque état pour réutiliser le résultat, pour l'instant naturel, d'avant. La complexité de calcul en devient alors réduite tout comme le temps de calcul.

L'outil de programmation dynamique peut être utilisé aussi bien avec une approche déterministe qu'avec une approche stochastique. Il est alors question de *Deterministic Dynamic Programming* (*DDP*) et de *Stochastic Dynamic Programming* (*SDP*) [45] [46] [47]. L'aspect stochastique intervient lorsque les valeurs des perturbations $\boldsymbol{w}$ à un instant ou pas de temps donné sont incertaines. Dès lors, ce n'est plus le coût qui est optimisé (1.3) ou (1.5) mais l'espérance du coût qui est maintenant calculée. [16]. Dans le reste du manuscrit, il ne sera fait référence qu'à la première catégorie de programmation dynamique dont les trois améliorations principales sont : l'*Iterative Dynamic Programming* (*IDP*), l'*Approximate Dynamic Programming* (*ADP*) et les *Boundary Lines* (*BL*). Ce choix ce justifie par la connaissance de toutes les perturbations des systèmes étudiés dans ce manuscrit.

***Iterative Dynamic Programming* (*IDP*)** L'amélioration *Iterative Dynamic Programming* consiste à réaliser une succession de programmation dynamique pour un même problème [48] [49]. L'objectif est alors de diminuer la complexité de calcul de l'algorithme en considérant des maillages plus grossiers. Pour arriver à ce résultat, l'amélioration commence par une programmation dynamique ordinaire avec un maillage uniforme entre les valeurs minimale et maximale admissibles par le(s) état(s) comme sur le maillage noir de la figure 1.4. Une première trajectoire est alors déterminée à partir de ce maillage. Cette dernière va alors servir de base pour le maillage de la seconde programmation dynamique. Dans les faits, pour chaque pas de temps, le maillage est alors centré autour de l'état du système à l'instant k. Les bornes extremums du maillage à ce pas de temps sont alors différentes. Un nombre identique de maille est utilisé à chaque pas de temps et également à chaque nouvelle itération de

la programmation dynamique. Le nombre d'itération dépend alors du temps que va mettre l'algorithme à donner deux solutions successives identiques. La figure 1.2 issue de [50] illustre l'espace exploré en rose en fonction de la trajectoire de référence en vert.

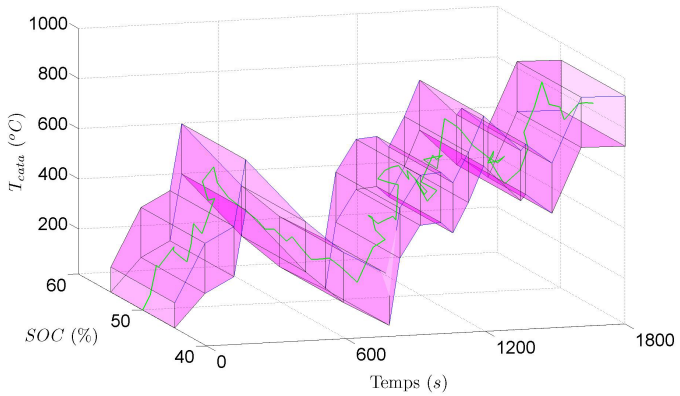

FIGURE 1.2 – Figure illustrative de l'*IDP* avec la trajectoire de référence en vert et l'espace exploré en rose. (Figure 8.4 [50])

Approximate Dynamic Programming L'*Approximate Dynamic Programming* est une amélioration de la programmation dynamique qui vise à amoindrir le fléau de la dimension induit par la présence d'un nombre de variables important dans la programmation dynamique [51] [52]. L'objectif est alors de réduire la complexité numérique de la partie *backward* de l'algorithme, section 1.4.1.1, qui représente la majeur partie du temps de calcul de la programmation dynamique. L'estimation classique de la matrice de coût optimal V est alors remplacée par une estimation plus rapide à calculer de cette dernière. Cette estimation de V peut être effectuée par plusieurs méthodes : réseau de neurones, approximation paramétrique, etc... [53]. Ainsi, à chaque pas de temps de chaque itération, V est mise à jour pour tendre vers l'optimalité. C'est pourquoi, l'intérêt principal de cet outil est la résolution de problèmes trop complexes pour une programmation dynamique classique dans un temps de calcul raisonnable. L'inconvénient de cette amélioration cependant, est la performance de la solution obtenue qui est toujours inférieure à celle d'une programmation dynamique classique quand la comparaison est faisable [54].

Boundary lines L'amélioration *Boundary lines* a pour objectif d'améliorer la qualité du résultat de la programmation dynamique avec un seul état [55]. L'objectif est de déterminer deux frontières (une haute et une basse) qui évoluent dans le temps en fonction des contraintes imposées au modèle. La construction de ces frontières s'effectue de façon rétrograde en temps. La connaissance des limites du modèle permet alors d'améliorer la précision des résultats enregistrés dans la matrice V. Les inconvénients principaux de l'outil sont d'ajouter du temps de calcul au calcul initial ainsi que d'être restreint à des modèles à un seul état uniquement. Cependant, l'avantage majeur de l'outil est d'améliorer grandement la performance des résultats obtenus.

L'intérêt porté à l'*Iterative Dynamic Programming* et aux *boundary lines* dans ce manuscrit est plus important que celui porté à l'*Approximate Dynamic Programming*. En effet, cette dernière amélioration repose sur une approximation du coût optimal pour réduire le temps de calcul. Or cette thèse a pour objectif de conserver les méthodes les plus précises tout en réduisant leurs temps de calcul.

1.4.1 Principe de la Programmation Dynamique

1.4.1.1 Méthode et Algorithme initiaux

L'algorithme 1 illustre le principe de la programmation dynamique à un état et une commande. Pour rappel, la *DP* est une méthode numérique qui implique une discrétisation ou maillage des états $\boldsymbol{x}$, des commandes $\boldsymbol{u}$ et du temps. L'algorithme de la programmation dynamique est décomposé en deux étapes. La première, appelée *backward* (boucle rétroactive), consiste à définir les solutions pour chaque sous problème possible [56] [19] [57]. L'algorithme va donc traverser le vecteur temps discrétisé de façon rétrograde. La discrétisation des variables d'états et de commandes peut être réalisée de plusieurs façon différentes : uniforme ou irrégulière, homogène ou hétérogène et permanente ou adaptative. La discrétisation d'une variable est qualifiée d'uniforme si chaque point de la discrétisation est équidistant avec ces voisins proches ; si ce n'est pas le cas, alors la discrétisation est irrégulière. L'homogénéité d'un maillage est qualifiée par le nombre de points disponibles pour chaque variable du problème. Ainsi, un maillage est dit homogène si toutes les variables utilisent le même nombre de mailles ; sinon, le maillage est dit hétérogène. Un maillage est dit permanent si ce dernier n'évolue pas en fonction du temps. Au contraire, un maillage est dit adaptatif s'il est modifié en fonction de l'indice temporel considéré. Par ailleurs, la complexité numérique, qui dépend du maillage utilisé, est un point d'étude non négligeable de la *DP*. En effet, souvent décrite comme étant $O(N.{q_u}^{n_u}.{q_s}^{n_s})$ selon [58][59], cette description sous-entend que les n_u commandes et les n_s états sont discrétisés avec le même nombre de mailles respectivement q_u et q_s. Or, il n'est pas toujours judicieux que le maillage des commandes et des états soit le même. C'est pourquoi, l'écriture de la complexité numérique suivante est proposée : $O(N.\prod_{i=1}^{n_u} q_{u_i}.\prod_{j=1}^{n_s} q_{s_j})$ [60]. Par comparaison, la complexité de calcul de la méthode exhaustive est : $O((\prod_{i=1}^{n_u} q_{u_i}.)^{N-1}\prod_{j=1}^{n_s} q_{s_j})$.

Pour commencer l'algorithme, il faut commencer par initialiser la matrice V pour le pas de temps final N pour chaque maille d'état. Cette initialisation se base sur les contraintes finales que le système doit respecter. La détermination de ces zones est un sujet majeur, notamment dans les cas où des contraintes finales sur les états sont imposées. En effet, lors de l'initialisation du calcul, plusieurs solutions sont possibles pour pénaliser les coûts des points du maillage qui ne respectent pas les contraintes finales :

- Imposer zéro pour les mailles d'état respectant les contraintes tandis que pour les autres mailles, une forte pénalité $\Phi(\boldsymbol{x}_N)$ est imposée pour exclure ces solutions pour l'algorithme de la *DP*.
- Mettre en place des fonctions pénalisant de plus en plus les mailles qui s'éloignent de

la zone cible.

Dans le cas de la première solution, $\Phi(\boldsymbol{x}_N)$ peut prendre n'importe quelle valeur ou fonction tant qu'elle agit comme une pénalité dans le processus de minimisation. Cette valeur peut aussi bien être finie qu'infinie.

Algorithme 1 Algorithme de Programmation Dynamique pour un système à un état et une commande

$\boldsymbol{u} = u_{min} : (u_{max} - u_{min})/(q_u - 1) : u_{max}$
$\boldsymbol{x} = x_{min} : (x_{max} - x_{min})/(q_s - 1) : x_{max}$
Pour $p = N - 1 : -1 : 1$ **Faire** ▷ boucle *Backward*
  Pour $j = 1 : 1 : q_u$ **Faire**
    Pour $i = 1 : 1 : q_s$ **Faire**
      $x_{p+1}^{i,j} = f_p(\boldsymbol{x}^j, \boldsymbol{u}^i, \boldsymbol{w}_p)$
      $g = L(p, \boldsymbol{x}^j, \boldsymbol{u}^i).\Delta t$
      $V_{interp} = \nu(\boldsymbol{x}, V(p+1, \boldsymbol{x}), x_{p+1}^{i,j})$
      $J^i = g + V_{interp} + \mu$ ▷ μ est une variable qui notifie des uplets infaisables
    Fin de Pour
    $V(p, j) = \min_{\boldsymbol{u}}(J)$
    $U(p, j) = \boldsymbol{u}(\underset{\boldsymbol{u}}{argmin}(J))$
  Fin de Pour
Fin de Pour
$Cost = 0$
Pour $t = 1 : 1 : N - 1$ **Faire** ▷ boucle *Forward*
  $u_t = \nu(\boldsymbol{x}, U(t, \boldsymbol{x}), x_t)$
  $x_{t+1} = f_t(x_t, u_t, w_t)$
  $Cost = L(t, x_t, u_t) + Cost$
Fin de Pour

Nomenclature : $\boldsymbol{x}$: vecteurs d'états ; $\boldsymbol{u}$: vecteurs des commandes ; $\nu(a_{ref}, b, a_{que})$ est une fonction d'interpolation qui estime des valeurs pour les coordonnées a_{que} à partir des valeurs connues b pour les coordonnées a_{ref}.

La boucle rétrograde en temps (*backward*) commence alors au pas de temps $N - 1$. De cet instant, pour chaque point du maillage de l'espace des états, tous les n-uplets possibles des commandes sont appliqués. Ces n-uplets sont évalués à partir de l'équation de la programmation dynamique (1.25) en se basant sur le coût instantané $L(k, \boldsymbol{x}_k, \boldsymbol{u}_k)$ et des *cost to go* V associés aux états estimés $\boldsymbol{x}_{k+1}$ à l'instant $k + 1$. Les *cost to go* sont évalués par une interpolation V_{interp} dans l'algorithme. V_{interp} représente le terme $V(k + 1, \boldsymbol{x}_{k+1}))$ de l'équation (1.25) pour des soucis de cohérence algorithmique. Pour finir, en pratique, il est possible d'ajouter une troisième variable μ ainsi montrée dans l'algorithme. Cette variable, permet de notifier si le n-uplet évalué respecte toutes les contraintes imposées au système ou non. Si une contrainte n'est pas respectée, alors la valeur de μ devient celle d'une pénalité pour non respect des contraintes. Cette valeur doit être largement supérieure au coût instantané le plus élevé afin de ne jamais tendre naturellement vers une solution qui ne respecte pas toutes les contraintes. L'équation (1.25) est résolue, ce qui permet d'obtenir les valeurs des *cost to go*

et les valeurs de commandes associées qui sont enregistrées respectivement dans les matrices V et U.

A la suite de cette première partie de l'algorithme, la seconde étape, appelée *forward* (ou sens naturel en français) va déterminer la loi de contrôle optimal en se basant sur les résultats du *backward*. Pour commencer, les conditions initiales de la simulation sont initialisées. Par la suite, la commande optimale est déterminée en se basant sur la matrice des commandes optimales U. Comme il est très probable que l'état du système $\boldsymbol{x}(t)$ à un instant t ne soit pas un nœud du maillage de l'espace des états, une interpolation est alors faite avec les valeurs proches de $\boldsymbol{x}(t)$. Ces commandes $\boldsymbol{u}(t)$ sont ensuite appliquées au système pour faire évoluer dynamiquement le système. Le coût de la solution C est calculé pour pouvoir évaluer cette dernière.

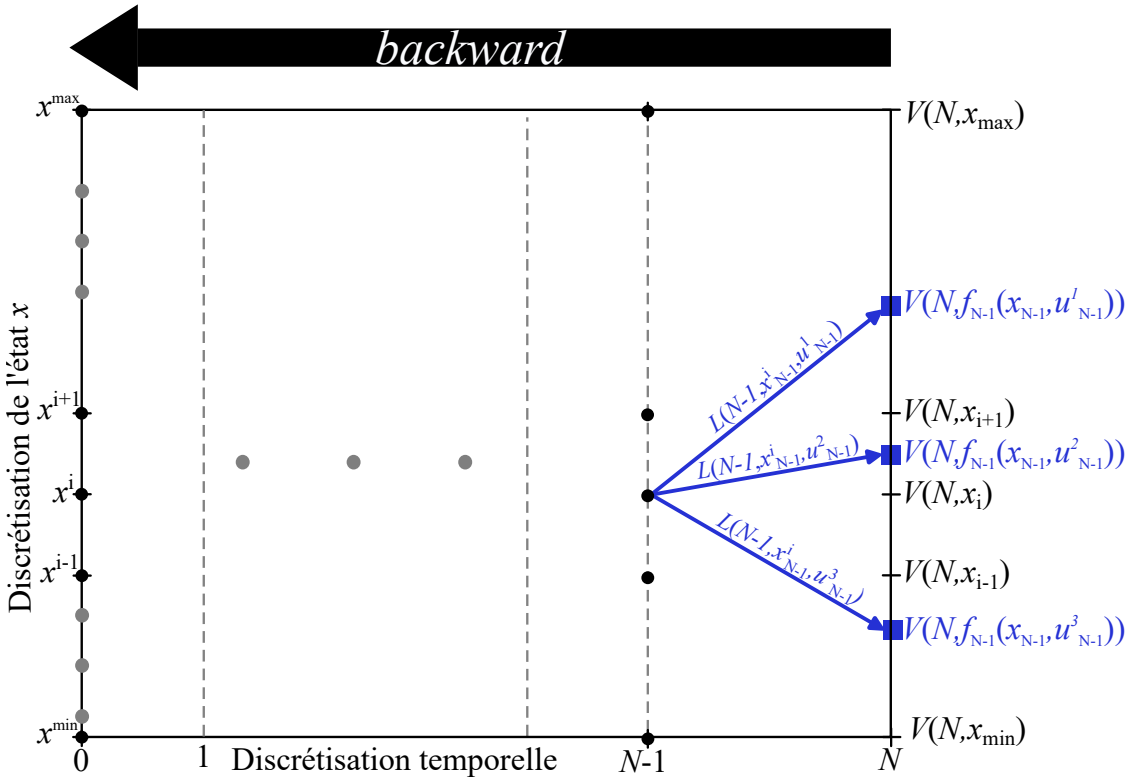

FIGURE 1.3 – Schéma explicatif de la partie *backward* de la Programmation Dynamique (*DP*) pour un problème à un état et une commande ($q_u = 3$).

La figure 1.3 permet d'illustrer le fonctionnement de l'algorithme pour un problème à un seul état et une seule commande pour $q_u = 3$. On lit donc le graphique comme allant d'une manière générale de la droite vers la gauche en suivant la flèche "*backward*". A chaque pas de temps parcouru, le modèle physique est calculé sur un seul pas de temps mais de manière naturelle, en *forward*. Cela permet alors de résoudre l'équation (1.25) pour chaque maille d'état. Une fois l'équation résolue, alors le processus en *backward* continue. Il est donc possible de transporter l'information initiale des *cost to go* finaux V_N^{all} vers les *cost to go* initiaux V_0^{all}. Pour ce faire, il faut donc pouvoir estimer les *cost to go* intermédiaires à chaque

pas de temps pour transporter l'information. Toutes les combinaisons d'états et de commandes possibles (couple état/commande dans l'exemple) sont alors calculées. Il est plus que possible que l'état futur estimé (carrés en bleu à l'instant N) ne soit pas une discrétisation de l'état et qu'il n'y ait pas l'information cherchée. Afin de pallier à ce problème, une interpolation entre les deux points connus les plus proches encadrant le point cherché peut être réalisée pour avoir accès à l'information désirée, V_{interp} dans l'algorithme.

Les flèches bleues représentent les trois commandes appliquées au système à l'état x^i_{N-1} tandis que les carrés bleus quant à eux représentent les trois états après application de ces commandes. Le coût instantané va donc uniquement varier en fonction des commandes étant donné que le nœud d'état considéré est le même dans l'exemple. Les états ne faisant pas parti du maillage de l'état, il faut alors faire une interpolation pour évaluer les *cost to go* des dynamiques à l'instant N. Par exemple, l'interpolation pour la seconde dynamique u^2_{n-1}, sera donc faite en utilisant les valeurs enregistrées dans V pour les mailles $i+1$ et i.

Des fonctions d'interpolation ν sont utilisées pour estimer V des dynamiques à l'instant $k+1$. Différentes fonctions d'interpolation sont possibles mais celle utilisée majoritairement durant la thèse est l'interpolation linéaire. En utilisant l'interpolation linéaire, il est alors considéré que l'évolution spatiale de V est linéaire entre deux points du maillage de l'espace des états. Cette considération est vraie si le maillage est suffisamment fin car localement les non-linéarités peuvent être linéarisées. L'équation (1.26) est un exemple de l'interpolation linéaire appliquée à la seconde commande de la figure 1.3.

$$V(N, f_{N-1}(x_{N-1}, u^2_{N-1})) = V(N, x_i) + \Big(f_{N-1}(x_{N-1}, u^2_{N-1}) - x_i\Big).\frac{V(N, x_{i+1}) - V(N, x_i)}{x_{i+1} - x_i} \tag{1.26}$$

D'autres méthodes d'interpolation sont possibles comme celles du plus proche voisin (*nearest* en anglais) [61], spline [62] ou cubique [63] en fonction des besoins du modèle. Pour finir, l'un des principaux intérêts de la méthode est le temps de calcul pour essayer différentes conditions initiales. En effet, plus le nombre de mailles misent en jeu pour le calcul de *backward* sera important, plus le temps de calcul sera long. Cependant, le temps de calcul du *forward* n'est pas impacté par le maillage. En effet, malgré l'utilisation d'interpolations pour obtenir la où les commandes à appliquer au système en fonction de ces conditions actuelles, ces interpolations ne sont pas excessivement plus longues en fonction du maillage. Il est alors rapide de faire une multitude de simulations avec des conditions initiales différentes.

Maillage adaptatif La notion de maillage adaptatif a été évoquée précédemment. Une explication plus approfondie est alors nécessaire pour une meilleure compréhension de son implémentation dans l'algorithme 1. Plusieurs définitions peuvent être données à ce concept mais pour le manuscrit, la définition suivante est utilisée : un maillage adaptatif est un maillage avec un nombre de points constant dont les valeurs minimales et maximales fluctuent dans le

temps.

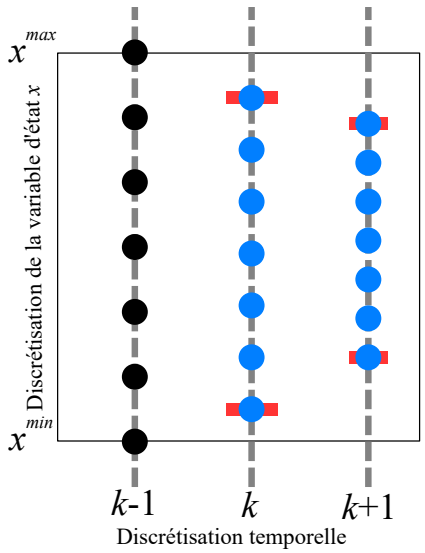

FIGURE 1.4 – Exemple de discrétisation uniforme permanente en noir à l'instant $k-1$ ainsi que deux exemples de maillage uniforme adaptatif aux instants k et $k+1$ en bleu. Les traits rouges représentent les états maximaux et minimaux possibles à chaque pas de temps.

La figure 1.4 illustre pour un problème à un état, l'application du maillage adaptatif. Les points en noir à l'instant $k-1$ représentent un maillage uniforme entre les contraintes minimales et maximales. C'est ce maillage qui est utilisé classiquement dans un algorithme de programmation dynamique. En utilisant l'*IDP* ou les *boundary lines*, il est possible d'avoir accès aux états maximaux et minimaux possibles à chaque pas de temps. Ces informations sont représentées par les traits rouge aux instant k et $k+1$. Il est ainsi possible d'obtenir des maillages adaptatifs, en bleu, entre ces deux bornes pour chaque pas de temps.

1.4.1.2 Amélioration *Iterative Dynamic Programming*

Pour répondre à la problématique de précision de calcul avec un temps de calcul et une utilisation de la mémoire raisonnable, l'idée d'adapter le maillage utilisé à des trajectoires d'états et de commandes déjà définies permet d'optimiser les choix faits autour de la solution de référence. Ceci est d'autant plus vrai si la solution de référence est obtenue par programmation dynamique et que tant que la solution obtenue par la nouvelle programmation dynamique n'est pas identique à la référence, alors un nouveau calcul est effectué. Ainsi, une boucle de convergence est mise en place afin de tendre vers la solution optimale.

Afin de déterminer si deux boucles consécutives sont identiques, le critère de convergence Ψ est alors calculé à partir de l'erreur moyenne sur chaque variable. Si $|\Psi| \leq \epsilon$, avec ϵ un seuil maximal d'erreur à ne pas dépasser, alors le calcul est considéré comme converger vers

la valeur finale.

$$\Psi = \prod_{a=1}^{n} \frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}_a^{i,r} - \boldsymbol{x}_a^i| \times \prod_{b=1}^{m} \frac{1}{N-1} \sum_{i=1}^{N-1} |\boldsymbol{u}_b^{i,r} - \boldsymbol{u}_b^i| \tag{1.27}$$

Nomenclature : σ_x : espace de recherche laissé autour de la trajectoire d'état de référence ; σ_u : espace de recherche laissé autour de la trajectoire de commande de référence.

L'algorithme 2 explicite la mise en place de l'outil qui nécessite l'utilisation d'un maillage adaptatif pour fonctionner. Afin de réaliser un maillage adaptatif uniforme, dans le cas de l'algorithme présenté, deux variables σ_x et σ_u sont utilisées pour déterminer l'espace d'exploration de la variable d'état et la variable de commande autour de la solution de référence dans les cas où au moins une programmation dynamique a déjà été effectuée (*compteur* ≥ 2). Plus ces deux valeurs sont grandes, plus l'espace de recherche est grand et les imprécisions de la programmation dynamique sont importantes.

1.4.1.3 Amélioration des *Boundary lines*

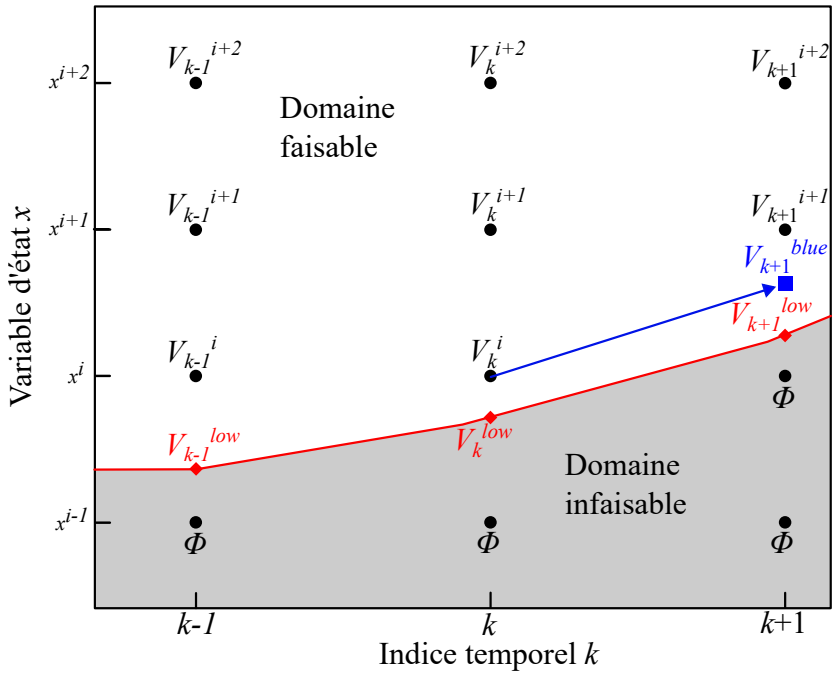

FIGURE 1.5 – Intérêt et application des *Boundary lines* représentés de manière schématique. La ligne rouge représente une *boundary line* (ici la *lower*) qui délimite la frontière entre zone infaisable en gris et faisable en blanc. En bleu est représenté la variation de l'état pour une commande donnée. La variable Φ représente un coût de pénalité.

Interpolation et Contamination La figure 1.5 montre, de manière schématique, l'intérêt des *boundary lines* pour la programmation dynamique. Nous cherchons alors à estimer V_{k+1}^{blue} qui est le *cost to go* de l'état x^{blue} après l'application d'une commande à partir de l'état

Algorithme 2 Algorithme de Programmation Dynamique avec l'outil des *Iterative Dynamic Programming* et le maillage adaptatif pour un système à un état et une commande.

$\boldsymbol{u} = u_{min} : (u_{max} - u_{min})/(q_u - 1) : u_{max}$
$\boldsymbol{x} = x_{min} : (x_{max} - x_{min})/(q_s - 1) : x_{max}$
$compteur = 1$
Tant que $\Psi > \epsilon$ **Faire**
  Si compteur=1 **Alors** ▷ Initialisation des contraintes finales
    $V(N, :) = \Phi_{pen}$ ▷ Pénalité finale
    $V(N, \boldsymbol{x} \in \boldsymbol{X}_N) = 0$
    $\boldsymbol{z} = \boldsymbol{x}$
  Sinon
    $\boldsymbol{x} = x_N - \sigma_x : (2.\delta_x)/(q - 1) : x_N + \sigma_x$ ▷ σ_x imposé arbitrairement
    $V(N, :) = \Phi_{pen}$ ▷ Pénalité finale
    $V(N, \boldsymbol{x} \in \boldsymbol{X}_N) = 0$
  Fin de Si
  Pour $p = N - 1 : -1 : 1$ **Faire** ▷ boucle *Backward*
    Si compteur>=2 **Alors** ▷ Adaptation du maillage à partir de la seconde boucle
      $\boldsymbol{x}_p = x_p - \sigma_x : (2.\delta_x)/(q_s - 1) : x_p + \sigma_x$
      $\boldsymbol{z} = x_{p+1} - \sigma_x : (2.\sigma_x)/(q_s - 1) : x_{p+1} + \sigma_x$
      $\boldsymbol{u} = u_p - \sigma_u : (2.\sigma_u)/(q_u - 1) : u_p + \sigma_u$ ▷ σ_u imposé arbitrairement
    Sinon
      $\boldsymbol{x}_p = \boldsymbol{x}$
    Fin de Si
    Pour $j = 1 : 1 : q$ **Faire**
      Pour $i = 1 : 1 : d$ **Faire**
        $x_{p+1}^{i,j} = f_p(\boldsymbol{x}_p^j, \boldsymbol{u}^i, \boldsymbol{w}_p)$
        $g = L(p, \boldsymbol{x}_p^j, \boldsymbol{u}^i).dt$
        $V_{interp} = \nu(\boldsymbol{z}, V(p + 1, \boldsymbol{z}), x_{p+1}^{i,j})$
        $J^i = g + V_{interp} + \mu$ ▷ μ est une variable qui notifie des uplets infaisables
      Fin de Pour
      $V(p, j) = \min_{\boldsymbol{u}}(J)$
      $U(p, j) = \boldsymbol{u}(\underset{\boldsymbol{u}}{argmin}(J))$
    Fin de Pour
  Fin de Pour
  $Cost = 0$
  Pour $t = 1 : 1 : N - 1$ **Faire** ▷ boucle *Forward*
    $u_t = \nu(\boldsymbol{x}_t, U(p, :), x_t)$
    $x_{t+1} = x_t + f_t(x_t, u_t).dt$
    $Cost = L(t, x_t, u_t) + Cost$
  Fin de Pour
  $compteur = compteur + 1$
  $\Psi = \frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}^{i,r} - \boldsymbol{x}^i| \times \frac{1}{N-1} \sum_{i=1}^{N-1} |\boldsymbol{u}^{i,r} - \boldsymbol{u}^i|$
Fin de Tant que

x^i à l'instant k mais qui n'appartient pas au maillage de $\boldsymbol{x}$. Si on utilise l'équation (1.26) pour l'estimation, nous obtenons (1.28). L'évaluation de cette équation dépend alors de la valeur de la pénalité Φ. Pour que cette valeur soit en accord avec le problème, il faut que Φ soit largement supérieur aux valeurs des intégrales des coûts instantanés sur la période de simulation pour que la pénalité soit efficace. On peut alors penser à la valeur infinie pour la pénalité.

$$V(k+1, x^{blue}) = \Phi + \left(x^{blue} - x^i\right) . \frac{V(k+1, x^{i+1}) - \Phi}{x^{i+1} - x_i} \tag{1.28}$$

Si effectivement on pose $\Phi = +\infty$, (1.28) donne alors le résultat (1.29).

$$V(k+1, x^{blue}) = +\infty \tag{1.29}$$

Ce résultat implique une contamination de la matrice de coût optimal V et donc une sous-optimalité possible du résultat donné par la programmation dynamique. Il est préférable de ne parler que de possibilité ici car la trajectoire peut ne pas être impactée par cette contamination mais la contamination n'assure plus l'optimalité de la solution.

Afin de palier à ce problème, les frontières entre le domaine faisable et infaisable peut être utilisées. En effet, si le *cost to go* est déterminé à chaque pas de temps, il est alors possible de l'utiliser lors des calculs d'interpolations. L'interpolation pour notre exemple devient alors l'équation (1.30). Étant donné que V_{k+1}^{low} est une valeur bien inférieure à Φ, la matrice de coût optimal V n'est plus contaminée par le domaine des infaisabilités apporté par les contraintes finales.

$$V_{k+1}^{blue} = V_{k+1}^{low} + (x_{k+1}^{blue} - x_{k+1}^{low}) . \frac{V_{k+1}^{i+1} - V_{k+1}^{low}}{x_{k+1}^{i+1} - x_{k+1}^{low}} \tag{1.30}$$

Construction des *Boundary Lines* Pour initier la construction des *boundary lines*, deux courbes partent des contraintes finales extrêmes pour délimiter les domaines faisables et infaisables. Il faut alors calculer quels sont les deux nouveaux états extrêmes au pas de temps précédent pour construire ces domaines. Il faut donc être capable d'inverser le modèle comme dans l'équation (1.31) et la construction se fait de manière rétroactive comme le *backward*. Si ce n'est pas directement possible à cause de la non linéarité du modèle, il existe des méthodes qui permettent de linéariser le modèle autour de points de fonctionnement [64] [65] et donc d'inverser le modèle. Ces méthodes sont particulièrement adaptées à la programmation dynamique et aux *boundary lines* étant donné que les points de fonctionnement sont définis par le maillage. L'état au pas de temps précédant x_{k-1} est alors exprimé par l'inverse de dynamique de l'état f_k^{-1} à l'instant k en fonction de la commande à cet état u_{k-1} et de l'état actuel x_k. En plus de ce calcul de dynamique, le calcul du *cost to go* aux points de la frontière est également calculé pour apporter de l'information supplémentaire au moment de l'interpola-

tion (1.30). D'une manière générale, un ensemble de commande est appliqué aux deux états extrêmes représentés par les *boundary lines* mais si la relation entre la variable de commande et la dynamique est monotone, alors seules les commandes extrêmes sont à appliquer.

$$x_{k-1} = f_k^{-1}(x_k, u_{k-1}) \tag{1.31}$$

Algorithme 3 Algorithme de Programmation Dynamique avec l'outil des *boundary lines* pour un système à un état et une commande.

$\boldsymbol{u} = u_{min} : (u_{max} - u_{min})/(q_u - 1) : u_{max}$
$\boldsymbol{x} = x_{min} : (x_{max} - x_{min})/(q_s - 1) : x_{max}$
Pour $k = N-1 : -1 : 1$ **Faire** ▷ boucle *boundary lines*
  Pour $i = 1 : 1 : d$ **Faire**
    $\alpha(i) = f_k^{-1}(\boldsymbol{x}_{k+1}^{up}, \boldsymbol{u}_k^i, \boldsymbol{w}_k)$
    $\beta(i) = f_k^{-1}(\boldsymbol{x}_{k+1}^{low}, \boldsymbol{u}_k^i, \boldsymbol{w}_k)$
  Fin de Pour
  $\boldsymbol{x}_k^{up} = \max(\alpha)$
  $u_k^{up} = \boldsymbol{u}_k(argmax(\alpha))$
  $V_{BL}^{up}(k) = L(k, \boldsymbol{x}_k^{up}, u_k^{up}) + V_{BL}^{up}(k+1)$
  $\boldsymbol{x}_k^{low} = \min(\beta)$
  $u_k^{low} = \boldsymbol{u}_k(argmax(\beta))$
  $V_{BL}^{low}(k) = L(k, \boldsymbol{x}_k^{low}, u_k^{low}) + V_{BL}^{low}(k+1)$
Fin de Pour
Pour $p = N-1 : -1 : 1$ **Faire** ▷ boucle *Backward*
  $\boldsymbol{z} = \underset{\boldsymbol{x}}{arg}(x_{p+1}^{low} \leq \boldsymbol{x} \leq x_{p+1}^{up})$
  Pour $j = \boldsymbol{z}_1 : 1 : \boldsymbol{z}_q$ **Faire** ▷ $\boldsymbol{z}_1$: première valeur de $\boldsymbol{z}$; $\boldsymbol{z}_q$: dernière valeur de $\boldsymbol{z}$
    Pour $i = 1 : 1 : d$ **Faire**
      $\boldsymbol{x}_{p+1}^{i,j} = f_p(\boldsymbol{x}_p^j, \boldsymbol{u}_p^i, \boldsymbol{w}_p)$
      $g = L(p, \boldsymbol{x}_p^j, \boldsymbol{u}_p^i).dt$
      $V_{interp} = \nu(\boldsymbol{x}, V(p+1, \boldsymbol{x}), \boldsymbol{x}_{p+1}^{i,j})$
      $J^i = g + V_{interp} + \mu$ ▷ μ est une variable qui notifie des uplets infaisables
    Fin de Pour
    $V(p,j) = \underset{\boldsymbol{u}}{min}(J)$
    $U(p,j) = \boldsymbol{u}(\underset{\boldsymbol{u}}{argmin}(J))$
  Fin de Pour
Fin de Pour
$Cost = 0$
Pour $t = 1 : 1 : N-1$ **Faire** ▷ boucle *Forward*
  $\boldsymbol{z} = \boldsymbol{x}(\underset{\boldsymbol{x}}{arg}(x_t^{low} \leq \boldsymbol{x} \leq x_t^{up}))$
  $u_t = \nu([x_t^{low}; \boldsymbol{z}; x_t^{up}], [u_t^{low}; U(\boldsymbol{z}); u_t^{up}], x_t)$
  $x_{t+1} = x_t + f_t(x_t, u_t).dt$
  $Cost = L(t, x_t, u_t) + Cost$
Fin de Pour

L'algorithme 3 décrit une solution d'intégration de l'outil à la programmation dynamique. Une nouvelle boucle temporelle est réalisée en amont du *backward* de la programmation dynamique afin de générer les *boundary lines*. La frontière supérieure entre les domaines faisable et infaisable est couramment appelée *upper boundary line*, et tout élément dans l'algorithme avec l'indice où l'exposant *up* se réfère à cette *boundary line*. La frontière inférieure est quant à elle dénommée *lower boundary line* et les indices ou exposants *low* la désignent. Toutes les commandes sont appliquées à partir des deux états frontières $\boldsymbol{x}_{k+1}^{up/low}$ à l'instant $k+1$ pour le modèle inversé pour déterminer quelle commande maximise la dynamique souhaitée (croissance pour la *upper boundary line* et décroissance pour la *lower boundary line*) ainsi que les coûts associés à ces deux commandes. Il est possible que cette maximisation de dynamique ne soit plus souhaitée car les *boundary lines* ne respectent plus les contraintes imposées au système. Si tel est le cas, la dynamique la plus faible est alors cherchée. Pour réutiliser les informations calculées dans la boucle *boundary lines* dans le *backward*, les données utilisées pour l'interpolation V_{interp} changent par rapport à l'algorithme 1. En effet, seulement la partie du vecteur d'état comprise entre les deux valeurs des *boundary lines* $\boldsymbol{z}$ et ces dernières sont utilisées pour les interpolations. Les valeurs des matrices de *cost to go* ainsi que de commandes optimales sont alors uniquement calculées pour les états compris dans le domaine faisable. Finalement dans le *forward*, lorsque la commande à appliquer au pas de temps t est calculée, l'information des commandes des *boundary lines* est également utilisée de la même manière que le *cost to go* l'est dans le *backward*.

L'algorithme 4 propose alors l'intégration du maillage adaptatif dans l'algorithme 3 de programmation dynamique avec l'outil *boundary lines*. Il n'est alors plus nécessaire de stocker l'information de l'état sur les *boundary* ainsi que les coûts associés dans des vecteurs spécifiques. En effet, il est possible d'enregistrer ces informations directement dans les matrices de coût optimal et de commande optimale étant donné que les deux valeurs frontières font parties intégrantes du maillage.

Algorithme 4 Algorithme de Programmation Dynamique avec l'outil des *boundary lines* et le maillage adaptatif pour un système à un état et une commande.

$\boldsymbol{u} = u_{min} : (u_{max} - u_{min})/(q_u - 1) : u_{max}$
$\boldsymbol{x} = x_{min} : (x_{max} - x_{min})/(q_s - 1) : x_{max}$
Pour $k = N-1 : -1 : 1$ **Faire** ▷ boucle *boundary lines*
  Pour $i = 1 : 1 : d$ **Faire**
    $\alpha(i) = f_k^{-1}(\boldsymbol{x}_{k+1}^{up}, \boldsymbol{u}_k^i, \boldsymbol{w}_k)$
    $\beta(i) = f_k^{-1}(\boldsymbol{x}_{k+1}^{low}, \boldsymbol{u}_k^i, \boldsymbol{w}_k)$
  Fin de Pour
  $\boldsymbol{x}_k^{up} = \max(\alpha)$
  $U(k,q) = \boldsymbol{u}_k(argmax(\alpha))$
  $V(k,q) = L(k, \boldsymbol{x}_k^{up}, u_k^{up}) + V_{BL}^{up}(k+1)$
  $\boldsymbol{x}_k^{low} = \min(\beta)$
  $U(k,1) = \boldsymbol{u}_k(argmax(\beta))$
  $V(k,1) = L(k, \boldsymbol{x}_k^{low}, u_k^{low}) + V_{BL}^{low}(k+1)$
Fin de Pour
Pour $p = N-1 : -1 : 1$ **Faire** ▷ boucle *Backward*
  $\boldsymbol{x}_p = \boldsymbol{x}_p^{low} : (\boldsymbol{x}_p^{up} - \boldsymbol{x}_p^{low})/(q_s - 1) : \boldsymbol{x}_p^{up}$
  Pour $j = 2 : 1 : q-1$ **Faire**
    Pour $i = 1 : 1 : d$ **Faire**
      $\boldsymbol{x}_{p+1}^{i,j} = \nu(\boldsymbol{x}_p^j, \boldsymbol{u}_p^i, \boldsymbol{w}_p)$
      $g = L(p, \boldsymbol{x}_p^j, \boldsymbol{u}_p^i).dt$
      $V_{interp} = f_{interp}(\boldsymbol{x}, V(p+1, \boldsymbol{x}), \boldsymbol{x}_{p+1}^{i,j})$
      $J^i = g + J_{interp} + \mu$ ▷ μ est une variable qui notifie des uplets infaisables
    Fin de Pour
    $V(p,j) = \min_{\boldsymbol{u}}(J)$
    $U(p,j) = \boldsymbol{u}(\underset{\boldsymbol{u}}{argmin}(J))$
  Fin de Pour
Fin de Pour
$Cost = 0$
Pour $t = 1 : 1 : N-1$ **Faire** ▷ boucle *Forward*
  $\boldsymbol{z} = \boldsymbol{x}_p^{low} : (\boldsymbol{x}_p^{up} - \boldsymbol{x}_p^{low})/(q-1) : \boldsymbol{x}_p^{up}$
  $u_t = \nu(\boldsymbol{z}, U(p,:), x_t)$
  $x_{t+1} = x_t + f_t(x_t, u_t).dt$
  $Cost = L(t, x_t, u_t) + Cost$
Fin de Pour

1.4.2 Application à l'exemple

Le modèle présenté en section 1.3 en considérant le problème posé durant l'introduction, dont les contraintes sont exprimées en (1.19) et le critère en (1.20), a été résolu en utilisant les différents outils de la programmation dynamique pour constater les différences. La table 1.2 présente les résultats obtenus avec différents maillages pour les quatre variantes de la programmation dynamique utilisées. Les trois variantes utilisant des outils sont : l'*IDP*, *DP*

BL pour l'utilisation des *boundary lines* et *DP BLMA* pour l'utilisation des *boundary lines* avec un maillage adaptatif. Ces outils affichent de bien meilleurs résultats que la programmation dynamique originale, notamment pour les maillages grossiers.

Pour information, les calculs ont été réalisés sur la version 2021b de Matlab avec l'ordinateur suivant : Intel(R) Core(TM) i7-6850K avec une fréquence de fonctionnement à 3.6 GHz et une mémoire vive de 128 Go de RAM. Pour les calculs de programmation dynamique qui seront présentés dans la suite du manuscrit, ce sont des programmes propres à la thèse, à l'exception de ceux présentés au sein du chapitre 2. Ces programmes développés en interne ont été développés en utilisant la parallélisation des calculs par une mise en forme des variables sous forme de tenseurs permise par Matlab. En effet, cette méthode de calcul est plus rapide qu'une cascade de boucle *for* mais requiert une mémoire vive plus importante pour le même résultat.

TABLE 1.2 – Table de résultats des quatre variantes de la programmation dynamiques étudiées pour différents maillages.

Méthode	Maillage ($q_u = q_s$)	SOC final (-)	Résultat (MJ)	temps de calcul (s)	*compteur* pour *IDP*	RAM (Mo)
DP	20	0,6945	7,96	0,19	-	0,87
	200	0,5763	3,65	1,90	-	9,56
	1000	0,5032	2,64	47,74	-	117,14
	2000	0,5009	2,61	181,42	-	410,02
IDP	20	0,5122	2,77	1,13	7	1,19
	200	0,5030	2,67	12,00	6	12,47
	1000	0,5017	2,62	94,45	2	131,60
	2000	0,5005	2,61	357,24	2	438,91
DP BL	20	0,5010	2,94	0,42	-	1,24
	200	0,5009	2,63	1,43	-	11,97
	1000	0,5000	2,60	31,89	-	117,96
	2000	0,5000	2,60	125,08	-	384,11
DP BLMA	20	0,5000	2,72	0,38	-	1,48
	200	0,5000	2,60	2,29	-	15,36
	1000	0,5000	2,60	48,58	-	146,03
	2000	0,5000	2,60	182,29	-	467,78

La figure 1.6 présente les trajectoires de la commande ainsi que de l'état des solutions trouvées par les quatre variantes de la programmation dynamique pour le maillage le plus fin utilisé. Étant donné que les solutions sont très similaires, la figure 1.7 illustre la différence des trois améliorations par rapport à la référence qu'est la programmation dynamique originale.Les solutions présentées sont celles obtenues pour les maillages à 2000 mailles par variable. La différence dans la trajectoire de commande est proche de zéro, ce qui explique des différences dans l'état de charge de la batterie également très faible.

Les figures 1.8 à 1.11 présentent les quatre matrices de coût optimal V de chacune des variantes de la programmation dynamique étudiée pour les maillages uniformes homogènes

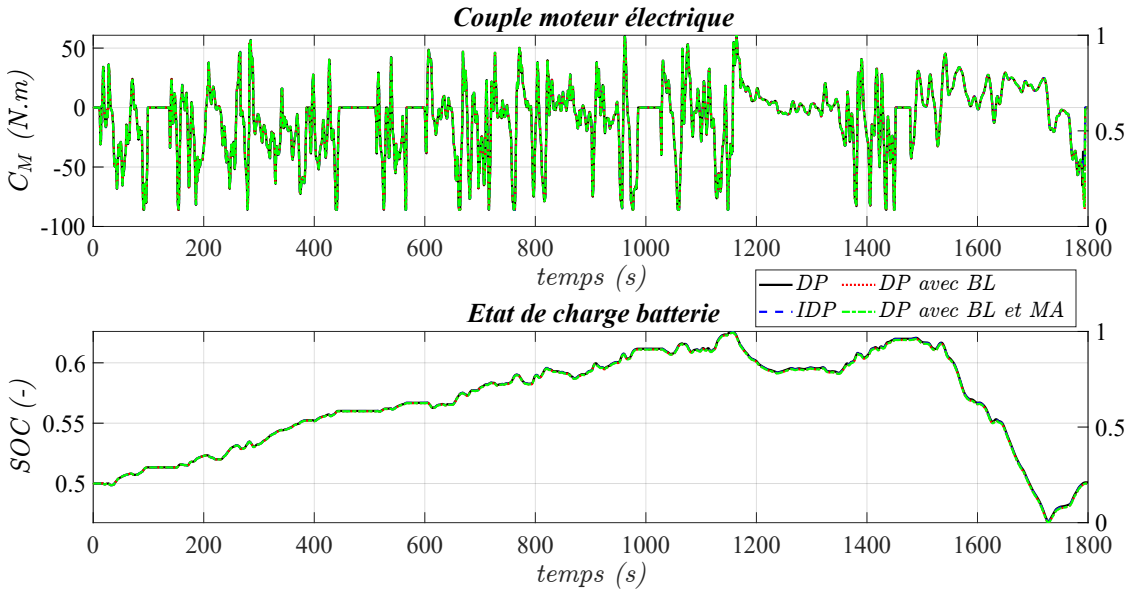

FIGURE 1.6 – Trajectoires des solutions pour les différentes variantes de la programmation dynamique avec des maillages homogènes uniformes de 2000 mailles.

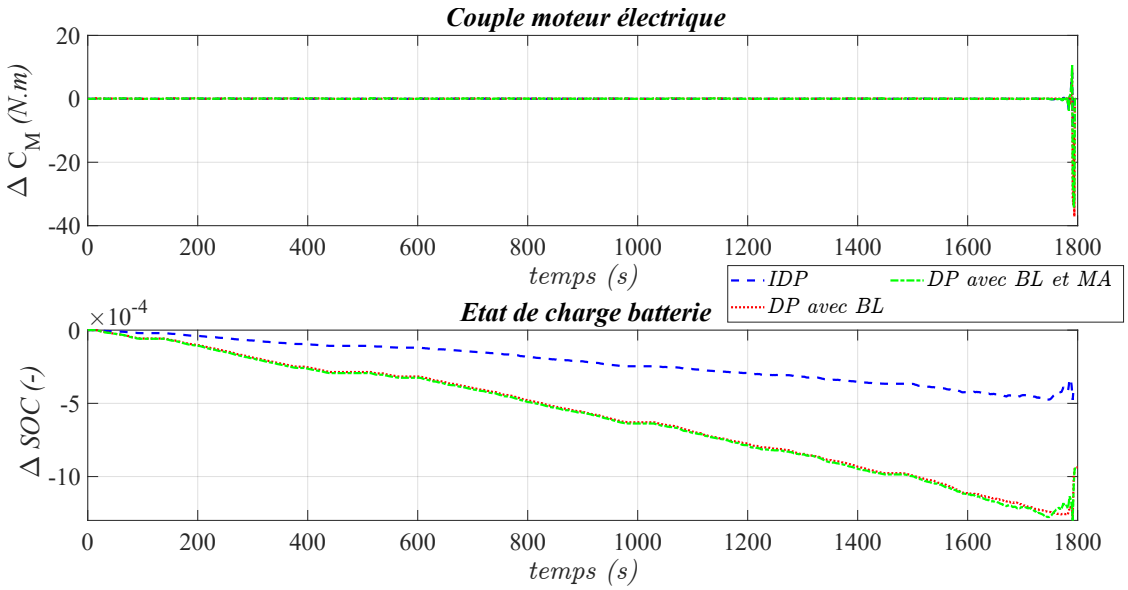

FIGURE 1.7 – Différence des trois améliorations (*IDP*,*DP BL* et *DP BLMA*) par rapport à la solution de référence (*DP*) avec des maillages homogènes uniformes de 2000 mailles.

de 2000 mailles par variable. La programmation dynamique originale est très dépendante de son maillage pour résoudre le problème de manière optimale, comme le montre la table 1.2. Une conséquence de ce phénomène se retrouve dans la zone à l'interface entre la zone faisable et infaisable au niveau de la *lower boundary line* sur la figure 1.8. Étant donné que les *boundaries* ne sont pas utilisées, cette zone d'interface fait tampon en limitant le phénomène

de contamination. Cependant, plus le maillage est fin, plus il est efficace pour conserver la matrice de coût optimal, ce qui explique les résultats obtenus pour la variante originale de la programmation dynamique. L'*Iterative Dynamic Programming* de son côté évolue dans une zone d'exploration plus restreinte à chaque itération, ce que explique les résultats obtenus qui évoluent uniquement dans un intervalle de $0,1$ de SOC. Les deux variantes utilisant les *boundary lines* sont très proches l'une de l'autre en terme de résultats, notamment pour des maillages fins. Cependant, le temps de calcul de la variante avec le maillage adaptatif est environ 50% supérieur à la version sans.

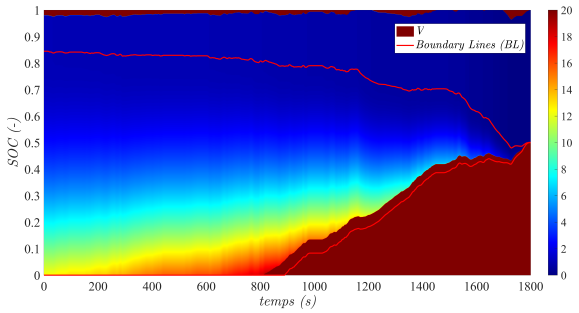

FIGURE 1.8 – Matrice de coût optimal *Dynamic Programming*.

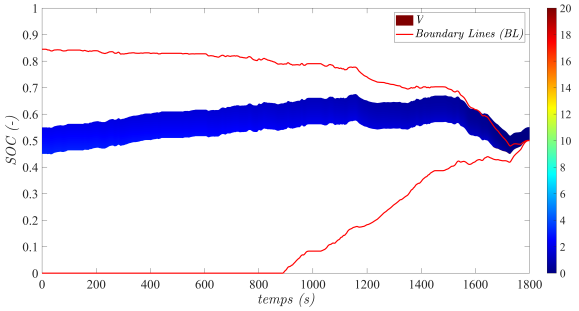

FIGURE 1.9 – Matrice de coût optimal *Iterative Dynamic Programming* à la dernière itération.

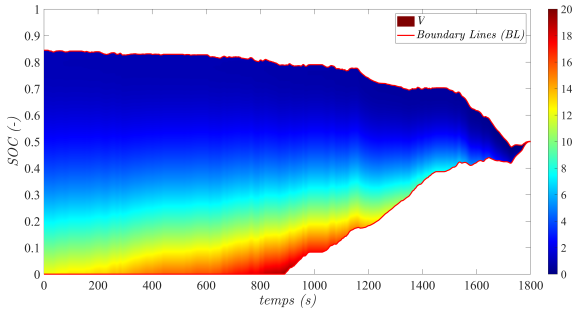

FIGURE 1.10 – Matrice de coût optimal *DP* avec *boundary lines*.

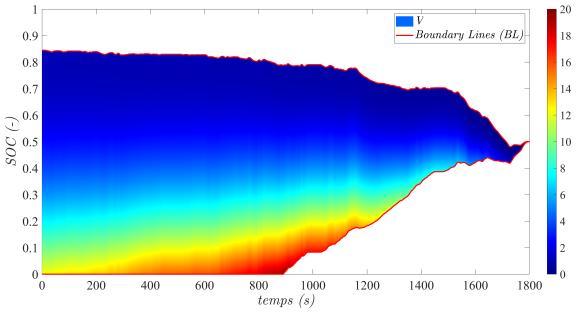

FIGURE 1.11 – Matrice de coût optimal *DP* avec *boundary lines* et maillage adaptatif.

1.5 Principe de Maximum de Pontriaguine

Cette section a pour objectif de montrer le lien entre la programmation dynamique présentée précédemment et le Principe de Maximum de Pontriaguine. De plus, l'application de la méthode au problème exemple du chapitre 1.3 permet de comparer le PMP à la méthode précédente.

1.5.1 Problème aux deux bouts

Le Principe de Maximum de Pontriaguine (PMP), qui est un résultat fondamental de la théorie du contrôle optimal [1], donne une condition nécessaire d'optimalité [66]. La résolution de ce principe se fait dans le domaine temporel en résolvant le problème (1.3) sous sa forme duale [67]. L'écriture du problème sous forme duale passe par l'écriture du Lagrangien $\bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))$ de la fonction coût $J(t, \boldsymbol{x}(t), \boldsymbol{u}(t))$ du problème étudié. L'écriture du Lagrangien permet alors de passer d'un problème avec contrainte, comme la forme primale, à un problème sans contrainte qui est appelé forme duale.

$$\bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = J(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) + \int_{t_0}^{t_f} \boldsymbol{\lambda}^T.(f(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) - \dot{\boldsymbol{x}}(t))dt \tag{1.32}$$

Le terme $\boldsymbol{\lambda}(t)$ désigne ici le vecteur des paramètres de Lagrange associé au vecteur d'état $\boldsymbol{x}(t)$. Pour faire apparaître le problème dual, il est d'abord nécessaire de faire apparaître l'opérateur Hamiltonien $H(.)$ dans l'équation (1.32). L'opérateur $H(.)$ utilisé est défini par l'équation (1.33) et permet d'exprimer le Lagrangien avec seulement une seule intégrale. Cet opérateur est fonction des états, des commandes, du vecteur des coefficients de Lagrange et du temps.

$$H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = L(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{\lambda}^T(t).f(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \tag{1.33}$$

C'est pourquoi nous injectons l'expression du critère de minimisation $J(.)$ du problème primal continu (1.3) dans l'expression du Lagrangien.

$$\bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = \int_{t_0}^{t_f} H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)).dt - \int_{t_0}^{t_f} \boldsymbol{\lambda}^T(t).\dot{\boldsymbol{x}}(t).dt + \Phi(\boldsymbol{x}(t_f)) \tag{1.34}$$

Alors, il est possible d'écrire (1.35) en intégrant par partie la seconde intégrale de (1.34) :

$$\begin{aligned}\bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = &\int_{t_0}^{t_f} (H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) + \dot{\boldsymbol{\lambda}}(t).\boldsymbol{x}(t)).dt \\ &- (\boldsymbol{\lambda}^T(t_f).\boldsymbol{x}(t_f) - \boldsymbol{\lambda}^T(t_0).\boldsymbol{x}(t_0)) + \Phi(\boldsymbol{x}(t_f))\end{aligned} \tag{1.35}$$

Pour trouver le minimum du Lagrangien $\bar{J}(.)$, il faut qu'une petite variation selon toutes les directions de ce dernier soit nulle. Cela implique donc l'équation (1.36). Il est toutefois important de noter ici que cette condition d'optimalité implique que le problème soit convexe.

$$\delta\bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = 0 \tag{1.36}$$

Afin de calculer ces petites variations, le Lagrangien est soumis à de petites variations des

commandes $\delta \boldsymbol{u}(t)$ et des états $\delta \boldsymbol{x}(t)$. Ces variations appliquées à (1.35) donnent :

$$\delta \bar{J}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = \boldsymbol{\lambda}^T(t_0).\delta \boldsymbol{x}(t_0) + \left(\frac{\partial \Phi(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}} - \boldsymbol{\lambda}^T(t_f) \right).\delta \boldsymbol{x}(t_f)$$

$$+ \int_{t_0}^{t_f} \left(\left(\frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{x}} + \dot{\boldsymbol{\lambda}}^T(t) \right) \delta \boldsymbol{x}(t) + \left(\frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} \right) \delta \boldsymbol{u}(t) \right) .dt \tag{1.37}$$

Afin que $\delta \bar{J}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t), t)$ soit nul, il est nécessaire que :

$$\dot{\boldsymbol{\lambda}}^T(t) = -\frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{x}} \tag{1.38}$$

$$\frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} = 0 \tag{1.39}$$

$$\boldsymbol{\lambda}^T(t_f) = -\frac{\partial \Phi(\boldsymbol{x}(t_f)}{\partial \boldsymbol{x}} \tag{1.40}$$

Étant donné que les variations sont petites, il peut alors être démontré que $\boldsymbol{\lambda}^T(t_0).\delta \boldsymbol{x}(t_0)$ est nul car $\delta \boldsymbol{x}(t_0) \approx 0$. Il devient alors possible d'écrire le problème dual suivant :

$$\begin{cases} \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \\ \dot{\boldsymbol{\lambda}}^T(t) = -\frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{x}} \\ \frac{\partial H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))}{\partial \boldsymbol{u}} = 0 \\ \boldsymbol{\lambda}^T(t_f) = -\frac{\partial \Phi(\boldsymbol{x}(t_f)}{\partial \boldsymbol{x}} \end{cases} \tag{1.41}$$

Le PMP correspond à l'équation (1.42), qui permet de dire que si $\boldsymbol{u}^*(t) \in U$ est une commande optimale, alors :

$$H(t, \boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)) \leq H(t, \boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t)) \tag{1.42}$$

avec respectivement $\boldsymbol{x}^*(t)$ et $\boldsymbol{\lambda}^*(t)$, le vecteur d'état optimal et le vecteur de paramètres de Lagrange optimal. Afin de déterminer les valeurs initiales du vecteur de coefficients de Lagrange $\boldsymbol{\lambda}^*(t_0)$, une méthode de résolution pour méthodes indirectes doit être utilisée. On peut citer la méthode par tir simple, méthode par tir multiple ou la méthode de Newton par exemple [1].

1.5.2 Lien à la Programmation Dynamique

Il est possible de faire un lien entre le PMP et la *DP*. Pour ce faire, nous repartons de l'expression (1.43) sur lequel se repose le principe de Bellman [59].

$$V(t, \boldsymbol{x}_t) = \min_{\boldsymbol{u}([t;t_f])} (\int_t^{t_f} L(\tau, \boldsymbol{x}_\tau, \boldsymbol{u}_\tau).d\tau + V(t_f, \boldsymbol{x}_{t_f})) \tag{1.43}$$

Le principe de Bellman énonce alors :

$$V(t, \boldsymbol{x}_t) = \min_{\boldsymbol{u}([t;t+\Delta t])} (\int_t^{t+\Delta t} L(\tau, \boldsymbol{x}_\tau, \boldsymbol{u}_\tau).d\tau + V(t+\Delta t, \boldsymbol{x}_{t+\Delta t})) \tag{1.44}$$

Il est possible d'approximer l'intégrale du coût instantané par :

$$\int_t^{t+\Delta t} L(\tau, \boldsymbol{x}_\tau, \boldsymbol{u}_\tau).d\tau \approx L(t, \boldsymbol{x}_t, \boldsymbol{u}_t).\Delta t \tag{1.45}$$

Ainsi, le coût optimal $V(t+\Delta t, \boldsymbol{x}_{t+\Delta t})$ est obtenu grâce à un développement de Taylor à l'ordre 1 :

$$V(t+\Delta t, \boldsymbol{x}_{t+\Delta t}) \approx V(t, \boldsymbol{x}_t) + \frac{\partial V}{\partial t}.\Delta t + \frac{\partial V}{\partial \boldsymbol{x}}.\Delta \boldsymbol{x} \tag{1.46}$$

En réinjectant (1.45) et (1.46) dans (1.44), il est possible d'écrire :

$$V(t, \boldsymbol{x}_t) \approx \min_{\boldsymbol{u}([t;t+\Delta t])} (L(t, \boldsymbol{x}_t, \boldsymbol{u}_t).\Delta t + V(t, \boldsymbol{x}_t) + \frac{\partial V}{\partial t}.\Delta t + \frac{\partial V}{\partial \boldsymbol{x}}.f(t, \boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_t).\Delta t) \tag{1.47}$$

En simplifiant (1.47), il est finalement possible d'écrire l'équation Hamilton-Jacobi-Bellman :

$$-\frac{\partial V}{\partial t} = \min_{\boldsymbol{u}([t;t_f])} (L(t, \boldsymbol{x}_t, \boldsymbol{u}_t) + \frac{\partial V}{\partial \boldsymbol{x}}.f(t, \boldsymbol{x}_t, \boldsymbol{u}_t), \boldsymbol{w}_t) \tag{1.48}$$

Il est possible de faire le lien entre programmation dynamique et PMP en définissant le vecteur de coefficient de Lagrange par l'équation (1.49). En effet, on s'aperçoit qu'il est possible d'intégrer cette nouvelle définition dans l'équation (1.48) qui nous donne après simplification l'équation (1.50).

$$\boldsymbol{\lambda}(t) = \frac{\partial V(t, \boldsymbol{x}_t)}{\partial \boldsymbol{x}} \tag{1.49}$$

$$\frac{\partial V}{\partial t} = \min_{\boldsymbol{u}([t;t_f])} (H(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t))) \tag{1.50}$$

Les équations (1.49) et (1.50) nous permettent alors de faire directement le lien entre la programmation dynamique et le PMP car la première équation exprime que la valeur des coefficients de Lagrange peut être directement obtenue par la variation de la matrice de coût

optimal V en fonction des états. La seconde équation décrit que la variation temporelle de la matrice de coût optimal déterminée par la DP est égale à l'opérateur Hamiltonien minimal qui est déterminé par le PMP.

1.5.3 Application du PMP à l'exemple

L'opérateur Hamiltonien défini en (1.33) appliqué au problème de commande optimale (1.3) étudié s'écrit alors :

$$H(t, SOC_{Bat}, C_{Mel}, \lambda_S) = p.(P_F^2 + P_{Bat}^2) + \lambda_S.f(t, SOC_{Bat}, C_{Mel}) \tag{1.51}$$

Le coefficient de Lagrange λ_S est une constante car :

$$\frac{\partial H(t, SOC_{Bat}, C_{Mel}, \lambda_S)}{\partial SOC_{Bat}} = 0 \tag{1.52}$$

Ainsi, le développement de l'opérateur H en remplaçant les différentes variables par leurs expressions permet d'obtenir la relation suivante :

$$\begin{aligned} H(t, SOC_{Bat}, C_{Mel}, \lambda_S) =& \frac{R^2}{r^4}.C_{Mel}^4 + \frac{2.R.\omega_{Mel}}{r^2}.C_{Mel}^3 - (2.\omega_{Mel}\frac{P_W}{\eta_{Eng}^2} + \frac{\lambda_S.\omega_{Mel}}{A_{max}.V_{Bat}}).C_{Mel} \\ &+ (\omega_{Mel}^2.(1 + \frac{1}{\eta_{Eng}^2}) - \frac{\lambda_S.R}{r.A_{max}.V_{Bat}}).C_{Mel}^2 + (\frac{P_W}{\eta_{Eng}})^2 \end{aligned} \tag{1.53}$$

Pour obtenir la commande optimale, la dérivée de H par rapport à cette dernière, C_{Mel}^*, doit être nulle :

$$\begin{aligned} \frac{\partial H(t, SOC_{Bat}, C_{Mel}^*, \lambda_S)}{\partial C_{Mel}^*} =& 4.\frac{R^2}{r^4}.C_{Mel}^{*^3} + \frac{6.R.\omega_{Mel}}{r^2}.C_{Mel}^{*^2} - (2.\omega_{Mel}\frac{P_W}{\eta_{Eng}^2} + \frac{\lambda_S.\omega_{Mel}}{A_{max}.V_{Bat}}) \\ &+ 2.(\omega_{Mel}^2.(1 + \frac{1}{\eta_{Eng}^2}) - \frac{\lambda_S.R}{r.A_{max}.V_{Bat}}).C_{Mel}^* = 0 \end{aligned} \tag{1.54}$$

Pour résoudre le problème (1.19) et (1.20) appliqué au modèle présenté en section 1.3, un schéma numérique d'Euler explicite est mis en place. En effet, la puissance P_W qui est requise par les roues pour avancer à la vitesse imposée varie dans le temps. De la même manière, la vitesse de rotation de la machine électrique ω_{Mel} varie également dans le temps. C'est pourquoi le PMP n'est pas résolu de manière analytique mais numériquement en se basant sur l'équation (1.54) pour déterminer la valeur de la commande à appliquer au modèle. Finalement, dans l'objectif d'atteindre l'objectif de SOC_{Bat} final à $0,5$, un algorithme de multi-tir [68] est utilisé. Les trajectoires obtenues par le PMP sont représentées en figure 1.12 avec la solution de la programmation dynamique originale obtenue précédemment pour comparer les deux

méthodes.

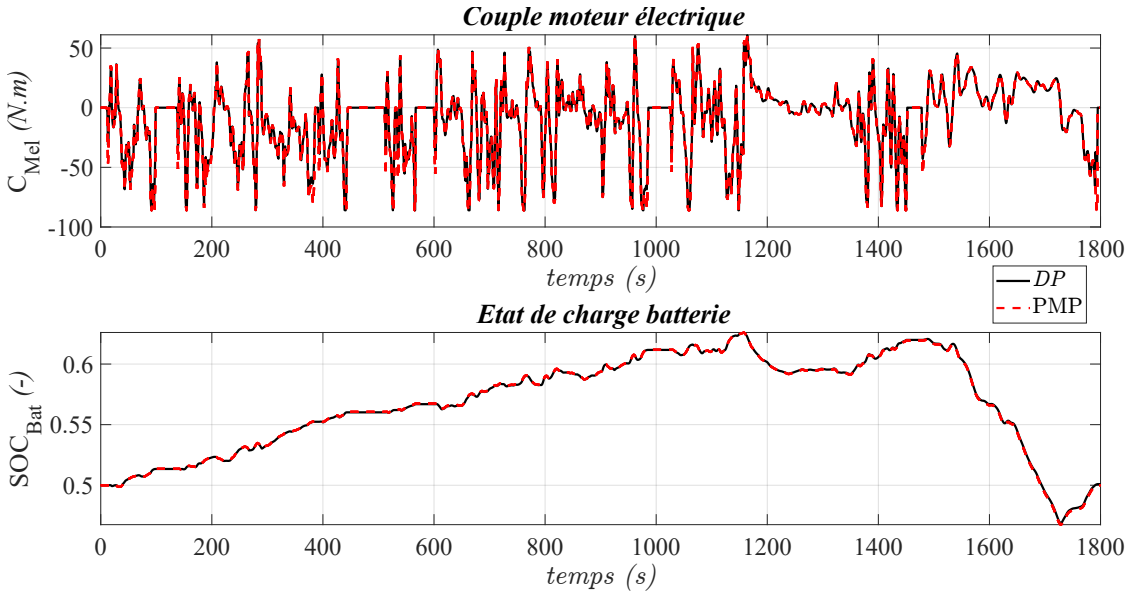

FIGURE 1.12 – Trajectoires de la solution déterminée par PMP et par Programmation Dynamique comme comparaison.

La comparaison des résultats entre la solution du PMP et de la programmation dynamique, une classique et l'autre avec les *boundary lines*, est présentée dans la table 1.3. Ce choix de comparaison s'explique Le PMP apparaît comme plus intéressante que la programmation dynamique à la vue du résultat affiché, du temps de calcul et de la RAM utilisée par le programme.

TABLE 1.3 – Table de résultats des quatre variantes de la programmation dynamiques étudiées pour différents maillages.

Méthode	SOC final (-)	Résultat (MJ)	temps calc. (s)	RAM (Mo)
PMP	0,5000	2,60	1,21	0,84
DP	0,5009	2,61	181,42	410,02
DP BL	0,5000	2,60	125,08	384,11

En appliquant l'équation d'Hamilton-Jacobi-Bellman (1.50) à la matrice de coût optimal de la programmation dynamique utilisée pour la comparaison, il est possible de montrer, en figure 1.13, le lien entre la programmation dynamique et le PMP. La valeur de λ_S déterminée par l'algorithme de *multi-shot* et celle obtenue par l'estimation de l'équation (1.50) appliquée à la programmation dynamique originale et avec l'amélioration *boundary lines* sont très similaires comme attendu. Il peut cependant être noté que l'estimation du coefficient de Lagrange par la programmation dynamique avec les *boundary lines* est plus proche du PMP que la programmation dynamique originale, ce qui montre son intérêt.

A la fin de cette section, le PMP semble être plus intéressant que la programmation

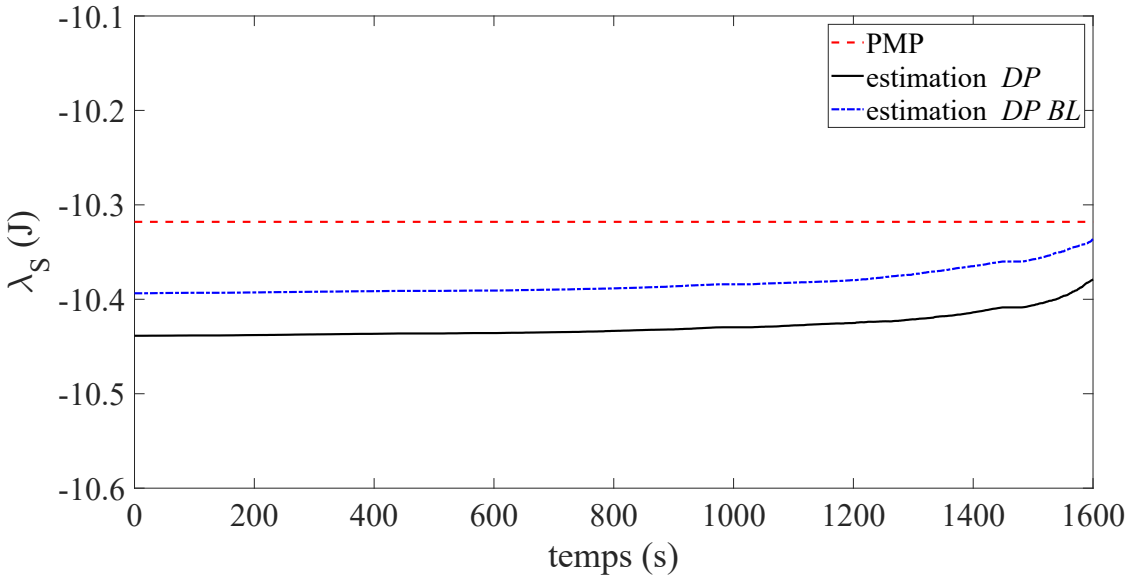

FIGURE 1.13 – Lien entre le PMP et la programmation dynamique via la valeur de λ_S.

dynamique comme méthodologie de résolution de problème de contrôle optimal. Cependant, afin de faire une comparaison entre les deux méthodes, le critère d'optimisation a été modifié pour appliquer le PMP au problème. Au lieu d'utiliser la variable ρ pour homogénéiser le coût instantané à une puissance, ici une puissance est réellement calculée en utilisant une racine carrée (1.55).

$$J(t_0, \boldsymbol{x}, \boldsymbol{u}) = \int_{t_0}^{t_f} \sqrt{(P_F^2(t) + P_B^2(t))}.dt + \Phi(\boldsymbol{x}(t_f)) \tag{1.55}$$

Le problème pour le PMP ici, réside dans l'écriture de l'opérateur Hamiltonien, de l'écriture des coefficients de Lagrange $\boldsymbol{\lambda}$ et dans la détermination de la commande optimale. De plus, le modèle utilisé dans ce chapitre n'est pas modélisé par des cartographies, qui rendent l'application du PMP moins précise. En effet, l'expression de ces cartographies sous forme de fonction au plus convexe peuvent induire des erreurs dans la modélisation comme dit précédemment. Étant donné quand dans le domaine automobile, plusieurs systèmes sont modélisés par des cartographies (moteurs, batteries, etc...), la thèse se concentrera exclusivement à la programmation dynamique à partir du prochain chapitre.

1.6 Conclusion

A travers ce chapitre, la formulation du problème de contrôle optimal a été introduite au même titre que la définition des termes importants du domaine qui seront réutilisés dans le reste de la thèse. Les méthodes de résolution des problèmes de contrôle optimal usuelles ont été présentées avec leurs avantages et inconvénients pour expliquer les choix réalisés au cours de cette thèse.

Étant donné que la méthode retenue est la programmation dynamique, l'algorithme original ainsi que ses outils principaux ont été développés plus longuement. La proximité entre le principe de maximum de Pontriaguine et la programmation dynamique est explicitée. L'application à un cas d'étude simplifié qui sera complexifié dans les chapitres 3 et 4 est réalisé pour montrer les performances de la programmation dynamique, ainsi que de ses améliorations principales. Les résultats montrent que l'amélioration *boundary lines* est très intéressant. C'est pourquoi cette amélioration sera utilisé pour résoudre un problème de loi de gestion de l'énergie pour un véhicule hybride série dans le chapitre 2. Cependant, la limitation d'application à des modèles à un seul état contraint les cas d'applications pouvant utiliser les *boundary lines*. C'est pourquoi, la généralisation de cet outil pour des modèles à deux états est fait dans le chapitre 3.

Chapitre 2

Gestion de l'énergie d'un véhicule hybride avec systèmes récupérateurs

Sommaire

L'accès aux énergies étant en pleine transition des énergies fossiles vers l'électrique [69], le domaine du transport doit s'adapter à ce changement. L'électrification totale ou partielle des véhicules est une réponse à ce évolution et également un sujet d'intérêt pour les constructeurs du domaine automobile [70] [59] [50] [71]. C'est pourquoi les véhicules hybrides ont fait leur apparition depuis plus de trente ans maintenant et que les véhicules électriques deviennent également de plus en plus présents dans le parc automobile mondial. Cependant, une problématique propre au véhicule électrique est son autonomie. En effet, les utilisateurs de ces véhicules peuvent expérimenter le phénomène de *range anxiety* [72] [73] qui exprime la peur d'arriver à cours d'énergie pendant le trajet. Cette peur s'explique par la rareté des stations de recharge pour ces véhicules mais également par le temps de recharge qui est plus long que celui des véhicules conventionnels. De plus, la baisse plus importante que prévue de l'autonomie du

véhicule à partir de certaines vitesses [74] ainsi que des conditions de température accentue ce phénomène. L'intérêt des véhicules hybrides électriques est d'aider à la transition énergétique du parc automobile dans le but d'adapter les véhicules au mix énergétique du futur. Dans l'objectif d'améliorer les rendements des chaînes de traction hybride, une des solutions est de revaloriser les pertes du moteur thermique. C'est pourquoi, l'objectif premier du chapitre est d'étudier l'intérêt de l'ajout de machines récupératrices de puissance dans des véhicules hybrides par rapport aux technologies actuelles. Le second objectif est de modéliser le système de climatisation sous la forme d'une machine thermodynamique dont la consommation électrique du compresseur est intégrée à la dépense énergétique du véhicule.

2.1 Hybridation électrique et machines thermodynamique

2.1.1 Architectures hybrides électrique

Cette sous-section permet au lecteur moins aguerri de se familiariser avec le concept du véhicule hybride électrique (*Hybrid Electric Vehicle* soit *HEV* en anglais) et de voir quelles sont les architectures possibles ainsi que leurs avantages et inconvénients. Un véhicule est défini comme hybride s'il a au minimum deux sources d'énergie différentes à sa disposition, dont au moins une réversible, qui peuvent être sollicitées pour tracter le véhicule. L'hybridation électrique est la plus populaire, mais il est possible d'avoir d'autres technologies comme l'hybride hydraulique, pneumatique, etc. [58] [75].

La figure 2.1 présente les trois architectures principales de véhicule hybride électrique : série, parallèle et série/parallèle. La première architecture présentée est celle de l'hybride série, au sein de laquelle le moteur thermique est totalement découplé des roues du véhicule. C'est la machine électrique qui propulse ou tracte le véhicule. Le moteur thermique est utilisé pour assurer une certaine autonomie en fonctionnant comme un groupe électrogène dans le véhicule. En effet, la puissance mécanique générée par le moteur thermique est convertie en puissance électrique par le biais de la génératrice et injectée dans la batterie. L'intérêt de cette architecture est d'avoir un fonctionnement de véhicule le plus proche possible d'un véhicule électrique mais avec une autonomie augmentée avec la présence du moteur thermique. La seconde architecture représente l'hybride parallèle. Le moteur thermique et la machine électrique sont tous les deux reliés aux roues par l'intermédiaire de la transmission. La machine électrique peut être placée à différentes positions sur la chaîne de traction [76] [71] ; L'architecture parallèle a pour but de déplacer le point de fonctionnement du moteur thermique sur le point de meilleur rendement possible compte tenu de la transmission ou encore de l'éteindre totalement si le véhicule doit fonctionner en mode zéro émission (ou *Zero Emission Vehicle* soit *ZEV* en anglais). La dernière architecture, l'hybride série/parallèle également appelée *power split*, permet de combiner les deux architectures précédentes en une seule. Cette architecture peut donc prendre les avantages et intérêts de l'hybride série et de l'hybride parallèle grâce à l'utilisation d'une transmission à base de trains épicycloïdaux.

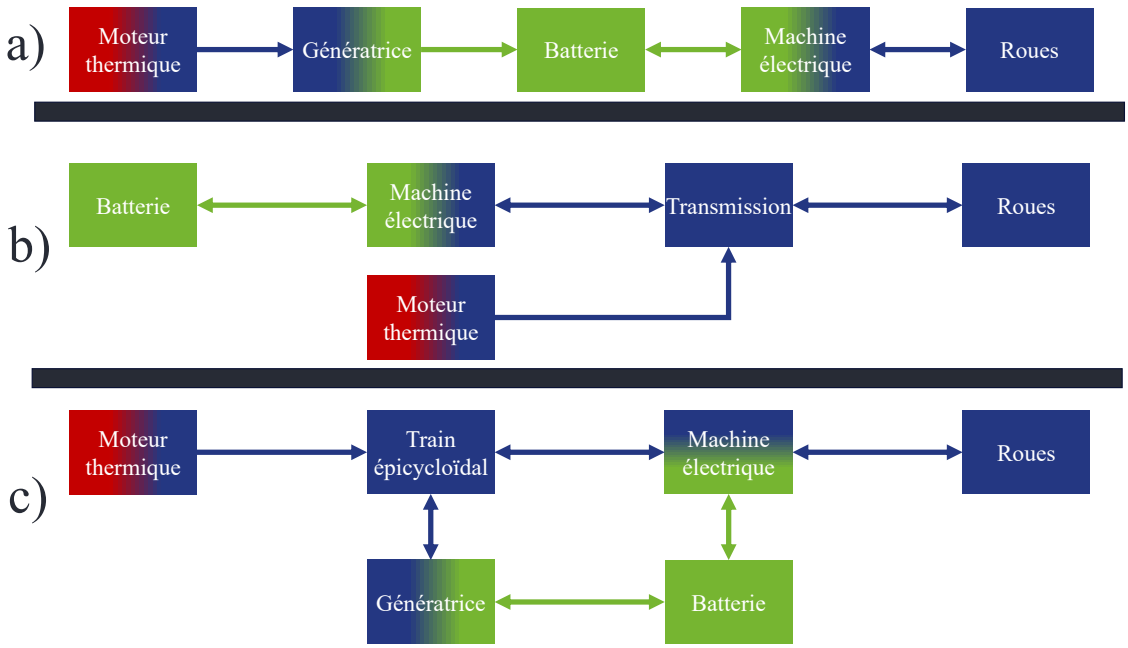

FIGURE 2.1 – Architectures de véhicules hybrides électriques. La première (a) illustre l'hybride série, la seconde (b) l'hybride parallèle et la dernière (c) l'hybride série/parallèle.En rouge sont représentés les éléments thermiques, en bleu les éléments et flux d'énergie mécaniques, en vert les éléments et flux d'énergie électriques.

2.1.2 Cycles thermodynamiques

Cette sous-section permet de faire un rappel des notions importantes de thermodynamique qui seront utilisées dans la suite du chapitre, ainsi que de justifier les choix technologiques réalisés.

Deux grandeurs physiques sont très importantes : l'enthalpie (h) et l'entropie (s). La première représente l'énergie contenue sous forme de chaleur par un fluide ou un solide. La seconde traduit l'énergie dissipée par un système qui n'est pas récupérable. Couplées respectivement à la pression et à la température, ces deux grandeurs physiques permettent la représentation des cycles thermodynamiques dans deux diagrammes nommés par les grandeurs utilisées : Ph et Ts.

Différentes solutions existent pour récupérer une énergie thermique comme le cycle de Joule-Brayton [77], le cycle de Rankine [78], etc. L'objectif de ces différents cycles thermodynamiques est de convertir un flux de chaleur, noté $\dot{Q}$ en une puissance mécanique, noté $\dot{W}$. Il est nécessaire d'avoir deux sources de chaleur (une chaude et l'autre froide) pour générer le flux de chaleur. L'avantage de ces cycles est qu'ils s'appuient sur une source de chaleur externe, ce qui n'est pas le cas du cycle de Carnot [79] ou d'un cycle de Beau de Rochas, sur lesquels sont basés les moteurs à combustion interne [80].

Les cycles à source de chaleur externe sont composés de trois ou quatre organes pour

réaliser la conversion de puissance : un organe de compression, un ou deux échangeurs et un organe de détente. Un fluide, dit de travail, traverse alors ces différents éléments pour générer la conversion de puissance. Parmi tous ces cycles, le cycle de Rankine a pour avantage d'exploiter le phénomène de changement de phase du fluide de travail pour augmenter la capacité d'absorption de chaleur du cycle.

Par exemple, à pression atmosphérique normale, il faut apporter $2256,4$ kJ pour évaporer un kg d'eau. Il faudrait 100 kg d'eau sous forme de vapeur pour absorber la même quantité d'énergie pour avoir une élévation de température de 10 °C. Cette masse très importante illustre la raison de l'exploitation des changements de phase. Par conséquent, l'utilisation de cycles de Rankine est privilégiée.

Toutefois, plusieurs variantes du cycle de Rankine existent : cycle de Rankine organique (*ORC*, *Organic Rankine Cycle*) [81], cycle de Hirn [82] ou le cycle de Kalina [83] par exemple. Sommairement, le cycle de Rankine organique utilise des fluides dits organiques dont les températures d'évaporation sont plus faibles que l'eau et donc permettent l'exploitation de sources chaudes à une température plus faible qu'avec de l'eau. Le cycle de Hirn est un cycle de Rankine avec une surchauffe ; ce qui implique que la température la plus haute du fluide de travail n'est pas celle d'évaporation. Le cycle de Kalina se base sur l'utilisation d'un mélange de deux fluides ainsi que sur une architecture plus complexe (plus de quatre organes) ; ce qui permet d'améliorer le rendement du cycle par rapport aux autres variantes du cycle de Rankine [84].

Dans le cas d'application considéré, deux sources de chaleur sont à exploiter : les gaz d'échappement ainsi que le liquide de refroidissement (LdR) du moteur thermique. Les températures de ces deux sources étant très différentes (la première étant proche de 800°C et la seconde de 100°C), deux cycles différents seront utilisés. Un cycle de Hirn fonctionnant avec de l'eau sera utilisé pour récupérer les pertes contenues dans les gaz d'échappement tandis qu'un cycle de Hirn fonctionnant avec un fluide organique recouvrera les pertes contenues dans le liquide de refroidissement. Par la suite, ces deux cycles seront appelés par simplification "cycles de Rankine". Le cycle de Kalina n'a pas été retenu pour les deux applications car sa complexité de conception semble moins adaptée pour l'application envisagée.

2.2 Architecture de véhicule considérée et modélisation

Une architecture hybride électrique série a été choisie eu égard aux avantages et inconvénients de chaque architecture d'hybride possible et de l'intégration d'une structure de systèmes récupérateurs de puissance dans un groupe moto propulseur (GMP) hybride. Le fonctionnement de la structure de systèmes récupérateurs sera expliqué dans la sous-section 2.2.2.

La modélisation de la chaîne de traction du véhicule présentée dans la section suivante met en lumière l'état de charge de la batterie comme la seule variable avec une dynamique à prendre en compte. C'est pourquoi, la programmation dynamique présentée dans le chapitre 1 sera utilisée comme méthode de commande optimale dans cette étude. Le modèle batterie

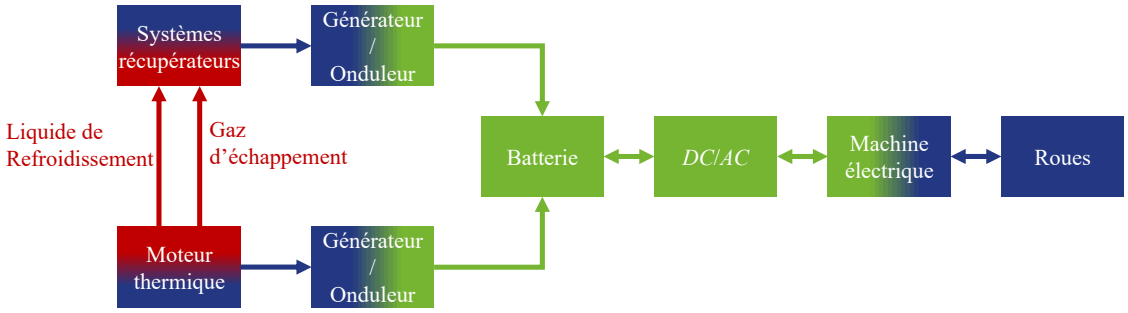

FIGURE 2.2 – Schéma bloc du véhicule hybride série considéré avec l'ajout des machines récupératrices de puissance. Les flux ou éléments de chaleur sont représentés en rouge, mécanique en bleu et électrique en vert.

explicité par la suite est alors adapté pour retirer la dépendance en température [85]. Pour limiter au plus le nombre de variables d'état, les hypothèses de fonctionnement en régime stabilisé des machines thermodynamique ainsi que d'une température constante du moteur thermique sont posées.

2.2.1 Châssis, batterie et moteurs

La deuxième loi de Newton [58] est utilisée pour représenter le bilan des forces exercées sur le châssis du véhicule :

$$\sum \boldsymbol{F} = m.\boldsymbol{a} \tag{2.1}$$

Étant donné que nous sommes intéressés uniquement par le déplacement dans une seule direction, nous pouvons nous affranchir de l'écriture vectorielle. De plus, cette somme des forces peut être perçue d'un point de vue macroscopique comme étant la force de traction $F_t(t)$ moins la force résistive au mouvement $F_{res}(t)$:

$$F_t(t) - F_{res}(t) = m.a(t) \tag{2.2}$$

La force de traction provient directement du couple de la machine électrique appliquée aux roues via la transmission tandis que la force résistive est exprimée selon la loi de route (2.3) dont les coefficients θ_0, θ_1 et θ_2 sont déduits expérimentalement [50]. Le véhicule considéré possède une masse $m_{véhicule} = 1100$ kg et des caractéristiques aérodynamiques $S.Cx = 0.63$ m^2.

$$F_{res}(t) = \theta_0 + \theta_1.v(t) + \theta_2.v^2(t) \tag{2.3}$$

Pour appliquer la force $F_t(t)$ au véhicule, la machine électrique doit fournir le couple $C_{Mel}(t)$ (2.4) aux roues avec ζ_{roues} le rayon des roues, $\eta_{transmission}$ le rendement de la chaîne de transmission entre la machine électrique et les roues et γ le rapport de démultiplication

entre les roues et la machine électrique.

$$\begin{cases} C_{Mel}(t) = \frac{F_t(t).\zeta_{roues}.\eta_{transmission}}{\gamma} \text{ si } F_t(t) \leq 0 \\ C_{Mel}(t) = \frac{F_t(t).\zeta_{roues}}{\gamma.\eta_{transmission}} \text{ sinon} \end{cases} \tag{2.4}$$

La vitesse de rotation de la machine électrique est définie par l'équation (2.5).

$$\omega_{Mel}(t) = \frac{v(t).\gamma}{\zeta_{roues}} \tag{2.5}$$

Le rendement de la machine électrique a été évalué expérimentalement en fonction du couple et du régime de rotation $\eta_{Mel}(C_{Mel}, \omega_{Mel})$ ainsi illustré figure 2.3.

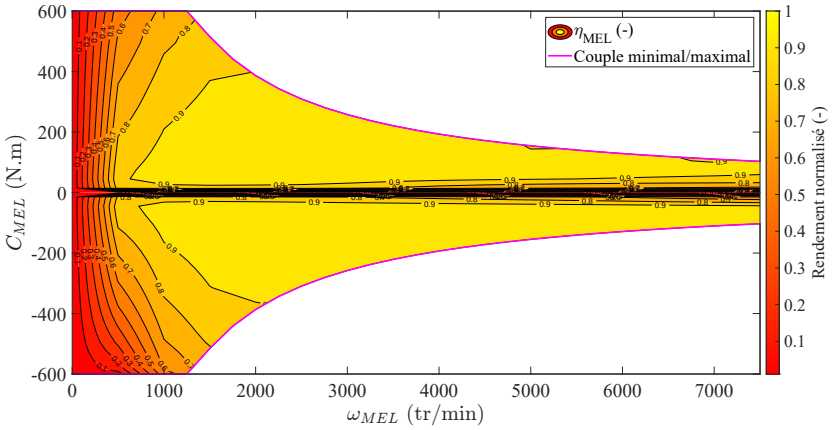

FIGURE 2.3 – Rendement normalisé en fonction du couple et du régime de rotation de la machine électrique de traction du véhicule.

La batterie doit alors fournir la puissance électrique demandée par la machine électrique pour respecter la vitesse demandée. Cependant, la batterie n'assure pas seulement les besoins en puissance de la machine électrique. En effet, la puissance électrique nécessaire au compresseur du système de climatisation $\dot{W}_{comp}$, ainsi que les demandes auxiliaires P_{AUX} d'électricité sont également assurées par la batterie. Le moteur thermique qui produit la puissance mécanique P_{Eng} permet de recharger la batterie grâce à la génératrice d'un rendement $\eta_{Gen} = 0,95$ considéré constant. L'équation (2.6) décrit la puissance dépensée ou perçue par la batterie.

$$P_{Bat}(t) = \frac{P_{Mel}(t)}{\eta_{Mel}(C_{Mel}(t), \omega_{Mel}(t))} + \dot{W}_{comp}(t) + P_{AUX}(t) - P_{Eng}(t).\eta_{Gen} \tag{2.6}$$

$$\tag{2.7}$$

Avec $P_{Mel}(t) = \omega_{Mel}(t).C_{Mel}(t)$ et $P_{Eng}(t) = \omega_{Eng}(t).C_{Eng}(t)$ qui est composée des deux commandes du système. Le débit de carburant consommé $\dot{m}_{carb}(t)$ pour obtenir cette puis-

sance P_{Eng} est obtenu avec l'équation suivante :

$$\dot{m}_{carb}(t) = \frac{P_{Eng}(t)}{\eta_{Eng}(\omega_{Eng}(t), C_{Eng}(t)).LHV} \tag{2.8}$$

avec η_{Eng} qui est le rendement du moteur thermique en fonction du couple et du régime de rotation du moteur thermique et le pouvoir calorifique du carburant LHV exprimé en J/kg. La variation de l'état du système est celle de l'état de charge de la batterie $S\dot{O}C_{Bat}(t)$:

$$\begin{aligned} S\dot{O}C_{Bat}(t) &= \frac{-I_{Bat}(t)}{A_{max}} \\ S\dot{O}C_{Bat}(t) &= \frac{-P_{Bat}(t)}{A_{max}.V_{Bat}(SOC_{Bat}(t))} \end{aligned} \tag{2.9}$$

La tension de la batterie V_{Bat} est obtenue avec le modèle batterie [85] [86] illustré en figure 2.4. Il est considéré ici que la tension à vide (OCV, *Open Circuit Voltage*) ainsi que la résistance interne de la batterie R dépendent uniquement de l'état de charge de la batterie.

$$V_{Bat}(t) = OCV(SOC_{Bat}(t)) - R(SOC_{Bat}(t)).I(t) \tag{2.10}$$

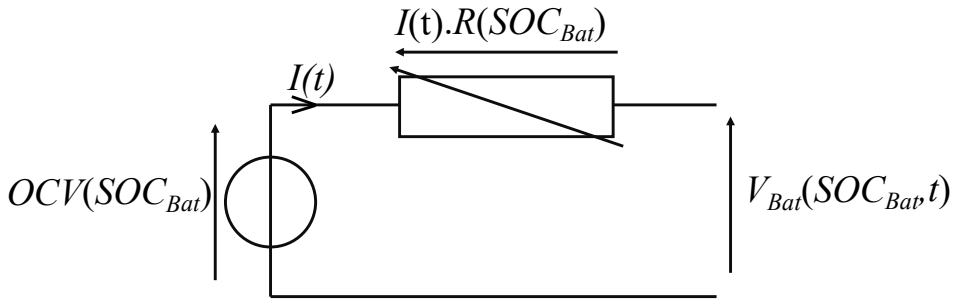

FIGURE 2.4 – Modèle batterie simplifié exprimant la tension en fonction de la tension à vide OCV, de la résistance interne R et du courant de la batterie I.

La figure 2.5 présente l'évolution de la tension à vide et de la résistance interne en fonction de l'état de charge de la batterie. Il apparaît que la résistance de la batterie est considérée constante sur toute la plage de SOC lorsque la batterie se recharge ; ce qui n'est pas le cas de la résistance dans le cas d'une décharge. En effet, on peut observer une plage de fonctionnement préférentielle sur l'état de charge de la batterie durant laquelle la résistance de décharge est minimum, entre 30% et 70% de l'état de charge de la batterie. De plus, le tension à vide augmente avec l'état de charge de la batterie. Cette évolution n'est cependant pas linéaire.

La génératrice est modélisée par un rendement fixe $\eta_{Gen} = 0.95$. Le moteur thermique est quant à lui modélisé par trois cartographies obtenues expérimentalement :

— une cartographie de rendement du moteur en fonction du régime et du couple moteur (figure 2.6) .

— une cartographie de température des gaz d'échappement (T_{1hf}) après le système de dépollution également en fonction du régime et du couple moteur (figure 2.8).
— une cartographie de débit de liquide de refroidissement uniquement en fonction du régime moteur (figure 2.7).

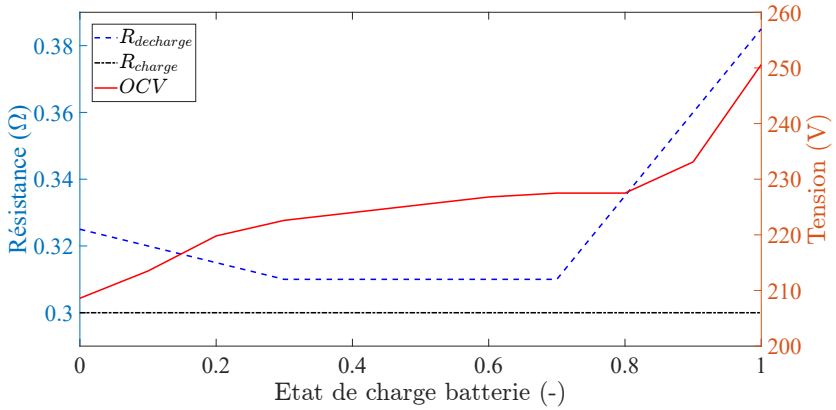

FIGURE 2.5 – Courbes de tension à vide et de résistance en fonction de l'état de charge de la batterie et du sens du courant (charge ou décharge).

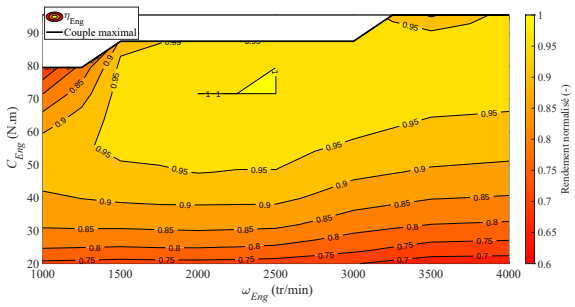

FIGURE 2.6 – Rendement normalisé du moteur.

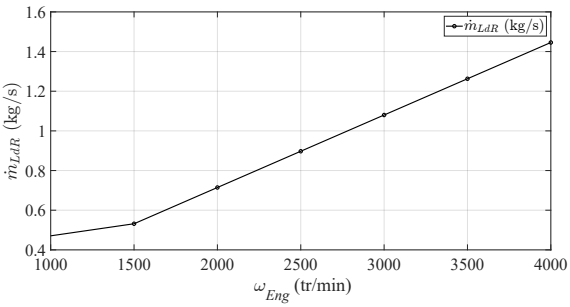

FIGURE 2.7 – Débit de liquide de refroidissement.

2.2.2 Architecture véhicule

Un des inconvénients majeurs des moteurs thermiques réside dans leur rendement. En effet, le rendement est, au mieux, proche des 40% sur les points de meilleurs rendements. Le reste est évacué sous forme de chaleur dans les gaz d'échappement ou dans le liquide de refroidissement. L'utilisation des architectures hybrides, notamment série ou série/parallèle, permet d'exploiter le moteur uniquement sur les plages de meilleur rendement. Pour améliorer le rendement global d'un moteur thermique, plusieurs solutions sont possibles. Une première solution est de réaliser une détente plus longue que la compression. On parle alors de cycle de Miller ou d'Atkinson [87] [88]. Pour mettre au point ce cycle moteur, il est alors nécessaire d'utiliser des technologies particulières durant la conception du moteur. C'est notamment le cas du moteur EB2DTS qui utilise un double variateur d'arbre à cames pour réaliser ces détentes prolongées. Une seconde solution serait d'utiliser des machines récupératrices de

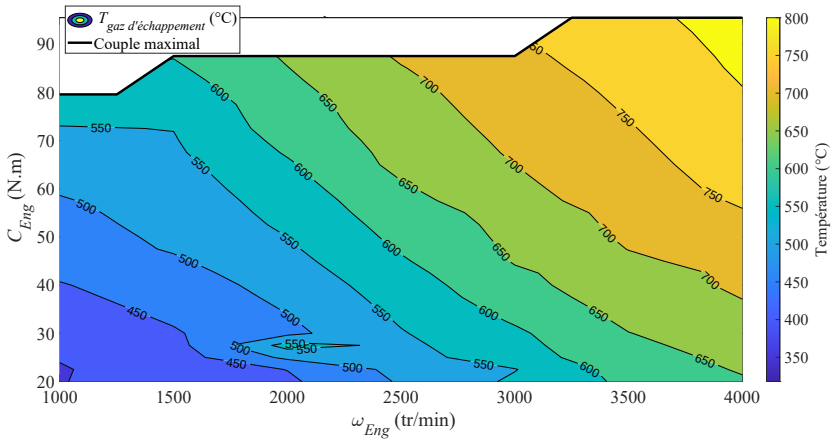

FIGURE 2.8 – Température des gaz d'échappement après systèmes de dépollution.

puissance basée sur des cycles de Rankine ou de Hirn qu'on appelle machine de Rankine. Ces machines thermiques permettent de transformer une puissance sous forme de chaleur en puissance mécanique. Le fonctionnement de ces machines sera expliqué dans la section 2.2.3. Une troisième solution serait d'utiliser des thermocouples, basés sur l'effet Seebeck, pour également récupérer les pertes du moteur [89] [90]. Cependant, selon [3], les rendements des machines de Rankine sont supérieurs aux thermocouples ; c'est pourquoi les machines de Rankine ont été privilégiées dans cette étude. Étant donné que les machines de Rankine ou les thermocouples ne récupèrent la puissance qu'à travers une source de chaleur, il peut être envisagé de remplacer le moteur thermique par une pile à combustible chaude, appelée *Solid Oxide Fuel Cell* [91], pour une architecture hybride série uniquement.

2.2.2.1 Structure de systèmes récupérateurs de puissance

La structure de systèmes récupérateurs est composée de deux machines récupératrices en orange sur la figure 2.9. La première machine récupératrice, dite de hautes températures et dénotée par l'acronyme *WRM* en exposant pour *Water Rankine Machine*, utilise de l'eau comme fluide de travail afin de récupérer l'énergie perdue dans les gaz d'échappement. La seconde machine, dite de moyennes températures et dénotée par l'acronyme *ORM* en exposant pour *Organic Rankine Machine*, utilise quant à elle un fluide organique pour récupérer l'énergie évacuée par le liquide de refroidissement. Ce choix de fluide pour les deux machines est motivé par les caractéristiques thermodynamiques des différents fluides. Le fluide organique utilisé dans cette étude est le r1234yf. Ce choix a été poussé par les restrictions et interdictions de certains fluides dans le secteur automobile. L'objectif de ces deux machines est alors de récupérer le plus de puissance possible à leurs turbines pour maximiser le rendement global du groupe motopropulseur moteur thermique plus machines récupératrices. Il faut cependant retrancher les puissances que la batterie doit fournir aux pompes pour que les machines puissent récupérer les puissances aux turbines.

La machine consommatrice, également appelée climatisation, en bleue sur la figure 2.9

dénotée par l'acronyme *iORC* pour *inversed Organic Rankine Cycle* utilise également le fluide organique r1234yf comme fluide de travail. L'objectif de cette machine est de climatiser l'air au sein de l'habitacle dans un but de confort thermique pour les usagers. Une dépense énergétique non négligeable est faite pour alimenter le compresseur de la machine afin de réaliser l'échange de chaleur [92].

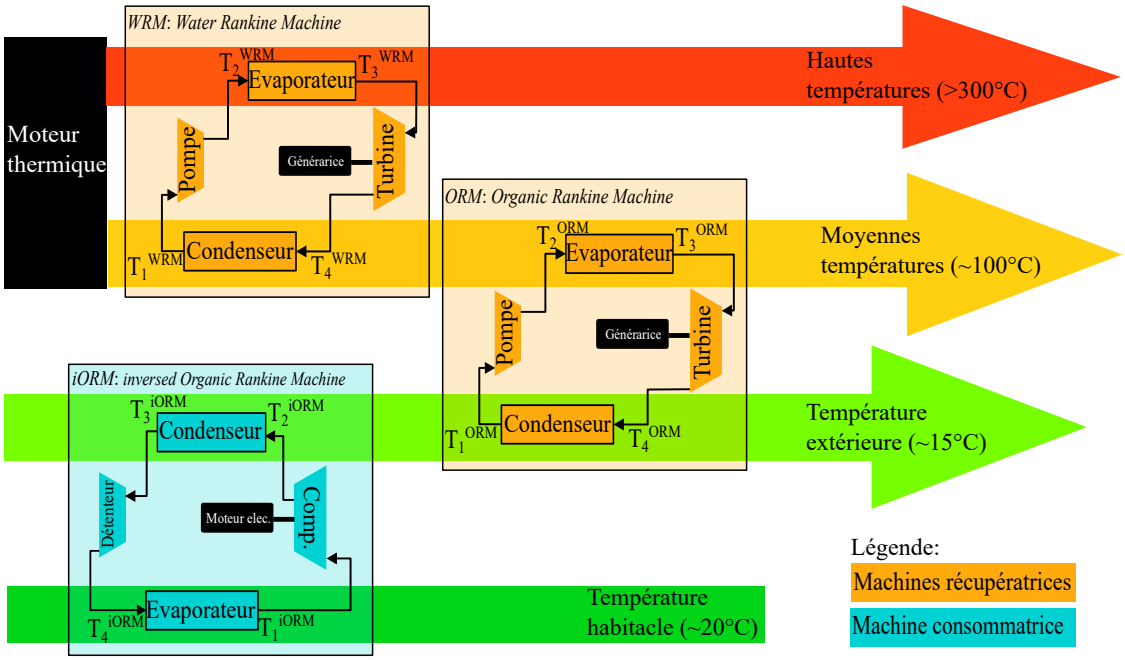

FIGURE 2.9 – Représentation graphique de la structure de machines récupératrices de puissance en orange et de la machine consommatrice en bleu.

2.2.2.2 Hypothèses

Les hypothèses utilisées pour la construction des modèles physiques des machines thermodynamiques sont les suivantes :

- La structure de machines récupératrices de puissance n'affecte pas le fonctionnement du moteur car la chaleur est récupérée après le système de dépollution.
- Les pertes de charge sont négligée dans l'échappement.
- Les modèles moteurs, batterie et véhicule sont des modèles sans dynamique. Aucun démarrage à froid ni de phase de chauffe du moteur ne sont considérés. Les moteurs et la batterie sont considérés à température constante.
- Les machines de Rankine fonctionnent avec des cycles de Rankine théoriques en régime stabilisé. Aucune perte de charge n'est considérée. Les échanges de chaleur sont parfaits dans les échangeurs. La compression et la détente sont non-isentropiques.
- Le débit massique du fluide chaud ($\dot{m}_{hf}$) est connu.
- Tous les échangeurs sont à contre-courants. Cette solution permet le meilleur compromis entre compacité de l'échangeur et efficacité [93].

De plus, les rendements des pompes, compresseurs et turbines utilisés dans cette étude sont considérés constants : $\eta_{pompe} = 0.75$, $\eta_{compresseur} = 0.75$ et $\eta_{turbine} = 0.82$. Toutes ces hypothèses placent alors l'étude dans une meilleure configuration que la réalité.

2.2.3 Fonctionnement des machines récupératrices

2.2.3.1 Description du fonctionnement du cycle de Rankine

La figure 2.10 représente un cycle théorique de Rankine dans un diagramme température/entropie, aussi noté T,s. Le cycle réalisé par le fluide de travail est représenté en noir, tandis que le fluide chaud, dont on récupère la puissance, est en rouge et le fluide froid, où les pertes sont rejetées, est en bleu. Comme on peut le voir sur les figures 2.9 et 2.10, une machine est composée de quatre éléments distincts : une pompe, une turbine et deux échangeurs (évaporateur et condenseur). La courbe en orange représente la courbe de saturation du fluide de travail. A droite de la cloche, le fluide est gazeux, liquide à gauche et diphasique à l'intérieur. La courbe de saturation représente ici un fluide dit humide, mais il existe également des fluides secs et isentropiques [94]. La différence entre ces trois types de fluides réside dans l'allure de la courbe de saturation du côté vapeur. En effet, si l'entropie augmente avec la diminution de la température, comme illustré sur la figure, alors le fluide est humide. Si l'entropie diminue avec la diminution de la température, le fluide est alors sec. Dans le cas où l'entropie est constante avec la diminution de la température, le fluide est isentropique. Dans le cas où le fluide utilisé est humide, il est possible que le fluide devienne diphasique lors de la détente dans la turbine ce qui est dangereux pour le bon fonctionnement de la machine récupératrice. Le fluide organique r1234yf est isentropique tandis que l'eau est un fluide humide.

La pompe est l'élément qui permet d'imposer le débit du fluide de travail souhaité et de faire passer le fluide de travail de la pression P_{cond} à la pression P_{evap} (de 1 à 2). Le travail nécessaire à cette élévation de pression est faible car le fluide est sous forme liquide. La puissance électrique nécessaire à la pompe pour réaliser ces deux objectifs doit cependant être fournie par une batterie pour faire fonctionner la machine. La turbine est l'élément qui, au contraire, permet de détendre le fluide de la pression P_{evap} à la pression P_{cond} (de 3 à 4). C'est au passage du fluide de travail sous forme gazeuse dans la turbine qu'il est possible de récupérer la puissance de ce dernier. Lors de la détente, le fluide de travail doit tout le temps être sous forme gazeuse.

Lors de la détente, le fluide de travail doit tout le temps être sous forme gazeuse. Au contraire, lors de la compression, le fluide de travail doit tout le temps être sous forme liquide. Ces deux contraintes sont imposées pour ne pas détériorer prématurément les organes de compression et de détente. [95]

L'évaporateur est l'échangeur qui permet l'échange de puissance entre le fluide chaud et le fluide de travail (de 2 à 3). Cela se traduit par une montée en température du fluide de travail au détriment d'une baisse de température du fluide chaud. Le condenseur est l'échangeur

qui permet l'échange de puissance entre le fluide de travail et le fluide froid (de 4 à 1). La température du fluide de travail diminue alors tandis que celle du fluide froid augmente.

Il est important de souligner ici qu'il faut faire attention aux différences de température entre les deux fluides dans l'échangeur. En effet, pour considérer que l'échange de chaleur est possible, il faut dans un premier temps qu'il n'y ait pas de croisement de températures. Dans cette étude, il est considéré qu'il faut une certaine différence entre les températures pour que l'échange de chaleur soit réalisable. Cette différence minimale est appelée température de pincement T^{pinch}. La table 2.1 décrit les valeurs à respecter pour chaque échangeur des deux machines (WRM et ORM). Si une température de pincement n'est pas respectée, deux solutions sont possibles. Soit la machine est considérée comme à l'arrêt donc aucune récupération de puissance est faite mais aucune dépense également. Sinon, les températures d'adaptation T_{cond}^{adapt} ou T_{evap}^{adapt} introduites dans les équations (2.11) et (2.12) peuvent être utilisées pour assurer l'écart de température minimum afin d'avoir un échange de chaleur.

TABLE 2.1 – Températures de pincement à respecter dans chacun des quatre échangeurs des machines récupératrices

	WRM	ORM
évaporateur	50°C	5°C
condenseur	5°C	5°C

Cependant, pour se rapprocher de la réalité, il serait également important de fixer les quatre échangeurs de l'architecture et de voir quelle configuration permettrait d'être le plus proche de l'optimalité en général. Les températures T_{cond}^{adapt} et T_{evap}^{adapt} seront alors des indicateurs de points de fonctionnement où l'échangeur sera contraint pour réaliser l'échange de chaleur.

La construction du cycle thermodynamique débute par le placement des points 1 et 3 de la figure 2.10 à partir des équations (2.11) et (2.12). La température T_3 au point 3 est limitée par la température T_{1hf} de la source chaude à l'entrée de l'évaporateur et par la température de pincement T_{evap}^{pinch} pour avoir un échange de chaleur réalisable. Cependant, T_3 peut être inférieure à $T_{1hf} - T_{evap}^{pinch}$, c'est pourquoi la température T_{evap}^{adapt} est introduite. Cette température permet de faire varier T_3 si l'inégalité $T_3 \leq T_{1hf} - T_{evap}^{pinch}$ n'est pas respectée. La démarche est identique pour la température au point 1 T_1 vis-à-vis de la température d'entrée de la source froide T_{1cf}, de la température de pincement T_{cond}^{pinch} et de la température d'adaptation T_{cond}^{adapt}. Les deux températures d'adaptation apparaissent alors comme des variables permettant le dimensionnement de l'échangeur lequel devra réaliser l'échange thermique.

$$T_3 = T_{1hf} - T_{evap}^{pinch} - T_{evap}^{adapt} \tag{2.11}$$

$$T_1 = T_{1cf} + T_{cond}^{pinch} + T_{cond}^{adapt} \tag{2.12}$$

De ces deux premières températures du cycle, il est possible de déterminer les températures de changement de phase si les températures de surchauffe $T_{surchauffe}$ ou de sous-refroidissement $T_{sous-refroidissement}$ sont connues. Dans notre étude, $T_{sous-refroidissement}$ est

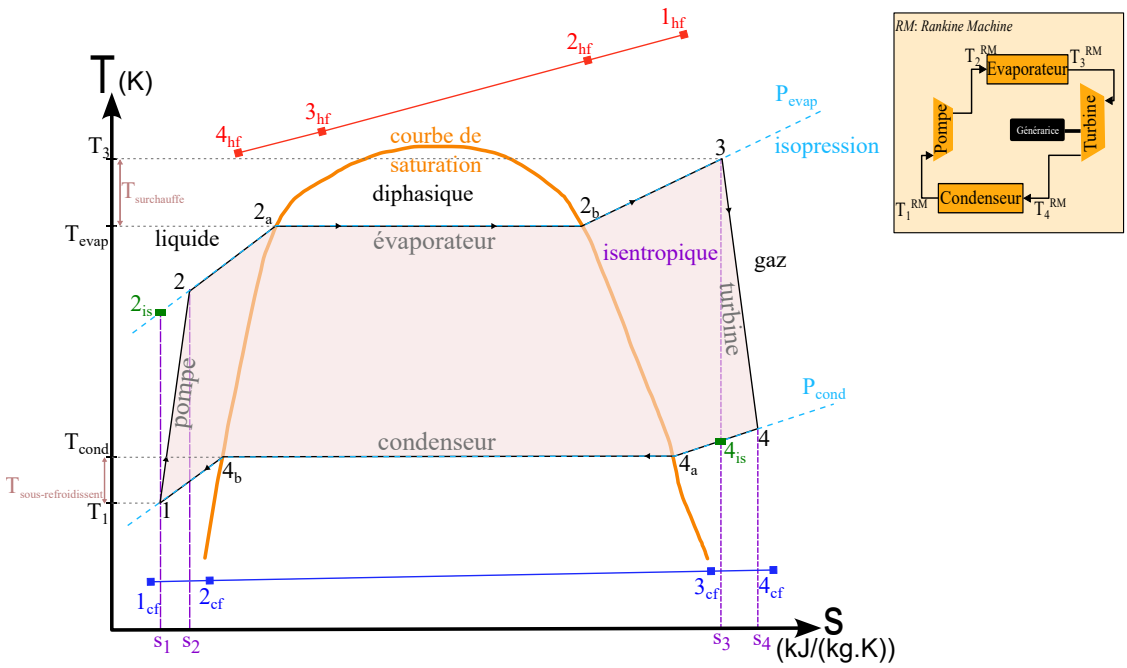

FIGURE 2.10 – Représentation d'un cycle thermodynamique Rankine sur un diagramme T,s. La source chaude est représentée en rouge avec les indices hf, la source froide en bleu avec les indices cf.

imposée à 1°C pour les machines récupératrices afin d'assurer que le fluide compressé est totalement liquide. Cette restriction est imposée pour ne pas dégrader la pompe utilisée pour la phase de compression. $T_{surchauffe}$ est imposée lors de la construction du cycle. La valeur de cette température est variable pour permettre une récupération de puissance plus ou moins importante en fonction des limitations imposées au cycle de Rankine vis-à-vis des conditions extérieures.

$$T_{evap} = T_{2a} = T_{2b} = T_3 - T_{surchauffe} \tag{2.13}$$

$$T_{cond} = T_{4a} = T_{4b} = T_1 + T_{sous-refroidissement} \tag{2.14}$$

Ces deux températures de changement de phase induisent les pressions d'évaporation P_{evap} et de condensation P_{cond} en fonction du fluide de travail. Il est possible de les obtenir grâce à des tables de données [96] [97].

$$Pr_{evap} = \varphi(T_{evap}) \tag{2.15}$$

$$Pr_{cond} = \varphi(T_{cond}) \tag{2.16}$$

Grâce à l'information sur les pressions, il est désormais possible de calculer les enthalpies et entropies aux points 1, 3 (h_1, h_3, s_1, s_3) et également les quatre points sur la courbe de saturation, 2_a, 2_b, 4_a et 4_b, grâce à une lecture dans les tables de données. La lecture de la

table est représentée par l'équation (2.17) avec κ la grandeur désirée (température, pression, enthalpie ou entropie) , f_κ la lecture de la table voulue en fonction de deux grandeurs physiques β_1 et β_2 qui peuvent être température, pression, enthalpie, entropie ou titre vapeur et du fluide, noté F, utilisé.

$$\kappa = \varphi_\kappa(\beta_1, \beta_2, F) \tag{2.17}$$

L'équation (2.18) montre, à titre d'exemple, l'utilisation de l'équation précédente. Pour obtenir l'enthalpie au point 1, les deux grandeurs connues sont utilisées : température et pression. La température est calculée en (2.12) tandis que sa pression est égale à la pression de condensation : $Pr_1 = Pr_{cond}$. Si le calcul est fait pour le *WRM*, alors le fluide utilisé est de l'eau : $F = eau$.

$$h_1 = \varphi_h(T_1, Pr_1, eau) \tag{2.18}$$

Comme les entropies s_1 et s_3 sont connues, il est possible de connaître les caractéristiques thermodynamiques des points théoriques à la sortie de la compression 2_{is} et de la détente 4_{is} isentropique. Avec ces points théoriques et les efficacités de la pompe et de la turbine, il est alors possible de déterminer les enthalpies aux points h_2 et h_4.

$$h_2 = h_1 + \frac{(h_{2is} - h_1)}{\eta_{pompe}} \tag{2.19}$$

$$h_4 = h_3 - \eta_{turbine}.(h_3 - h_{4is}) \tag{2.20}$$

A partir des enthalpies et des pressions aux points 2 et 4, il est possible de connaître totalement les points avec l'équation (2.17). Alors le cycle en entier est connu. En fonction de la puissance à récupérer du fluide chaud $\dot{Q}_{in}$, il est alors possible de déterminer le débit du fluide de travail $\dot{m}_{wf}$ en kg/s.

$$\dot{m}_{wf} = \frac{\dot{Q}_{in}}{(h_3 - h_2)} \tag{2.21}$$

Avec les enthalpies h_1, h_2, h_3, h_4 et le débit du fluide de travail, il est possible de connaître la puissance à fournir à la pompe $\dot{W}_{pompe}$, la puissance récupérée par la turbine $\dot{W}_{turbine}$ et la puissance évacuée sous forme de chaleur dans le fluide froid $\dot{Q}_{out}$.

$$\dot{W}_{pompe} = \dot{m}_{wf}.(h_2 - h_1) \tag{2.22}$$

$$\dot{W}_{turbine} = \dot{m}_{wf}.(h_4 - h_3) \tag{2.23}$$

$$\dot{Q}_{out} = \dot{m}_{wf}.(h_1 - h_4) \tag{2.24}$$

La vérification que la somme des quatre puissances ($\dot{Q}_{in}$, $\dot{W}_{pompe}$,$\dot{W}_{turbine}$,$\dot{Q}_{out}$) est nulle est faite pour s'assurer de la loi de conservation de l'énergie. Pour vérifier que le machine puisse fonctionner, il faut également s'assurer que les deux échangeurs puissent assurer leurs fonctions. Pour ce faire, il faut comparer les températures des deux fluides (chaud ou froid et travail) aux quatre endroits spécifiques de l'échangeur (figure 2.11). Cette figure représente un évaporateur dans lequel le fluide de travail en bleu récupère la puissance du fluide

chaud en rouge. Les points thermodynamiques considérés durant l'échange de chaleur ont été positionnés pour expliquer la notion de température de pincement. Le long de l'échangeur, l'échange de chaleur est dicté par la différence de température entre les deux fluides [93]. La plus petite différence de température à une position donnée correspond alors à la température de pincement dans l'échangeur.

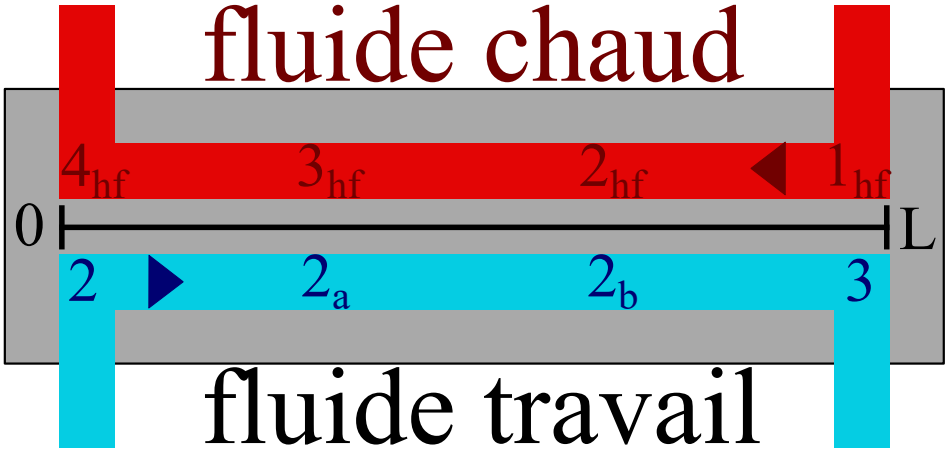

FIGURE 2.11 – Représentation spatiale d'un évaporateur avec en rouge le fluide chaud et en bleu le fluide de travail. Les positions des huit points thermodynamiques sont indiquées par leur dénomination. Les deux flèches indiquent le sens des écoulements qui donnent la configuration d'un échangeur contre-courants.

Dans le cas de l'évaporateur, il faut comparer T_3 avec T_{1hf}, T_{2b} avec T_{2hf}, T_{2a} avec T_{3hf} et T_2 avec T_{4hf}.

$$T_{evap}^{pinch} = min \begin{pmatrix} T_{1hf} - T_3 \\ T_{2hf} - T_{2b} \\ T_{3hf} - T_{2a} \\ T_{4hf} - T_2 \end{pmatrix} \tag{2.25}$$

Avec :

$$T_{2hf} = T_{1hf} - \frac{h_3 - h_{2b}}{Cp_{hf}.\dot{m}_{hf}} \tag{2.26}$$

$$T_{3hf} = T_{1hf} - \frac{h_3 - h_{2a}}{Cp_{hf}.\dot{m}_{hf}} \tag{2.27}$$

$$T_{4hf} = T_{1hf} - \frac{h_3 - h_2}{Cp_{hf}.\dot{m}_{hf}} \tag{2.28}$$

La plus petite différence doit être au moins supérieure à la température de pincement minimum imposée qui permet de définir si un échange est réalisable ou non dans l'échangeur.

Le même calcul est fait pour le condenseur.

$$T_{cond}^{pinch} = min \begin{pmatrix} T_4 - T_{4cf} \\ T_{4a} - T_{3cf} \\ T_{4b} - T_{2cf} \\ T_1 - T_{1cf} \end{pmatrix} \tag{2.29}$$

Toutefois, les températures T_{4cf}, T_{3cf} et T_{2cf} ne sont calculées que si le fluide froid est le liquide de refroidissement. En effet, dans le cas de l'*ORM*, le fluide froid est l'air extérieur qui est considéré comme étant à température constante.

$$T_{2cf}^{WRM} = T_{1cf}^{WRM} + \frac{\dot{m}_{wf}^{WRM}.(h_{4b}^{WRM} - h_1^{WRM})}{\dot{m}_{cf}^{WRM}.Cp_{cf}^{WRM}} \tag{2.30}$$

$$T_{3cf}^{WRM} = T_{1cf}^{WRM} + \frac{\dot{m}_{wf}^{WRM}.(h_{4a}^{WRM} - h_1^{WRM})}{\dot{m}_{cf}^{WRM}.Cp_{cf}^{WRM}} \tag{2.31}$$

$$T_{4cf}^{WRM} = T_{1cf}^{WRM} + \frac{\dot{m}_{wf}^{WRM}.(h_4^{WRM} - h_1^{WRM})}{\dot{m}_{cf}^{WRM}.Cp_{cf}^{WRM}} \tag{2.32}$$

$$T_{1cf}^{ORM} = T_{2cf}^{ORM} = T_{3cf}^{ORM} = T_{4cf}^{ORM} \tag{2.33}$$

Lorsque toutes les grandeurs physiques sont calculées, il faut maintenant s'assurer que les cycles thermodynamiques respectent toutes les contraintes.

Afin de calculer successivement les différents points thermodynamiques des fluides de travail, la table 2.2 récapitule l'ordre à suivre pour calculer les variables thermodynamiques souhaitées et les équations utilisées. Une couleur identique indique que les variables ont la même valeur. Pour commencer, il faut calculer les variables indiquées par un 'A' grâce aux équations (2.11) et (2.12). A partir de ces variables et d'autres informations, il est alors possible de calculer les variables indiquées par un 'B' et ainsi de suite jusqu'à la complétion du tableau.

Pour assurer que le fluide de travail conserve le même état durant la compression ou la détente, une vérification sur les enthalpies est réalisée. D'un côté, lors de la compression, il faut alors assurer que $h_1 \leq h_{4b}$ et $h_2 \leq h_{2a}$. Si ces deux inéquations sont vérifiées, alors le fluide est toujours liquide lors de la compression. D'un autre côté, lors de la détente si $h_3 \geq h_{2b}$ et $h_4 \geq h_{4a}$ sont vérifiées, alors le fluide est toujours gazeux lors de la détente.

TABLE 2.2 – Table explicative de l'ordre à suivre pour calculer les variables thermodynamiques pour les cycles de Rankine. Les variables dont les cellules sont de même couleur (excepté blanc) sont identiques.

	T (K)	Pr (Pa)	h (kJ/kg)	s (kJ/(kg.K))
1	A (2.12)	C (2.16)	D (2.17)	D (2.17)
2is		C (2.15)	E (2.17)	D (2.17)
2	G (2.17)	C (2.15)	F (2.19)	
2a	B (2.13)	C (2.15)	D (2.17)	
2b	B (2.13)	C (2.15)	D (2.17)	
3	A (2.11)	C (2.15)	D (2.17)	D (2.17)
4is		C (2.15)	E (2.17)	D (2.17)
4	G (2.17)	C (2.15)	F (2.20)	
4a	B (2.14)	C (2.16)	D (2.17)	
4b	B (2.14)	C (2.16)	D (2.17)	

2.2.3.2 Optimisation statique de la structure de systèmes récupérateurs

L'objectif est de maximiser la puissance récupérée par la structure de systèmes récupérateurs :

$$\dot{W}_{nette} = \dot{W}^{WRM}_{turbine} + \dot{W}^{ORM}_{turbine} - \dot{W}^{WRM}_{pompe} - \dot{W}^{ORM}_{pompe} \tag{2.34}$$

Pour chaque point d'étude, trois variables peuvent être modifiées afin de maximiser le rendement global de la chaîne de traction du véhicule. Tout d'abord, un point d'étude est défini par quatre variables :

— La température extérieure T_{ext} qui varie de -20 à 45°C.
— La température du liquide de refroidissement T_{LdR} qui varie de 80 à 110°C.
— Le couple du moteur thermique C_{Eng} qui varie de 0 à 95 N.m.
— La régime de rotation du moteur thermique ω_{Eng} qui varie de 1000 à 4000 tr/min.

Ces variables influent directement sur le dimensionnement des échangeurs ainsi que sur la chaleur utilisée pour alimenter le *WRM*. Cela affecte alors le fonctionnement de la structure de machines récupératrices. Les trois variables d'optimisation sont :

— La température de surchauffe du cycle de Rankine haute température $T^{WRM}_{surchauffe}$ qui varie de 100 à 600°C.
— La température de surchauffe du cycle de Rankine basse température $T^{ORM}_{surchauffe}$ qui varie de 0 à 6°C.
— La température des gaz d'échappement après évaporation T^{WRM}_{4hf} qui varie de 150 à 800°C.

Pour déterminer le triplet qui maximise (2.34), la méthode exhaustive est utilisée 1 : une discrétisation des trois variables est faite et chaque solution est évaluée pour ne garder que la meilleure à la fin. Si aucun triplet n'est rentable, $\dot{W}_{nette} < 0$, alors la structure ne récupère aucune puissance.

La figure 2.12 expose les trois variables d'optimisation des machines récupératrices. Dans l'exemple utilisé, la température des gaz d'échappement après l'évaporateur du *WRM*, T^{WRM}_{4fc}

est la même quelque soit le point de fonctionnement du moteur. C'est une des températures les plus basses possibles ; ce qui montre la possibilité du système à pouvoir récupérer le plus de puissance possible des gaz d'échappement. Les deux températures de surchauffe $T^{WRM}_{surchauffe}$ et $T^{ORM}_{surchauffe}$ évoluent de manière similaire. Elles augmentent avec la puissance récupérable totale disponible. Il est toutefois possible d'observer un phénomène pour $T^{WRM}_{surchauffe}$ qui décroît légèrement au centre de la plage possible du régime de rotation ω_{Eng}. Ceci s'explique par le phénomène de maximisation de la puissance totale récupérée car ce qui n'est pas récupéré par *WRM* est rejeté dans *ORM*.

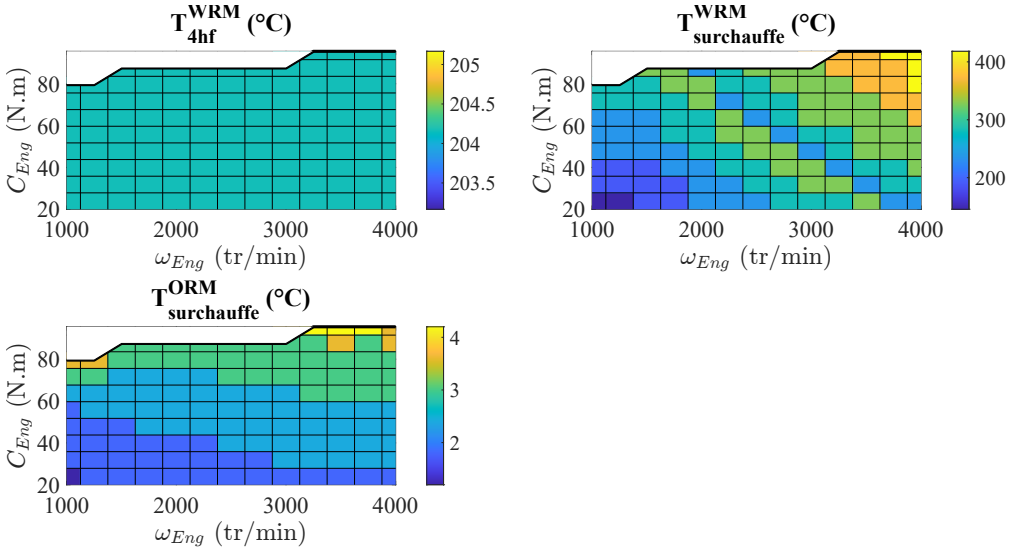

FIGURE 2.12 – Exemple de résultats des variables d'optimisation pour T_{ext} = 20°C et T_{LdR} = 90°C en fonction du couple et régime de rotation du moteur thermique.

La figure 2.13 illustre que les conditions de changement de phase s'adaptent également pour que la température de pincement soit respectée dans chaque échangeur. Il est possible d'observer ces conditions grâce aux températures de changement de phase montrées sur la droite de la figure 2.13. La température de condensation du cycle organique T^{ORM}_{cond} est la seule température à être constante sur toute la plage de fonctionnement. Cette température, égale à 26°C dans notre exemple, est la plus froide possible. En effet, c'est la température extérieure, qui est le fluide froid du condenseur de l'*ORM*, qui impose une température minimale à ne pas dépasser dans l'échangeur. L'optimisation estime qu'il n'y a aucun intérêt à augmenter cette température car cela reviendrait à diminuer la puissance récupérée sans raison valable. Les températures de condensation T^{WRM}_{cond} du *WRM* et d'évaporation T^{ORM}_{evap} du cycle organique évoluent relativement peu sur toute la plage de fonctionnement. En effet, la température de condensation est comprise entre 96 et 98 °C tandis que celle d'évaporation se situe entre 86 et 90°C. Ces températures s'adaptent aux flux de chaleur à libérer ou à absorber. Enfin, la température évoluant le plus est celle d'évaporation du *WRM* : T^{WRM}_{evap}. Cela s'explique par la grande variation de la température des gaz d'échappement dont l'eau récupère la puissance.

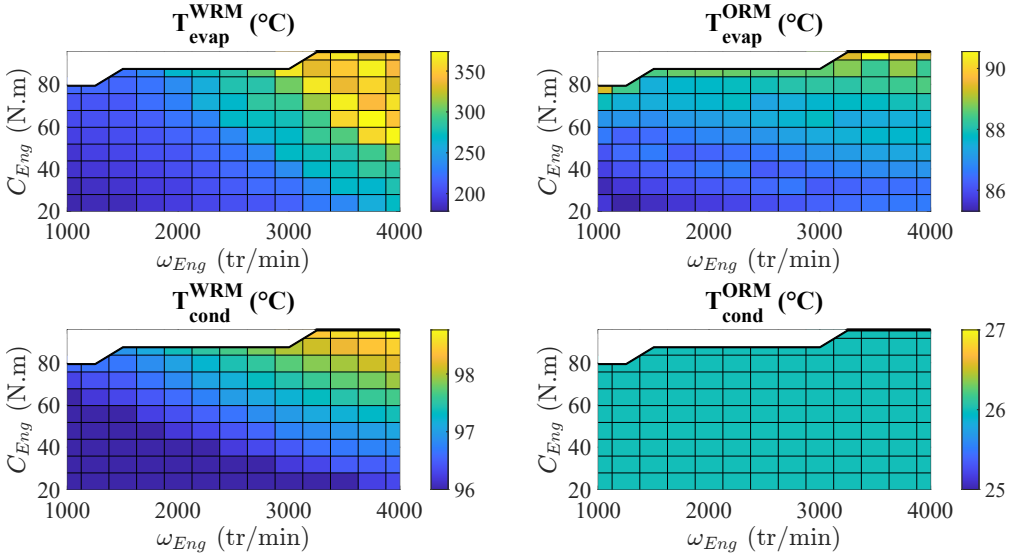

FIGURE 2.13 – Exemple de résultats des températures de changement de phase des quatre échangeurs pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ en fonction du couple et du régime de rotation du moteur thermique. $T^{WRM}_{surchauffe}$ décroît légèrement au centre de la plage possible du régime de rotation ω_{Eng} car pour maximiser la puissance totale récupérée, il est préférable de récupérer moins dans la première machine pour récupérer plus dans la seconde.

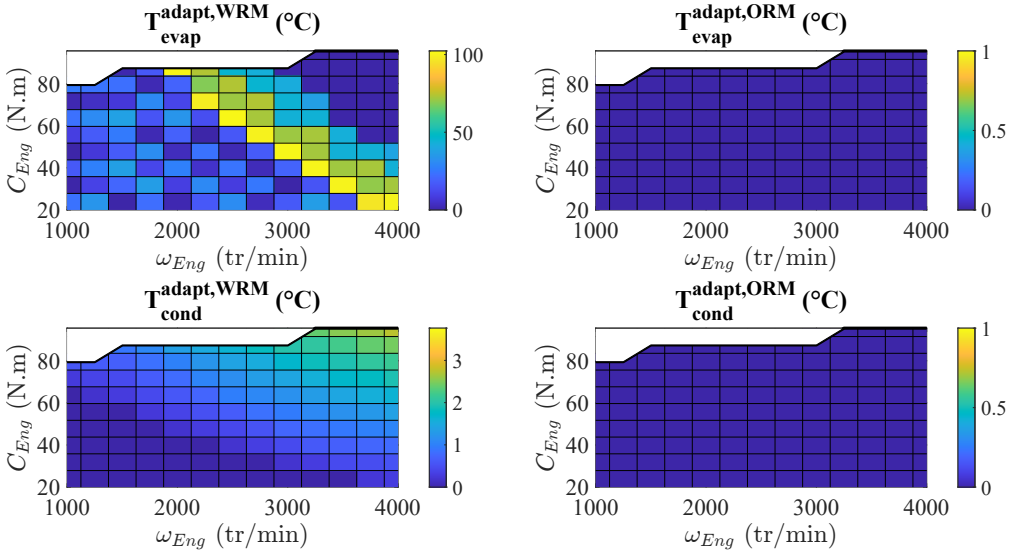

FIGURE 2.14 – Températures d'adaptation des quatre échangeurs. La ligne du haut présente les températures d'évaporation des deux machines et la ligne du bas les températures de condensation. La colonne de gauche présente le *WRM* et la colonne de droite l'*ORM*. Ces graphiques sont obtenus pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$. La température $T^{ORM}_{condenseur}$ est constante car la température extérieure l'est également.

La figure 2.14 illustre les quatre températures d'adaptation. Les échanges sont toujours

réalisable sans contraintes supplémentaire pour la machine organique, c'est pourquoi les températures d'adaptation sont nulles. Ce n'est cependant pas le cas pour la *WRM* car la température d'adaptation dans le condenseur de la *WRM* évolue linéairement en fonction de la chaleur dégagée par le moteur, ce qui n'est pas vrai pour le cas de l'évaporateur. Un certain nombre de température d'adaptation sont estimées à 100°C pour avoir une récupération de puissance maximale dans les conditions imposées.

La figure 2.15 explicite la puissance perdue ainsi que la puissance utile du moteur thermique utilisé. Les deux puissances évoluent globalement de la même manière : les pertes augmentent avec la puissance utile. Une partie des pertes va alimenter la structure de systèmes récupérateurs.

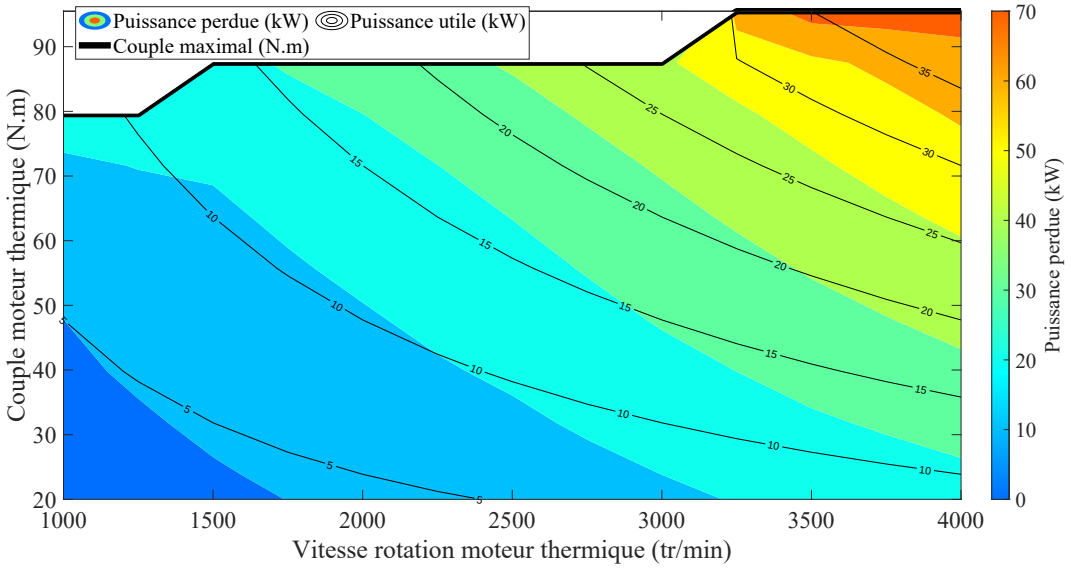

FIGURE 2.15 – Cartographie des pertes du moteur en couleur et cartographie de la puissance produite par le moteur en contour noir. La courbe de couple maximal du moteur en fonction du régime de rotation est représentée en noir.

Les figures 2.16 illustrent les puissances récupérées pour trois températures de liquide de refroidissement différentes et une température extérieure de 20°C, ainsi que les rendements de la structure de machines récupératrices.

Pour les températures de liquide de refroidissement à 80 et 90°C, les deux machines récupératrices fonctionnent et récupèrent une puissance allant jusqu'à 13 kW sur la zone de haute puissance libérée du moteur thermique. Cependant, pour une température de 110°C de liquide de refroidissement, on peut voir la puissance récupérée diminuer. Cela s'explique par la mise en arrêt de l'*ORM* car la température du liquide de refroidissement, qui est la source chaude de cette machine, est à trop haute température pour le fluide de travail car les limites du fluide organique sont atteintes. Au delà de ces températures, le fluide se dégrade de manière irréversible.

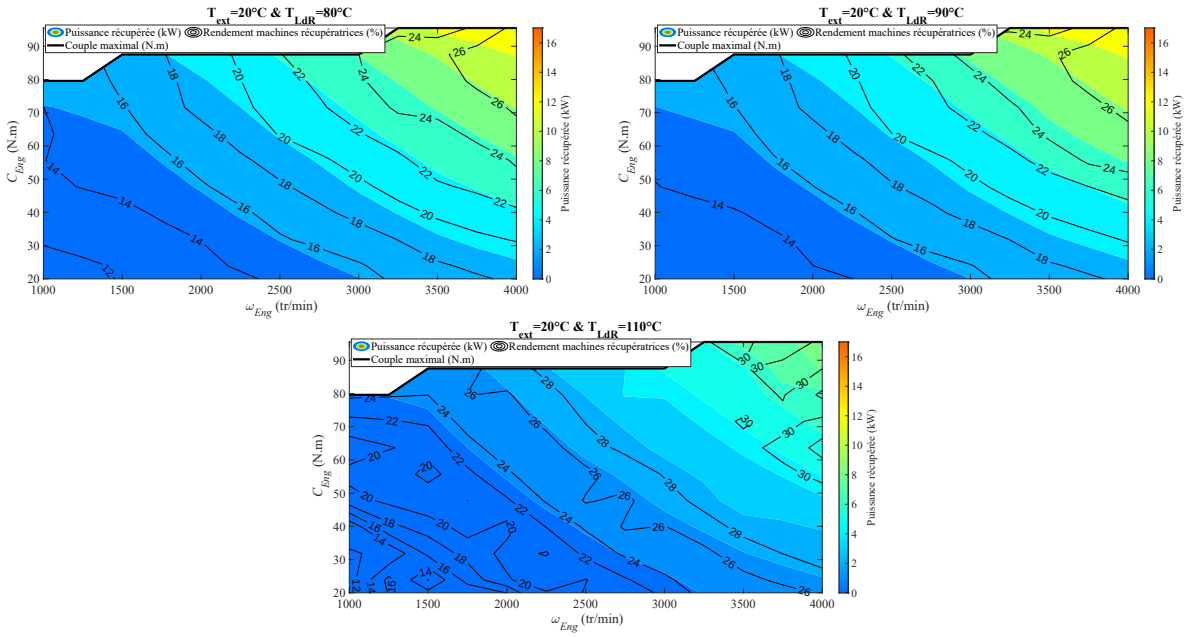

FIGURE 2.16 – Rendement et puissance récupérée par la structure de machine récupératrice pour trois température de liquide de refroidissement différentes.

2.2.4 Fonctionnement de la machine consommatrice

2.2.4.1 Description du fonctionnement du cycle de Rankine inversé

La climatisation représentée en bleu sur la figure 2.9 permet d'assurer le confort thermique des passagers au sein de la cabine d'habitacle. Elle est modélisée selon le cycle thermodynamique illustré en bleu sur la figure 2.17.

A partir du point 1, le fluide sous forme gazeuse est compressé jusqu'au point 2, passant de la pression Pr_1 à la pression Pr_2 avec une compression non isentropique. Le fluide est alors réchauffé par la compression. Le fluide est refroidi jusqu'au point 3 grâce au condenseur. Au point 3, le fluide est détendu dans un vase d'expansion jusqu'au point 4. Il est alors plus froid que l'air dans la cabine. Étant donné que la détente n'est pas motrice, contrairement aux cycles précédents, alors il est considéré que l'enthalpie au point 3 et au point 4 est la même : $h_4 = h_3$. On dit que la détente est isenthalpique. Afin d'évacuer la puissance de climatisation (*Heat Ventilation Air Conditioning* en anglais) $\dot{Q}_{HVAC}$ de l'habitacle, le débit du fluide de travail $\dot{m}_{wf}^{iORC}$ doit être adapté en conséquence. Enfin, le flux de chaleur à retirer de la cabine est échangé avec la machine consommatrice à la pression Pr_1. Le fluide de travail (*working fluid* soit *wf* en anglais) monte en température dans l'évaporateur du point 4 vers le point 1, soit de l'enthalpie h_4 à l'enthalpie h_1.

$$\dot{m}_{wf}^{iORC} = \frac{\dot{Q}_{HVAC}}{h_1^{iORC} - h_4^{iORC}} \tag{2.35}$$

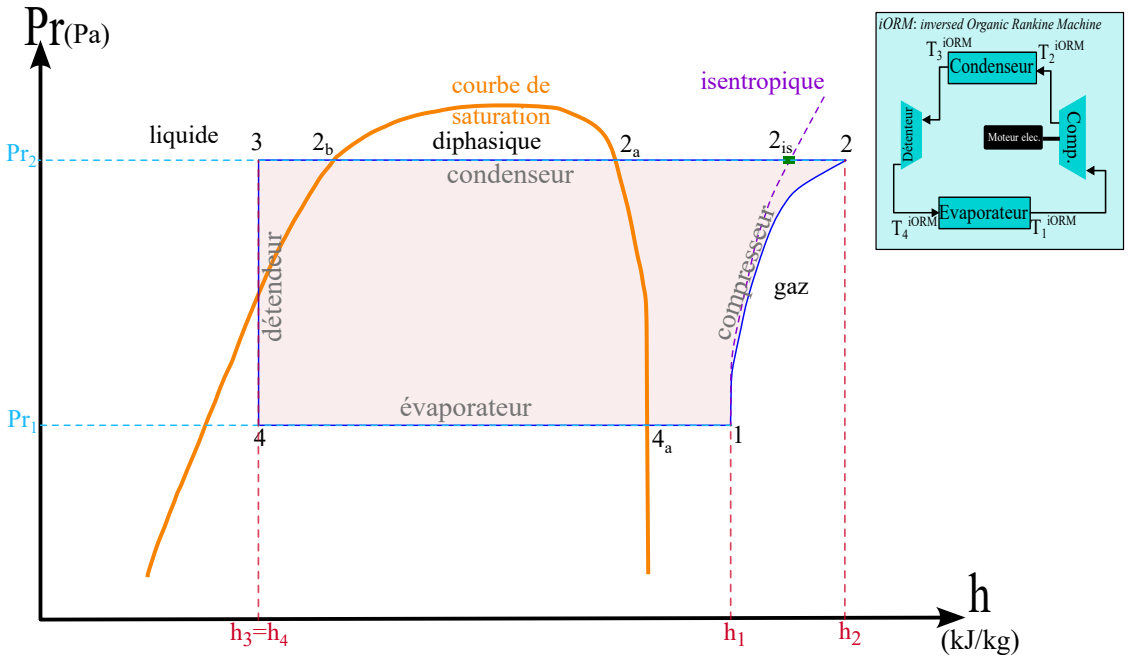

FIGURE 2.17 – Diagramme Pression enthalpie pour cycle de Rankine inversé.

La puissance $\dot{Q}_{HVAC}$ évolue en fonction de la température extérieure comme illustré dans [98]. Seule la puissance nécessaire pour refroidir l'habitacle est considérée. En effet, la puissance pour réchauffer l'habitacle est considérée fournie par les pertes du moteur thermique non récupérées par l'*ORM*. Il n'est alors pas nécessaire de fournir une puissance électrique pour chauffer l'habitacle. C'est la puissance électrique fournie au compresseur $\dot{W}_{comp}^{iORC}$ qui permet de réaliser le cycle de Rankine inversé.

$$\dot{W}_{comp}^{iORC} = \dot{m}_{wf}^{iORC}.(h_2^{iORC} - h_1^{iORC}) \tag{2.36}$$

TABLE 2.3 – Table explicative de l'ordre à suivre pour calculer les variables thermodynamiques pour le cycle de Rankine inversé. Les variables dont les cellules sont de même couleur (excepté blanc) sont identiques.

	T (K)	Pr (Pa)	h (kJ/kg)	s (kJ/(kg.K))
1	A (2.38)	A (2.15)	B (2.17)	C (2.17)
2is		C (2.16)	D (2.17)	C (2.17)
2	F (2.17)	C (2.16)	E (2.41)	
2a	B (2.40)	C (2.16)	D (2.17)	
2b	B (2.40)	C (2.16)	D (2.17)	
3	A (2.39)	C (2.16)	D (2.17)	
4	E (2.17)	A (2.15)	D (2.17)	
4a		A (2.15)	A (2.17)	

Enfin, la machine doit évacuer le flux de chaleur de l'habitacle vers l'extérieur. Le flux de

chaleur rejeté à l'extérieur $\dot{Q}_{out}^{iORC}$ est calculé suivant la relation (2.37).

$$\dot{Q}_{out}^{iORC} = \dot{m}_{wf}^{iORC}.(h_3^{iORC} - h_2^{iORC}) \tag{2.37}$$

La température d'évaporation est imposée dans notre modèle pour effectuer une régulation de température à 20°C. On a alors $T_{evap}^{iORC} = 5$°C. Étant donné que ce point appartient nécessairement à la courbe d'ébullition (courbe de saturation du côté vapeur), alors il est possible d'obtenir la pression ainsi que l'enthalpie à ce point grâce à l'équation (2.17). La température du fluide de travail dans l'évaporateur est définie comme :

$$T_1^{iORC} = T_{evap}^{iORC} + T_{surchauffe}^{iORC} \tag{2.38}$$

Pour obtenir l'entropie s_1^{iORC} au point 1, l'utilisation de l'enthalpie a été préférée à l'utilisation de la pression dans l'équation (2.17) pour éviter des problèmes de lecture dans les tables. C'est pourquoi l'entropie est obtenue après l'enthalpie au point 1.

En parallèle, il est également possible de calculer la température au point 3, qui est la température la plus basse possible du condenseur imposée par une température de pincement minimum, $T_{pinch}^{iORC} = 5$°C. La température au point 3 s'obtient alors par la relation suivante :

$$T_3^{iORC} = T_{ext} + T_{pinch}^{iORC} \tag{2.39}$$

Il est possible de calculer la température de condensation à partir de la température au point 3 et de la température de sous-refroidissement :

$$T_{cond}^{iORC} = T_3^{iORC} + T_{sous-refroidissement}^{iORC} \tag{2.40}$$

La température de condensation (2.40) permet d'obtenir la pression de condensation avec l'équation (2.16). Pour obtenir l'enthalpie au point 2, une compression non-isentropique est considérée avec un rendement de compresseur $\eta_{compresseur} = 0.85$.

$$h_2^{iORC} = h_1^{iORC} + \frac{(h_{2is}^{iORC} - h_1^{iORC})}{\eta_{compresseur}} \tag{2.41}$$

2.2.4.2 Optimisation statique de la climatisation

Une optimisation statique de la climatisation basée sur la méthode exhaustive est également faite. Les paramètres d'optimisation sont la température de surchauffe et celle de sous refroidissement. L'objectif est de minimiser la puissance électrique à fournir au compresseur. Étant donné que ce cycle thermodynamique ne dépend que de l'air extérieur (celle de l'habitacle est fixée à 20°C), alors l'optimisation est réalisée pour chaque température extérieure considérée :

— La température de sous refroidissement, $T_{sous-refroidissement}^{iORC}$, varie de 0 à 5°C.

— La température de surchauffe, $T^{iORC}_{surchauffe}$, varie de 0 à 5°C.

Les courbes obtenues du travail d'optimisation illustrées sur la figure 2.18 relatent qu'en dessous de 0°C, il n'y a pas besoin de refroidir la cabine. Au delà de cette température, il est nécessaire d'assurer un refroidissement de la cabine qui s'illustre sur la figure 2.18 par la puissance compresseur en rouge. Plus la température est élevée, plus la puissance à fournir est importante. Cependant, on remarque que les températures d'optimisation (sous refroidissement en bleu et surchauffe en noir) ne varient pas en fonction de la température extérieure. Les résultats connus sont alors obtenus : la première température est toujours au minimum possible tandis que la seconde est la plus chaude possible. On retrouve alors des résultats attendus, soit le plus froid possible pour la température de sous-refroidissement, soit le plus chaud possible pour la température de surchauffe.

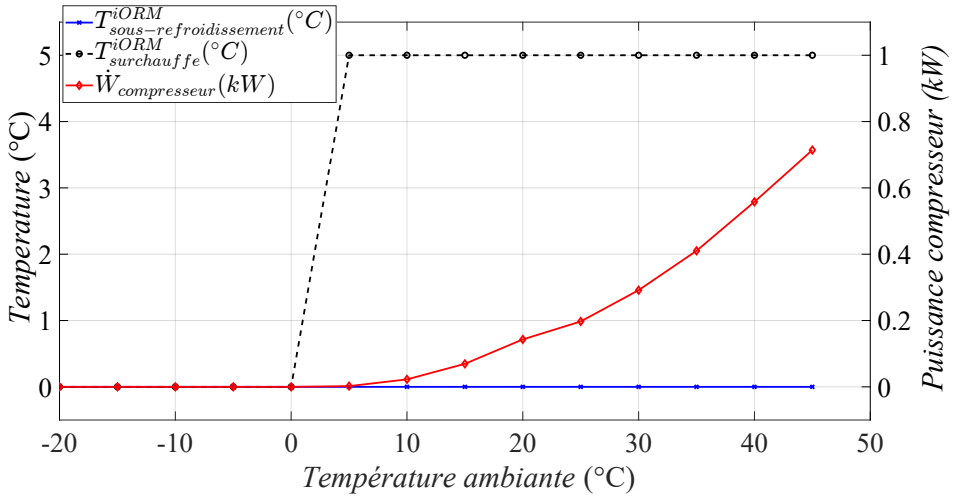

FIGURE 2.18 – Minimisation de la puissance compresseur en optimisant les températures de surchauffe et de sous-refroidissement en fonction de la température extérieure.

2.3 Incorporation de la structure de systèmes récupérateurs dans le véhicule hybride électrique série

2.3.1 Comparaison des groupes motopropulseurs et scénarios utilisés

Trois groupes moto propulseurs (GMP) véhicules seront utilisés pour cette étude. Le premier correspond au GMP original du véhicule hybride électrique série illustré sur a) de la figure 2.1, appelé GMP original par la suite. Le second GMP reprend la même architecture véhicule que le premier mais le moteur thermique est remplacé par une version améliorée du moteur initial, nommé GMP amélioré par la suite. Cette amélioration est une estimation d'un moteur optimisé pour une utilisation en mode *range extender* d'un véhicule hybride électrique. Dans les faits, cette estimation se résume à la réduction des valeurs de consommation spécifique du moteur de 10%, soit une amélioration globale du rendement normalisé

du GMP de 11%. Cette estimation de la réduction de consommation du moteur thermique semble cohérente vis-à-vis de la littérature [99] [100]. Finalement, le dernier GMP est composé du moteur thermique initial avec la structure de systèmes récupérateurs comme l'architecture véhicule exposée en figure 2.2, nommée GMP avec *WHRS* par la suite. Les rendements normalisés des trois groupes moto propulseurs sont affichés en figure 2.19. Sur la gauche, la cartographie d'efficacité du GMP original, au centre une des cartographies d'efficacité du GMP avec *WHRS* et sur la droite la cartographie du GMP amélioré. Les points de meilleur rendement du GMP original et amélioré se trouvent dans une zone relativement restreinte entre 70 et 80 N.m et 2000 et 2500 tr/min. Le GMP avec *WHRS* ne possède pas ces points de meilleur rendement là où sont ceux du moteur thermique. En effet, on peut voir que grâce à la récupération des pertes émises par le moteur thermique, le rendement du GMP avec *WHRS* a tendance à augmenter avec la puissance mécanique produite par le moteur thermique parce qu'il y a plus de pertes à revaloriser. Il est possible de constater que le rendement de ce groupe motopropulseur a un rendement normalisé supérieur au GMP amélioré sur toute la plage de fonctionnement du moteur thermique et va jusqu'à 31 %. La référence pour la normalisation du rendement de tous les GMP est le point de meilleur fonctionnement du GMP original.

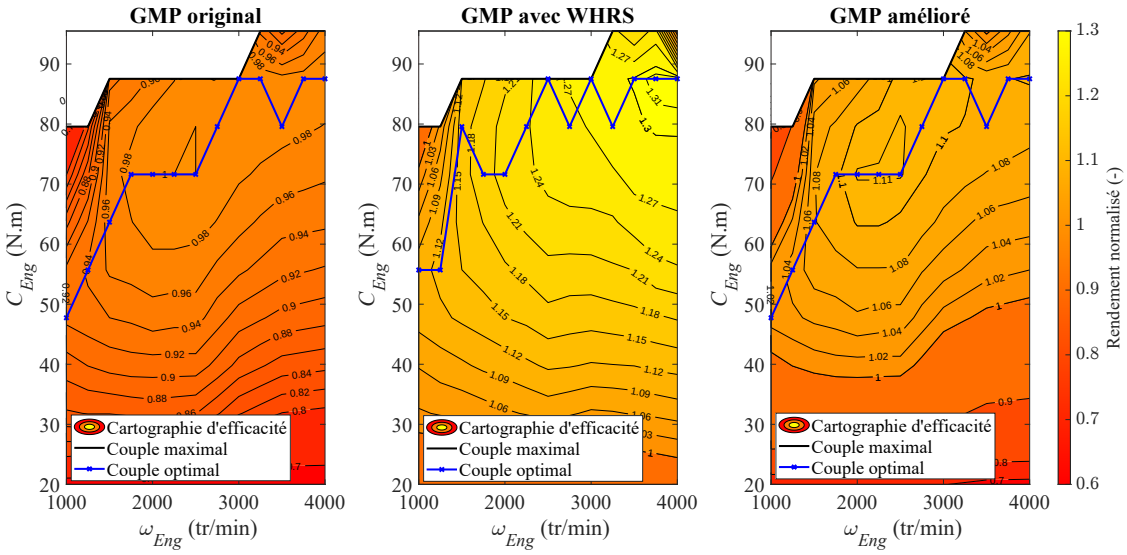

FIGURE 2.19 – Cartographie d'efficacité moteur original sur la gauche, du groupe motopropulseur avec les machines récupératrices pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ au centre et cartographie d'efficacité moteur amélioré sur la droite. Figure en noir la courbe de couple maximal du moteur en fonction du régime de rotation. La puissance utile dans chacun des cas est celle developpée par le moteur thermique (cas de gauche et droite) ainsi que la puissance récupérée par le *WHRS* pour le cas central.

Le véhicule doit réaliser un cycle de conduite représentatif de son utilisation pour estimer sa consommation de carburant. Deux cycles ont été utilisés. Le premier est le *WLTC* qui est le cycle d'homologation des véhicules en Europe comme décrit en dans l'avant-propos. Le second cycle représente un profil de vitesse sur autoroute. La vitesse sera donc majoritairement constante sur le cycle à haute vitesse. Chaque groupe motopropulseur sera alors testé sur les

deux cycles de vitesse.

En plus de la puissance nécessaire à la traction du véhicule et au compresseur de climatisation, une consommation de puissance électrique constante représentant les accessoires électriques est également prise en compte [101].

2.3.2 Problème de Commande Optimale

Pour réaliser cette évaluation, un problème de contrôle optimal sous sa forme discrète primale est écrit comme dans le chapitre 1. Les commandes du système $\boldsymbol{u}$ sont le couple C_{Eng} et le régime de rotation ω_{Eng} du moteur thermique tandis que l'état considéré est l'état de charge de la batterie SOC_{Bat}.

$$\boldsymbol{u} = \begin{bmatrix} C_{Eng} \\ \omega_{Eng} \end{bmatrix} \tag{2.42}$$

$$\boldsymbol{x} = \begin{bmatrix} SOC_{Bat} \end{bmatrix} \tag{2.43}$$

Le coût instantané du problème est le débit du carburant injecté dans le moteur qui dépend du régime et couple moteur :

$$L(x_k, \boldsymbol{u}_k) = \dot{m}_{fuel}(C_{Eng_k}, \omega_{Eng_k}) \tag{2.44}$$

L'écriture du problème de contrôle optimal (1.5) est rappelée ici :

$$\begin{cases} \min\limits_{\boldsymbol{u}} J(k, \boldsymbol{x}, \boldsymbol{u}) = \sum_{k=0}^{N-1} \dot{m}_{fuel_k}(C_{Eng_k}, \omega_{Eng_k}).\Delta t + \Phi\left(\boldsymbol{x}_N\right) \\ SOC_{Bat_{k+1}} = SOC_{Bat_k} + \frac{P_{Bat_k}(\boldsymbol{u})}{A_{max}.V_{Bat_k}(SOC_{Bat_k})}.\Delta t \\ \boldsymbol{x}_k \in \boldsymbol{X}_k, \forall k \geq 0 \\ \boldsymbol{u}_k \in \boldsymbol{U}_k, \forall k \geq 0 \end{cases} \tag{2.45}$$

L'expression de la puissance batterie P_{Bat} dans le domaine continue (2.6) est discrétisée en (2.46) pour être utilisée et la tension sera évaluée grâce au modèle illustré sur la figure 2.4. Le pas de temps du problème est fixé à une seconde ($\Delta t = 1$s) et la capacité de la batterie est de soixante ampère heure ($A_{max} = 60$ A.h).

$$P_{Bat_k} = \frac{P_{Mel_k}}{\eta_{Mel}(C_{Mel_k}, \omega_{Mel_k})} + \dot{W}_{comp_k}(T_{ext}) + P_{Aux} - P_{Eng_k}(\boldsymbol{u}).\eta_{Gen} \tag{2.46}$$

La programmation dynamique sera utilisée pour résoudre ce problème. Le problème ne possédant qu'un seul état, alors il est possible d'utiliser la méthode des *boundary lines* [55] décrit en section 1. L'outil de programmation dynamique de l'ETH Zurich [57] a été utilisé.

De plus, pour comparer équitablement les solutions, un bilan batterie nul est souhaité. Cela permet ainsi d'estimer la consommation d'énergie réelle tout en conservant la même

quantité d'énergie contenue dans la batterie. Il est alors imposé que l'état de charge final soit égal à celui initial : $SOC_{Bat_N} = SOC_{Bat_0}$. Cette condition finale est utilisée pour construire les *boundary lines*. Dans tous les cas d'étude, l'état de charge de la batterie initial sera fixé à $SOC_{Bat_0} = 50\%$.

$$\begin{cases} \Phi\left(SOC_{Bat_N} = SOC_{Bat_0}\right) = 0 \\ \Phi\left(SOC_{Bat_N} \neq SOC_{Bat_0}\right) = 1000 \end{cases} \tag{2.47}$$

Le maillage de la programmation dynamique pour les deux commandes et l'état sont les suivants :

— 38 points pour la vitesse de rotation du moteur thermique entre 1000 et 4000 tr/min.
— 42 points pour le couple du moteur thermique entre 20 et 95 N.m.
— 51 points pour l'état de charge de la batterie entre 30 et 70 %.

En plus de ce maillage uniforme, la commande particulière de puissance nulle est également considérée. Cette commande permet l'arrêt du moteur thermique et donc une consommation de carburant instantanée nulle.

A noter que des maillages plus fins ont été appliqués au problème, mais les résultats obtenus n'étaient pas significativement meilleurs. C'est pourquoi, grâce à l'utilisation des *boundary lines*, les calculs sont précis tout en étant acceptables d'un point de vue du temps de calcul. Ce sont les *boundary lines* qui permettent l'utilisation d'un maillage si peu fin, car, sans cet outil, l'algorithme n'arrive pas à trouver de solution à cause de la propagation d'infini dans la matrice de coût optimal, notamment à cause de la contrainte finale comme le présente le chapitre 1. Une programmation dynamique avec le maillage présenté est calculée entre 70 secondes. Au vu du nombre de cas à étudier, un temps de calcul plus long n'est pas intéressant s'il n'y a pas d'amélioration majeure.

2.4 Résultats

2.4.1 1er cycle : *Worldwide harmonized Light vehicles Test Cycle*

La table 2.4 expose les résultats de la consommation de carburant du véhicule avec les trois groupes motopropulseurs. Les consommations du GMP amélioré et avec *WHRS* sont évaluées par rapport à la consommation du GMP original à la même température. La température du GMP original augmente avec la température extérieure car la puissance requise par le compresseur de climatisation augmente comme en témoigne la figure 2.18. La solution du GMP original avec la température extérieure à 20°C utilisée pour normaliser les consommations du GMP original. Les réductions de consommations affichées en pourcentage sont quant à elles évaluées par rapport à la consommation du GMP à la même température extérieure.

D'un côté, les consommations relatives du GMP amélioré sont inférieures de 10% à celles du GMP original, ce qui correspond à l'amélioration de l'efficacité du moteur thermique. D'un autre côté, les consommations relatives du GMP avec *WHRS* sont inférieures à celles du GMP amélioré. Toutefois, la température de régulation du liquide de refroidissement (LdR) pos-

TABLE 2.4 – Consommation de carburant normalisée des trois GMP en fonction de la température extérieure et du liquide de refroidissement pour un *WLTC*.

		GMP avec *WHRS*				GMP original	GMP amélioré
		Température LdR (°C)					
		80	90	100	110		
Température extérieure (°C)	-20	-25%	-25%	-23%	-14%	0,98	-10%
	-15	-24%	-24%	-23%	-14%	0,98	-10%
	-10	-24%	-24%	-22%	-14%	0,98	-10%
	-5	-23%	-23%	-22%	-14%	0,98	-10%
	0	-23%	-23%	-21%	-14%	0,98	-10%
	5	-22%	-22%	-21%	-14%	0,98	-10%
	10	-22%	-22%	-20%	-14%	0,99	-10%
	15	-21%	-21%	-20%	-14%	0,99	-10%
	20	-21%	-21%	-19%	-14%	1.00	-10%
	25	-20%	-21%	-19%	-14%	1.01	-10%
	30	-20%	-20%	-18%	-14%	1.02	-10%
	35	-19%	-20%	-18%	-14%	1.03	-10%
	40	-19%	-19%	-18%	-14%	1.05	-10%
	45	-18%	-19%	-17%	-14%	1.07	-10%

sède un impact sur la consommation. Cette variation de consommation de carburant peut être directement liée à la variation de la puissance récupérée par la structure de machines récupératrices. Plus la température du liquide de refroidissement est élevée, plus la consommation de carburant est élevée. Le rendement des machines s'effondre entre 100 et 110 °C. Le résultat de la réduction de puissance récupérée par la structure de machines récupératrices à haute température de liquide de refroidissement illustré en figure se manifeste dans les résultats de consommation du véhicule à cette température de liquide de refroidissement. C'est pourquoi la configuration du GMP avec *WHRS* pour une température de liquide de refroidissement à 110°C est la moins performante des quatre températures testée. Cependant, même cette configuration reste meilleure que le GMP amélioré, ce qui montre l'intérêt de la solution technique.

La cartographie de rendement du GMP original avec les points de fonctionnement du moteur thermique sont présentés sur la gauche de la figure 2.20, tandis que la cartographie du GMP avec *WHRS* avec les points de fonctionnement du moteur thermique sont présentés sur la droite. La simulation est faite pour une température de liquide de refroidissement à 90°C et une température extérieure de 20°C. Plus un point de fonctionnement est large, plus le moteur a passé de temps dans cette région de sa plage de fonctionnement. Les deux solutions passent la majeure partie du trajet arrêté (environ 70% du temps). Les points de fonctionnement des 30 % restant réalisés par les deux GMP sont situés proche de la courbe de couple optimal en bleu. Toutefois, les points de fonctionnement privilégiés par le GMP avec *WHRS* sont à des puissances plus élevées comparé au GMP original. Cela s'explique par la différence de rendement qui est à une puissance plus faible pour le GMP original alors que la zone de rendement optimal du GMP avec *WHRS* correspond à la zone de haute puissance.

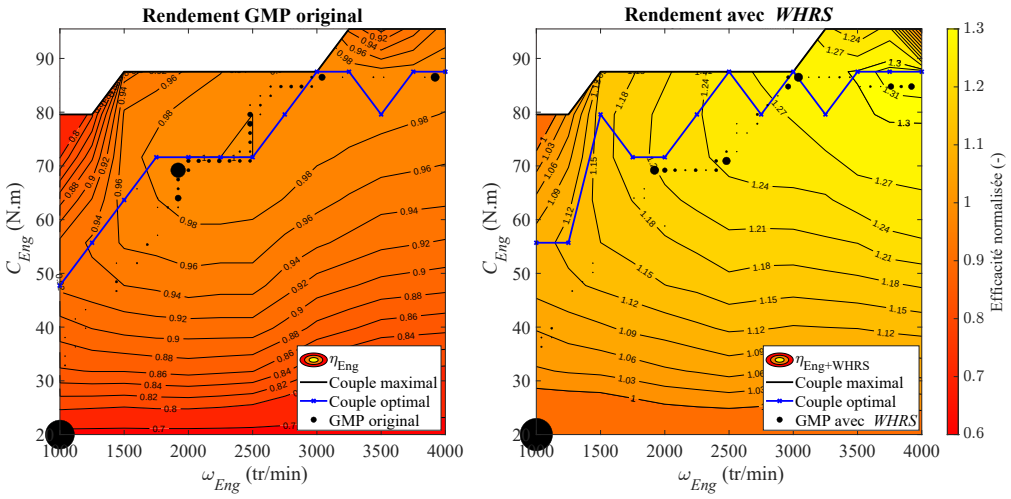

FIGURE 2.20 – Efficacité du GMP original en fonction du régime et couple moteur sur la gauche et efficacité du GMP avec *WHRS* également en fonction du régime et couple moteur sur la droite pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ pour un *WLTC*. Les points de fonctionnement des deux GMP sont en noir sur leur cartographie respective. Les points de fonctionnement placés à 20 N.m et 1000 tr/min présentent l'arrêt du moteur.

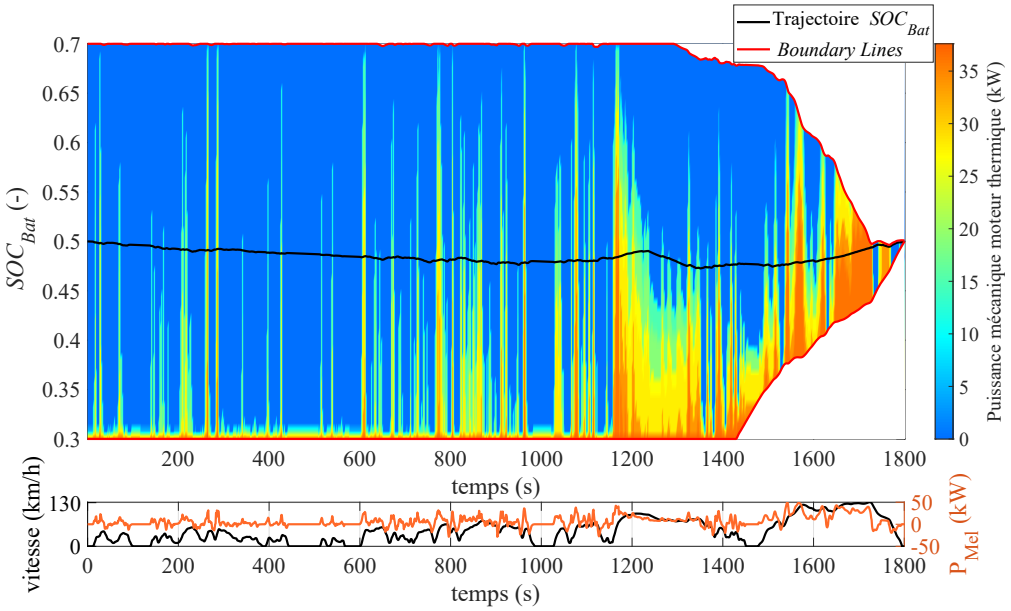

FIGURE 2.21 – Matrice de puissance optimale en fonction de l'état de charge de la batterie calculée par la *DP* pour le GMP avec *WHRS*, évolution de l'état de charge de la batterie dans le temps en noir pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ et profil de vitesse du *WLTC* en noir.

Il est également possible de voir une représentation des matrices de commandes optimales en fonction de l'état de charge de la batterie et du temps sur la figure 2.21 pour le GMP avec *WHRS*. L'utilisation des *boundary lines* est visible car une partie des états est exclue car

ils ne permettent pas de respecter la contrainte finale (équation (2.47)). La courbe en noir représente la trajectoire de l'état de charge de la batterie dans le temps tandis que la puissance du moteur thermique est représentée par le dégradé de couleur. Au plus la couleur est chaude, au plus la puissance mécanique est importante. Cette cartographie provient des matrices de commandes optimales calculées durant le *backward* de la programmation dynamique. Sur l'intervalle regardé de *SOC*, on peut noter que les commandes sont les mêmes car la puissance est la même. Les événements de grandes puissances pour le moteur interviennent lorsque la demande de puissance par les roues sont également les plus importantes ; ce qui peut paraître non naturel du fait que le moteur thermique ne propulse pas le véhicule. Ceux-ci s'expliquent pour permettre le respect des contraintes tout en maximisant le rendement de la chaîne de traction. On peut également souligner que le moteur est arrêté majoritairement lorsque le véhicule est à basse ou moyenne vitesses, voire à l'arrêt.

2.4.2 2nd cycle : profil de vitesse autoroute

Le profil de vitesse autoroute considéré dans cette étude peut être aperçu au bas de la figure 2.23. Il est composé d'une accélération constante de 0.5 m/s^2 jusqu'à ce que la vitesse du véhicule atteigne 130 km/h. A ce moment là, la vitesse est conservée constante jusqu'à la fin du trajet. Les résultats des différentes technologies considérées peuvent être observés dans le tableau 2.5. Les résultats obtenus ont les mêmes tendances que ceux obtenus pour le *WLTC*.

Table 2.5 – Consommation de carburant normalisé des trois GMP en fonction de la température extérieure et du liquide de refroidissement pour le profil de vitesse autoroutier.

		GMP avec *WHRS*				GMP original	GMP amélioré
		Température LdR (°C)					
		80	90	100	110		
Température extérieure (°C)	-20	-39%	-38%	-36%	-24%	1,00	-10%
	-15	-38%	-38%	-36%	-24%	1,00	-10%
	-10	-37%	-37%	-35%	-24%	1,00	-10%
	-5	-37%	-36%	-34%	-24%	1,00	-10%
	0	-36%	-36%	-34%	-24%	1,00	-10%
	5	-35%	-35%	-33%	-24%	1,00	-10%
	10	-35%	-34%	-32%	-24%	1,00	-10%
	15	-34%	-34%	-32%	-24%	1,00	-10%
	20	-33%	-33%	-31%	-24%	1.00	-10%
	25	-32%	-32%	-30%	-24%	1.00	-10%
	30	-31%	-31%	-29%	-24%	1.00	-10%
	35	-30%	-31%	-28%	-24%	1.01	-10%
	40	-29%	-30%	-27%	-24%	1.01	-10%
	45	-28%	-29%	-26%	-24%	1.02	-10%

La table 2.5 expose les résultats de la consommation de carburant du véhicule avec les trois groupes motopropulseurs pour le profil de vitesse autoroute. La table est construite de

la même manière que la précédente. Étant donné que la demande en énergie de ce cycle est plus importante que le précédent, les consommations du GMP original, toujours normalisées par celle à 20°C, évoluent moins en fonction de la température extérieure. Les gains du GMP amélioré sont les mêmes que pour le cycle *WLTC*, ce qui n'est pas le cas du GMP avec *WHRS*. En effet, la réduction de consommation est encore plus importante pour ce GMP. Cela induit alors que la technologie étudiée est d'autant plus intéressante pour les trajets qui ont une demande de puissance plus importante.

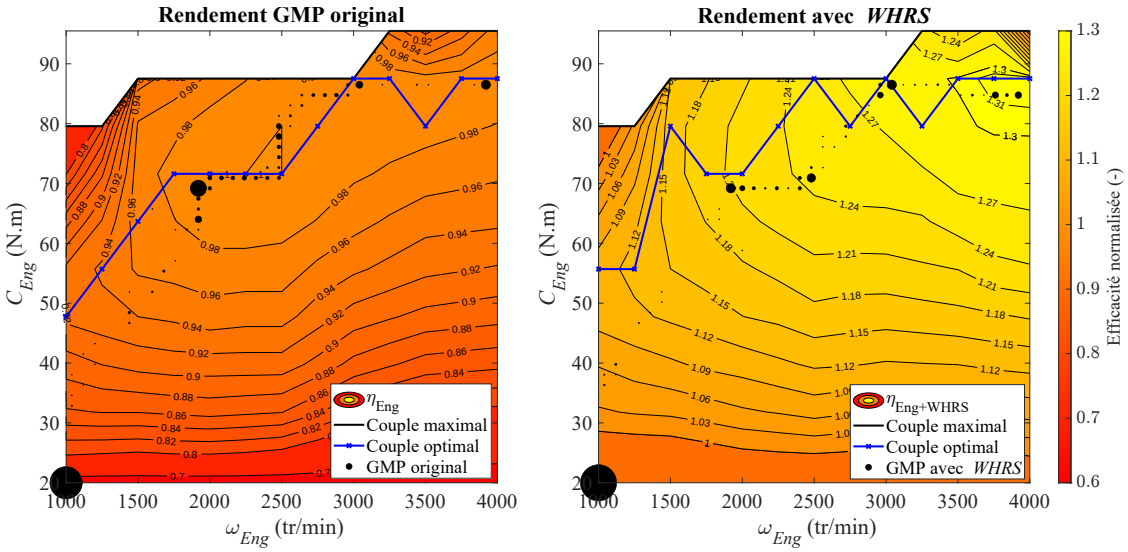

FIGURE 2.22 – Efficacité du GMP initial en fonction du régime et couple moteur sur la gauche et efficacité du GMP équipé des machines récupératrices également en fonction du régime et couple moteur sur la droite pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ pour le cycle autoroutier. Les points de fonctionnement des deux GMP sont en noir sur leur cartographie respective. Les points de fonctionnement placés à 20 N.m et 1000 tr/min présentent l'arrêt du moteur.

La figure 2.22 illustre les points de fonctionnement du moteur thermique pour le GMP original et le GMP avec *WHRS*. Il est possible d'observer que le moteur thermique fonctionne majoritairement sur des zones de hautes et très hautes puissances afin de répondre à la demande énergétique du véhicule. En effet, la demande énergétique du cycle autoroutier est environ quatre fois plus importante que celle du cycle *WLTC* ; ce qui implique des phases d'arrêt du moteur thermique très faibles mais également un repositionnement des points de fonctionnement du moteur thermique des deux groupes motopropulseurs afin de répondre à la demande énergétique. Ce repositionnement des points de fonctionnement, toujours le long de la courbe de couple optimal, est à l'avantage du GMP avec *WHRS* car le rendement de ce dernier est beaucoup plus intéressant sur cette plage de fonctionnement (différence de presque 30% de rendement normalisé), ce qui permet une réduction moyenne de la consommation d'environ 30%.

La figure 2.23 dévoile également la puissance mécanique délivrée par le moteur thermique en fonction de l'état de charge de la batterie en couleur pour le GMP avec *WHRS*. La trajectoire de l'état de charge de la batterie sur la partie supérieure de la figure nous indique alors

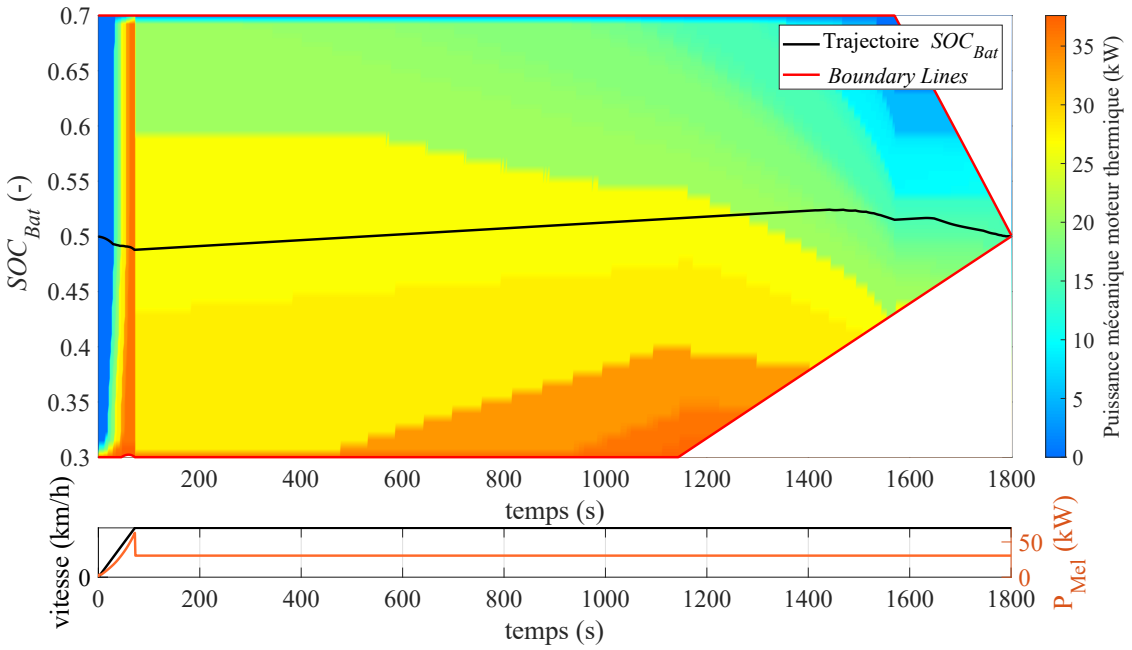

FIGURE 2.23 – Matrice de puissance optimale en fonction de l'état de charge de la batterie calculée par la *DP* pour le GMP avec *WHRS*, évolution de l'état de charge de la batterie dans le temps en noir pour $T_{ext} = 20°C$ et $T_{LdR} = 90°C$ et profil de vitesse autoroutier en noir.

quelle est la puissance mécanique moteur délivrée par ce dernier. Il apparaît alors qu'avec les conditions initiales imposées sur la batterie, le moteur est constamment sollicité à une puissance élevée afin de réaliser l'objectif de bilan batterie nul. On peut cependant constater un comportement particulier adopté par la loi de gestion de l'énergie déterminé par l'algorithme de programmation dynamique. En effet, lors de la phase de vitesse constante, l'algorithme produit plus de puissance au début du cycle pour maximiser le rendement de la chaîne de traction du véhicule. La cartographie de couleur qui représente la puissance mécanique du moteur thermique en fonction de l'état de charge de la batterie et du temps illustre bien ce phénomène.

2.5 Conclusion et perspectives

Ce chapitre a évalué un système technique basé sur des machines de Rankine qui permettent de récupérer la puissance perdue sous forme de chaleur du moteur thermique pour améliorer le rendement global du groupe motopropulseur. La prise en compte de la consommation du compresseur climatisation est également faite pour avoir des résultats de simulation numérique plus proche d'un fonctionnement réel du véhicule.

Une première partie est employée pour décrire la modélisation du véhicule considéré pour mener l'étude. C'est un véhicule hybride électrique série donc le moteur thermique n'est pas

relié aux roues du véhicule, c'est uniquement la machine électrique qui assure la traction du véhicule. Cette architecture permet de considérer une revalorisation des pertes du moteur thermique via des machines thermodynamiques.

C'est pourquoi, une seconde partie est consacrée à la modélisation des machines de Rankine qui permettent la revalorisation des pertes thermiques. Deux sources de chaleur sont considérées dans un moteur thermique : les gaz d'échappement et le liquide de refroidissement. Deux machines sont donc modélisées, l'une récupérant la puissance disponible des gaz d'échappement avec une machine de Rankine fonctionnant avec de l'eau tandis que la seconde récupère la puissance disponible du liquide de refroidissement avec une machine de Rankine fonctionnant avec un fluide organique, le r1234yf. L'agencement des deux machines de Rankine donne lieu à une structure de machines récupératrices.

Le rendement du groupe motopropulseur du moteur avec les machines récupératrices est donc déterminé comme la puissance générée par le moteur thermique plus la puissance récupérée par la structure de machines récupératrices divisé par la puissance libérée par la combustion du carburant. Ce groupe motopropulseur est comparé au groupe motopropulseur original et à un troisième groupe motopropulseur composé d'un moteur thermique amélioré par rapport à celui du GMP original. Le moteur de ce dernier groupe motopropulseur est une amélioration du moteur original qui pourrait être obtenue par une optimisation du cycle thermodynamique de combustion.

Les résultats obtenus par programmation dynamique permettent d'évaluer avec équité les trois solutions techniques proposées par les différents groupes motopropulseurs. En effet, avec cette approche, nous sommes assurés d'avoir à chaque simulation la meilleure solution possible. La programmation dynamique est utilisée ici car les non linéarités du système sont facilement prisent en compte. Les résultats obtenus par simulation sur le cycle *WLTC* et le cycle autoroutier montrent un réel intérêt des machines récupératrices avec un gain moyen de 20% par rapport à la technologie actuelle et de 10% par rapport au groupe motopropulseur équipé du moteur amélioré.

Une autre perspective de ce chapitre est de considérer un jeu de quatre échangeurs fixes. En effet, étant donné que les configurations varient en fonction des différents éléments extérieurs qui jouent sur le rendement des machines récupératrices, il y a nécessairement une configuration pour chaque échangeur qui soit meilleure que les autres pour minimiser en moyenne la consommation de carburant. Cette étude sur le dimensionnement de l'architecture en général pourrait également être appuyée par une étude d'intérêt de l'implémentation des véhicules équipés de cette technologie. Par exemple, si on considère que cette technologie ne sera pas vendue dans les milieux trop chauds en raison d'un rendement moins intéressant, alors on pourrait pondérer les résultats obtenus pour ces conditions climatiques.

Enfin, pour avoir une étude plus approfondie, il faudrait prendre en compte plus de dynamiques. La dynamique de la température du moteur thermique qui est considérée constante dans cette étude est la dynamique à ajouter en priorité dans cette étude. En effet, une variation des consommations de carburant en fonction de la température de liquide de refroidissement qui est directement liée à la température du moteur thermique peut être observée. Les dyna-

miques de température au sein des machines récupératrices sont également très importantes pour se rapprocher de la réalité. Par la suite, d'autres dynamiques comme la température ou le vieillissement de la batterie pourraient être ajoutées afin d'avoir une modélisation plus proche de la réalité mais également de pouvoir faire une optimisation multi-critères.

Une prise en compte des grandeurs physiques liées aux machines récupératrices est également envisageable. Afin de rendre cela possible, les chapitres 3 et 4 visent à réduire la complexité de calcul des méthodologies de contrôle optimal et plus particulièrement de la programmation dynamique.

Chapitre 3

Optimisation des résultats de la Programmation Dynamique

Sommaire

Il a pu être constaté au cours du chapitre 1 et 2 que l'exploitation de l'amélioration *boundary lines* de la programmation dynamique permet de grandement améliorer la qualité du résultat obtenu. Cela provient du fait que la délimitation entre le domaine faisable et infaisable est parfaitement connue, ce qui évite la contamination de la matrice de coût optimal liée aux pénalités sur les contraintes finales. En parallèle, le chapitre 2 met également en lumière l'intérêt des modèles à un nombre de variables d'état et/ou de commande important pour que les modèles soient le plus réaliste possible. C'est pourquoi, la généralisation des *boundary lines* à n états est identifiée comme un axe de développement prometteur pour répondre à la problématique posée.

Dans la littérature, il est possible de retrouver les travaux de [102] qui ont développé une méthode proche des *boundary lines* fonctionnant pour une programmation dynamique à deux états. Pour ce faire, un contour convexe entre le domaine faisable et infaisable est déterminé pour concentrer le maillage adaptatif exclusivement dans cette zone d'intérêt. Les travaux présentés dans la première partie du chapitre se différencient par l'exploitation des informations transportées par le contour pour améliorer les interpolations faites durant le

backward pour conserver un maillage qualifié de permanent en reprenant les termes du chapitre 1. Ainsi la précision des calculs est améliorée par la qualité des interpolations et non par une meilleure exploitation des ressources numériques. Les différentes étapes de la méthode sont décrites par la suite pour expliquer quels sont les avantages et inconvénients de chacune des solutions présentées.

En utilisant l'outil de généralisation des *boundary lines*, qui sera appelé *boundary surfaces* dans le reste du manuscrit, il sera également possible d'apporter des éléments de réponse quant à la mise en place d'un maillage hétérogène permanent. Ce type de maillage permet d'exploiter au mieux possible les ressources numériques mises à disposition de l'algorithme pour tendre vers la meilleure qualité de résultat possible. Ainsi, ces maillages possèdent un compromis temps de calcul ou puissance de calcul et précision de calcul si l'hétérogénéité du maillage est bien mise en place.

Pour présenter ces travaux, le chapitre commence d'abord par le détail du modèle utilisé. Ce modèle est une version améliorée du modèle utilisé dans le chapitre 1 en ajoutant la prise en compte de la dynamique de la température de la machine électrique ainsi que son effet sur le critère d'optimisation étudié.

3.1 Application à la gestion de l'énergie d'un vehicule hybride électrique avec dynamique de température

3.1.1 Description du modèle

Le modèle considéré pour ce chapitre ainsi que le suivant est illustré en figure 3.1. Ce modèle possède deux commandes : le couple de la machine électrique C_{Mel} et le débit du liquide de refroidissement de la machine électrique $\dot{m}_c$; et deux états : l'état de charge de la batterie SOC_{Bat} et la température de la machine électrique T_{Mel}. Les hypothèses considérées pour réaliser le modèle sont les suivantes :

- Le frein moteur du moteur thermique est négligé, donc $P_{Eng} \geq 0$.
- La dynamique de réponse du moteur thermique est considérée suffisamment rapide pour répondre instantanément aux besoins du contrôleur *ECU*.
- Le rendement du moteur thermique est considéré constant : $\eta_{Eng} = 0{,}3$.
- La puissance dissipée durant les freinages est entièrement récupérée par la machine électrique.
- La batterie est considérée parfaite, sa résistance interne est donc nulle : $R_{Bat} = 0$.
- La tension de la batterie varie uniquement en fonction de son état de charge.
- La température du liquide de refroidissement de la machine électrique est considérée constante : $T_c = 27°\text{C}$.

Étant donné que ce chapitre à pour objectif de travailler sur les erreurs numériques induites par l'utilisation de la Programmation Dynamique, certaines des hypothèses permettent de retirer une modélisation par cartographie qui provoquerait une source d'erreur numérique supplémentaire.

Pour rappel, voici les équations présentent dans le modèle simplifié qui sont réutilisées pour ce nouveau modèle :

$$P_{Mel} + P_{Eng} = P_W \tag{3.1}$$

$$0 \leq P_{Eng} \leq 15 \text{ kW} \tag{3.2}$$

$$-86 \leq C_{Mel} \leq 86 \text{ N.m} \tag{3.3}$$

$$P_F = \frac{P_{Eng}}{\eta_{Eng}} \tag{3.4}$$

Ces quatre équations rappellent la relation de conservation de puissance entre les deux convertisseurs d'énergie et les roues, les contraintes de puissance du moteur thermique et de la machine électrique ainsi que la puissance carburant P_F nécessaire au moteur thermique pour développer la puissance mécanique P_{Eng}. Les pertes P_L ainsi que l'intensité de fonctionnement de la machine électrique I_{Mel} sont calculées de la même façon que dans le modèle simplifié, avec les mêmes constantes ($R = 48{,}47$ mΩ et $r = 0{,}6143$ N.m/A) :

$$P_L = R.I_{Mel}^2 \tag{3.5}$$

$$I_{Mel} = \frac{C_{Mel}}{r} \tag{3.6}$$

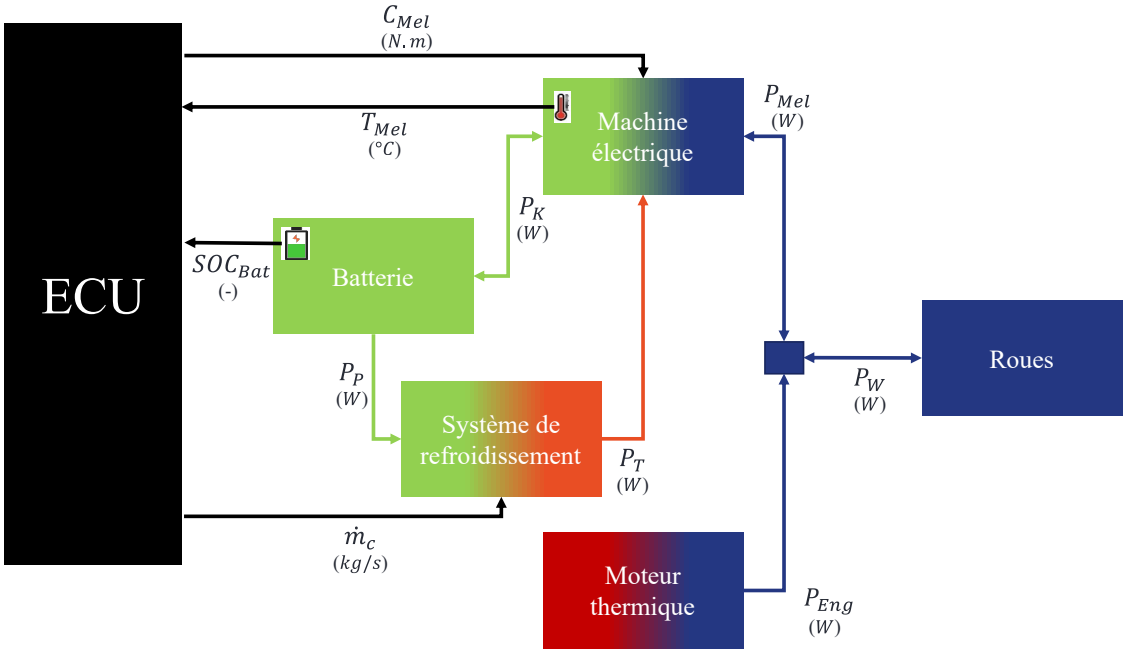

FIGURE 3.1 – Schéma bloc du modèle véhicule utilisé. Les blocs et flèches en bleu marine représentent les éléments mécaniques, en vert clair sont représentés les éléments ou flux électriques et en orange les éléments ou flux de chaleur. L'*ECU* et les flux associés sont représentés en noir.

La tension de la batterie n'est plus considérée comme constante. Le modèle de batterie

parfaite utilisé est composé d'une tension à vide constante E_0 et d'une tension e_0 qui dépend de l'état de charge de la batterie. La batterie peut stocker une capacité A_{max} et plus la batterie est chargée, plus la tension de la batterie V_{Bat} est haute comme le décrit l'équation (3.7). Les valeurs de ce modèle de batterie sont $E_0 = 529$ V et $e_0 = 92$ V.

$$V_{Bat} = E_0 + e_0.SOC_{Bat} \tag{3.7}$$

La dynamique de l'état de charge de la batterie (3.8), qui est le seul état du problème initial, est alors adaptée à la nouvelle écriture de la tension de la batterie (3.7), ainsi qu'à la consommation électrique de la pompe du circuit de refroidissement P_P décrite ultérieurement.

$$\dot{SOC}_{Bat}(t) = -\frac{\dot{m}_c(t).Ws + R.(\frac{C_{Mel}(t)}{r})^2 + C_{Mel}(t).\omega_{Mel}(t)}{A_{max}.(E_0 + e_0.SOC_{Bat}(t))} = \frac{P_{Bat}(t)}{A_{max}.V_{Bat}(t)} \tag{3.8}$$

L'équation (3.8) décrit la variation d'état de charge de la batterie dans le temps. Cette dernière dépend de la dépense énergétique électrique de la batterie dans le véhicule. Certaines des puissances expliquées précédemment sont alors impliquées dans ce calcul. L'état de charge de la batterie est donc en réalité un rapport entre la charge actuelle A_{act} de la batterie et la charge maximale. L'état de charge est une variable qui évolue entre 0 et 1. La variation est directement liée à la quantité de charge électrique entrant ou sortant de la batterie.

$$SOC_{Bat}(t) = \frac{A_{act}(t)}{A_{max}} = \int_{t_0}^{t} \frac{P_{Bat}(\tau)}{A_{max}.V_{Bat}(\tau)} d\tau + SOC_{Bat}(t_0) \tag{3.9}$$

Il est souhaité que la machine électrique fonctionne à 30°C pour des raisons d'efficacité et de dégradation. C'est pour cette raison que la température de la machine électrique est considérée comme second état du problème. De plus, cela explique pourquoi il a été choisi de piloter le couple de la machine électrique plutôt que la puissance du moteur thermique et c'est également pour cette raison que la seconde commande ajoutée au système est le débit de liquide de refroidissement $\dot{m}_c$, afin de contrôler la température de la machine électrique. Enfin, la manipulation du débit permet de contrôler la puissance électrique P_P allouée au refroidissement par la relation :

$$P_P = \dot{m}_c.Ws \tag{3.10}$$

où $Ws = 900$ J/kg est le travail spécifique de la pompe. La puissance électrique transitant par la batterie P_{Bat} devient alors :

$$P_{Bat} = P_K + P_P \tag{3.11}$$

avec P_K la puissance électrique provenant ou demandée par la machine électrique qui s'exprime par la relation (3.12) qui comprend la puissance mécanique P_{Mel} et les pertes de la machine électrique P_L.

$$P_K = P_{Mel} + P_L \tag{3.12}$$

Deux puissances sont en jeu quant à la variation de température de la machine électrique :

les pertes par effet Joule (3.5) et la puissance d'échange thermique avec le liquide de refroidissement P_T (3.13). Cet échange thermique dépend du débit, de la capacité calorifique du liquide de refroidissement $Cp_c = 4180$ J/(kg.K) et de la différence de température entre la machine électrique et le liquide de refroidissement.

$$P_T = \dot{m}_c.Cp_c.(T_{Mel} - T_c) \tag{3.13}$$

L'équation (3.14) décrit alors la variation de température pour la machine électrique dans le temps. Afin de transformer la puissance thermique en température, les puissances sont divisées par le produit de la masse de la machine électrique $m_{Mel} = 80$ kg et par son pouvoir calorifique $Cp_{Mel} = 900$ J/(kg.K).

$$\dot{T}_{Mel}(t) = \frac{R.(\frac{C_{Mel}(t)}{r})^2 - \dot{m}_c(t).Cp_c.(T_{Mel}(t) - T_c)}{m_{Mel}.Cp_{Mel}} \tag{3.14}$$

L'objectif de l'*ECU* est donc de minimiser la consommation d'énergie globale du véhicule. Contrairement au chapitre 1 où le critère est convexe, il a été choisi de retirer le coefficient ρ qui permettait la convexité de la fonction coût car ce dernier était arbitraire. Ainsi, pour exprimer réellement la fonction coût en puissance, la fonction racine carrée est utilisée.

$$J = \int_{t_0}^{t_f} \sqrt{P_F^2(t) + \alpha_{Mel}(T_{Mel}).P_{Bat}^2(t)}.dt + \Phi(SOC_{Bat}(t_f), T_{Mel}(t_f)) \tag{3.15}$$

Le coefficient α_{Mel}, sans unité, permet d'intégrer la dégradation de la machine électrique dans le critère de minimisation. Il est construit sous la forme d'un polynôme du second degré dont les coefficients sont les suivants : $\theta_a = 1{,}1.10^{-3}$ $(°C)^{-2}$, $\theta_b = -6{,}67.10^{-2}$ $(°C)^{-1}$ et $\theta_c = 30$.

$$\alpha_{Mel}(T_{Mel}) = \theta_a.T_{Mel}^2 + \theta_b.T_{Mel} + \theta_c \tag{3.16}$$

Ainsi, le coût associé à la partie électrique de la chaîne de traction est proche de la partie thermique. De plus, à la température cible $T_{Mel} = 30$ °C, le coût associé à la partie électrique est deux fois moins important qu'à la température extrême $T_{Mel} = 60$°C.

3.1.2 Formulation du problème de commande optimale avec contraintes finales

Les contraintes déjà présentes pour le modèle simplifié du chapitre 1 sont conservées (1.19) mais de nouvelles contraintes, associées au système de refroidissement sont ajoutées :

$$\begin{cases} \underline{SOC_{Bat}} \leq SOC_{Bat}(t) \leq \overline{SOC_{Bat}} \\ \underline{C_{Mel}} \leq C_{Mel}(t) \leq \overline{C_{Mel}} \\ \underline{P_{Eng}} \leq P_{Eng}(t) \leq \overline{P_{Eng}} \\ \underline{T_{Mel}} \leq T_{Mel}(t) \leq \overline{T_{Mel}} \\ \underline{\dot{m}_c} \leq \dot{m}_c(t) \leq \overline{\dot{m}_c} \end{cases} \tag{3.17}$$

La table 3.1 récapitule l'ensemble des valeurs numériques de ces contraintes.

TABLE 3.1 – Table récapitulative des contraintes appliquées au système véhicule hybride

Variable	Contrainte minimale	Contrainte maximale
SOC_{Bat} (-)	$\underline{SOC_{Bat}} = 0$	$\overline{SOC_{Bat}} = 1$
C_{Mel} (N.m)	$\underline{C_{Mel}} = -86$	$\overline{C_{Mel}} = 86$
P_{Eng} (kW)	$\underline{P_{Eng}} = 0$	$\overline{P_{Eng}} = 15$
T_{Mel} (°C)	$\underline{T_{Mel}} = 20$	$\overline{T_{Mel}} = 60$
$\dot{m}_c$ (kg/s)	$\underline{\dot{m}_c} = 0$	$\overline{\dot{m}_c} = 0{,}05$

En plus de ces contraintes, des contraintes spécifiques sont à respecter pour les deux variables d'état à la fin du cycle de conduite imposé qu'est le *WLTC* dans cette étude. Les contraintes sont les suivantes :

$$\begin{cases} SOC_{Bat_0} = 0{,}61 \leq SOC_{Bat_N} \leq 0{,}8 \\ T_{Mel_0} = 37 \leq T_{Mel_N} \leq 50 \end{cases} \tag{3.18}$$

Si l'une des quatre contraintes finales n'est pas respectée, il faut alors pénaliser l'algorithme de programmation dynamique à travers $\Phi(\boldsymbol{x}_N)$. Cette pénalité est alors définie comme :

$$\begin{cases} \Phi\left((SOC_{Bat_N} < 0{,}61 + SOC_{Bat_N} > 0{,}8) \ + (T_{Mel_N} < 37 + T_{Mel_N} > 50)\right) = 10^{20} \\ \Phi\left((SOC_{Bat_N} \geq 0{,}61 \times SOC_{Bat_N} \leq 0{,}8) \ + (T_{Mel_N} \geq 37 \times T_{Mel_N} \leq 50)\right) = 0 \end{cases} \tag{3.19}$$

Ainsi, en repartant de (1.5), l'écriture du problème de commande optimale pour le modèle considéré est le suivant :

$$\begin{cases} \min\limits_{C_{Mel},\dot{m}_c} J(k, \begin{pmatrix} SOC_{Bat_k} \\ T_{Mel_k} \end{pmatrix}, \begin{pmatrix} C_{Mel_k} \\ \dot{m}_{c_k} \end{pmatrix}) = \sum_{i=k}^{N-1} \sqrt{P_{F_i}^2 + \alpha_{Mel}(T_{Mel_i}).P_{Bat_i}^2}.\Delta t + \Phi\left(\begin{pmatrix} SOC_{Bat_N} \\ T_{Mel_N} \end{pmatrix}\right) \\ \begin{pmatrix} SOC_{Bat_{i+1}} \\ T_{Mel_{i+1}} \end{pmatrix} = \begin{pmatrix} SOC_{Bat_i} \\ T_{Mel_i} \end{pmatrix} + \begin{pmatrix} f_{SOC_{Bat_i}}(\begin{pmatrix} SOC_{Bat_i} \\ T_{Mel_i} \end{pmatrix}, \begin{pmatrix} C_{Mel_i} \\ \dot{m}_{c_i} \end{pmatrix}) \\ f_{T_{Mel_i}}(\begin{pmatrix} SOC_{Bat_i} \\ T_{Mel_i} \end{pmatrix}, \begin{pmatrix} C_{Mel_i} \\ \dot{m}_{c_i} \end{pmatrix}) \end{pmatrix} \\ P_{Eng_i} \in [\underline{P_{Eng}}; \overline{P_{Eng}}] \\ \begin{pmatrix} SOC_{Bat_i} \\ T_{Mel_i} \end{pmatrix} \in \begin{pmatrix} [\underline{SOC_{Bat}}; \overline{SOC_{Bat}}] \\ [\underline{T_{Mel}}; \overline{T_{Mel}}] \end{pmatrix} \\ \begin{pmatrix} C_{Mel_i} \\ \dot{m}_{c_i} \end{pmatrix} \in \begin{pmatrix} [\underline{C_{Mel}}; \overline{C_{Mel}}] \\ [\underline{\dot{m}_c}; \overline{\dot{m}_c}] \end{pmatrix} \end{cases} \tag{3.20}$$

L'équation (3.20) intègre au problème de commande optimale appliqué au modèle véhicule simplifié toutes les contraintes imposées au système définies précédemment.

Le choix d'une pénalité forte, très importante devant le coût attendu, est fait pour mettre en difficulté l'algorithme face à la problématique de contamination de la matrice de coût

optimal. Cependant un coût infini $\Phi(\boldsymbol{x}_N \notin \boldsymbol{X}_N) = +\infty$ n'a pas été retenu car aucune comparaison entre l'algorithme classique et l'algorithme avec les *boundary surfaces* n'aurait été possible. En effet, avec cette pénalité finale, l'algorithme classique n'est pas capable de trouver de solution car la matrice de coût optimal est remplie d'infinis.

Les deux contraintes finales minimales pour l'état de charge de la batterie et la température de la machine électrique sont égales aux conditions initiales des problèmes. Cela s'explique par la volonté d'avoir un trajet à variation de condition nulle : $SOC_{Bat_N} = SOC_{Bat_0}$ et $T_{Mel_N} = T_{Mel_0}$. Cela provient du fait qu'il est souhaité que le trajet soit répétable dans le temps mais également de pouvoir préparer la partie électrique de la chaîne de traction à une utilisation après arrêt du véhicule. On peut penser à une recharge rapide ou à l'utilisation du véhicule pour suppléer un apport d'énergie dans une configuration *V2G* (*Vehicle to Grid*).

3.1.3 Résultats par Programmation Dynamique classique

Les résultats affichés sont en deux parties : l'une pour une trajectoire à des conditions initiales précises définies dans la partie précédente et l'autre à bilan batterie nul. Cela permet de comparer les solutions de deux points de vue différents. Le premier est d'observer comment se comporte les différents algorithmes aux bords des contraintes finales tandis que le second permet de comparer chaque algorithme à un bilan d'énergie nul, c'est-à-dire que le véhicule conserve la même quantité d'énergie entre le début et la fin du cycle de conduite. Cependant, si aucune condition initiale permet ce résultat, alors une interpolation est réalisée comme l'illustre la figure 3.2. Les conditions initiales utilisées pour déterminer le bilan batterie nul ne varient pas en température mais uniquement l'état de charge de la batterie change. Cette température initiale est la même que pour les conditions initiales spécifiques : $T_{Mel_0} = 37°\text{C}$.

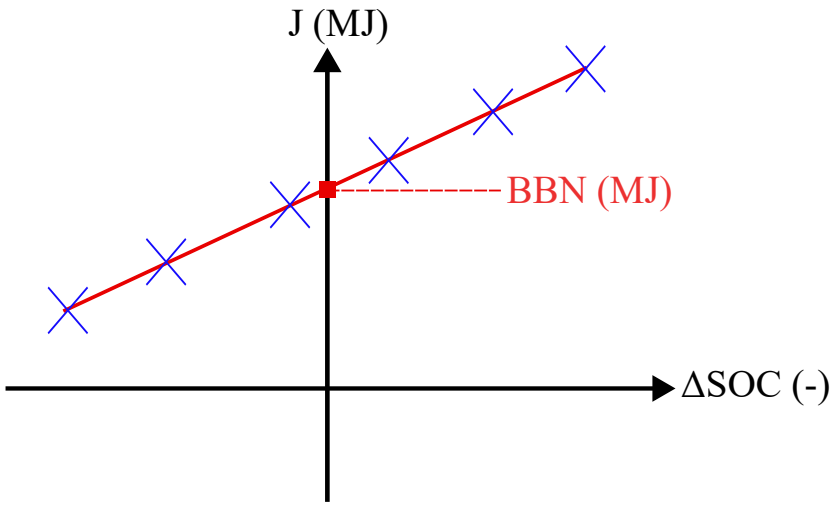

FIGURE 3.2 – Graphique du coût J déterminé en fonction de la différence d'énergie électrique $\Delta SOC = SOC(t_f) - SOC(t_0)$ contenue dans la batterie. Les points en bleus sont les résultats à partir de diverses conditions initiales.

Les tables 3.2 et 3.3 notent les résultats où un phénomène contre-intuitif est observable. Le coût des solutions trouvées aussi bien pour la trajectoire particulière qu'au Bilan Batterie Nul (BBN) augmente avec le raffinage du maillage jusqu'à un maillage critique (121 mailles par variables). Pour comprendre ce phénomène, une observation des matrices de coût optimal (figures 3.3 à 3.5) est requise. Premièrement, toutes les matrices sont fortement contaminées par les contraintes finales. Il peut être observé que la contamination de la matrice est réduite lorsque le maillage est raffiné. Cette observation est toutefois contraire aux résultats des tables 3.2 et 3.3.

TABLE 3.2 – Résultats obtenus avec une Programmation Dynamique "classique" avec contraintes finales pour les conditions initiales particulières.

Maillage (-)	Coût (MJ)	Tps calc (s)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)
21	46,03	20,2	0,7156	38,6	0,04
41	46,23	191,2	0,6931	38,1	0,35
61	51,67	883,8	0,6804	38,6	1,50
81	55,77	2699,9	0,6780	38,6	4,44
101	58,88	6524,3	0,6748	38,9	10,48
121	60,72	13269	0,6779	39,4	21,30
141	56,59	24775	0,6757	39,4	38,94
161	37,97	41232	0,6801	37,5	65,82
181	35,47	67911	0,6871	37,1	104,74

TABLE 3.3 – Résultats obtenus avec une Programmation Dynamique "classique" avec contraintes finales au bilan batterie nul.

Maillage (-)	BBN (MJ)	BBN (°C)
21	42,99	38,4
41	43,89	37,8
61	47,40	38,3
81	50,23	37,2
101	53,20	38,4
121	55,06	37,2
141	52,89	38,7
161	37,66	37,5
181	34,18	37,1

Les tranches des matrices montrent les coûts optimaux calculés pour chaque couple de température de machine électrique et d'état de charge de la batterie qui sont considérés dans le problème de contrôle optimal. Par soucis d'affichage et de lisibilité, pour les matrices des trois premiers maillages, l'échelle graphique n'est pas celle de du critère d'optimisation, mais le logarithme en base 10 de ce dernier : $log_{10}(J_k(\boldsymbol{x}_k))$. La matrice du coût optimal pour le maillage le plus fin est quant à elle représentée par le critère d'optimisation car cette dernière n'est pas aussi contaminée que les trois autres.

Le maillage le plus fin proposé permet d'assurer une certaine qualité du résultat car la matrice de coût optimale semble avoir réussi à préserver l'ensemble ou une partie du domaine faisable, ce qui n'est pas le cas des maillages moins fins. La méconnaissance du vrai domaine faisable justifie la remise en question de l'optimalité de la solution. De plus, le temps et les ressources en calcul requises pour obtenir le meilleur résultat sont suffisamment grandes pour investiguer la mise en place d'une généralisation de l'amélioration des *boundary lines* pour des programmations dynamiques à deux états.

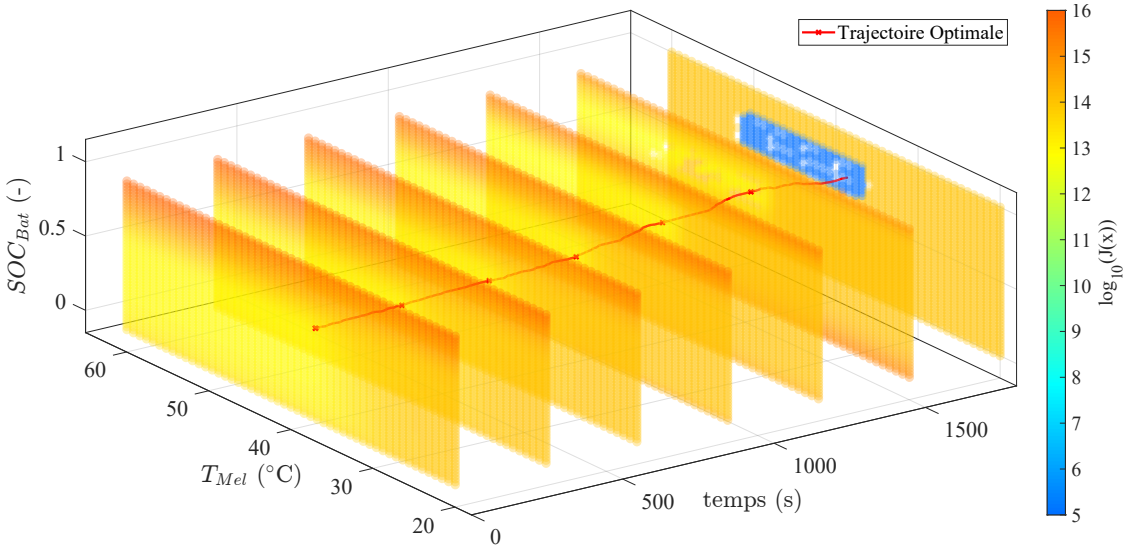

FIGURE 3.3 – Maillage homogène à 21 mailles

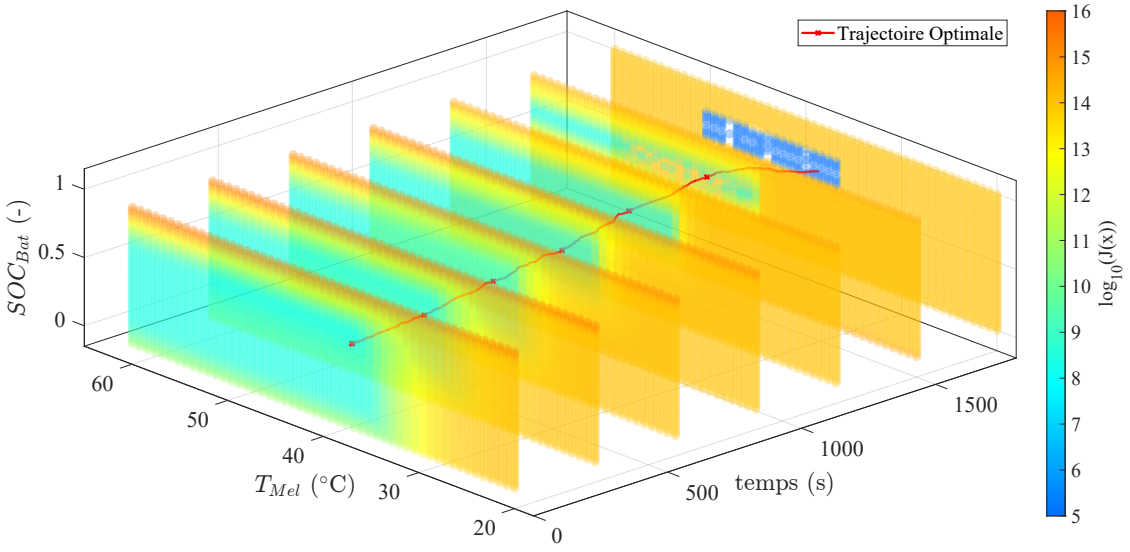

FIGURE 3.4 – Maillage homogène à 81 mailles

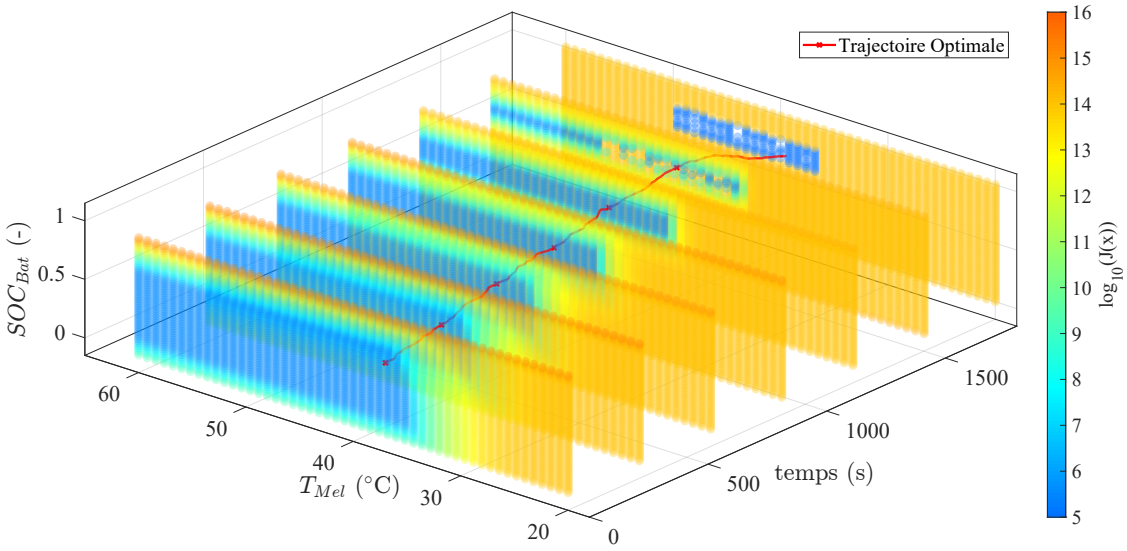

FIGURE 3.5 – Maillage homogène à 121 mailles

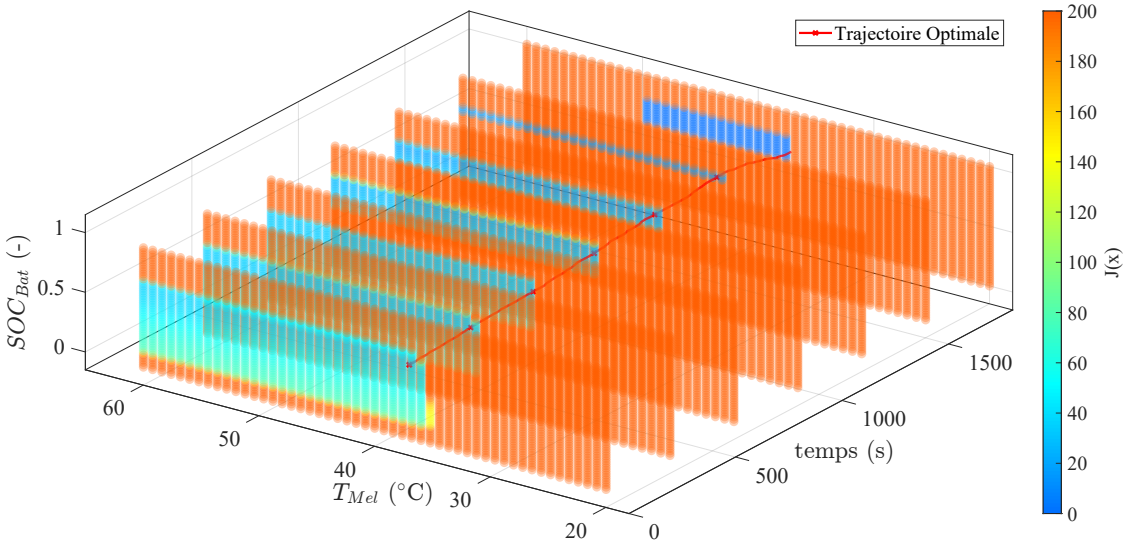

FIGURE 3.6 – Maillage homogène à 181 mailles

La figure 3.7 illustre les trajectoires des différentes commandes et états. Pour les quatre solutions présentées, l'état de charge de la batterie augmente durant toute la mission à tel point que le bilan batterie est positif. De plus, la température de la machine électrique augmente également le long de la mission, ce qui induit une augmentation du critère d'optimisation. Si d'un côté les trajectoires d'état de charge de la batterie sont similaires, ce n'est pas le cas de la température de la machine électrique. En effet, cette dernière augmente lorsque le maillage est plus fin. Cela explique la dégradation des résultats obtenus dans la table 3.2.

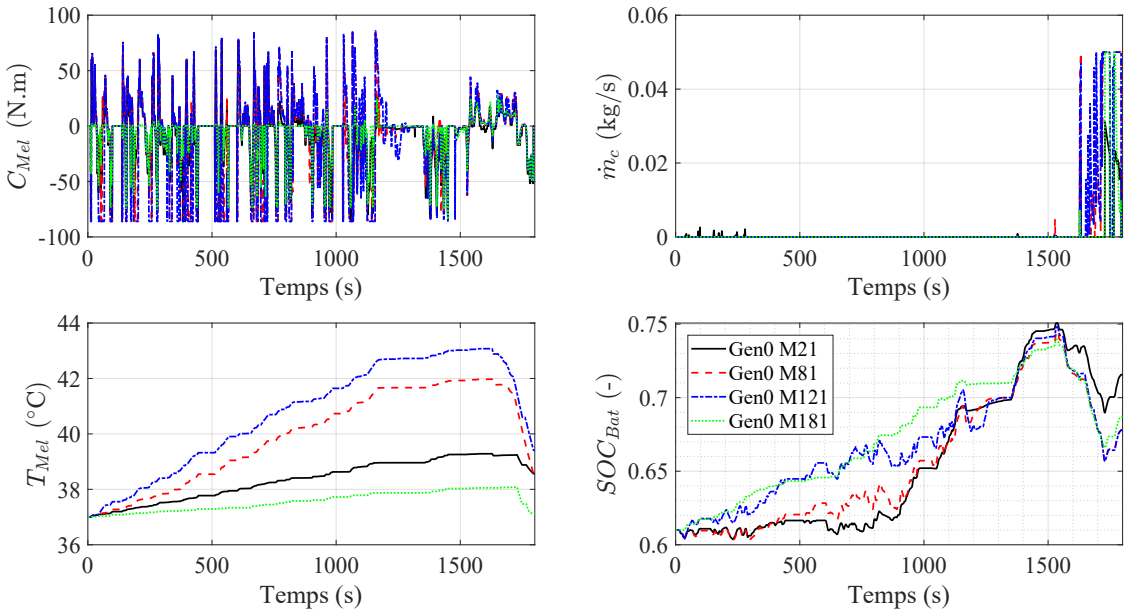

FIGURE 3.7 – Trajectoires des états et des commandes de quatre solutions de la *DP* classique (Gen0) pour les maillages à 21, 81, 121 et 181 mailles par variables respectivement en noir, rouge, bleu et vert pour le problème (3.20) avec contraintes finales (3.19) sur un cycle WLTC.

3.2 Des *boundary lines* aux *boundary surfaces*

Le chapitre 1 montre que l'amélioration des *boundary lines* repose sur la connaissance des états les plus hauts/bas possible à chaque pas de temps pour protéger la matrice de coût optimal de la contamination (figure 3.8) . En effet, à partir de l'état maximal admissible à la fin de la simulation, il est possible de reconstruire la courbe d'état maximal à chaque instant. Cependant, lorsque le problème est constitué de deux états, la définition du couple d'état qui est le plus haut ou le plus bas n'est pas évidente. L'objectif de l'outil *boundary surfaces* est alors de déterminer la surface et son contour qui contient tous les couples possibles qui permettent d'accéder à la surface suivante.

3.2.1 Algorithme et solution initiale

La première version développée de la méthodologie donne accès à la version la plus précise de la méthodologie mais c'est également la plus lente de celles présentées dans le manuscrit à iso-maillage. L'algorithme utilise trois fonctions déjà présentes dans Matlab qui sont les fonctions *boundary*, *griddata* et *inpolygon*. La première permet de déterminer pour un ensemble de points dans un espace à deux ou trois dimensions quels sont les points qui constituent la frontière de l'ensemble. Un paramètre, appelé *shrink factor* (facteur de forme), influence grandement la forme du contour. Afin de déterminer quels points appartiennent à cette frontière, la fonction s'appuie sur un algorithme appelé *Alpha shapes* [103]. Cet algorithme détermine

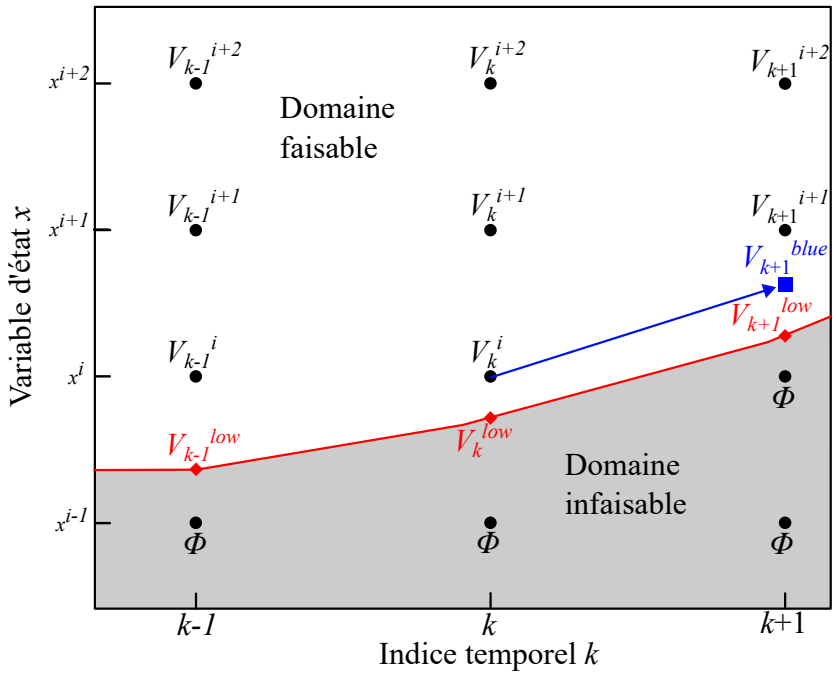

FIGURE 3.8 – Rappel de la construction des *boundary lines* expliquée dans le chapitre 1.

quel est le plus grand cercle possible avec pour centre chaque point de l'ensemble sans aucun autre point de cet ensemble dans ce cercle. Ainsi, les cercles les plus grands sont ceux retenus pour tracer le contour. Le facteur de forme, sf, permet d'ajuster la taille critique des cercles. La seconde fonction permet de faire des interpolations de Delaunay [104] à partir d'ensemble de points qui ne sont pas organisés sous forme de tenseur avec des coordonnées définies à l'avance. La dernière fonction permet de déterminer si un point est à l'intérieur ou non d'un contour. Cette fonction est utilisée ici afin de ne pas faire les interpolations sur tous les points, ce qui permet de réduire le temps de calcul nécessaire aux interpolations.

Premièrement, la construction d'une nouvelle surface à l'indice de temps k se base sur la surface déjà existante à l'indice de temps $k+1$. La méthode décrite ici est faite de manière numérique pour éviter d'approximer analytiquement la frontière entre les domaines faisable et infaisable. La variable $\boldsymbol{X}_k$ représente l'ensemble des points appartenant à la frontière du domaine faisable et infaisable. En appliquant toutes les commandes possibles $\{\boldsymbol{u}\}_k$ à partir de cet ensemble de points, il est possible de déterminer les points d'origines possibles de ces points B_k grâce à l'inversion du modèle (1.31) appliqué au contour $\boldsymbol{X}_{k+1}$.

$$B_k = f^{-1}(\boldsymbol{X}_{k+1}, \{\boldsymbol{u}\}_k) \tag{3.21}$$

Les points ne respectant pas les contraintes du système D_k sont retirés de l'ensemble B_k pour ne pas considérer des points infaisables durant la construction des surfaces. Cela donne lieu à l'ensemble F_k qui représente l'ensemble de tous les points possibles sans les points considérés comme impossibles.

$$G_k = B_k \backslash D_k \tag{3.22}$$

Pour calculer le contour $\boldsymbol{X}_k$ de l'ensemble F_k, il faut retirer l'intérieur de cet ensemble $\overset{\circ}{F}_k$ pour en déterminer le contour.

$$\boldsymbol{X}_k = F_k \backslash \overset{\circ}{F}_k \tag{3.23}$$

En prenant la figure 3.9 comme exemple, il est possible d'expliquer cette dernière phrase. Les points bleus représentent l'intérieur de l'ensemble $\overset{\circ}{F}_k$ tandis que l'addition des points bleus et des losanges noirs représente la totalité de l'ensemble F_k. Ainsi, le contour de l'ensemble $\boldsymbol{X}_k$ est représenté uniquement par les losanges noirs.

La figure 3.9 illustre le fonctionnement de la fonction *boundary* en fonction du coefficient de facteur de forme sf. Ce paramètre peut donner alors une forme convexe au contour, le contour le plus compact ou un entre-deux. Le choix de la forme convexe a été fait car le gros défaut du domaine compact réside dans les points sélectionnés pour définir le contour.

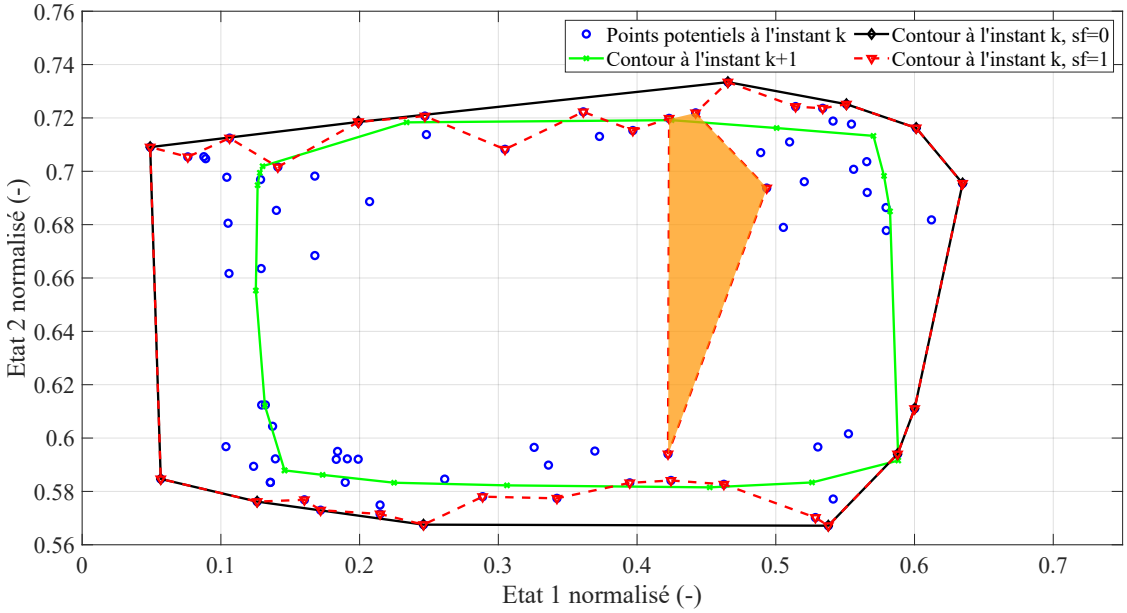

FIGURE 3.9 – Illustration de l'effet du coefficient *shrink factor*, sf, sur la détermination des contours par la fonction *boundary*. Le contour à l'instant $k+1$ est en vert, les points en bleu sont les origines possibles à l'instant k des points du contour à l'instant $k+1$. Les contours rouge et noir sont les contours déterminés par la fonction *boundary* pour un sf respectivement de 1 et 0.

Premièrement, la détermination du contour à l'instant k s'appuie sur l'information du contour à l'instant $k+1$ en vert. A partir du contour de l'instant $k+1$ et du modèle inversé (3.21), il est possible de calculer tous les points bleus qui représentent toutes les origines possibles à l'instant k des points du contour à l'instant $k+1$. La détermination du contour à l'instant k revient alors à déterminer quels sont les points bleus qui permettent de définir un contour qui encerclerait ces derniers. La fonction *boundary* permet alors de modifier le facteur de forme pour ajuster la forme du contour. En effet, ce coefficient, compris entre 0 et

1, permet d'avoir respectivement l'ensemble le plus convexe (en noir) ou le plus compact (en rouge) possible. La différence entre les deux contours s'effectue alors sur la sélection des points pour générer le contour mais également sur la surface occupée par l'ensemble de points. Cette surface permet par la suite d'évaluer si un point appartient ou non à l'ensemble des points permettant de résoudre le problème de contrôle optimal tout en respectant les contraintes finales. A titre d'exemple, l'analogie avec l'existence du point dans l'intervalle entre la *upper* et la *lower boundary lines* peut être faite. On peut alors comprendre pourquoi le choix du facteur de forme est imposé $sf = 0$. En effet, l'espace déterminé par le contour compact n'inclut pas une partie de l'espace qui est importante (en orange) car des points de maillage seront dans cette zone. Ces points seront probablement des solutions infaisables car ils ne pourront pas atteindre la zone faisable. Un dysfonctionnement semble alors apparaître avec la construction des espaces les plus compacts possibles.

En plus de la méthode de détermination des contours, la question du nombre de points maximum admissible sur un contour est également de rigueur. En effet, sans limitation, rien n'empêche l'algorithme de considérer un grand nombre de points pour les contours, ce qui allongera très fortement le calcul. C'est pourquoi, une limitation arbitraire du nombre de points placés a été mise en place. La solution retenue est la suivante : diviser le contour en n_p parties, allouer N_{max}/n_p points pour chacune. Si le nombre de points dans une partie est inférieur à N_{max}/n_p alors tous les points sont conservés. A défaut, une sélection uniforme est effectuée pour obtenir N_{max}/n_p points à la fin du processus pour la partie.

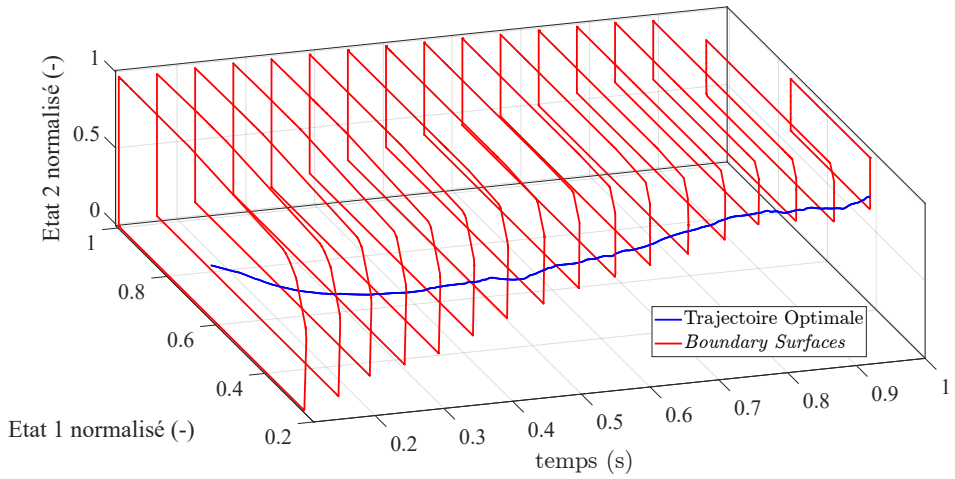

FIGURE 3.10 – Exemple de rendu des *Boundary Surfaces*. Des tranches de contour sont représentées en rouge et la trajectoire optimale du problème est représentée en bleue.

Un exemple de résultat de la création des *boundary surfaces* est illustré sur la figure 3.10. Les contours successifs en rouge sont calculés à partir du contour initial tracé à l'instant final pour définir les conditions dans lesquelles le système doit finir le problème. On peut ainsi voir le contour s'élargir au fur et à mesure que la formation des surfaces se fait. Par la suite, la trajectoire des états du système en bleu, est initialisée aux conditions voulues, qui se trouve ici appartenir à l'ensemble décrit par la *boundary surface* sur l'instant initial.

Ces conditions initiales sont alors considérées comme appartenant au domaine faisable et il existe alors au moins une loi de commande qui permet au système de respecter toutes les contraintes, notamment les finales. La loi de commande est alors déterminée de manière à respecter le critère de minimisation.

L'algorithme 5 est alors décomposé en trois parties tout comme l'algorithme 3 de programmation dynamique utilisant l'outil des *boundary lines*.

Algorithme 5 Algorithme de Programmation Dynamique avec l'outil des *boundary surfaces*

Pour $k = N-1 : -1 : 1$ **Faire** ▷ boucle *boundary surfaces*
$B_k = f_k^{-1}(\boldsymbol{X}_{k+1}, \boldsymbol{u}_k, \boldsymbol{w}_k).dt$
$G_k = B_k \backslash D_k$
$\boldsymbol{X}_k = F_k \backslash \overset{\circ}{F}_k$ ▷ réalisé par la fonction *boundary* en Gen.1, fonction contour en Gen.2
$\boldsymbol{V}_k^{BdS} = L(k, \boldsymbol{X}_k, \boldsymbol{u}_k).dt + \boldsymbol{V}_{k+1}^{BdS}$
$\boldsymbol{u}_k^{BdS} = \boldsymbol{u}_k(\underset{\boldsymbol{u}_k}{arg}(F_k))$
Fin de Pour
Pour $p = N-1 : -1 : 1$ **Faire** ▷ boucle *Backward*
$\boldsymbol{x}_{p+1} = \boldsymbol{x}_p + f_p(\boldsymbol{x}_p, \boldsymbol{u}_p).dt$
$g = L(p, \boldsymbol{x}_p, \boldsymbol{u}_p).dt$
$V_{interp} = \nu([\boldsymbol{x}_{p+1}; \boldsymbol{X}_{k+1}], [V(p+1, \boldsymbol{x}_{p+1}); \boldsymbol{V}_{p+1}^{BdS}], \boldsymbol{x}_{p+1})$
$J = g + V_{interp} + \mu$ ▷ μ est une variable qui notifie des uplets infaisables
$V(p, \boldsymbol{x}_p) = \underset{\boldsymbol{u}}{min}(J)$
$U(p, \boldsymbol{x}_p) = \boldsymbol{u}(\underset{\boldsymbol{u}}{argmin}(J))$
Fin de Pour
$Cost = 0$
Pour $t = 1 : 1 : N-1$ **Faire** ▷ boucle *Forward*
$u_t = f_i([\boldsymbol{x}_t; \boldsymbol{X}_t], [\boldsymbol{u}_t^{BdS}; \boldsymbol{U}_t], \boldsymbol{x}_t)$
$x_{t+1} = x_t + f_t(x_t, u_t).dt$
$C = L(t, x_t, u_t) + Cost$
Fin de Pour

Dans la première partie, pour générer les *boundary surfaces*, le même principe que les *boundary lines* est conservé : il y a des contraintes initiales et/ou finales à respecter. Les contraintes initiales ou conditions initiales ne seront pas utilisées ici. La figure 3.11 permet d'illustrer l'application des outils décrits précédemment. Dans un premier temps, il est nécessaire de discrétiser ce contour. A partir de ces points, toutes les commandes sont appliquées dans le modèle inversé du système pour savoir d'où les points peuvent venir pour former le contour final en un seul pas de temps. De là, les points formant le nouveau contour sont déterminés avec la fonction *boundary*. Les commandes permettant le lien entre tous les points sont alors stockées ainsi que le coût instantané lié aux commandes et aux états. Ces actions sont répétées jusqu'à avoir traversé tous les pas de temps de la simulation.

Lors du *backward*, seconde partie de l'algorithme, il nous faut réutiliser les résultats obtenus pendant la génération des *boundaries* pour propager l'information de manière analogue

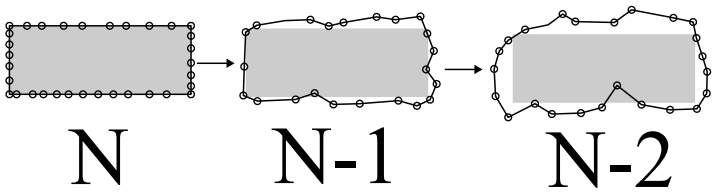

FIGURE 3.11 – Évolution des contours, le fond gris indique le contour généré pour le pas de temps N.

aux *boundary lines*. Cependant, la méthode d'interpolation ne change pas entre une programmation dynamique à un état qui utilise l'outil des *boundary lines* et une autre programmation dynamique qui n'utilise pas cet outil. En effet, c'est toujours l'interpolation linéaire qui est utilisée. Toutefois, l'utilisation des *boundary surfaces* nécessite de mettre de côté les interpolations bilinéaires pour des interpolations basées sur la triangulation de Delaunay [105], ce qui est effectué par la fonction *griddata*. Cette nécessité s'explique par l'utilisation des points du contour dans les interpolations. Ces points n'ont aucune disposition à respecter, donc il est très probable que ces derniers n'appartiennent pas au maillage du problème. De ce fait, il n'est pas envisageable de les intégrer aux interpolations bilinéaires.

Enfin, durant le *forward*, dernière partie de l'algorithme, les commandes pour les états extrêmes qui sont sur le contour sont réutilisées en même temps que les matrices de commandes optimales calculées pendant le *backward* lors de la détermination des commandes à appliquer au système.

Les résultats affichés par les tables 3.4 et 3.5 démontrent l'intérêt de la méthode, notamment les résultats qui sont bien meilleurs que ceux obtenus dans les tables 3.2 et 3.3 pour le même problème. En effet, les résultats obtenus avec les *boundary surfaces* sont équivalents aux meilleurs résultats obtenus avec la programmation dynamique classique (respectivement $J = 37{,}97$ et $J = 35{,}47$ pour le coût particulier des maillages 161 et 181). Cependant, le Le temps de calcul augmente énormément comparé à la programmation dynamique classique car en plus de la génération des surfaces, la méthode d'interpolation utilisée ici est plus lente car les informations disponibles ne sont désormais plus uniquement placées sur le maillage. Toutefois, il est possible d'obtenir une qualité de résultat équivalente en moins de temps de calcul avec les *boundary surfaces*. Pour une qualité de résultat équivalente, il faut 40 000 secondes et 66 Go de RAM environ pour effectuer le calcul pour le maillage à 161 mailles par variables là où il ne faut que 5000 secondes et 0,2 Go de RAM pour le maillages à 31 mailles par variables avec l'amélioration.

Les trajectoires des quatre maillages avec les *boundary surfaces* sont présentées sur la figure 3.12. Il est alors possible de comprendre les écarts de précision de résultats et l'amélioration des résultats avec le raffinage du maillage. En effet, le maillage le plus grossier, en noir, a une dépense d'énergie plus importante que les autres car l'état de charge finale est plus élevé tout comme la température de la machine électrique. La réduction du coût passe alors par une dépense d'énergie moins importante mais également par une meilleure gestion de la température de la machine électrique.

TABLE 3.4 – Résultats obtenus avec une Programmation Dynamique avec *boundary surfaces* première génération (Gen.1) pour le problème avec contraintes finales pour les conditions initiales particulières.

Maillage (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	Temps de calcul (s)			
					BdS	*BW*	*FW*	Total
21	38,42	0,6997	38,7	0,06	729,0	762,0	0,7	1491,7
31	35,25	0,6741	37,9	0,18	1629,2	3345,2	0,6	4975,0
41	34,32	0,6635	37,0	0,46	3304,4	10662	0,7	13967
51	33,92	0,6567	37,9	0,99	5275,6	25548	0,7	30824

TABLE 3.5 – Résultats obtenus avec une Programmation Dynamique avec *boundary surfaces* première génération (Gen.1) pour le problème avec contraintes finales pour le bilan batterie nul.

Maillage (-)	Coût BBN (MJ)	T_{Mel_N} BBN (°C)
21	38,79	37,8
31	35,43	37,6
41	34,37	37,9
51	34,12	37,9

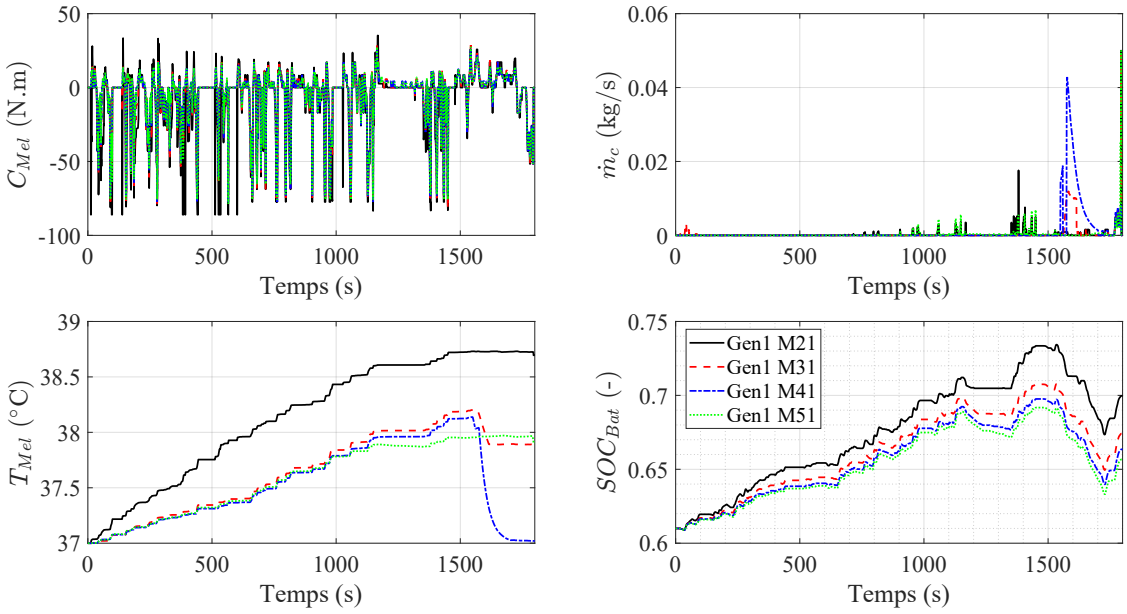

FIGURE 3.12 – Trajectoires des états et des commandes des trois solutions de *DP* avec les *BdS* pour les maillages de 21, 31, 41 et 51 mailles par variables respectivement en noir, rouge, bleu et vert pour le problème (3.20) avec contraintes finales (3.19).

La figure 3.13 permet d'illustrer l'évolution de la trajectoire déterminée par la programmation dynamique basée sur l'algorithme 5 avec un maillage homogène de 51 mailles par

variable. Il est également possible d'observer des tranches de la matrice de coût optimal $\boldsymbol{V}$ à plusieurs pas de temps au cours de la mission. Ces tranches de matrice mettent en lumière le fait que d'une manière générale, les coûts des points au début de la simulation sont plus élevés que ceux de la fin, ce qui est attendu. Il est possible de voir que la surface s'est agrandit rapidement au début (pour les pas de temps à la fin) mais n'évolue que très peu par la suite. Cela s'explique par le fait que l'algorithme est capable de trouver les couples de commande qui permettent une conservation des contours notamment proche des contraintes. La tranche de $\boldsymbol{V}$ à l'instant initial illustre que certaines conditions initiales sont avantagées par rapport à d'autres. La trajectoire déterminée dans le cas illustré ici évolue au centre du domaine faisable sauf à la fin car la solution optimale est d'arriver dans le coin en bas à droite de la surface finale. Il peut être constaté que les valeurs dans la matrice de coût optimal sont cohérentes avec celles présentées dans la table 3.4. La figure B.1 en annexe B présente l'évolution du temps de calcul en fonction du coût à bilan batterie nul.

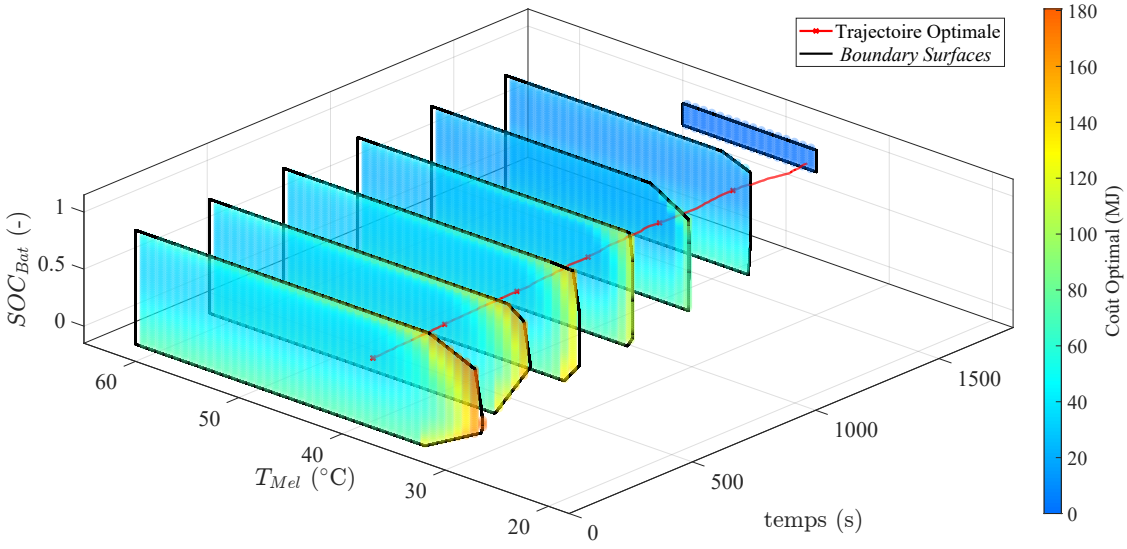

FIGURE 3.13 – Trajectoire de la solution trouvée par la programmation dynamique en rouge avec des tranches de la matrice de coût optimal $\boldsymbol{V}$ en couleur dans la région définie faisable par les *boundary surfaces* pour un maillage homogène de 51.

La figure 3.14 permet de comparer une programmation dynamique avec l'outil des *boundary surfaces* en bleu par rapport à une programmation dynamique classique en noir. Pour comparer les deux solutions, deux critères sont regardés : la qualité du résultat puis le temps de calcul. Comme ici l'algorithme avec les *boundary surfaces* est plus à gauche que l'algorithme classique, cela veut dire que la qualité de ses résultats est meilleure. Pour un même temps de calcul, il est d'ailleurs préférable de choisir l'algorithme proposé car même si le maillage est plus grossier, la qualité du résultat est supérieure à l'algorithme de référence.

Cependant, une étude du temps de calcul opérée avec l'outil *profile viewer* de Matlab, table 3.6, permet de montrer que le temps de calcul est concentré essentiellement par les trois fonctions *boundary*, *griddata* et *inpolygon* de Matlab qui permettent de mettre en place l'outil des *boundary surfaces*. Cette table affiche le temps de calcul consacré aux trois fonctions

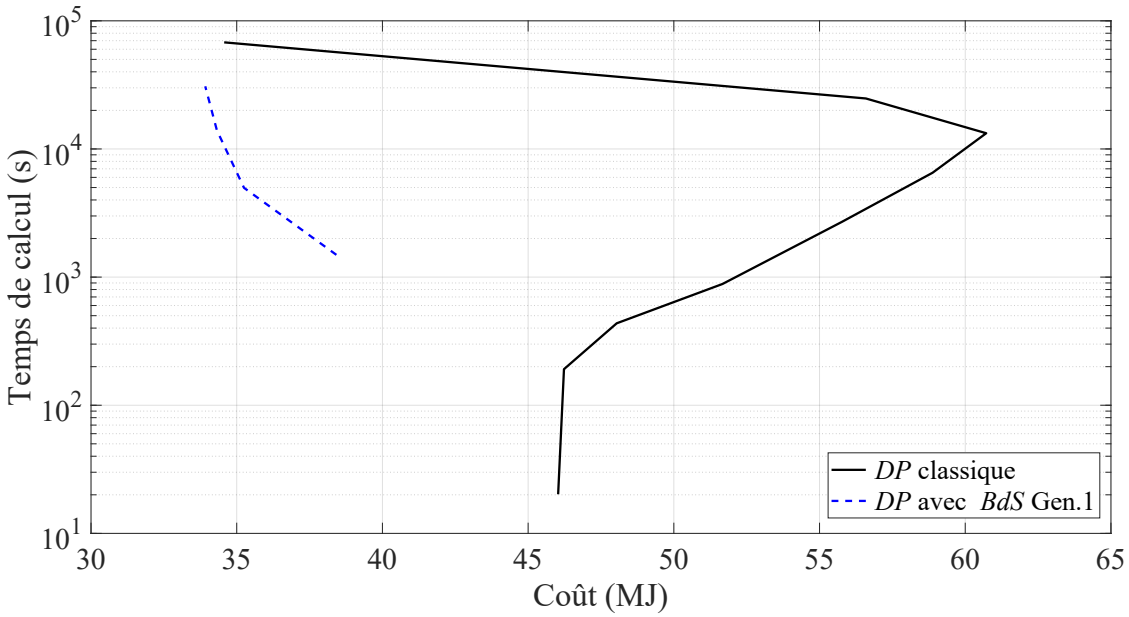

FIGURE 3.14 – Temps de calcul en fonction du coût pour les conditions initiales particulières pour le problème avec contraintes finales (3.19) de la génération 1 des *boundary surfaces* et de la programmation dynamique classique.

principales ainsi qu'une répartition du temps de calcul total entre ces trois fonctions. Cette analyse est effectuée pour différents maillages homogènes pour estimer si les résultats obtenus pour un maillage se répètent pour les autres maillages. Les résultats montrent que plus le maillage est fin, plus le temps de calcul de chaque fonction est élevé ; ce qui est attendu. Cependant, la répartition du temps de calcul évolue fortement en fonction du maillage. En effet, la fonction *boundary* est la plus consommatrice pour les faibles maillages (près de 50 % du temps de calcul total) mais cette répartition du temps de calcul diminue avec les maillages plus fins. Cela s'explique par une évolution plus forte du temps de calcul des fonctions *griddata* et *inpolygon*. Là où la fonction *inpolygon* demandait 40 % du temps de calcul total pour le maillage homogène à 21 mailles, la fonction nécessite plus de 55 % du temps de calcul pour les maillages homogènes supérieurs ou égaux à 31 mailles. Cela fait de *inpolygon* la fonction la plus consommatrice en temps de l'algorithme.

Ces résultats sont obtenus pour un nombre de points maximal sur le contour $N_{max} = 100$ points. Les résultats et temps de calculs montrés dans cette partie sont également dépendant de ce paramètre. L'annexe A illustre une étude de sensibilité à ce paramètre.

Malgré les bons résultats de l'amélioration *boundary surfaces*, le temps de calcul a été augmenté d'un facteur proche de 75 pour un même maillage. Pour réduire ce temps de calcul, deux axes de travail sont à explorer : remplacer la fonction *boundary* par une nouvelle fonction qui permet de générer le contour des surfaces plus rapidement ; remplacer la fonction d'interpolation utilisée. En effectuant ce travail, en plus de la réduction de temps de calcul, une généralisation de la méthode à plus de deux états est également visée. Il n'est effectivement

TABLE 3.6 – Analyse du temps de calcul des trois fonctions les plus chronophages pour la programmation dynamique avec l'amélioration des *boundary surfaces* pour différents maillages homogènes permanents.

Maillage		Autres	*boundary*	*griddata*	*inpolygon*	Total
21	temps (s)	41,9	732,7	110,7	631,4	1516,7
	Répartition (%)	2,8	48,3	7,3	41,6	-
31	temps (s)	138,1	2231,5	796,2	2595	5760,8
	Répartition (%)	2,4	38,7	13,8	45,0	-
41	temps (s)	308,2	3720,1	2284,4	7655,3	13968
	Répartition (%)	2,2	26,6	16,4	54,8	-
51	temps (s)	659,4	6069,9	6439,7	18104	31273
	Répartition (%)	2,1	19,4	20,6	57,9	-

pas possible d'utiliser la version de l'algorithme pour des systèmes à quatre états ou plus à cause des limitations des fonctions *boundary* et *griddata* qui ne peuvent être utilisées que dans des espaces à deux ou trois dimensions.

Malgré le temps de calcul plus important pour la fonction *inpolygon*, c'est la fonction *boundary* est la première à être remplacée. En effet, le temps de calcul de la fonction *inpolygon* est dépendant de celui de la fonction *boundary*. Plus le nombre de points déterminé par la fonction *boundary* est important, plus le temps consacré aux deux autres fonctions (*griddata* et *inploygon*) sera important. C'est pourquoi, le remplacement de la fonction *boundary* donne lieu à la seconde génération des *boundary surfaces* qui est détaillée dans la sous-section suivante.

3.2.2 Réduction du temps de calcul par une nouvelle fonction de génération des surfaces

L'unique différence entre la première et la seconde génération des *boundary surfaces* est la fonction utilisée pour déterminer le contour (3.23). Cela comporte deux objectifs : réduire le temps total du calcul mais également d'amorcer la généralisation de l'outil à n états. En effet, avec les fonctions utilisées dans la première génération, il est possible d'appliquer l'algorithme 5 pour un problème à trois états car les fonctions *boundary*, *griddata* et *inpolygon* fonctionnent en deux et trois dimensions mais pas au-delà. C'est pourquoi, la fonction appelée "contour" présentée par la suite est construite pour déterminer le contour d'un ensemble dans un espace à n dimensions.

Cette fonction repose alors sur une exploration dans l'espace des états. Pour déterminer les points décrivant le contour de la surface, la méthode de recherche suivante est appliquée :

- Déterminer le barycentre des points de recherche en calculant la moyenne des coordonnées des points pour chaque dimension.
- Définir des cônes de recherches d'un angle α à partir du barycentre
- Sélectionner parmi les points de recherche dans le cône celui qui est à la distance la plus grande du barycentre.

— Faire la sélection pour tous les cônes qui, ensemble, décrivent l'espace de recherche.

Dans la méthode de recherche décrite ci-dessus, le choix de l'angle α n'est pas aléatoire. En effet, α est calculé suivant l'équation (3.24).

$$\alpha = \frac{2.\pi}{N_{max}} \tag{3.24}$$

Au plus N_{max} sera grand, au plus α sera petit, ce qui aura pour effet de rendre l'ensemble plus compact tandis qu'avec un nombre de points plus faible, le comportement du contour se rapproche d'un ensemble convexe. Il est cependant important d'avoir un nombre de points suffisamment grand pour avoir le plus d'informations utiles que possible, sans augmenter grandement le temps de calcul.

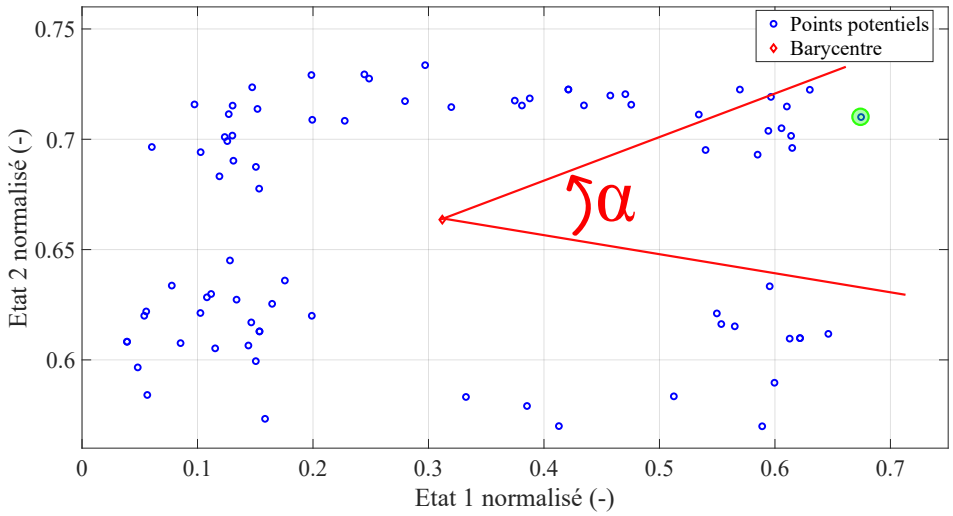

FIGURE 3.15 – Schéma de fonctionnement de la fonction contour avec la fonction proposée.

La figure 3.15 schématise la procédure décrite précédemment qui permet l'identification des points sur la frontière du contour. Les points potentiels Q_k sont représentés en bleu et leur barycentre b en rouge. Les points bleus existant dans le cône rouge décrit par l'angle α appartiennent à l'ensemble Γ et sont notés p_k. Parmi tous ces points p_k, celui qui est le plus loin en vert, appartient à l'ensemble $\boldsymbol{X}_k$. L'algorithme de la fonction est détaillé dans l'algorithme 6.

Algorithme 6 Algo remplacement fonction *boundary* appelée fonction contour

$\alpha = \frac{2.\pi}{N_{max}}$
Pour $i = 1:1:N_{max}$ **Faire**
  $p_k = Q_k \in \Gamma_i$
  $d = \sqrt{(p_k^{x_1} - b^{x_1})^2 + (p_k^{x_2} - b^{x_2})^2}$
  $\boldsymbol{X}_k(i) = p_k(argmax(d))$
Fin de Pour

Pour appliquer cette fonction de recherche pour un problème à trois états, il suffit d'ajouter un nouvel angle se comportant comme α et sur un plan orthogonal à celui sur lequel se trouve α pour générer des cônes de recherches en trois dimensions. Ainsi dans chaque cône, le point le plus loin est détecté et appartient au contour $\boldsymbol{X}_k$. La généralisation de la fonction contour à n états consiste alors à déterminer le point le plus loin de l'hyper-cône déterminé à partir des $n-1$ angles de détection, toujours sur des plans orthogonaux.

Étant donné que cette méthode de détection des points sur le contour fonctionne de manière localisée (un cône de détection après l'autre sans recouvrement), une protection est effectuée pour éviter que des trous apparaissent dans le contour. Une première détection des points est réalisée avec les contraintes du système (carré en vert sur la figure 3.16) et une seconde avec un dépassement léger des contraintes (carré en bleu). Si dans un cône de détection il existe des points à l'intérieur du carré bleu et à l'extérieur du vert, le point le plus proche représenté en jaune est sélectionné pour conserver de l'information dans cette région.

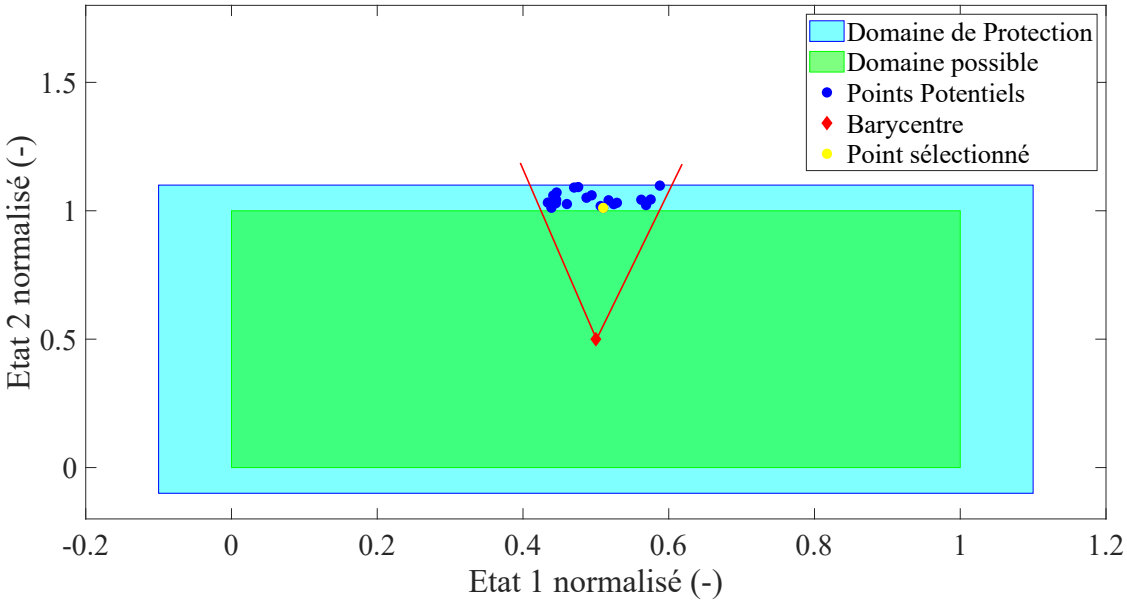

FIGURE 3.16 – Fonctionnement de la protection par double détection du contour.

Les résultats obtenus par la programmation avec l'outil seconde génération sont affichés en tables 3.7 à 3.9. La première table explicite le temps passé dans chacune des trois boucles de calcul tandis que la dernière indique le temps utilisé par chaque fonction importante. Le temps a été réduit dans les deux premières boucles. La réduction de temps dans la première boucle s'explique par l'utilisation de la nouvelle fonction contour. Dans le *backward*, la réduction de temps provient de la rapidité d'exécution des fonctions *griddata* et *inpolygon* car la quantité de points sur les contours a diminué. Cette diminution est liée à l'utilisation de la fonction contour. Le temps de calcul de la boucle *boundary surfaces* n'évolue que très peu avec le raffinement du maillage. La deuxième table explicite les résultats obtenus à bilan batterie nul.

TABLE 3.7 – Résultats obtenus avec une Programmation Dynamique avec la seconde génération (Gen.2) des *boundary surfaces* pour le problème avec contraintes finales pour les conditions initiales particulières.

Maillage (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	Temps de calcul (s)			
					BdS	*BW*	*FW*	Total
21	39,63	0,7248	38,2	0,06	17,7	634,7	0,7	653,1
31	35,46	0,6780	37,9	0,17	28,3	2701,9	0,6	2730,8
41	34,33	0,6642	37	0,45	54,2	8811,3	0,7	8866,3
51	33,94	0,6566	38	0,98	83,9	22114	0,7	22199

TABLE 3.8 – Résultats obtenus avec une Programmation Dynamique avec la seconde génération (Gen.2) des *boundary surfaces* pour le problème avec contraintes finales pour le bilan batterie nul.

Maillage (-)	Coût BBN (MJ)	$T_{Mel_N}BBN$ (°C)
21	37,5	37,5
31	35,56	37,2
41	34,37	37,2
51	34,1	37,9

TABLE 3.9 – Résultats du *profile viewer* pour la programmation dynamique avec la seconde génération de l'outil *boundary surfaces* pour différents maillages homogènes permanents.

Maillage		Autres	contour	*griddata*	*inpolygon*	Total
21	temps (s)	42,5	12,4	137,2	538,2	730,3
	Répartition (%)	5,8	1,7	18,8	73,7	-
31	temps (s)	135,3	21,6	814,9	2006,8	2978,6
	Répartition (%)	4,5	0,7	27,4	67,4	-
41	temps (s)	318,2	39,1	2446,4	6353,9	9157,6
	Répartition (%)	3,5	0,4	26,7	69,4	-
51	temps (s)	668,2	58,9	6510,9	15835	23073
	Répartition (%)	2,9	0,3	28,2	68,6	-

Il est possible de voir que le temps de calcul est réduit grâce à l'utilisation de la nouvelle fonction contour. En effet, la fonction contour consomme uniquement 2 % du temps de calcul au maillage le plus grossier et cette part du temps de calcul diminue également avec l'augmentation du nombre de mailles. Cela s'explique par le temps de calcul de la fonction qui n'augmente que très peu avec l'accroissement du nombre de mailles comparé aux deux autres fonctions principales. Afin d'obtenir les résultats présentés ici, le choix de la seconde détection de protection a été appliquée suivant les valeurs indiquées dans la figure 3.16 (entre 0 et 1 pour le domaine possible et entre $-0,1$ et $1,1$ pour le domaine de protection). Les résultats illustrés par les tables 3.7 et 3.8 et la figure 3.17 montrent des résultats entre les deux générations de l'amélioration très similaires. En effet, le temps de calcul plus faible (réduction du temps de calcul entre 60% et 30%) pour la seconde génération pour une qualité

de résultat identique. L'utilisation de la fonction contour permet ainsi d'avoir une matrice de coût optimal (figure figure 3.18) très proche de celle obtenue avec la premier génération des *boundary surfaces* (figure 3.13). La figure B.2 en annexe B présente l'évolution du temps de calcul en fonction du coût à bilan batterie nul.

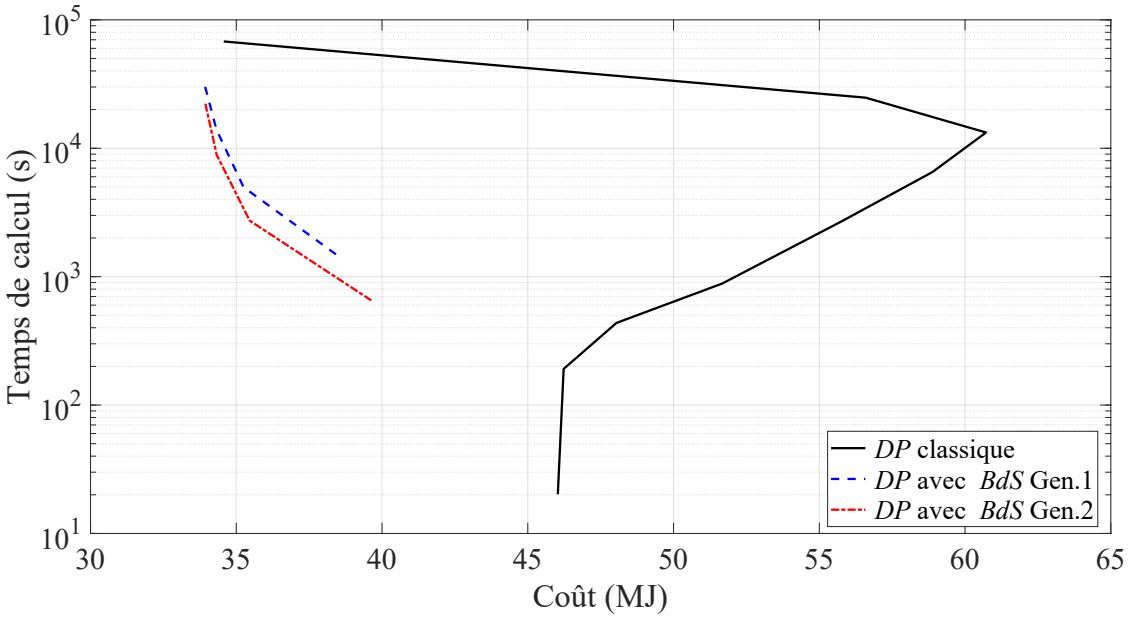

FIGURE 3.17 – Temps de calcul en fonction du coût pour les conditions initiales particulières pour le problème avec contraintes finales (3.19) des générations 1 et 2 des *boundary surfaces* et de la programmation dynamique classique.

La figure 3.19 présente les trajectoires des quatre maillages avec la nouvelle version de l'amélioration des *boundary surfaces*. En comparaison avec la première version de l'algorithme, les trajectoires à iso-maillage sont très différentes tout en ayant une qualité de résultat proche. Cela implique donc que plusieurs chemins mènent à un résultat similaire.

Dans l'objectif de toujours réduire le temps de calcul de l'outil, qui commence à être compétitif, une troisième génération de *boundary surfaces* est mise au point. Cette troisième génération doit se détacher de la fonction d'interpolation *griddata* pour réutiliser la fonction d'interpolation bilinéaire *interp2*. Le développement de la méthode est détaillé dans la sous-section suivante.

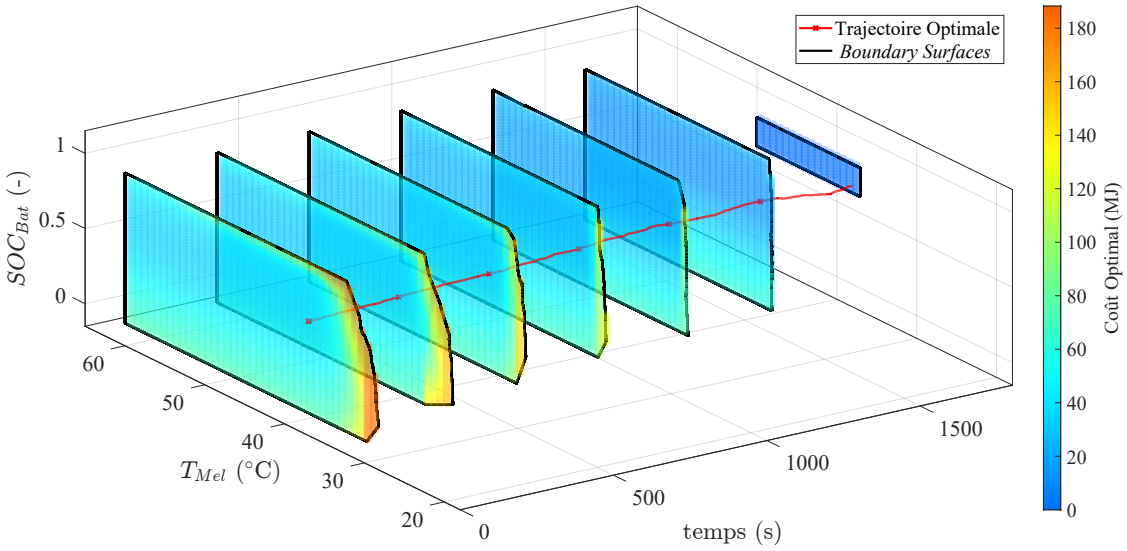

FIGURE 3.18 – Trajectoire de la solution trouvée par la programmation dynamique en rouge avec des tranches de la matrice de coût optimal $\boldsymbol{V}$ en couleur dans la région définie faisable par les *boundary surfaces* seconde génération pour un maillage homogène de 51.

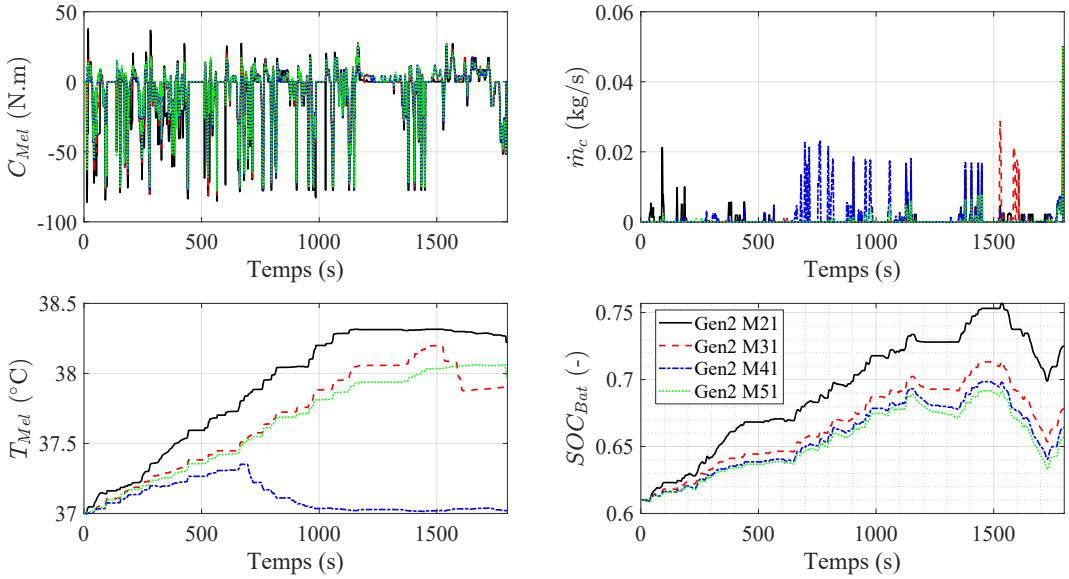

FIGURE 3.19 – Trajectoires des états et des commandes des trois solutions de *DP* avec les *BdS* seconde génération pour les maillages de 21, 31, 41 et 51 mailles par variables respectivement en noir, rouge, bleu et vert pour le problème (3.20) avec contraintes finales (3.19).

3.2.3 Réduction du temps de calcul par une différente méthode d'interpolation

Pour remplacer la fonction *griddata*, il faut repenser la manière d'intégrer les informations contenues par les *boundary surfaces* au *backward* et plus précisément à la matrice de coût optimal et aux matrices de commandes optimales. Pour rappel, l'objectif premier des *boundaries* est d'empêcher la contamination de la matrice de coût optimal à cause des pénalités d'infaisabilité. Afin de réduire le temps de calcul, il est proposé ici d'exploiter l'information du contour uniquement comme vérification de la non contamination de la matrice de coût optimal. De cette manière, il n'est plus nécessaire d'intégrer les points du contour aux interpolations et il est alors possible de réutiliser les interpolations bilinéaires comme une programmation dynamique "classique" à la place de la fonction *griddata*.

L'idée est alors de faire deux calculs en parallèle et de les comparer. Le premier provient de la programmation dynamique usuelle de l'estimation de $\boldsymbol{V}$ à l'instant $k+1$, le second est une estimation de ce même *cost to go* en utilisant uniquement les points du contour les plus proches. Cette estimation est faite en utilisant une interpolation de Pondération Inverse à la Distance (PID) ou *Inverse Distance Weighting* (*IDW*) en anglais qui est définie par les équations (3.25) et (3.26). La valeur estimée $V_{estimée}$ est une somme de produits des valeurs b_i des n_p points considérés avec leur importance p_i. L'importance est calculée en se basant sur la distance du point d_i au point estimé par rapport à la somme de toutes les distances des n_p points.

$$V_{estimée} = \sum_{i=1}^{n_p} p_i.b_i \tag{3.25}$$

$$p_i = \frac{d_i}{\sum_{i=1}^{n_p} d_i} \tag{3.26}$$

Afin d'utiliser cette interpolation sur des points proches de celui estimé uniquement, un paramètre appelé rayon de détection d_d est alors introduit. Ce paramètre permet de n'utiliser que les points du contour qui ont une distance inférieure au rayon de détection dans l'*IDW*. Après comparaison des deux estimations, deux cas sont possibles : dans le premier, le résultat de la *DP* classique est meilleur, d'un point de vue du critère d'optimisation, que celui donné par l'*IDW*. Dans ce cas, il est estimé que le point n'était pas contaminé et que l'utilisation des informations du contour n'était pas nécessaire. Dans le second cas, le résultat donné par l'*IDW* est meilleur que celui donné par la *DP* classique. Alors, on considère que le point est contaminé et qu'il faut remplacer la solution déterminée par la *DP* classique par celle donnée par l'*IDW*. Les commandes optimales à sauvegarder sont estimées avec la même méthode d'interpolation qui remplace également les commandes optimales déterminées par la *DP* classique. Finalement, les matrices de coût et de commandes optimales sont protégées des contaminations tout en ayant un temps de calcul réduit. La fonction *inpolygon* est toujours utilisée malgré son temps de calcul important dans les générations précédentes car le temps de calcul nécessaire dans cette génération est très acceptable. Le nombre de points évalués est drastiquement réduit car au lieu d'évaluer si les points potentiels à l'indice de temps $k+1$

dans le *backward* existent ou pas dans le contour $\boldsymbol{X}_{k+1}$, ici seuls les points du maillage sont évalués quant à leur existence ou non dans le contour $\boldsymbol{X}_k$.

La figure 3.20 illustre le fonctionnement de l'utilisation des *boundary surfaces* avec l'interpolation *IDW*. Dans l'exemple présenté, les points du maillage à l'intérieur du contour des *boundary surfaces* sont représentés avec les points bleus. Afin de vérifier que la matrice de coût optimal n'est pas contaminée pour ces points là uniquement, une estimation basée sur l'information du contour des *boundary surfaces* à l'indice de temps k est réalisée. Afin de n'utiliser que des points qui ont un sens, un cercle basé sur le rayon de détection en vert est tracé à partir de tous les points. Dans l'exemple, seul le point estimé en rouge est illustré. Tous les points du contour en noir qui sont dans ce cercle de détection sont utilisés pour l'interpolation *IDW*. Ces points sont représentés par les carrés cyan. Une valeur du coût optimal est alors estimée $\boldsymbol{V}_{estimée}(k+1, \boldsymbol{x}_{k+1})$ et comparée à celle calculée par la programmation dynamique $\boldsymbol{V}_{DP}(k+1, \boldsymbol{x}_{k+1})$. Deux résultats sont possibles à cette comparaison. Si $\boldsymbol{V}_{DP}(k+1, \boldsymbol{x}_{k+1}) \leq \boldsymbol{V}_{estimée}(k+1, \boldsymbol{x}_{k+1})$, alors la valeur $\boldsymbol{V}_{DP}$ est considérée comme non contaminée. Dans le cas contraire, $\boldsymbol{V}_{DP}(k+1, \boldsymbol{x}_{k+1}) > \boldsymbol{V}_{estimée}(k+1, \boldsymbol{x}_{k+1})$, la valeur $\boldsymbol{V}_{DP}(k+1, \boldsymbol{x}_{k+1})$ est considérée comme contaminée et il faut utiliser la valeur $\boldsymbol{V}_{estimée}(k+1, \boldsymbol{x}_{k+1})$ à la place dans la matrice de coût optimal $\boldsymbol{V}(k+1, \boldsymbol{x}_{k+1})$. Des interpolations *IDW* sont également réalisées si besoin pour la détermination des valeurs de commandes optimales à stocker dans les matrices de commandes optimales.

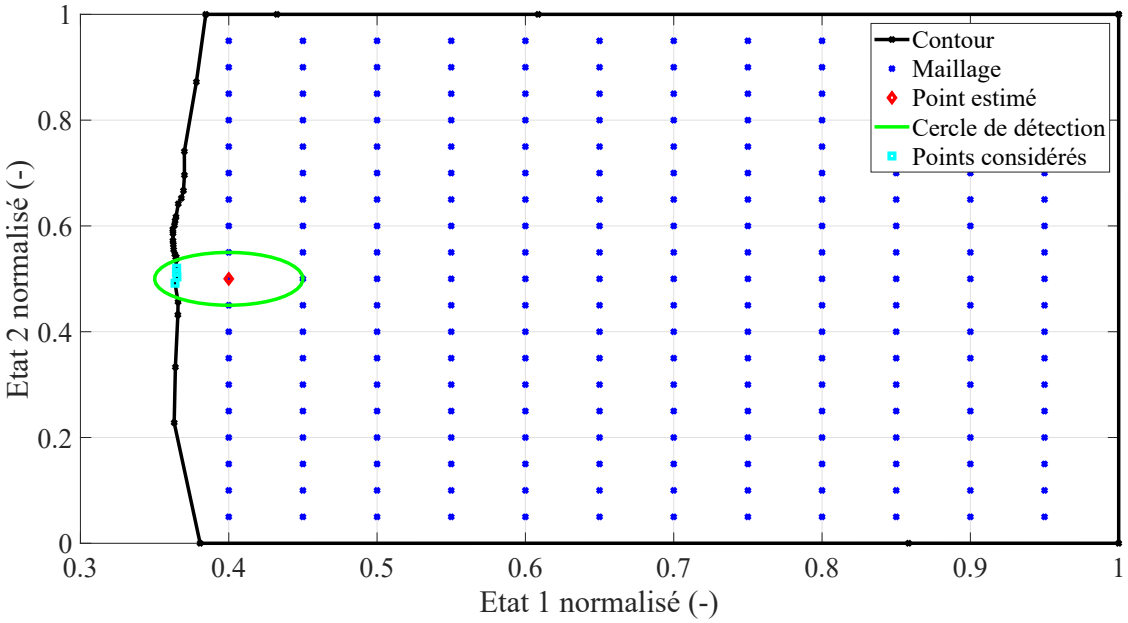

FIGURE 3.20 – Fonctionnement de la troisième génération des *boundary surfaces* : utilisation de l'*IDW* durant le *backward* de la programmation dynamique.

La figure 3.21 illustre l'allure des *boundary surfaces* troisième génération qui sont identiques à celle de la seconde génération. La différence entre les deux versions de l'algorithme qui réside dans la méthode d'interpolation utilisée dans le *backward* est néanmoins visible sur la figure ci-dessus. En effet, les valeurs dans la matrice de coût optimal à l'instant initial sont

moins élevées aux abords du contour pour les basses températures de la machine électrique.

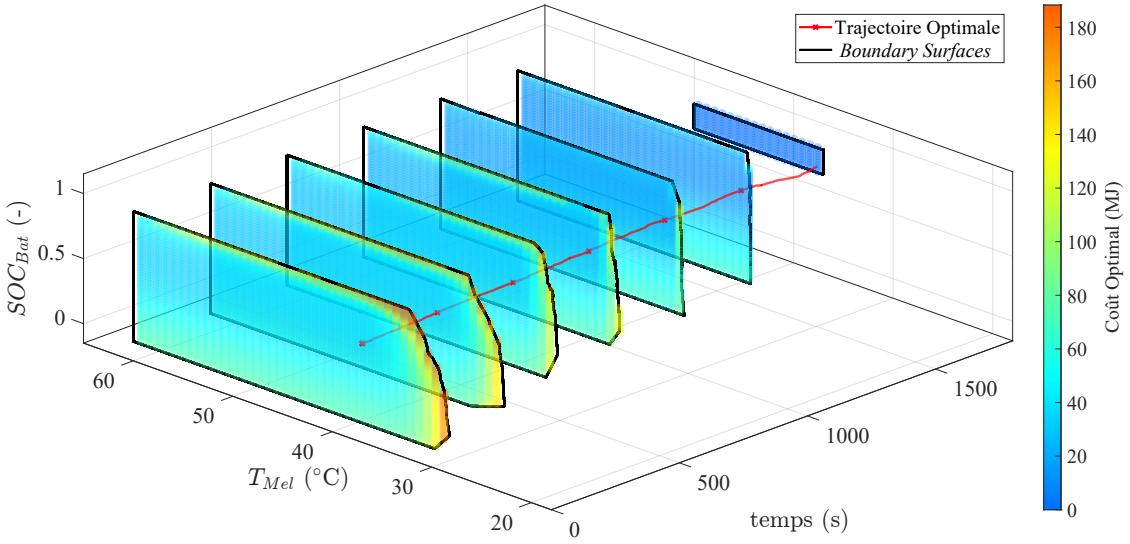

FIGURE 3.21 – Trajectoire de la solution trouvée par la programmation dynamique en rouge avec des tranches de la matrice de coût optimal $\boldsymbol{V}$ en couleur dans la région définie faisable par les *boundary surfaces* troisième génération pour un maillage homogène de 51.

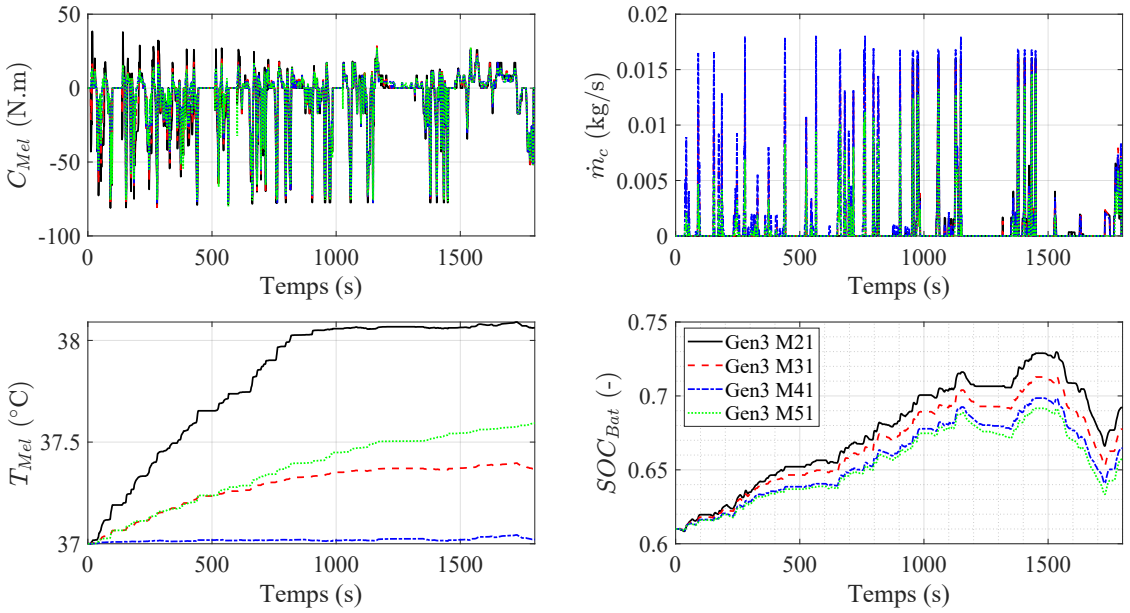

FIGURE 3.22 – Trajectoires des états et des commandes des trois solutions de *DP* avec les *BdS* troisième génération pour les maillages de 21, 31, 41 et 51 mailles par variables respectivement en noir, rouge, bleu et vert pour le problème (3.20) avec contraintes finales (3.19).

Les trajectoires obtenues pour les quatre maillages (figure 3.22) diffèrent légèrement des trajectoires précédentes. En effet, la minimisation du critère de minimisation est effectuée

majoritairement en réduisant la quantité d'énergie stockée par la batterie le long du cycle. La gestion de la température de la machine électrique est alors effectuée en conséquence de l'utilisation de cette dernière pour minimiser au global le critère.

TABLE 3.10 – Résultats obtenus avec une Programmation Dynamique avec la troisième génération (Gen.3) des *boundary surfaces* pour le problème avec contraintes finales pour les conditions initiales particulières.

Maillage (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	Temps de calcul (s)			
					BdS	*BW*	*FW*	Total
21	38,20	0,6921	38,1	0,05	18	29,5	0,6	48,2
31	35,77	0,6775	37,4	0,15	30,7	97,7	0,6	129
41	34,32	0,6646	37	0,38	52,1	257,8	0,6	310,5
51	33,92	0,6571	37,6	0,82	82,3	584,1	0,7	667,1

TABLE 3.11 – Résultats obtenus avec une Programmation Dynamique avec la troisième génération (Gen.3) des *boundary surfaces* pour le problème avec contraintes finales pour le bilan batterie nul.

Maillage (-)	Coût BBN (MJ)	T_{Mel_N} BBN (°C)
21	38,2	38,1
31	35,67	37,4
41	34,34	37,1
51	34,05	37,6

Les résultats affichés par les tables 3.10 et 3.11 et la figure 3.23 illustrent l'efficacité de la méthode. En effet, le temps nécessaire à l'ensemble de l'algorithme est du même ordre de grandeur qu'une programmation dynamique "classique" pour des résultats bien meilleurs. Les résultats obtenus avec cette troisième génération sont encore plus performants que ceux de la seconde génération. La figure B.3 en annexe B présente l'évolution du temps de calcul en fonction du coût à bilan batterie nul.

L'analyse de la répartition du temps de calcul entre les différentes fonctions principales affichées table 3.12 montre l'optimisation du temps de calcul. En effet, nous retrouvons le fait que la génération des contours à maillage grossier est la plus grande dépense de temps et cette répartition se réduit pour des maillages plus fins. Les fonctions utilisées pour les interpolations sont des consommateurs de temps très réduit par rapport à la première génération de l'amélioration. Cela implique donc que pour minimiser le temps de calcul, il faudrait maintenant identifier les nouveaux consommateurs principaux de temps.

Pour totalement généraliser l'outil des *boundary surfaces* à n dimensions, il faut retirer la fonction *inpolygon*. Cette fonction qui permet de déterminer si un point est dans une surface ou un volume peut être remplacer par une approximation du contour par un parallélotope droit définit par les valeurs extrémales du contour dans chaque direction.

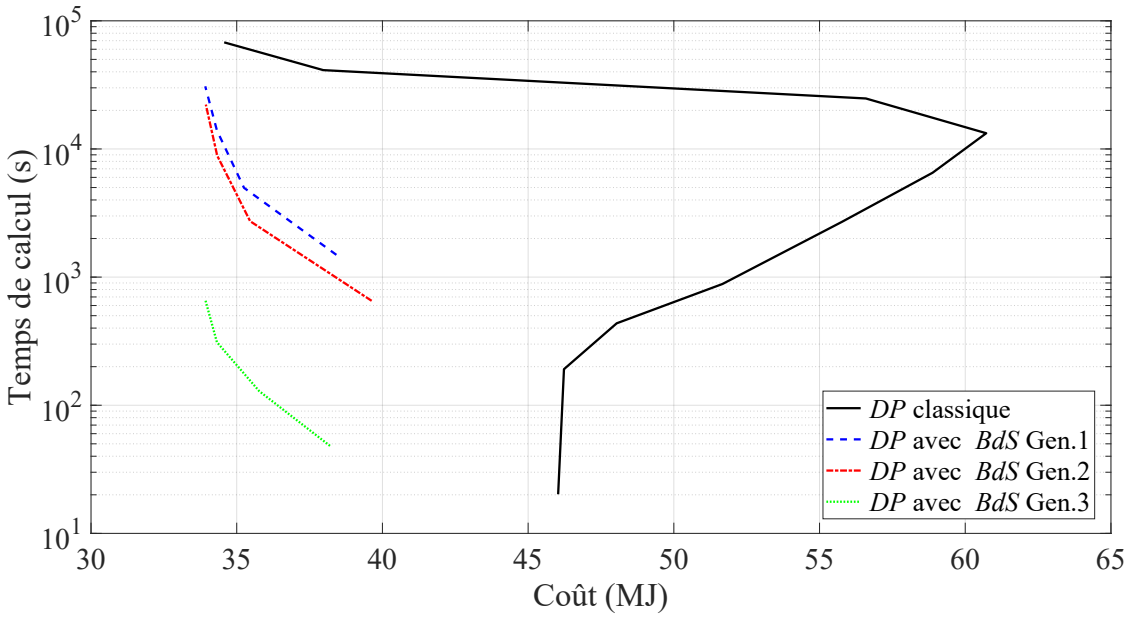

FIGURE 3.23 – Temps de calcul en fonction du coût pour les conditions initiales particulières pour le problème avec contraintes finales (3.19) des générations 1 à 3 des *boundary surfaces* et de la programmation dynamique classique.

TABLE 3.12 – Résultats du *profile viewer* pour la programmation dynamique avec la troisième génération de l'outil *boundary surfaces* pour différents maillages homogènes permanents.

Maillage		Autres	contour	*interp2*	*inpolygon*	Total
21	temps (s)	39,7	12,2	3,8	1,9	57,6
	Répartition (%)	68,9	21,2	6,6	3,3	-
31	temps (s)	107,5	21,4	13,1	3,0	145,0
	Répartition (%)	74,1	14,8	9,0	2,1	-
41	temps (s)	266,9	41,9	38,0	5,1	351,9
	Répartition (%)	75,8	11,9	10,8	1,4	-
51	temps (s)	558,8	61,4	93,7	7,8	721,7
	Répartition (%)	77,4	8,5	13,0	1,1	-

3.3 Mise en place des maillages hétérogènes

Dans la première partie de ce chapitre, il a pu être montré la dépendance du résultat donné par la programmation dynamique et le maillage utilisé. Seulement des maillages homogènes avaient été considérés pour ne pas impliquer des biais dans le résultat des maillages afin de comparer les différentes solutions équitablement. Maintenant qu'une assurance de l'optimalité est établie, il est intéressant de réduire la complexité de calcul à travers une répartition des ressources de calcul en adéquation avec le problème et le système considéré. Pour mener cette étude, les maillages considérés sont qualifiés de permanents et uniformes selon le lexique mis en place au chapitre 1. Pour mener cette étude, plusieurs calculs de programmation

dynamique seront réalisés avec différents maillages. En exploitant les résultats obtenus, il sera ainsi possible de déterminer les maillages les plus avantageux.

Étant donné que les résultats de la programmation dynamique classique pour le problème avec contraintes finales sont contaminés par les contraintes finales, une première étude de la répartition des ressources de calcul sera menée sur le même problème mais sans contraintes finales. Ainsi, il sera possible de voir si un maillage hétérogène sera capable d'endiguer le phénomène de contamination de la matrice de coût optimal. Enfin, des maillages hétérogènes seront également utilisés avec l'amélioration des *boundary surfaces* pour le bénéfice d'utiliser ces deux leviers pour obtenir le compromis temps de calcul et qualité de résultat le plus intéressant possible.

3.3.1 Étude sans l'amélioration *boundary surfaces*

3.3.1.1 Problème sans contraintes finales

Pour commencer, il faut introduire le problème sans contraintes finales. Ce dernier s'écrit de la même façon que celui avec contraintes finales (3.20). La modification entre les deux problèmes réside dans la valeur de $\Phi(\boldsymbol{x}_N)$. Ainsi le problème sans contraintes finales se doit de respecter uniquement les contraintes appliquées aux états à chaque instant. L'écriture du terme de contraintes finales est alors la suivante :

$$\Phi\left((0 \leq SOC_{Bat_N} \leq 1) \ \times \ (20 \leq T_{Mel_N} \leq 60)\right) = 0 \tag{3.27}$$

Naturellement les solutions qui mènent à un non respect de ces contraintes sont écartées car les interpolations bilinéaires ne permettent pas d'évaluer $\boldsymbol{V}(k+1, \boldsymbol{x}_{k+1})$ étant donné que l'état futur n'est pas dans le maillage considéré.

Premièrement, il faut s'assurer que la matrice de coût optimale n'est pas contaminée. Pour ce faire, plusieurs maillages homogènes uniformes permanents sont essayés. Les tables 3.13 et 3.14 montre que les coûts diminuent avec le raffinage du maillage.

Table 3.13 – Résultats obtenus avec une Programmation Dynamique "classique" sans contraintes finales.

Maillage (-)	Coût (MJ)	Tps calc (s)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)
21	45,95	20,2	0,5041	28,3	0,04
41	39,66	191,3	0,5188	27,9	0,35
51	37,96	435,4	0,5478	28,1	0,77
101	32,89	6576	0,6296	29,8	10,48

TABLE 3.14 – Résultats obtenus avec une Programmation Dynamique "classique" sans contraintes finales.

Maillage (-)	Coût à BBN (MJ)	T_{Mel_N}àBBN (°C)
21	46,22	28,6
41	38,34	28,1
51	34,99	28,9
101	34,99	28,9

De plus, les figures 3.24 à 3.27 permettent d'illustrer la décontamination de la matrice de coût optimal avec le raffinage du maillage. En effet, les tranches de matrice de coût optimal sont toutes orange dans le cas le moins maillé car une saturation d'affichage est mise à 100 MJ. Lorsque le maillage est raffiné, une bande bleue commence à apparaître sur toutes les tranches de matrice. Cette bande bleue correspond aux réelles valeurs du critère d'optimisation. Plus le maillage est raffiné, plus la zone à exploiter augmente. A 101 mailles par variables, le point des conditions initiales particulières existe dans la zone non contaminée, ce qui est corroboré par le coût obtenu qui est à l'optimal.

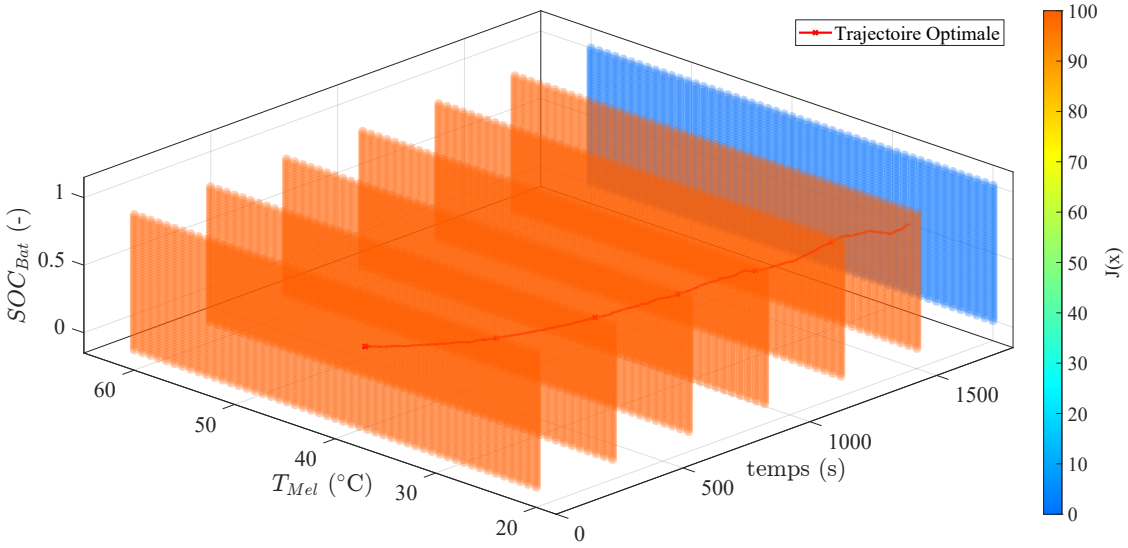

FIGURE 3.24 – Maillage homogène à 21 mailles

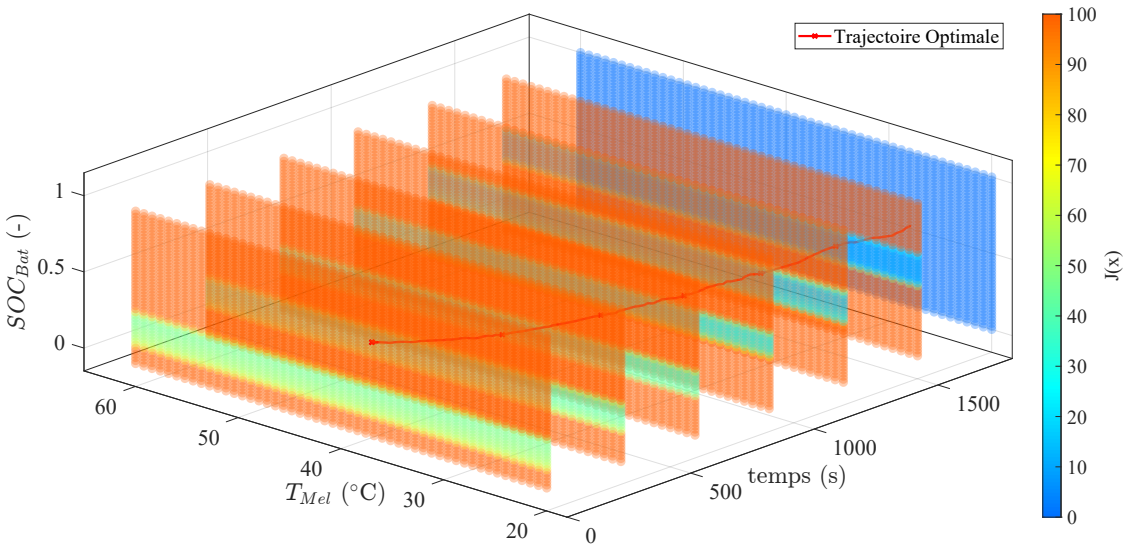

FIGURE 3.25 – Maillage homogène à 41 mailles

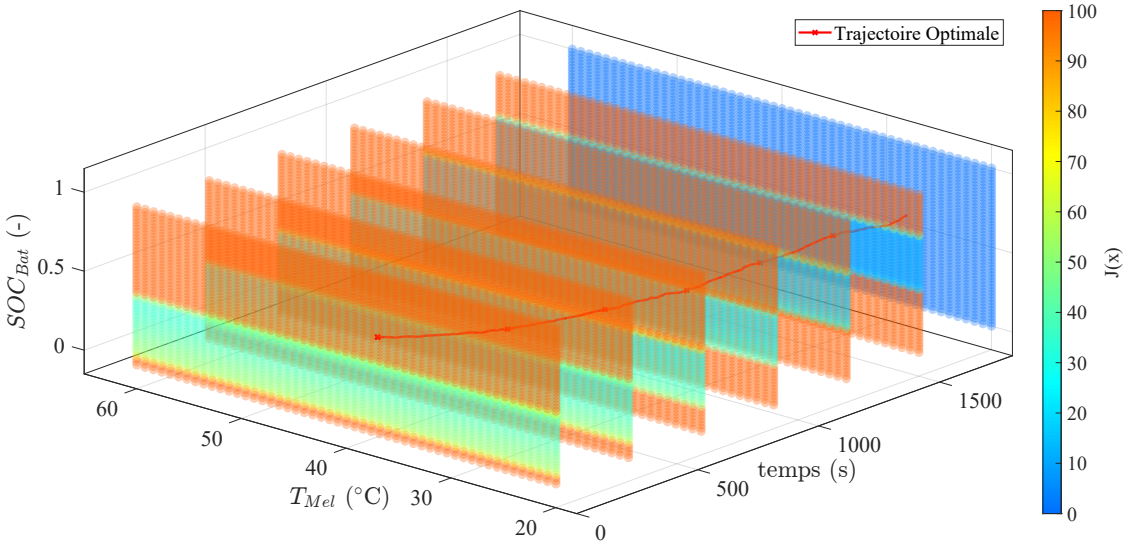

FIGURE 3.26 – Maillage homogène à 51 mailles

Les trajectoires affichées en figure 3.28 diffèrent beaucoup du problème avec contraintes. En effet, la grande différence entre les deux variantes du problème réside dans la gestion de la température T_{Mel}. Lorsqu'il n'y a pas de contraintes finales, l'algorithme sollicite énormément la commande $\dot{m}_c$ du débit de liquide de refroidissement pour abaisser la température de la machine électrique et l'amener vers la température voulue qui est de 30°C. La seconde différence est le bilan d'énergie déployé par la batterie. Le bilan batterie n'est plus forcément positif car il n'y a plus de contraintes. Alors l'algorithme exploite les deux moteurs de la chaîne de traction de telle manière à minimiser le critère d'optimisation.

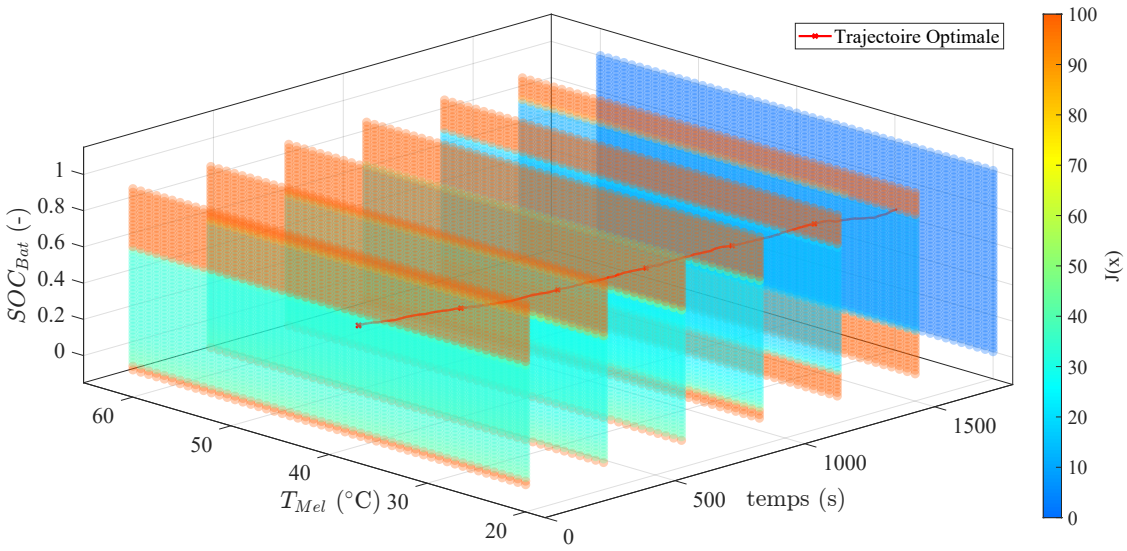

FIGURE 3.27 – Maillage homogène à 101 mailles

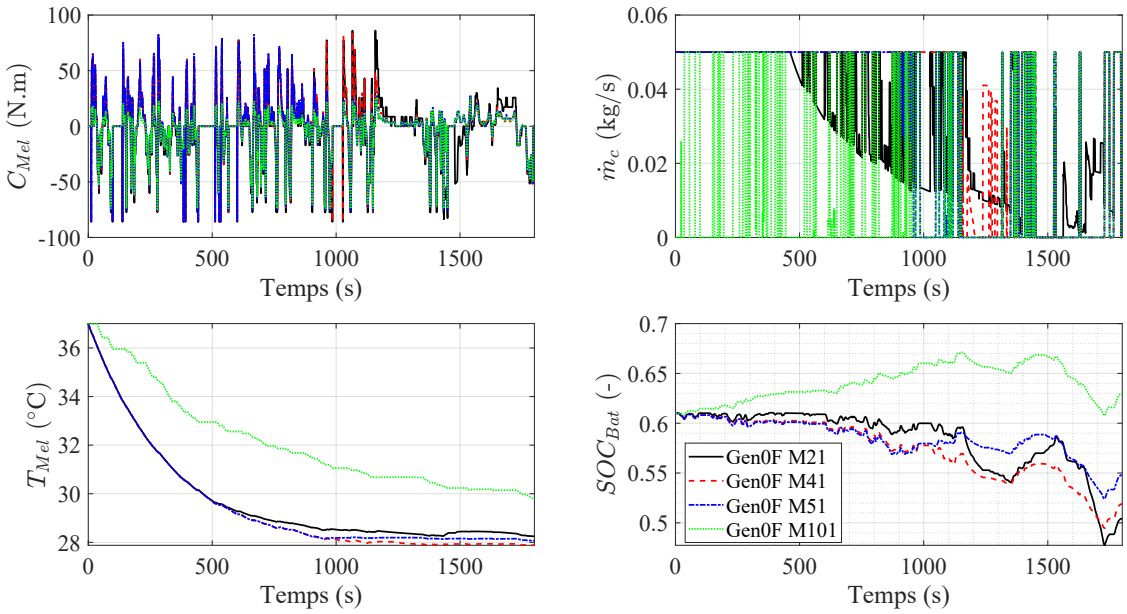

FIGURE 3.28 – Trajectoires des états et des commandes des quatre solutions de *DP* classique pour les maillages de 21, 41, 51 et 101 mailles par variables respectivement en noir, rouge, bleu et vert pour le problème (3.20) sans contraintes finales (3.27).

Pour mener l'analyse de sensibilité, uniquement le résultat de la condition initiale particulière est considérée. L'analyse de sensibilité, dont les résultats sont présentés dans la table 3.15, permet de mettre en évidence qu'il est préférable de raffiner le maillage pour la variable de commande C_{Mel} et pour la variable d'état SOC_{Bat}, avec une préférence pour la variable d'état. Ainsi, le maillage hétérogène proposé ($U1 = 81$, $U2 = 21$, $E1 = 21$, $E2 = 121$)

permet de diviser le temps de calcul par 20 tout en ayant une sous-optimalité de $0,5$ % par rapport à la référence.

TABLE 3.15 – Résultats obtenus à partir du maillage homogène de référence de 21 mailles par variables. Problème (3.20) sans les contraintes finales (3.27)

U1 C_{Mel}	U2 $\dot{m}_c$	E1 T_{Mel}	E2 SOC_{Bat}	Coût (MJ)	Erreur (%)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	Tps calc (s)	Nbre mailles	RAM (Go)
21	21	21	21	45,95	39,7	0,5041	28,3	20,2	1,94E+05	0,04
41	21	21	21	45,05	37,0	0,4795	28,7	32,4	3,80E+05	0,06
21	41	21	21	45,95	39,7	0,5041	28,25	32,6	3,80E+05	0,06
21	21	41	21	46,03	40,0	0,5038	27,7	33,3	3,80E+05	0,07
21	21	21	41	42,29	28,6	0,5250	28,2	33	3,80E+05	0,07
81	21	21	121	33,05	0,5	0,6310	29,8	283,2	4,32E+06	0,53
101	101	101	101	32,89	-	0,6296	29,8	6576	1,04E+08	10,48

Maintenant qu'une répartition des ressources numériques à été déterminée, est-ce que cette dernière permettra d'améliorer la qualité des résultats de la programmation dynamique "classique" sans utiliser l'amélioration des *boundary surfaces* ?

3.3.1.2 Problème avec contraintes finales

Afin de répondre à la question précédente, deux maillages hétérogènes différents sont appliqués au problème (3.20) avec contraintes finales (3.19).

TABLE 3.16 – Résultats obtenus à partir du maillage homogène de référence de 21 mailles par variables. Problème (3.20) avec les contraintes finales (3.19)

U1 C_{Mel}	U2 $\dot{m}_c$	E1 T_{Mel}	E2 SOC_{Bat}	Coût (MJ)	Erreur (%)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	Tps calc (s)	Nbre mailles	RAM (Go)
21	21	21	21	46,03	-	0,7156	38,6	20,2	1,94E+05	0,04
81	21	21	81	54,68	18,79	0,6781	39,4	196,5	2,89E+06	0,36
81	21	21	121	59,4	29,05	0,6786	40,4	285,8	4,32E+06	0,53
141	21	21	201	39,79	-13,56	0,6934	38,1	803,5	1,25E+07	1,40

La table 3.16 présente les résultats des maillages hétérogènes appliqués au problème avec les contraintes (3.19). Grâce aux indications données par l'étude de sensibilité faite pour le problème sans contraintes finales, il est possible d'obtenir un maillage hétérogène meilleur que le meilleur maillage homogène montré. Le dernier maillage hétérogène présenté s'approche des meilleurs résultats obtenus par la programmation dynamique classique avec un temps et une puissance de calcul divisés par 50. Cela démontre l'intérêt des maillages hétérogènes qui permettent d'obtenir des résultats de meilleure qualité comparé aux maillages homogènes pour un temps de calcul raisonnable.

Pour finir, il est pertinent de combiner l'amélioration des *boundary surfaces* et les maillages hétérogènes.

3.3.2 Étude avec l'amélioration *boundary surfaces*

Les résultats présentés dans cette section sont obtenus avec la dernière version de l'amélioration des *boundary surfaces*.

TABLE 3.17 – Résultats obtenus à partir du maillage homogène de référence de 21 mailles par variables. Problème (3.20) avec les contraintes finales (3.19).

U1 C_{Mel}	U2 $\dot{m}_c$	E1 T_{Mel}	E2 SOC_{Bat}	Coût (MJ)	Erreur (%)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	Tps calc (s)	Nbre mailles	RAM (Go)
21	21	21	21	38,20	6,8	0,6921	38,1	48,2	1,94E+05	0,05
41	21	21	21	36,90	3,2	0,6805	38,1	64,2	3,80E+05	0,07
21	41	21	21	38,20	6,8	0,6919	38,1	65,3	3,80E+05	0,07
21	21	41	21	36,60	2,3	0,6853	37	64,9	3,80E+05	0,09
21	21	21	41	39,60	10,7	0,6969	38,0	64,8	3,80E+05	0,09
41	21	41	21	34,40	-3,8	0,6719	37	102,3	7,41E+05	0,13
31	31	31	31	35,77	-	0,6775	37,4	129	9,24E+05	0,15

La table 3.17 présente une étude de sensibilité au maillage de la programmation dynamique avec la dernière version de l'amélioration des *boundary surfaces* (Gen.3) pour résoudre le problème (3.20) avec les contraintes finales (3.19). La table ci-dessus indique qu'un nombre de mailles doublé pour la commande du couple de la machine électrique C_{Mel} où l'état de la température de la machine électrique T_{Mel} permet d'améliorer significativement les résultats par rapport à la référence (21 mailles par variables). Cette indication n'est pas la même que celle obtenue pour la programmation dynamique classique. Une conclusion de l'étude est que l'étude de sensibilité réalisée dans des conditions spécifiques de calcul (avec ou sans amélioration) n'est pas transposable à d'autres conditions de calcul. En prenant en compte les indications obtenues et en appliquant un maillage hétérogène spécifique, il est possible d'obtenir la meilleure qualité de résultat présentée ici, tout en réduisant le temps de calcul de 20%.

Il est finalement possible de dresser une liste de préconisation pour mettre au point un maillage hétérogène pour un problème donné :

- L'étude de sensibilité doit être effectuée avec les pénalités finales du problème étudié (il n'est pas possible d'effectuer l'analyse de sensibilité avec le problème sans contraintes finales alors que c'est le problème avec contraintes finales qui doit être résolu).
- Il faut s'assurer que l'étude de sensibilité ne s'effectue pas dans la zone critique du maillage (augmenter le nombre de mailles revient à dégrader la qualité du résultat).
- Effectuer l'analyse de sensibilité
- Répartir les ressources numériques en fonction du résultat obtenu par l'analyse de sensibilité.

3.4 Conclusions et perspectives

Au cours de ce chapitre, deux axes d'amélioration de la programmation dynamique ont été explorés : la généralisation de l'amélioration des *boundary lines* à deux états et l'exploitation des maillages hétérogènes. Pour illustrer ces travaux, un modèle véhicule hybride parallèle à deux commandes (couple machine électrique C_{Mel} et débit de liquide de refroidissement $\dot{m}_c$) et deux états (état de charge batterie SOC_{Bat} et température machine électrique T_{Mel}) a été utilisé.

La première partie de ce chapitre développe l'amélioration appelée *boundary surfaces*. Différentes versions de l'algorithme sont présentées, allant de la version la plus lente à la version la plus rapide. Un second objectif des différentes versions de l'algorithme est de permettre une généralisation de l'algorithme à n états. Cette généralisation à pu être validée pour un problème à deux états. Ainsi, la dernière version de l'amélioration peut être utilisée pour des programmations dynamiques à n états, ce qui n'est pas le cas des deux premières générations qui sont limitées à trois états. Cette application n'a cependant pas été vérifiée durant cette thèse, laissant en perspective cet axe de travail. Un autre axe de travail important est d'appliquer un maillage adaptatif à l'amélioration des *boundary surfaces*, comme il est possible de faire avec les *boundary lines* (montré dans le chapitre 1).

Les résultats obtenus par l'amélioration des *boundary surfaces* sont très satisfaisants car ils permettent de protéger le domaine faisable dans la matrice de coût optimal. Cela induit une meilleure qualité des résultats mais également un sens physique à la matrice elle-même. En effet, lorsqu'une matrice de coût optimal est dite contaminée, il n'est plus possible d'en retirer la moindre information.

La seconde partie de ce chapitre investigue la mise en place et l'intérêt des mailles hétérogènes. Au lieu de répartir les ressources de calcul de façon égale entre toutes les variables, il est préférable de laisser un maximum de ressources aux variables qui ont un poids plus important sur le calcul. Une étude de sensibilité est mise en place pour identifier les variables d'importance. Un point d'étude non négligeable serait de réussir à déterminer *a priori* le maillage. Ainsi, en se basant sur les caractéristiques du modèle étudié par exemple, il serait possible de déterminer le maillage adéquat au problème avant tout calcul de programmation dynamique.

Enfin, il a été montré qu'il était possible et intéressant de coupler les deux axes d'amélioration de la programmation dynamique développés lors de ce chapitre pour obtenir des résultats de qualité avec un temps de calcul le plus petit possible.

Chapitre 4

Amélioration de la Programmation Dynamique par analyse fréquentielle

Sommaire

L'optimisation des résultats de la programmation dynamique par l'utilisation des *boundary surfaces* comme elles ont été développées dans le chapitre 3, montrent une augmentation du temps de calcul modérée mais tout de même existante par rapport au modèle simplifié utilisé dans le chapitre 1. Cette augmentation du temps de calcul sera d'autant plus importante si le nombre de variable augmente. C'est pourquoi, une nouvelle approche peut être à privilégier pour les modèles avec un grand nombre de variable de commande et/ou d'état.

Dans un problème à plusieurs états et plusieurs commandes, il est très probable que chacune des commandes n'ait pas le même poids par rapport aux autres commandes pour toutes les dynamiques du système. En effet, si un problème est divisé en une multitude de sous-problèmes plus faciles et/ou plus rapides à résoudre, alors il devient envisageable de résoudre de manière quasi optimale des problèmes bien plus importants que ceux qui peuvent être traités actuellement dans des temps de calcul raisonnable. Il faut cependant que la sous-optimalité de la résolution successive des sous-problèmes soit maîtrisée et faible pour que la méthodologie ait un réel intérêt. C'est la raison pour laquelle, le choix de la division doit être faite en fonction de l'importance des commandes par rapport aux dynamiques, au critère d'optimisation et aux contraintes. Des outils d'analyse fréquentielle permettent de répondre à

ces questions. Les deux principaux outils utilisés durant la thèse sont détaillés dans la section suivante.

La méthodologie est par la suite appliquée au modèle véhicule hybride électrique parallèle détaillé dans la section 3.1 du chapitre précédent. Ainsi les résultats obtenus par la méthodologie seront évalués à la référence définie par l'amélioration des *boundary surfaces* pour évaluer la sous-optimalité de la solution obtenue. Les résultats de références variant en fonctions du problème étudié, ces derniers seront indiqués en dernière ligne des tableau par l'acronyme "*FIFO*" pour la programmation dynamique classique ou par "*FIFO BdS* Gen.3" pour la dernière version de la programmation dynamique avec les *boundary surfaces*.

4.1 Méthodologie d'analyse fréquentielle

Afin de présenter l'intérêt du découplage fréquentiel, la complexité numérique de l'algorithme de la programmation dynamique, explicité dans le chapitre 1, est utilisée comme mesure du nombre de calcul à réaliser durant l'algorithme. Pour rappel, la complexité numérique de la programmation dynamique est :

$$\xi_0 = O\left(N.\prod_{i=1}^{n_u} q_{u_i}.\prod_{j=1}^{n_s} q_{s_j}\right) \tag{4.1}$$

Avec N le nombre de pas de temps, n_u le nombre de commandes, q_{u_i} le nombre de mailles allouées à la commande i, n_s le nombre d'états et q_{s_j} le nombre de mailles allouées à l'état j.

Un intérêt au découplage fréquentiel peut alors être détecté si et seulement si, la complexité numérique ξ_1 d'une série de problèmes plus petits est inférieure à la complexité numérique initiale ξ_0. Cette nouvelle complexité numérique peut être écrite comme :

$$\xi_1 = O\left(\sum_{k=1}^{\Lambda} N_k.\prod_{i=1}^{n_{u_k}} q_{u_i}.\prod_{j=1}^{n_{s_k}} q_{s_j}\right) \tag{4.2}$$

Avec Λ le nombre de sous-problèmes détectés ainsi que n_{u_k} et n_{s_k} respectivement le nombre de commandes et d'états pour le sous-problème k.

Pour déterminer si le découplage est intéressant, le rapport de ξ_1/ξ_0 est calculé. Si ce rapport est inférieur à 1, alors il y aura un gain en temps de calcul. Le rapport des deux complexités s'écrit alors :

$$\frac{\xi_1}{\xi_0} = \frac{\sum_{k=1}^{\Lambda} N_k.\prod_{i=1}^{n_{u_k}} q_{u_i}.\prod_{j=1}^{n_{s_k}} q_{s_j}}{N.\prod_{i=1}^{n_u} q_{u_i}.\prod_{j=1}^{n_s} q_{s_j}} \tag{4.3}$$

Pour illustrer ce calcul, un système de même dimension que le modèle véhicule dans le chapitre précédent est utilisé. Ce système, sans contraintes, est alors composé de deux commandes et deux états. Il est supposé, que le découplage fréquentiel indique une séparation du problème initial en deux sous-problèmes à un état et une commande. La complexité initial s'écrit $\xi_0 = N.p_1.p_2.q_1.q_2$ tandis que la nouvelle complexité est exprimée par $\xi_1 = N_1.p_1.q_1 + N_2.p_2.q_2$. En supposant que $N_1 = N_2 = N$, le rapport ξ_1/ξ_0 donne alors :

$$\frac{\xi_1}{\xi_0} = \frac{N.(p_1.q_1 + p_2.q_2)}{N.p_1.p_2.q_1.q_2} \tag{4.4}$$

qui peut se simplifier en (4.5) :

$$\frac{\xi_1}{\xi_0} = \frac{1}{p_2.q_2} + \frac{1}{p_1.q_1} \tag{4.5}$$

Il est possible de tracer la figure 4.1 qui illustre le rapport de complexité numérique des deux solutions techniques de calcul pour un problème à deux états et deux commandes séparables en une somme de deux problèmes à un état et une commande.

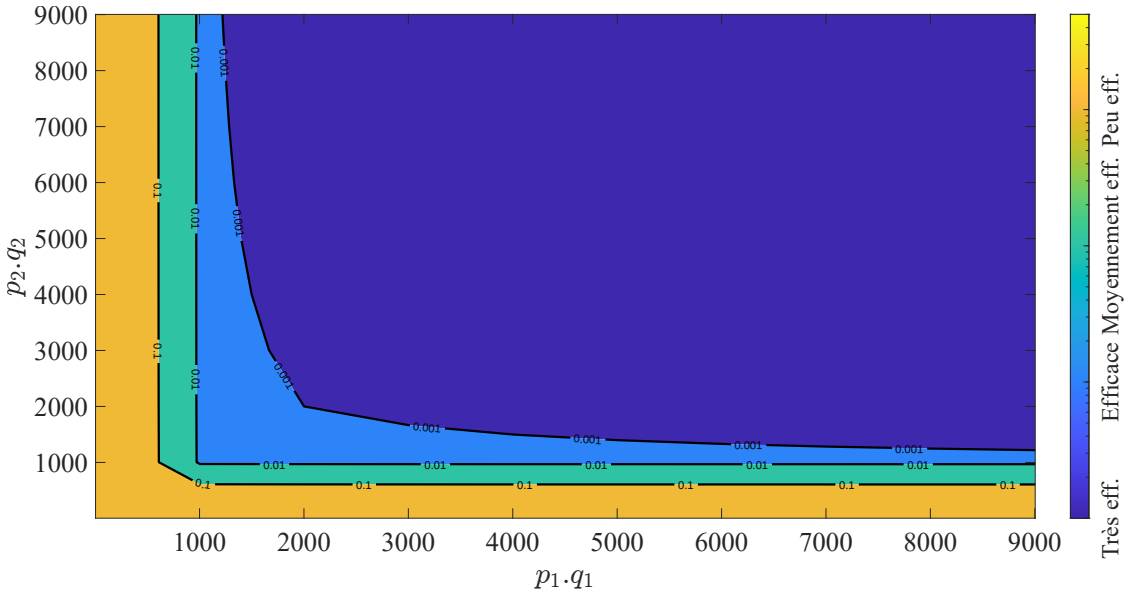

FIGURE 4.1 – Comparaison de complexité numérique $\frac{\xi_1}{\xi_0}$ d'un même problème entre une résolution avec le problème en entier et une résolution successive des sous problèmes pour un problème à deux états et deux commandes.

Le constat est largement favorable au découplage. En effet, si l'un des deux couples de variables doit être au moins égal à 5000, soit 70 mailles pour les deux variables si le maillage est homogène, la complexité numérique est réduite d'un facteur 10 000. C'est ce résultat qui confirme le découplage comme solution prometteuse. Maintenant que ce résultat de principe

est présenté, il est nécessaire d'appliquer une méthodologie de découplage pertinente pour déterminer si un système peut être fractionné en une somme de sous-systèmes. Il est proposé ici d'utiliser une méthode fréquentielle. Le premier outil de la méthodologie proposée est le *Relative Gain Array*.

4.1.1 *Relative Gain Array*

L'outil *Relative Gain Array* (*RGA*) a été développé par Bristol en 1966 [106] [27] [107]. Cet outil permet une mesure de l'interaction entre les entrées et les sorties d'un système lequel est représenté par la matrice de fonction de transfert G. Cet outil est exprimé par l'équation (4.6) pour des matrices G carrées.

$$RGA(G(\omega)) = G(\omega) \times (G(\omega)^{-1})^T \tag{4.6}$$

L'opérateur $\times$ indique ici un produit matriciel de Hadamard, ou produit de Schur, qui multiplie deux matrices de même dimension, élément par élément. Cette nouvelle matrice *RGA* possède de nombreuses propriétés. Une première propriété importante est que plus un élément de la matrice est proche de 1, plus l'interaction entre la commande et la sortie concernées est importante. Une seconde propriété est que dans le cas d'une matrice carrée à k lignes et colonnes, la somme des éléments sur une ligne, ou sur une colonne, est égale à 1 :

$$\begin{cases} \forall j,\ \sum_{i=1}^{k} RGA_{ij}(\omega) = 1 \\ \\ \forall i,\ \sum_{j=1}^{k} RGA_{ij}(\omega) = 1 \end{cases} \tag{4.7}$$

avec $RGA_{ij}(\omega)$ l'élément de la matrice *RGA* à la i$^{\text{ème}}$ ligne et à la j$^{\text{ème}}$ colonne. Pour une matrice G non carrée avec l lignes et m colonnes, la formule donnée précédemment pour calculer *RGA* doit être modifiée pour prendre en compte des matrices non carrées [108] [109]. L'opérateur † dans (4.8) représente l'opérateur pseudo-inverse [110] qui est à utiliser pour les matrices non carrées.

$$RGA(G(\omega)) = G(\omega) \times (G(\omega)^{-1})^{\dagger} \tag{4.8}$$

S'il y a autant ou plus de commandes que de sorties, alors la matrice est décrite comme *full row rank*, ce qui se traduit par : $r = rank(G) = l$. Dans ce cas, la somme des m éléments d'une même colonne est égale à 1, mais la somme des éléments d'une même ligne n'est plus forcément égale à 1. Cela se traduit par l'équation (4.9).

$$\forall i,\ \sum_{j=1}^{m} RGA_{ij}(\omega) = 1 \tag{4.9}$$

D'un autre côté, s'il y a moins de commandes que de sorties, la matrice est alors définie comme *full column rank*, ce qui se traduit par : $r = rank(G) = m$. A l'inverse du cas précédent, la somme des l éléments d'une même ligne est égale à 1, mais la somme des éléments d'une

même colonne n'est plus forcément égale à 1. Cela se traduit par l'équation (4.10).

$$\forall j,\ \sum_{i=1}^{l} RGA_{ij}(\omega) = 1 \tag{4.10}$$

Une commande j est importante dans le comportement d'une sortie i si le terme $RGA_{ij}(\omega)$ est proche de 1 à la pulsation étudiée ω [27]. Une valeur négative du terme $RGA_{ij}(\omega)$ dénote une instabilité dans le contrôle de la sortie i par la commande j.

Une fois l'analyse RGA réalisée, il est possible de réordonner la matrice de fonction de transfert G du système d'une telle manière que les éléments sur la diagonale de G soient les plus importants pour avoir la diagonale dominante. Un exemple de matrice à réordonner est explicité sur la figure 4.2. Dans cet exemple à trois commandes et trois sorties, si l'analyse RGA indique que la commande u_1 a plus d'importance pour la sortie y_2 et que la commande u_2 est plus importante pour la sortie y_1, alors la réorganisation de la matrice est donnée sur la droite.

$$\begin{matrix} u_1 & u_2 & u_3 \end{matrix}$$
$$\begin{pmatrix} h_{11}(\omega) & h_{21}(\omega) & h_{31}(\omega) \\ h_{12}(\omega) & h_{22}(\omega) & h_{32}(\omega) \\ h_{13}(\omega) & h_{23}(\omega) & h_{33}(\omega) \end{pmatrix} \begin{matrix} y_1 \\ y_2 \\ y_3 \end{matrix} \Rightarrow \begin{matrix} u_1 & u_2 & u_3 \end{matrix}$$
$$\begin{pmatrix} h_{12}(\omega) & h_{22}(\omega) & h_{32}(\omega) \\ h_{11}(\omega) & h_{21}(\omega) & h_{31}(\omega) \\ h_{13}(\omega) & h_{23}(\omega) & h_{33}(\omega) \end{pmatrix} \begin{matrix} y_2 \\ y_1 \\ y_3 \end{matrix}$$

FIGURE 4.2 – Réorganisation de la matrice en fonction de l'analyse RGA.

La question de la possibilité d'appliquer cette analyse sans connaître la matrice de fonction de transfert peut être intéressante. Il est possible d'avoir à disposition le système réel, ou un modèle fermé. Pour ces deux situations, il est possible de caractériser la matrice de fonction de transfert à travers une excitation fréquentielle des entrées [111]. Plusieurs types de signaux peuvent être utilisés pour réaliser cette tâche : *chirp*, multisinus ou séquence binaire pseudo-aléatoire. Le premier signal, *chirp*, est un signal sinusoïdal dont la fréquence augmente au cours du temps. Le second signal, multi-sinus, est un signal composé d'une somme de sinus à des fréquences et des déphasages différents. Le dernier signal, séquence binaire pseudo-aléatoire, est une succession d'échelon montant et descendant à différents moments, pour varier les fréquences excitées.

Ces signaux ont pour but de stimuler l'ensemble des sorties sur une plage de fréquence définie afin de mesurer les variations des sorties du système causées par ces stimulations. Il est alors possible de retrouver la fonction de transfert $G_{ij}(\omega)$ entre la commande i et la sortie j grâce à (4.11) :

$$G_{ij}(\omega) = \frac{fft\left(y_j(\omega)\right)}{fft\left(u_i(\omega)\right)} \tag{4.11}$$

Avec $fft(.)$ représentant l'opération de transformée de Fourier rapide [112], $y_j(\omega)$ la réponse de la sortie j dans le domaine fréquentiel et $u_i(\omega)$ la réponse fréquentielle de la com-

mande i. Une autre méthode est l'utilisation du théorème de Wiener-Khintchine qui permet de faire un lien entre la densité spectrale d'un signal et la transformée de Fourrier de l'autocorrélation du même signal. Ainsi pour retrouver le transfert de la commande vers la sortie, le rapport de la densité spectrale de puissance du signal d'intercorrélation entre l'entrée u_i et la sortie y_j et la densité spectrale de puissance du signal d'autocorrélation de l'entrée est calculé.

$$G_{ij}(\omega) = \frac{psd(u_i(\omega), y_j(\omega))}{psd(u_i(\omega), u_i(\omega))} \tag{4.12}$$

Afin d'illustrer l'application de l'outil *RGA*, un exemple d'un système à deux entrées et deux sorties est utilisé. La matrice de fonction de transfert de ce système, G, peut être écrite comme :

$$G(\omega) = \begin{pmatrix} h_{11}(\omega) & h_{21}(\omega) \\ h_{12}(\omega) & h_{22}(\omega) \end{pmatrix} \tag{4.13}$$

En supposant que les quatre fonctions de transfert composant la matrice G sont inconnues, il faut alors dans un premier temps déterminer ces dernières. Pour ce faire, un signal multi-sinus défini par l'équation (4.14) est utilisé. [111] a montré que l'utilisation d'un signal multi-sinus permettait une identification des transferts plus fine et moins bruitée sur la plage de fréquence étudiée.

$$\begin{cases} x(t) = \sum_{k=1}^{F} B_k.cos(2.\pi.f_k.t + \Phi_k) \\ B_k = 1, \forall k \in [1; F] \\ f_k = (r+k).f_0, \forall r \in \mathbb{N} \\ \Phi_k = \frac{-k.(k-1).\pi}{F} \end{cases} \tag{4.14}$$

F étant le nombre de fréquences utilisées pour composer le signal multi-sinus $x(t)$, f_0 représentant la fréquence de discrétisation permettant de balayer les F fréquences entre f_{min} et f_{max}. Enfin, la constante r permet d'ajuster le coefficient devant la fréquence de discrétisation pour que : $f_{min} = (r+1).f_0$.

Les quatre fonctions de transfert obtenues en figure 4.3 sont des fonctions de transfert du 1er ordre. Ces dernières s'expriment en fonction de la variable de Laplace, notée s ici, comme le montre l'équation (4.15). Cette variable de Laplace est exprimée par : $s = \sigma + j\omega$.

$$h_{ij}(\omega) = \frac{K_{ij}}{1 + \tau_{ij}.(\omega)} \tag{4.15}$$

avec K_{ij} le gain statique de la fonction de transfert et τ_{ij} sa constante de temps. En utilisant les résultats de la figure 4.3, il est possible d'identifier les quatre fonctions de transfert.

$$G(s) = \begin{pmatrix} \frac{10}{1+0.1s} & \frac{5}{1+10s} \\ \frac{0.1}{1+0.5s} & \frac{1}{1+s} \end{pmatrix} \tag{4.16}$$

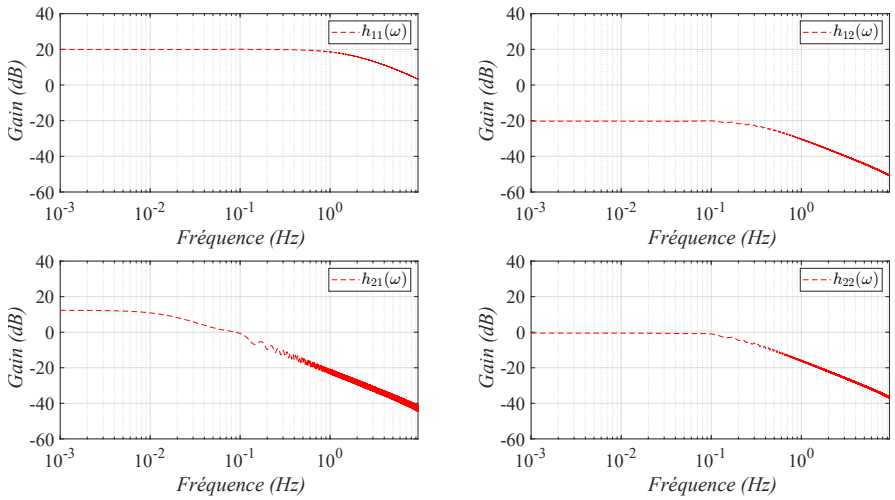

FIGURE 4.3 – Diagrammes de Bode d'identification des quatre transferts grâce au signal multi-sinus (4.14) en utilisant (4.11).

Il n'est cependant pas obligatoire de recalculer les fonctions de transfert pour effectuer l'analyse *RGA*. En effet, il est possible d'utiliser les valeurs obtenues dans la figure 4.3, qui correspondent à l'équation (4.11), directement dans l'équation (4.6). L'analyse fréquentielle obtenue est alors affichée en figure 4.4.

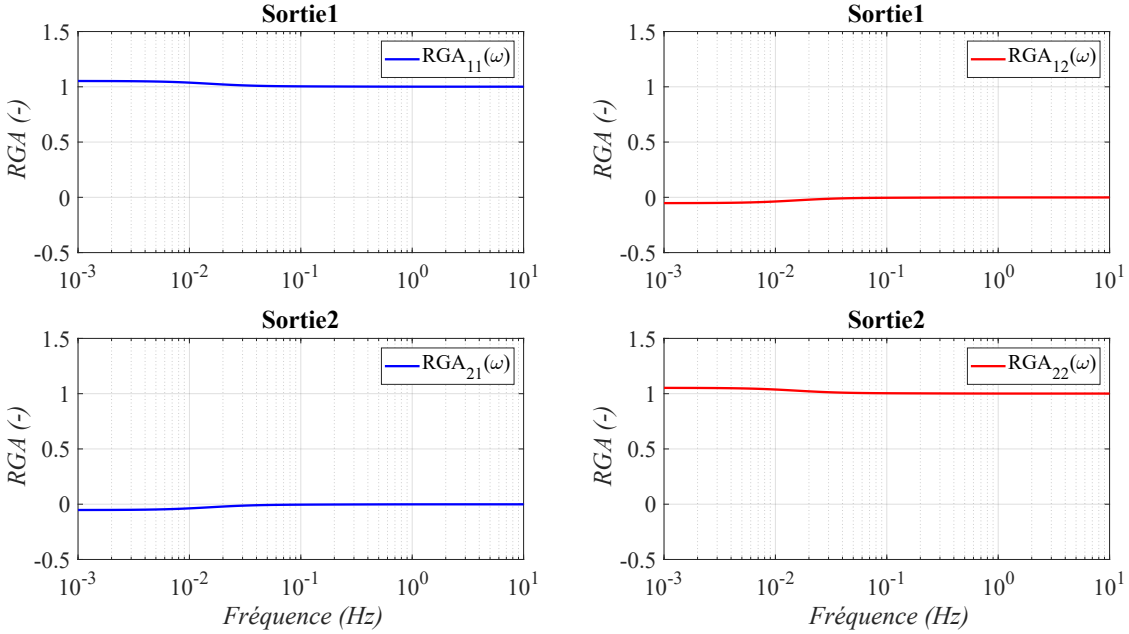

FIGURE 4.4 – Analyse *RGA* du système (4.13) identifié sur la figure 4.3.

La commande 1 sera utilisée pour contrôler la sortie 1 tandis que la commande 2 contrôlera la sortie 2 car les deux termes $RGA_{11}(\omega)$ et $RGA_{22}(\omega)$ sont proches de 1 sur toute la plage de

fréquence étudiée. De plus, la matrice de fonction de transfert n'a pas besoin d'être réarrangée car les éléments sur la diagonale de la matrice RGA sont proches de 1. Il est également possible de constater que la somme des éléments en ligne et les colonnes est égale à 1, et ce pour toute la plage de fréquence étudiée. Il est conclu de cette analyse que le système peut être découplé en deux sous-problèmes à un état et une commande.

4.1.2 *Column Diagonal Dominance Degree*

L'outil *Column Diagonal Dominance Degree* noté CD^3 permet d'évaluer le taux d'interaction d'une commande sur les sorties qui ne sont pas sur la diagonale formée précédemment par l'analyse RGA [113]. Il n'est à utiliser que si l'analyse précédente indique un découplage fréquentiel potentiel. Il se définit comme :

$$CD^3_{ci}(\omega) = \underset{G}{max} \left\{ \sum_{\substack{1 \leq j \leq n \\ j \neq i}} \frac{|G_{ij}(\omega)|}{|G_{ii}(\omega)|} \right\} \forall i \in [1; n] \tag{4.17}$$

En fonction du résultat donné par (4.17), il est alors possible d'envisager un découplage total ou non des sous-systèmes. Les critères sont les les suivants :

— Si $\forall i,\ CD^3_{ci}(\omega) \ll 1$, il est possible de totalement découpler les sous-problèmes
— Si $\exists i,\ CD^3_{ci}(\omega) \geq 1$, un découplage total des problèmes n'est pas possible.

En appliquant ce calcul sur l'exemple, il est possible d'obtenir ces courbes :

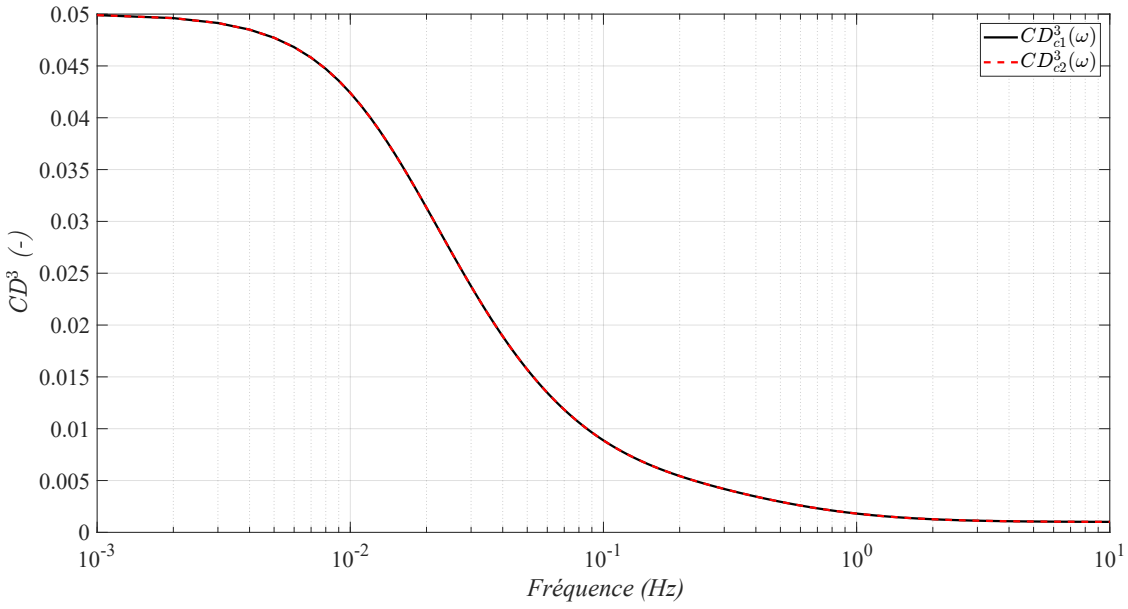

FIGURE 4.5 – Analyse CD^3 du système (4.16). Le système peut être découplé totalement.

Il peut alors être constaté que l'analyse fréquentielle CD^3 en figure 4.5 indique que ce

modèle peut être totalement découplé car toutes les conditions sont respectées.

4.1.3 Méthodologie de découplage proposée

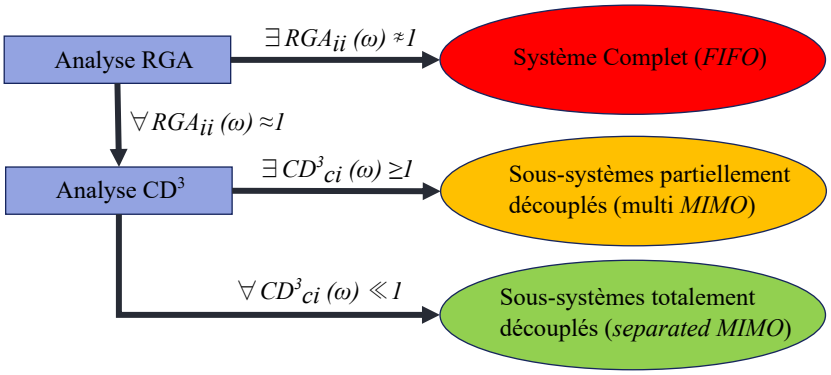

FIGURE 4.6 – Arbre des possibilités des résultats de l'analyse fréquentielle. Le cas le plus défavorable est identifié en rouge car le découplage du système n'est pas possible. Le cas intermédiaire est en orange car un découplage est possible mais partiel tandis que le cas en vert est le meilleur car le système peut être découplé totalement.

La figure 4.6 montre que l'analyse fréquentielle débute par l'analyse *RGA*. S'il est possible de réorganiser la matrice de fonction de transfert afin que la matrice de l'analyse *RGA* ait sa diagonale dominante et que les éléments de cette dernière soient proches de 1, alors il est possible de séparer le problème dans une somme de sous-problèmes. Si ce n'est pas le cas, alors l'analyse CD^3 n'est pas à appliquer au système car le résultat de l'analyse fréquentielle montre que le système doit être considéré dans son entièreté pour résoudre le problème de contrôle optimal. Le problème est catégorisé comme étant complet, ce qui donne l'acronyme *FIFO* pour *Full Inputs Full Outputs*, soit toutes les commandes et toutes les sorties. Dans le cas où le système peut être découplé, alors l'analyse CD^3 est réalisée sur le système. En fonction du résultat obtenu, le calcul de la détermination de la loi de contrôle optimale va légèrement différer. La différence réside dans la prise en compte ou non de la résolution des autres sous-problèmes pour la résolution d'un des sous-problèmes. Dans le cas où le découplage total n'est pas possible, on parlera alors du système multi *MIMO* tandis que s'il est possible, on parlera de *separated MIMO*. L'acronyme *MIMO* signifie *Multi Inputs Multi Outputs* pour plusieurs commandes et plusieurs sorties. Si la décomposition du problème donne lieu à des problèmes à une seule commande et une sortie, on parle alors de *Single Input Single Output* soit *SISO*. Dans le cas où il est possible de diviser le problème en somme de sous-problèmes, il est possible de réitérer l'analyse sur chaque sous-problème. La complexité des sous-problèmes peut ainsi être réduite encore plus si la nouvelle analyse fréquentielle indique un découplage plus important.

L'algorigramme illustré en figure 4.7 détaille la manière d'ordonner et de résoudre les sous-problèmes dans les cas multi *MIMO* (orange) et *separated MIMO* (vert) de l'analyse fréquentielle. Dans le premier cas, il y a deux possibilités pour sortir de la boucle de convergence

et obtenir la solution finale : soit le nombre de boucle λ_{glo} a été effectué, soit le critère de convergence ϵ_{glo} est atteint. Ces deux termes (λ_{glo} et ϵ_{glo}) sont indiqués par l'utilisateur en fonction de ses besoins.

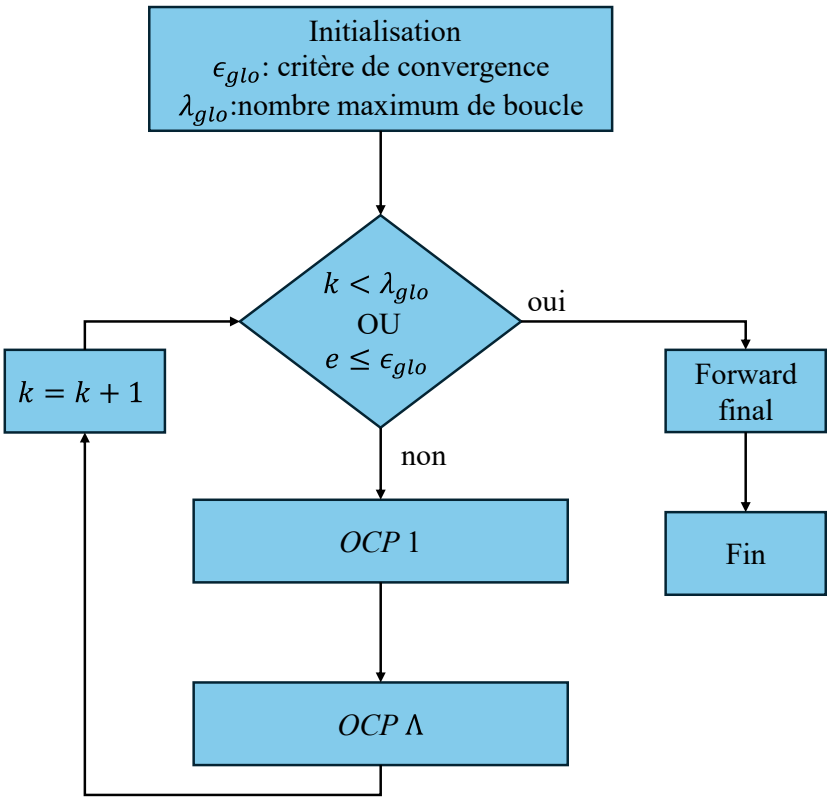

FIGURE 4.7 – Algorigramme de résolution des sous-problèmes lorsqu'il est possible de découpler (partiellement ou totalement) le problème global.

L'expression permettant de calculer l'erreur e est définie par (4.18). Dans le second cas, $\lambda = 1$ étant donné que les sous-problèmes sont totalement découplés.

$$e = \prod_{a=1}^{n} \frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}_a^{i,k-1} - \boldsymbol{x}_a^{i,k}| . \prod_{b=1}^{m} \frac{1}{N-1} \sum_{i=1}^{N-1} |\boldsymbol{u}_b^{i,k-1} - \boldsymbol{u}_b^{i,k}| \tag{4.18}$$

avec $\boldsymbol{x}_a^{i,k-1}$ et $\boldsymbol{x}_a^{i,k}$ la valeur de l'état a respectivement de la boucle d'avant $(k-1)$ ou de la boucle actuelle (k), les deux au pas de temps i. L'écriture est analogue pour les commandes $\boldsymbol{u}$, avec $\boldsymbol{u}_b^{i,k-1}$ et $\boldsymbol{u}_b^{i,k}$ la valeur de la commande b respectivement de la boucle d'avant $(k-1)$ ou de la boucle actuelle (k), les deux au pas de temps i.

Pour la résolution de notre exemple, il serait nécessaire de ne faire qu'une seule boucle de convergence : $\lambda = 1$ car le problème peut être totalement découplé en deux sous-problèmes.

4.1.4 Illustration de la méthodologie pour un système linéaire à 2 états et 2 commandes

Afin de montrer que le découplage est possible, il est demandé que les deux sorties du système x^1 et x^2 suivent deux signaux de consigne respectivement x^{r1} et x^{r2}. Pour déterminer la suite de commande à appliquer, un problème de contrôle optimal est posé :

$$\begin{cases} \min\limits_{\boldsymbol{u}} J(t_0, \boldsymbol{x}(t), \boldsymbol{u}(t)) = \Delta t. \sum_{k=0}^{N-1} |x_k^1 - x_k^{r1}| + |x_k^2 - x_k^{r2}| \\ \boldsymbol{x}_{k+1} = f_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{x}_k \in [-40; 40] \times [-20; 10], \forall k \geq 0 \\ \boldsymbol{u}_k \in [-1; 2] \times [-20; 1], \forall k \geq 0 \end{cases} \tag{4.19}$$

Il a été montré dans les sous-sections précédentes que le système pouvait être totalement découplé en deux sous problèmes : la commande u_1 pilote la sortie x_1 tandis que la commande u_2 pilote la sortie x_2. Le problème (4.19) peut alors être réécrit sous la forme :

$$\begin{cases} \min\limits_{u^1} J(t_0, \boldsymbol{x}(t), \boldsymbol{u}(t)) = \Delta t. \sum_{k=0}^{N-1} |x_k^1 - x_k^{r1}| \\ x_{k+1}^1 = f_k(x_k^1, u_k^1, u_k^2) \\ x_k^1 \in [-40; 40], \forall k \geq 0 \\ u_k^1 \in [-1; 2], \forall k \geq 0 \end{cases} \tag{4.20}$$

$$\begin{cases} \min\limits_{u^2} J(t_0, \boldsymbol{x}(t), \boldsymbol{u}(t)) = \Delta t. \sum_{k=0}^{N-1} |x_k^2 - x_k^{r2}| \\ x_{k+1}^2 = f_k(x_k^2, u_k^2, u_k^1) \\ x_k^2 \in [-20; 10], \forall k \geq 0 \\ u_k^2 \in [-20; 1], \forall k \geq 0 \end{cases} \tag{4.21}$$

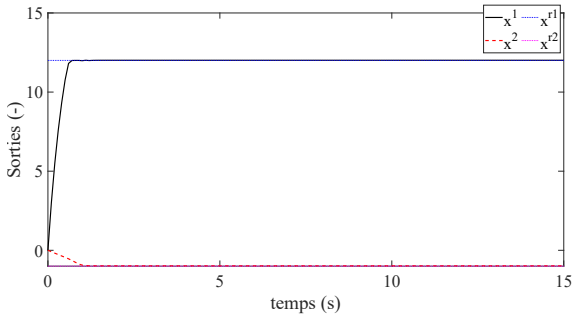

FIGURE 4.8 – Trajectoires des états x^1 et x^2 avec leurs références x^{r1} et x^{r2}.

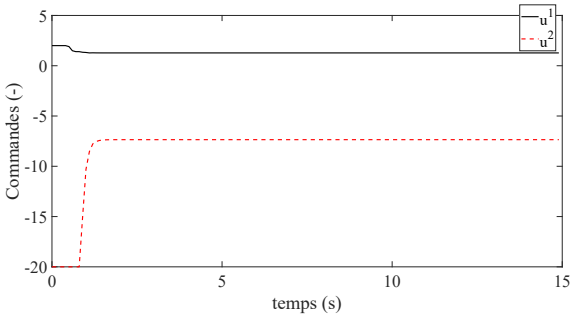

FIGURE 4.9 – Trajectoires des commandes u^1 et u^2.

Il est alors envisageable d'appliquer cette méthodologie sur un cas d'étude, plus complexe dont la résolution est moins évidente.

4.2 Application à la gestion de l'énergie d'un vehicule hybride électrique avec dynamique de température

Le modèle développé dans la section 3.1.1 est réutilisé dans ce chapitre pour évaluer la performance du découplage fréquentiel avec utilisation de la programmation dynamique pour résoudre les problèmes de commande optimale. L'évaluation de la méthodologie est faite en comparant les résultats de la résolution successive des sous-problèmes générés à ceux obtenus avec les *boundary surfaces* développées dans le chapitre précédent.

4.2.1 Analyse fréquentielle

Dans un premier temps, les outils d'analyse fréquentielle sont appliqués sur le système. Il faut commencer par définir la matrice de fonction de transfert G du système. Pour ce faire, nous allons commencer par écrire le système sous forme de représentation d'état :

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t).\boldsymbol{x}(t) + \boldsymbol{B}(t).\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}.\boldsymbol{x}(t) + \boldsymbol{D}.\boldsymbol{u}(t) \end{cases}$$

Les variables $\boldsymbol{x} \in \mathbf{R}^m$, $\boldsymbol{y} \in \mathbf{R}^p$ et $\boldsymbol{u} \in \mathbf{R}^n$ sont respectivement les états, les sorties et les commandes du système. Les matrices sont alors de tailles différentes : $\boldsymbol{A}(t) \in \mathbf{R}^{m\times m}$, $\boldsymbol{B}(t) \in \mathbf{R}^{m\times n}$, $\boldsymbol{C} \in \mathbf{R}^{p\times n}$, $\boldsymbol{D} \in \mathbf{R}^{p\times m}$. Pour le cas d'application, il peut être écrit :

$$\boldsymbol{x} = \boldsymbol{y} = \begin{bmatrix} T_{Mel} \\ SOC_{Bat} \end{bmatrix} \tag{4.22}$$

$$\boldsymbol{u} = \begin{bmatrix} C_{Mel} \\ \dot{m}_c \end{bmatrix} \tag{4.23}$$

Afin de déterminer les matrices $\boldsymbol{A}$ et $\boldsymbol{B}$, il faut repartir des équations (3.8) et (3.14) et procéder par identification. Toutefois, la présence de non linéarités dans ces deux équations liées aux termes au carré ou aux interactions entres deux variables de commandes ou d'états, ne permettent pas l'identification des éléments des deux matrices. Pour résoudre cette problématique, les équations sont linéarisées grâce à un développement de Taylor à l'ordre 1. Il devient donc possible de trouver les deux matrices $\boldsymbol{A}$ et $\boldsymbol{B}$, à un point de fonctionnement donné du système. Dans notre étude, nous imposons que $\boldsymbol{y} = \boldsymbol{x}$ car il est imposé que les variables d'états soient les variables de sortie. On en déduit alors que $\boldsymbol{C} = I_2$ et que $\boldsymbol{D} = [0\,0]^T$.

Les matrices A et B sont définies par :

$$A = \begin{pmatrix} \frac{\partial \dot{T}_{Mel}}{\partial T_{Mel}} & \frac{\partial \dot{T}_{Mel}}{\partial SOC_{Bat}} \\ \frac{\partial \dot{SOC}_{Bat}}{\partial T_{Mel}} & \frac{\partial \dot{SOC}_{Bat}}{\partial SOC_{Bat}} \end{pmatrix} \tag{4.24}$$

$$B = \begin{pmatrix} \frac{\partial \dot{T}_{Mel}}{\partial C_{Mel}} & \frac{\partial \dot{T}_{Mel}}{\partial \dot{m}_c} \\ \frac{\partial \dot{SOC}_{Bat}}{\partial C_{Mel}} & \frac{\partial \dot{SOC}_{Bat}}{\partial \dot{m}_c} \end{pmatrix} \tag{4.25}$$

Les matrices A et B obtenues pour le modèle étudié sont alors décrites par les équations (4.26) et (4.27). Les variables avec un indice L, sont des variables linéarisées autour d'une valeur de fonctionnement.

$$A = \begin{pmatrix} -\frac{\dot{m}_{c_L}.Cp_c}{m_{Mel}.Cp_{Mel}} & 0 \\ 0 & Q.e_0.\frac{\dot{m}_{c_L}.w_s + R.(C_{Mel_L}/k)^2 + C_{Mel_L}.\omega_{Mel_L}}{Q.(E+e_0.SOC_{Bat_L})^2} \end{pmatrix} \tag{4.26}$$

$$B = \begin{pmatrix} \frac{2.R.\frac{C_{Mel_L}}{k^2}}{m_{Mel}.Cp_{Mel}} & -\frac{Cp_c.(T_{Mel_L}-T_c)}{m_{Mel}.Cp_{Mel}} \\ -\frac{2.R.\frac{C_{Mel_L}}{k^2}+\omega_{Mel_L}}{Q.(E+e_0.SOC_{Bat_L})} & -\frac{w_s}{Q.(E+e_0.SOC_{Bat_L})} \end{pmatrix} \tag{4.27}$$

A partir des quatre matrices A, B, C et D, il est possible de déterminer la matrice de fonction de transfert du système comme étant :

$$G(s) = C.(s.I - A)^{-1}.B + D \tag{4.28}$$

L'analyse *RGA* est réalisée pour tous les points de fonctionnement du domaine faisable. Une partie représentative de l'ensemble des résultats sont présentés en figure 4.10. Les points de fonctionnement présentés sont les suivants : $\dot{m}_{c_L} = 0,025$ kg/s, $C_{Mel_L} = -86 : -17,2 : 86$ N.m, $\omega_{Mel_L} = 1000$ tr/min, $SOC_{Bat_L} = 0,6$ et $T_{Mel_L} = 40$°C. Plusieurs points de fonctionnement différents sont présentés en annexe C.

L'indication de la séparation possible des deux sous-problèmes suivants est obtenue :

— Le couple de la machine électrique C_{Mel} pilotera l'état de charge de la batterie SOC_{Bat}

— Le débit de liquide de refroidissement $\dot{m}_c$ pilotera la température de la machine électrique T_{Mel}

Il reste maintenant à savoir s'il est possible de totalement découpler les problèmes, ou si les interactions entre les deux sous-systèmes seront suffisamment importantes pour imposer la prise en compte de ces dernières dans la résolution des sous-problèmes. Nous devons donc maintenant faire la seconde partie de l'analyse fréquentielle avec l'outil *CD*[3].

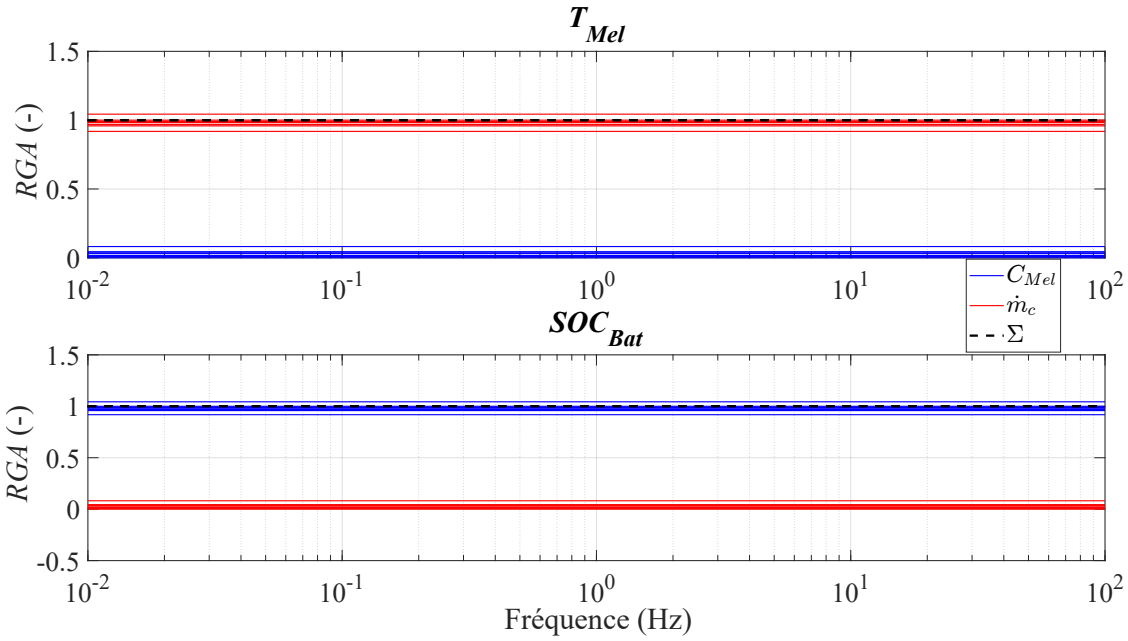

FIGURE 4.10 – Analyse *RGA* du modèle véhicule simplifié. L'analyse montre un découplage possible en liant $\dot{m}_c$ à T_{Mel} et C_{Mel} à SOC_{Bat}.

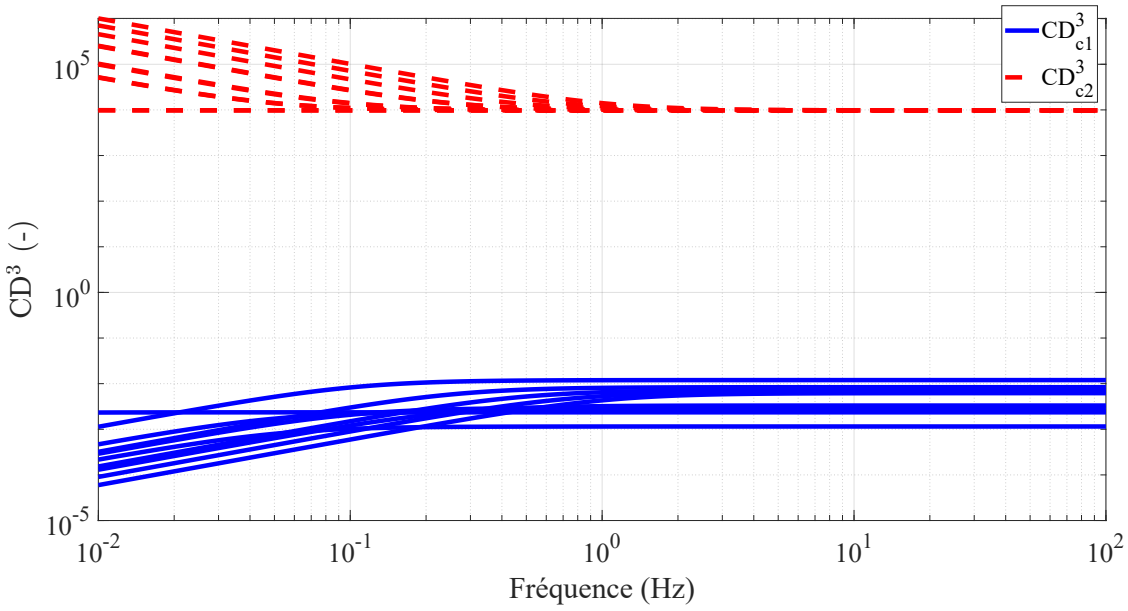

FIGURE 4.11 – Analyse CD^3 du modèle véhicule simplifié. L'analyse montre qu'il n'est pas possible de découpler totalement les deux sous problèmes. Le problème est donc multi *MIMO*.

La figure 4.11 illustre ici l'impossibilité de découpler totalement les problèmes car CD^3_{c1} n'est pas négligeable devant 1 pour au moins un point de linéarisation. C'est alors le cas multi *MIMO*, en orange au milieu de la figure 4.6, qui est indiqué par les résultats de l'analyse fré-

quentielle. Les résultats obtenus par la résolution successive de sous-problème seront comparés à ceux obtenus lors de la résolution du problème en un seul problème par une programmation dynamique avec et sans *boundary surfaces*.

4.2.2 Résultats

4.2.2.1 Division du problème (multi *MIMO*)

Avec le découplage fréquentiel, le problème (3.20) va être divisé en deux problèmes P_1 (4.29) et P_2(4.30). Le premier problème permet de résoudre le problème mettant en relation le couple de la machine électrique et l'état de charge de la batterie tandis que le second problème met en relation la température de la machine électrique et le débit de liquide de refroidissement comme l'illustre la figure 4.12.

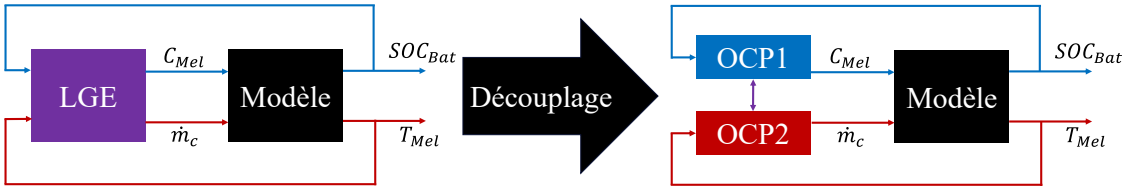

FIGURE 4.12 – Division du problème initial à deux états et deux commandes en deux sous-problèmes de un état et une commande liés par leurs interactions.

$$\begin{cases} \min\limits_{C_{Mel}} J_1(SOC_{Bat_k}, C_{Mel_k}) = \sum_{k=0}^{N-1} \sqrt{P_{F_k}^2 + \alpha(T_{Mel_k}).P_{Bat_k}^2}.\Delta t + \Phi_1\left(SOC_{Bat_N}\right) \\ SOC_{Bat_{k+1}} = SOC_{Bat_k} + f_{SOC_{Bat_k}}(SOC_{Bat_k}, C_{Mel_k}) \\ P_{Eng_k} \in [\underline{P_{Eng}}; \overline{P_{Eng}}] \\ SOC_{Bat_k} \in [\underline{SOC_{Bat}}; \overline{SOC_{Bat}}] \\ C_{Mel_k} \in [\underline{C_{Mel}}; \overline{C_{Mel}}] \end{cases} \tag{4.29}$$

$$\begin{cases} \min\limits_{\dot{m}_c} J_2(T_{Mel_k}, \dot{m}_{c_k}) = \sum_{k=0}^{N-1} \sqrt{P_{F_k}^2 + \alpha(T_{Mel_k}).P_{Bat_k}^2}.\Delta t + \Phi_2\left(T_{Mel_N}\right) \\ T_{Mel_{k+1}} = T_{Mel_k} + f_{T_{Mel_k}}(T_{Mel_k}, \dot{m}_{c_k}) \\ T_{Mel_k} \in [\underline{T_{Mel}}; \overline{T_{Mel}}] \\ \dot{m}_{c_k} \in [\underline{\dot{m}_c}; \overline{\dot{m}_c}] \end{cases} \tag{4.30}$$

Le critère de minimisation est le même dans les deux problèmes. Dans le premier, le terme $\alpha(T_{Mel_k})$ est déterminé aux termes P_{Eng_k} et P_{Bat_k} qui sont déterminées en grande partie par le second problème. Avec cette écriture, les deux problèmes générés sont des problèmes à une seule commande et un seul état.

4.2.2.2 Résultats du problème sans contraintes finales

Ce problème déjà résolu sans découplage fréquentiel dans le chapitre précédent (table 3.13) permet d'évaluer l'intérêt du découplage d'un problème en somme de sous-problèmes sans considérer les pénalités finales qui dégradent fortement la qualité des résultats. Ainsi les tables 4.1 et 4.2 présentent différents résultats de maillage homogènes de la résolution des deux sous-problèmes (4.29) et (4.30) pour respectivement les conditions initiales particulières et au bilan batterie nul.

La résultats obtenus par la résolution successive des sous-problèmes de commande optimale sont d'une qualité similaire à ceux obtenus par la résolution du problème de commande optimale complet. Cela montre que la sous-optimalité attendue par la méthodologie est maîtrisée pour la minimiser. La division en somme de sous-problèmes permet d'atteindre *in fine* une meilleure qualité de résultat car il est possible de raffiner énormément le maillage pour un temps de calcul acceptable et une puissance de calcul très faible.

TABLE 4.1 – Résultats du problème sans contraintes finales obtenus avec découplage fréquentiel pour la trajectoire particulière.

Maillage (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	temps calcul (s)	Nbre boucle (-)
21	46,07	0,5043	29,8	0,00	0,8	3
101	32,89	0,6293	29,8	0,01	6	3
501	32,38	0,6258	29,8	0,06	82,8	3
1001	32,32	0,6254	29,8	0,19	309,8	3
101 *FIFO*	32,89	0,6296	29,8	10,48	6576	-

TABLE 4.2 – Résultats du problème sans contraintes finales obtenus avec découplage fréquentiel pour le Bilan Batterie Nul.

Maillage (-)	Coût (MJ)	T_{Mel_N} (°C)
21	46,07	29,5
101	33,79	29,9
501	32,79	29,8
1001	32,62	29,8
101 *FIFO*	34,99	28,9

Les résultats présentés par les deux tables montrent que la qualité de la solution s'améliore et la mémoire vive ainsi que le temps de calcul augmentent avec le maillage. Cependant, l'augmentation du temps de calcul et de la mémoire vive sont faibles comparé à une solution *Full Input Full Output*. En effet, il est possible d'avoir une meilleure qualité de résultat comparé à la référence qui est ici la résolution du problème à deux états et deux commandes à 101 mailles par variables pour un temps de calcul et une mémoire vive réduite. Enfin, il peut être souligné qu'il faut trois boucles pour faire converger le résultat, et ce pour tous les maillages présentés.

Les trajectoires des quatre maillages sont présentées en figure 4.13. Ces dernières sont très proches de la solution de référence calculée dans le chapitre précédent à l'exception du maillage le plus grossier. En effet, toutes les solutions contrôlent la température de la machine électrique de manière importante pour atteindre la température cible de 30°C à la fin du trajet. La différence majeure du maillage grossier avec les autres est la gestion de l'état de charge de la batterie. Les meilleures solutions conservent un état de charge de la batterie relativement proche de celui initial, la solution déterminée par la *DP* découplée à 21 mailles par variables consomme plus d'énergie de la batterie, ce qui est sous-optimal selon le critère d'optimisation.

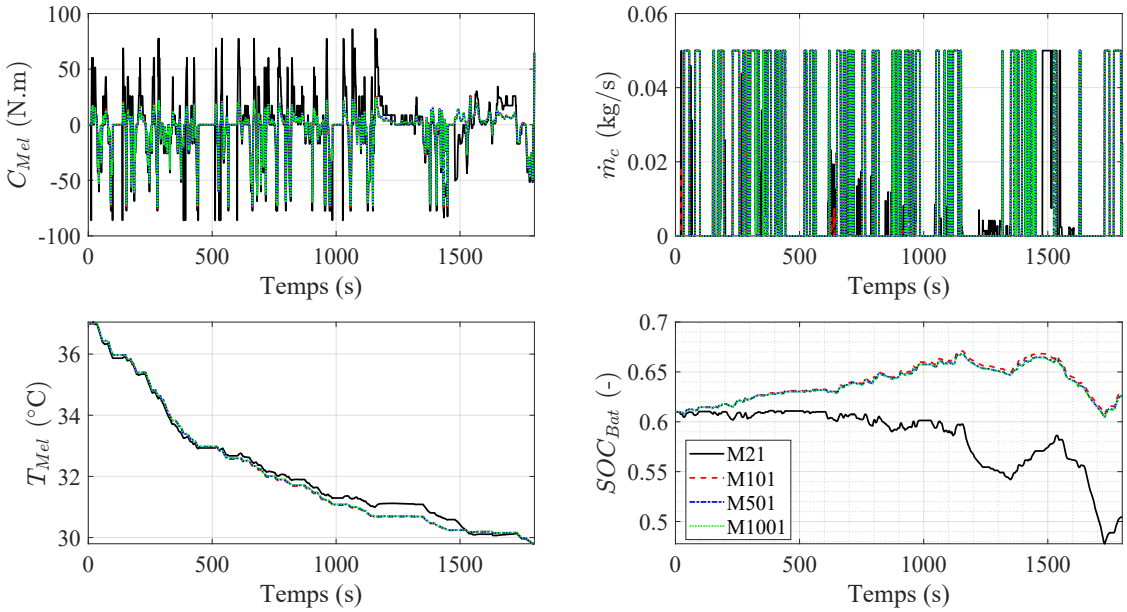

FIGURE 4.13 – Trajectoires des états et des commandes des quatre solutions de *DP* avec le découplage fréquentiel pour les maillages de 21, 101, 501 et 1001 mailles par variables respectivement en noir, rouge, bleu et vert pour les problèmes (4.29) et (4.30) sans contraintes finales (3.27).

Les matrices de coût optimal des deux sous-problèmes pour les deux maillages homogènes extrêmes (21 mailles et 1001 mailles par variables) sont présentées ci-dessous. Il est possible de voir un phénomène intéressant qui est que l'une des deux matrices est contaminée par des pénalités liées à des choix impossibles mais que la seconde n'est absolument pas polluée. Cela fait écho à l'apparition des bandes de zone décontaminées lors de l'étude du même problème dans le chapitre précédent. Les valeurs dans les deux matrices sont plus basses lorsque le maillage est plus fin, ce qui explique la meilleure qualité de résultat.

Les résultats obtenus par la résolution successive des sous-problèmes étant très pertinents sans les contraintes finales, il faut maintenant voir si ces dernières limiteront l'intérêt de la méthodologie.

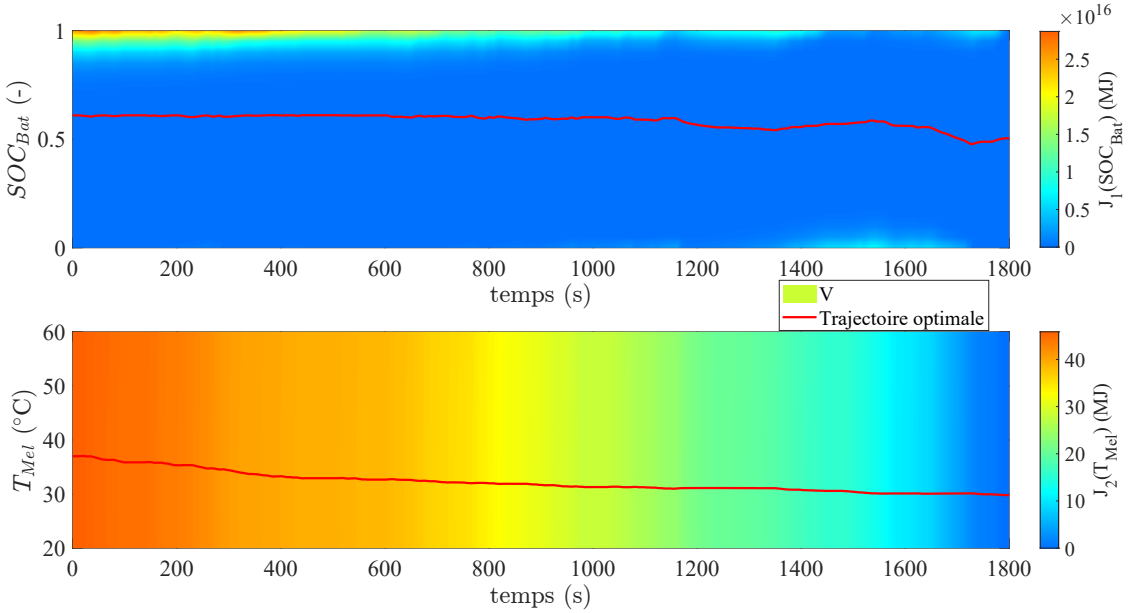

FIGURE 4.14 – Matrices de coût optimal des deux sous-problèmes pour un maillage homogène à 21 mailles

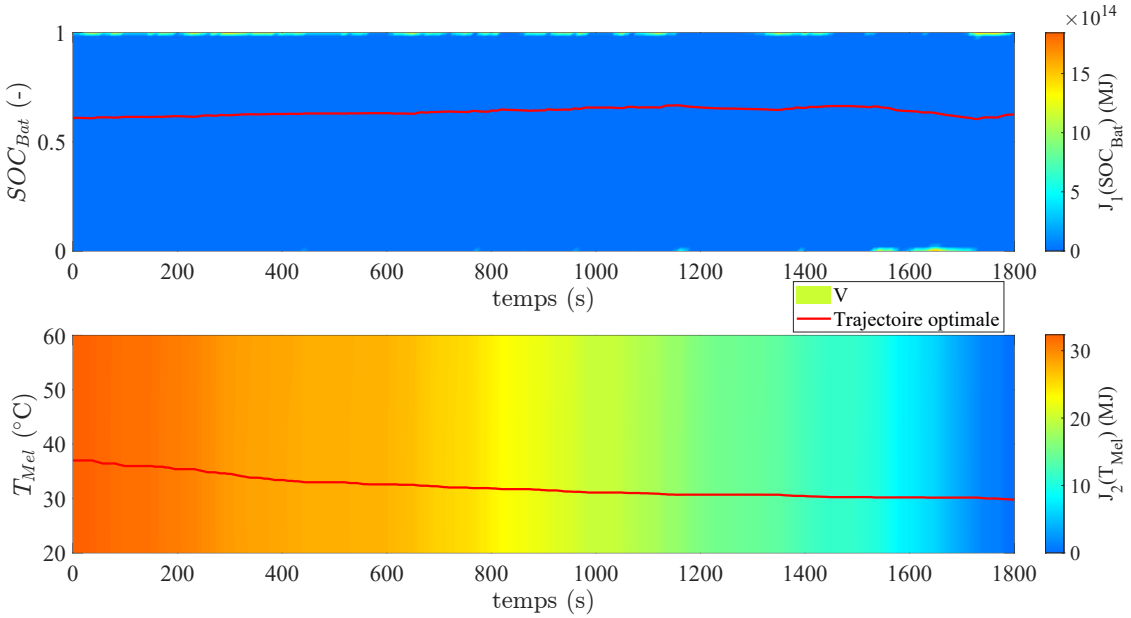

FIGURE 4.15 – Matrices de coût optimal des deux sous-problèmes pour un maillage homogène à 1001 mailles

4.2.2.3 Résultats du problème avec contraintes finales

Ce problème déjà résolu sans découplage fréquentiel dans le chapitre précédent (table 3.10) avec l'amélioration des *boundary surfaces* prend désormais en compte des contraintes finales pour observer comment la méthodologie se comporte vis-à-vis de la contamination des matrices de coût optimal. Étant donné que deux programmation dynamiques à un seul état et une seule commande découlent de la division du problème initial, l'exploitation de l'amélioration des *boundary lines* peut être envisagée. Ainsi, une résolution du même problème avec l'amélioration des *boundary lines* est effectuée pour évaluer la qualité des résultats obtenus lorsque des contraintes finales dures sont appliquées au problème de commande optimale.

Les tables 4.3 et 4.4 explicitent les résultats obtenus pour la méthodologie pour le problème avec contraintes avec l'amélioration des *boundary lines*.

TABLE 4.3 – Résultats obtenus avec découplage fréquentiel pour le problème avec contraintes avec l'utilisation des *boundary lines* pour les conditions initiales particulières.

Maillage (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	temps calcul (s)	Nbre boucle (-)
21	36,64	0,6874	37	0,00	2,4	3
101	33,08	0,6463	37	0,00	8,3	3
501	32,51	0,6314	37	0,06	92	3
1001	32,45	0,6280	37	0,19	327,8	3
101 *FIFO BdS* Gen.3	33,09	0,6466	37,2	10,65	8331,2	-

TABLE 4.4 – Résultats obtenus avec découplage fréquentiel pour le problème avec contraintes avec l'utilisation des *boundary lines* pour le bilan batterie nul.

Maillage (-)	Coût (MJ)	T_{Mel_N} (°C)
21	37,09	37
101	33,17	37
501	32,61	37
1001	32,54	37
101 *FIFO BdS* Gen.3	33,19	37,2

Les trajectoires des quatre solutions sont très proches les unes des autres. La différence majeure du maillage le plus grossier réside dans la gestion de l'état de charge de la batterie qui perçoit plus d'énergie, ce qui est sous-optimal. La gestion de la température de la machine électrique est très différente des solutions avec les *boundary surfaces*. En effet, la machine est refroidi au début du cycle de conduite pour minimiser le critère d'optimisation. Puis, le refroidissement de la machine est arrêté pour atteindre la température minimale acceptée par les contraintes finales. L'annexe D illustre une solution obtenue avec les *boundary surfaces* qui possède cette trajectoire.

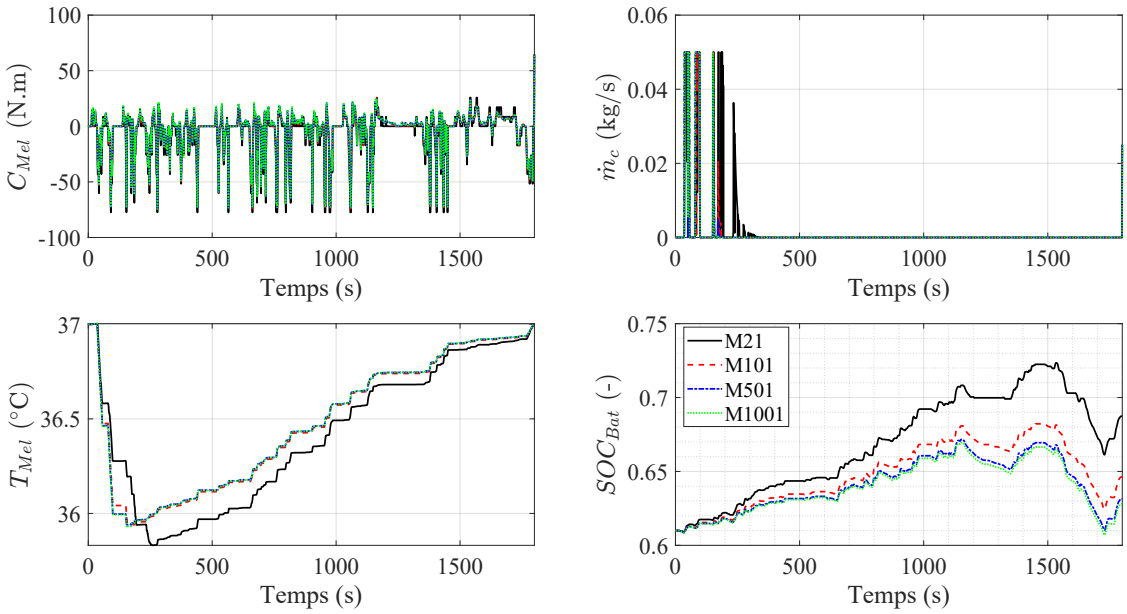

FIGURE 4.16 – Trajectoires des états et des commandes des quatre solutions de *DP* avec le découplage fréquentiel pour les maillages de 21, 101, 501 et 1001 mailles par variables respectivement en noir, rouge, bleu et vert pour les problèmes (4.29) et (4.30) avec contraintes finales (3.19).

Toutes les matrices de coût optimal sont totalement protégées de la pollution grâce aux *boundary lines*. Il est intéressant de voir ici que la répartition des valeurs dans les matrices de coût optimal des deux sous problèmes est différente. En effet, dans le premier sous-problème, le coût optimal dépend de la valeur de l'état de charge de la batterie et du temps tandis que dans le second sous-problème, le coût optimal ne semble dépendre que du temps et très peu de la température de la machine électrique.

Pour conclure, la figure 4.19 récapitule toutes les méthodes de résolution du problème avec contraintes du chapitre actuel et précédent. D'après les résultats présentés, il est important d'effectuer la méthodologie basée sur une analyse fréquentielle pour diviser le problème en somme de sous-problèmes les plus petits possible pour réduire le temps et la puissance de calcul nécessaire pour résoudre le problème de commande optimale avec une bonne qualité de résultat. Dès lors que les sous-problèmes sont définis, alors l'utilisation des améliorations de la programmation dynamique sont à utiliser pour la résolution de chaque sous-problème de commande optimale. Il est possible d'utiliser les *boundaries* comme le montre les résultats précédents, mais l'application de l'*Iterative Dynamic Programming* présentées dans le chapitre 1 est également possible. La figure B.4 en annexe B présente l'évolution du temps de calcul en fonction du coût à bilan batterie nul.

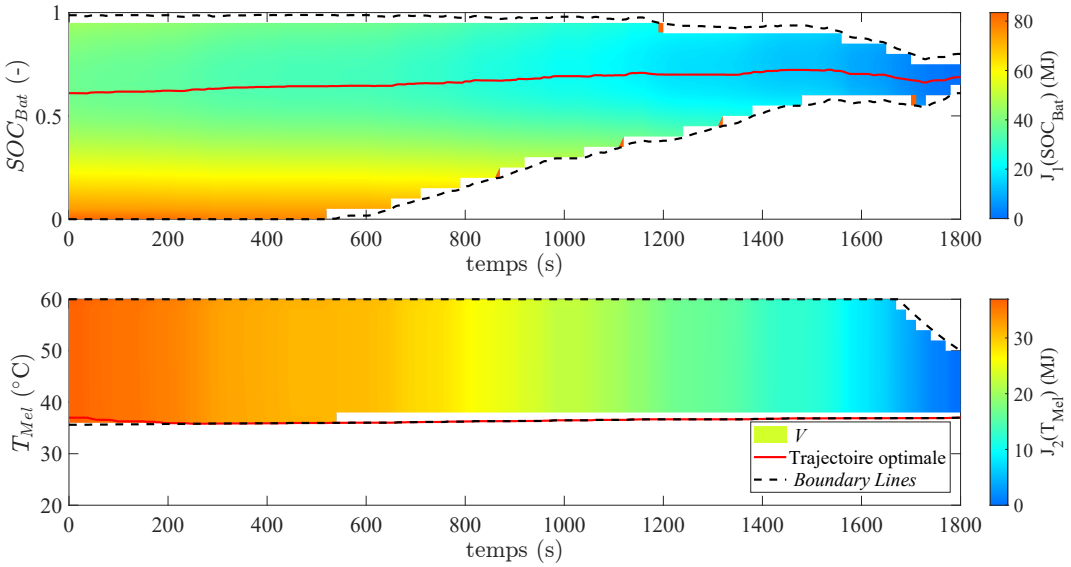

FIGURE 4.17 – Matrices de coût optimal des deux sous-problèmes pour un maillage homogène à 21 mailles

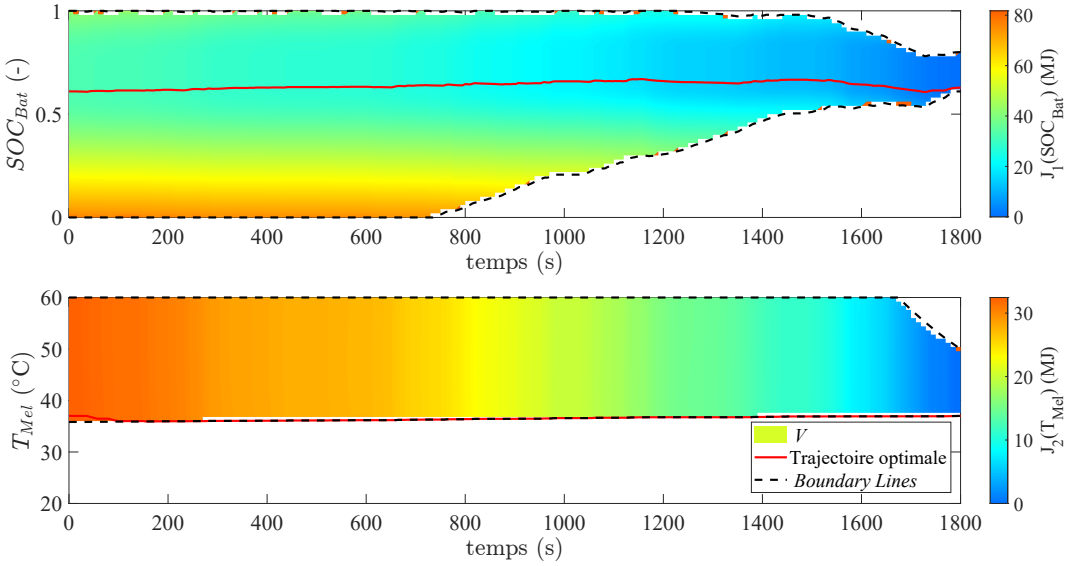

FIGURE 4.18 – Matrices de coût optimal des deux sous-problèmes pour un maillage homogène à 1001 mailles

4.3 Conclusion

La méthodologie développée durant la première partie du chapitre est comparée à la référence définie par l'amélioration des *boundary surfaces* du chapitre 3 sur un modèle véhicule hybride électrique parallèle.

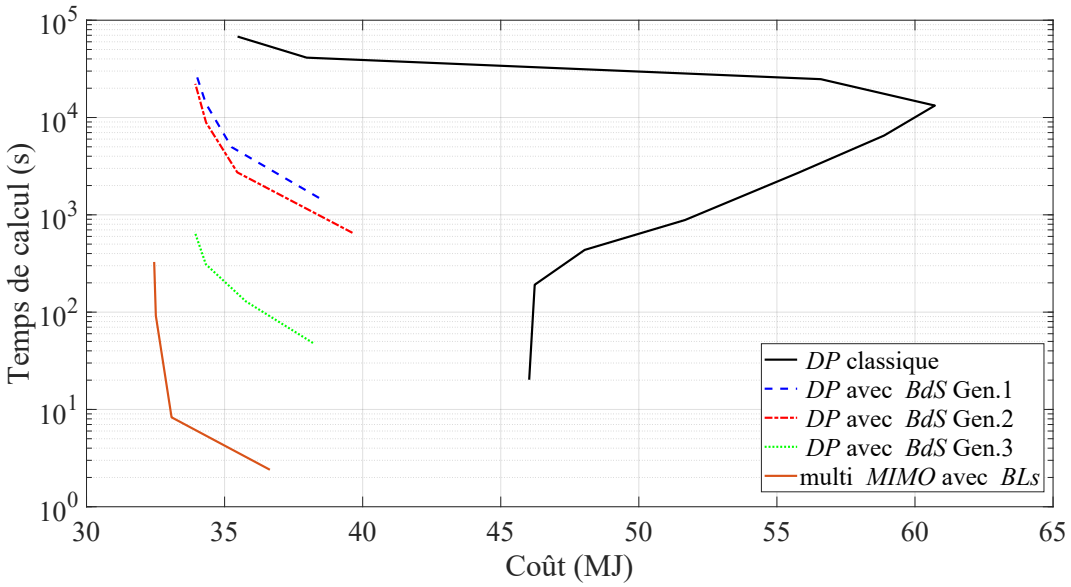

FIGURE 4.19 – Temps de calcul en fonction du coût pour les conditions initiales particulières pour le problème avec contraintes finales (3.19) des générations 1 à3 des *boundary surfaces*, de la programmation dynamique classique et de la résolution basée sur l'analyse fréquentielle (multi *MIMO*).

La méthodologie se base sur les outils mathématiques *RGA* et CD^3 qui indiquent s'il est possible de découpler un problème en une somme de sous-problèmes. Une décomposition du problème en somme de sous-problèmes a pour objectif de réduire la complexité de calcul de ce dernier. Pour ce faire, la méthodologie indique quelles sont les variables de commande qui sont les plus importantes pour chaque variable d'état ou de sortie. De plus, l'erreur générée par la non prise en compte d'une commande pour une sortie est également calculée pour savoir si la division du problème peut être totale ou partielle. Il existe alors trois issues à l'analyse fréquentielle : la première est que le découplage n'est pas possible et qu'il est obligatoire de considérer le problème dans son entièreté pour le résoudre ; la seconde est que le découplage est partiel, ce qui implique une résolution successive des sous-problèmes générés ; la dernière possibilité est que le découplage est total et que chaque sous-problème peut être résolu de manière indépendante.

La division d'un problème en somme de sous-problèmes permet de mettre plus de raffiner le maillage des programmations dynamiques résolvant les sous-problèmes de commande optimale dans le cas traité dans ce chapitre. Les résultats obtenus au cours de ce chapitre démontrent tout l'intérêt du découplage fréquentiel. En effet, les résultats sont autant, voire plus, qualitatifs que ceux donnés par la référence pour un temps de calcul divisé par 25.

Cette méthodologie ainsi que l'amélioration des *boundary surfaces* développée au cours de chapitre 3 seront exploitées pour permettre la résolution d'un problème de charge rapide d'une batterie de véhicule électrique dont le nombre de variables considérées ne permet pas la résolution par une programmation dynamique classique.

Chapitre 5

Application des contributions à la recharge rapide de batterie de véhicule électrique

Sommaire

La problématique de la recharge batterie dans le domaine automobile est une question de plus en plus importante. En effet, cette problématique apparaît avec l'arrivée massive des véhicules électriques (*BEV* pour *Battery Electric Vehicle*) qui requièrent un certain nombre de recharges pour faire un long trajet. Or, pour limiter le temps de trajet, le temps de la recharge de la batterie doit être le plus faible possible. Cependant, c'est lors de cette phase de vie de la batterie que la dégradation de cette dernière peut être importante, si la gestion thermique de cette dernière n'est pas bien assurée [114]. Pour limiter ce problème de hautes températures, deux solutions complémentaires sont possibles : la première est d'améliorer les performances de l'échangeur pour refroidir plus fortement la batterie ; la seconde est de mettre en place une loi de commande prenant en compte diverses informations liées à la batterie. C'est ce second axe qui est étudié dans cette thèse.

La problématique rencontrée actuellement est que la puissance de recharge est limitée à cause de la non homogénéité de la température dans l'ensemble de la batterie. Étant donné que l'eau permettant le refroidissement de la batterie se réchauffe le long de l'échangeur au contact de la batterie, le refroidissement en fin d'échangeur est moins performant qu'au début

car la différence de température est plus faible. Le problème d'homogénéité des températures dans la batterie limite alors la charge de la batterie. En effet, si cette dernière est trop froide, le courant maximal admissible par la batterie est très faible pour limiter sa dégradation tandis que si la batterie est trop chaude, le courant maximal admissible est important mais la dégradation est très importante.

Pour tendre vers une température homogène dans la batterie, il est plusieurs typologies d'échangeurs sont possibles [115]. Dans le cadre de cette étude, une architecture simple en série est utilisée pour mettre en place un problème de commande optimale. Ce dernier est résolu par programmation dynamique en exploitant les contributions des chapitres 3 et 4 et porte sur la charge rapide de batterie d'un véhicule électrique.

5.1 Présentation du problème de recharge rapide d'une batterie automobile

5.1.1 Modèle batterie étudié

Une batterie est composée d'un certain nombre de cellule en série et en parallèle pour atteindre la plage de tension et de courant maximal désirées. La tension de la batterie V_{Bat} est exprimée par l'équation (5.1), avec n_s le nombre de cellule en série et V_{cel} la tension d'une cellule.

$$V_{Bat} = n_s.V_{cel} \tag{5.1}$$

L'intensité traversant la batterie I_{Bat} est définie par l'équation (5.2), avec n_p le nombre de branche de cellule en parallèle et I_{cel} l'intensité traversant une cellule.

$$I_{Bat} = n_p.I_{cel} \tag{5.2}$$

Le modèle d'une cellule est alors représenté par la figure 5.1. La cellule est modélisée de la même façon que la batterie du chapitre 2, c'est-à-dire avec une tension à vide et une résistance interne, mais la prise en compte de la température dans la tension à vide et la résistance interne de la batterie est effectuée.

Spatialement, les cellules sont mises les unes à côté des autres, ce qui implique que les $N_{cel} = n_s.n_p$ cellules seront refroidies par le même écoulement comme illustré en figure 5.2. Le liquide utilisé dans ce système de régulation de température est de l'eau qui est mise en déplacement par une pompe. Afin d'assurer un bon contrôle de la température des cellules composant la batterie, un système de chauffage de l'eau appelé *heater* ainsi qu'un système de refroidissement appelé *chiller* sont implémentés dans la boucle de circulation de l'eau.

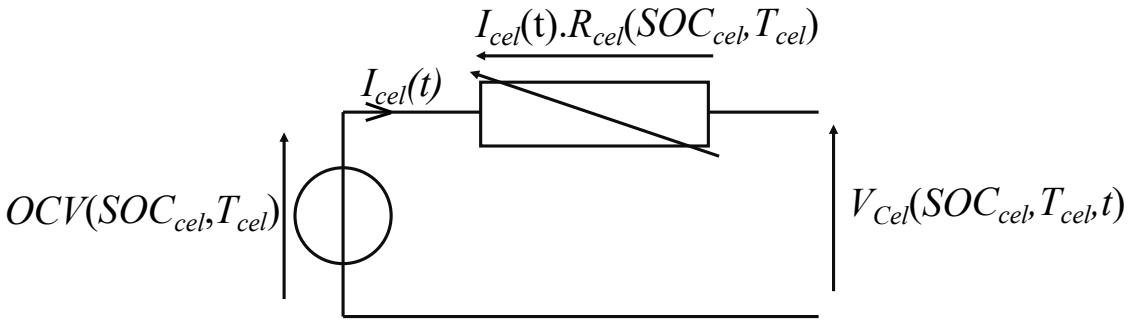

FIGURE 5.1 – Modèle électrique de cellule batterie simplifié exprimant la tension en fonction de la tension à vide OCV_{cel}, de la résistance interne R_{cel} et du courant de la cellule I_{cel}.

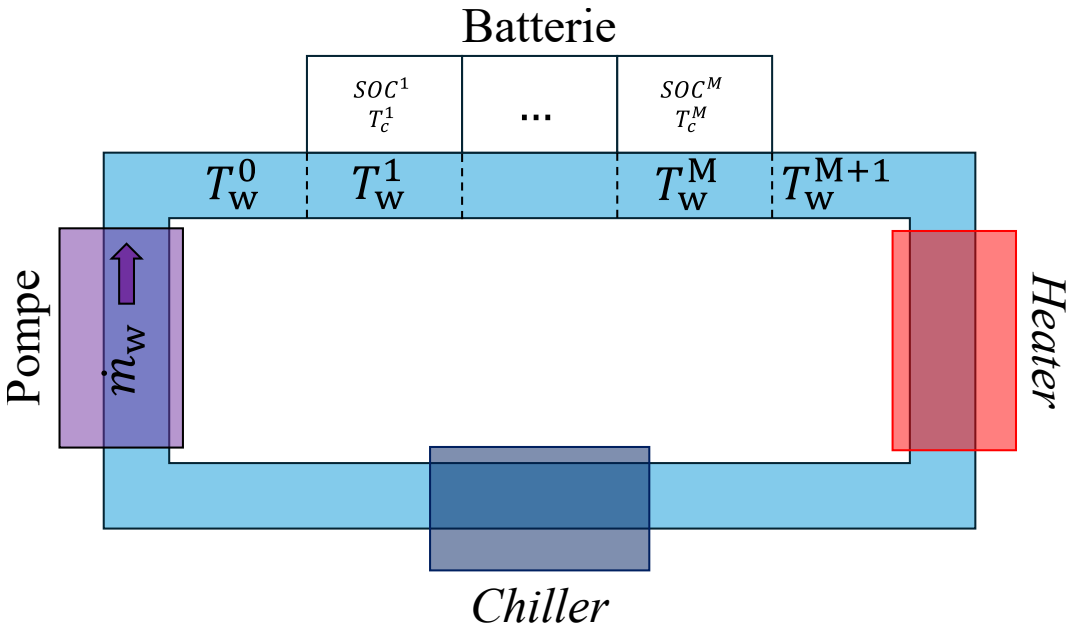

FIGURE 5.2 – Schéma du modèle considéré.

L'alimentation électrique de ces trois composants (pompe, *heater* et *chiller*) provient directement de la batterie, ce qui requiert une dépense d'énergie durant la recharge de cette dernière. Le bilan électrique à la batterie est alors le suivant :

$$P_{bat} = V_{bat}.I_{Alim} - P_{pump} - \frac{P_{reg}}{\eta_r} \tag{5.3}$$

avec I_{Alim}, le courant fourni par la borne de recharge à la batterie ainsi que P_{pump}, P_{reg} les deux puissances électrique d'alimentation des éléments consommateurs d'énergie. Premièrement, la consommation électrique de la pompe est exprimée par la même équation

que dans les chapitres précédents : $P_{pump} = Ws.\dot{m}_w$, avec Ws également fixé à 900 J/kg. Deuxièmement, les deux puissances électriques liées aux flux de chaleur sont évaluées comme suit : $P_{reg} = h.S.(T_w^0 - T_w^{M+1})$. La différence entre l'utilisation du *heater* ou du *chiller*, qui ne peut pas être simultanée, provient de leur efficacité énergétique ($\eta_h = 1$ et $\eta_c = 3$) :

$$\begin{cases} \eta_r = \eta_h \text{ si } T_w^0 - T_w^{M+1} > 0 \\ \eta_r = \eta_c \text{ sinon} \end{cases} \tag{5.4}$$

Ces deux systèmes n'ont pas la même efficacité énergétique car le réchauffement de l'eau est assuré électriquement par l'effet Joule supposé parfait tandis que le refroidissement est assuré par une pompe à chaleur dont le coefficient de performance est supposé de 3. Naturellement, ces deux puissances sont comprises entre des valeurs minimales et maximales, il apparaît alors les deux inéquations suivantes qui font office de contraintes au système :

$$\begin{cases} 0 \leq \frac{P_{reg}}{\eta_h} \leq 3000 \\ -3000 \leq \frac{P_{reg}}{\eta_c} \leq 0 \end{cases} \tag{5.5}$$

Dans le cadre de cette étude, la batterie est divisée en deux partie égales : $M = 2$.

5.1.2 Contrôle de référence du système

La loi de commande de référence tout-ou-rien est basée sur des règles. Le courant d'alimentation appliqué au système est toujours le courant maximal possible pouvant être appliqué au système. En fonction de la température des cellules de la batterie et de l'eau dans l'échangeur, le débit et la température de l'eau en entrée de l'échangeur varie. Ainsi, tant que la température de l'eau sous la deuxième cellule T_w^2 de la batterie est inférieure à 20°C, le débit et la température de l'eau en entrée sont adaptées pour maximiser le chauffage de la batterie. Si T_w^2 est supérieure à 20°C, alors le système de régulation est mis de telle sorte à réduire la consommation d'énergie du système de régulation de température. Enfin, si la température T_c^1 de la première cellule est supérieure à 35°C, alors un refroidissement maximal est appliqué à la batterie.

C'est donc à cette stratégie que la loi de commande optimale déterminée par les différents outils développés durant la thèse sera comparée.

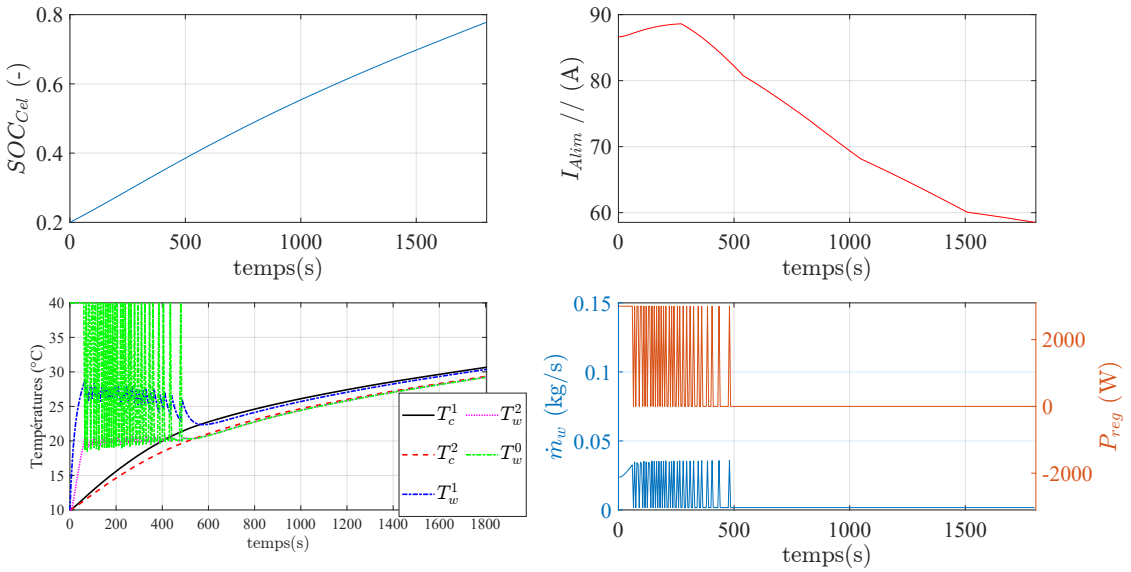

FIGURE 5.3 – Stratégie de regulation de température actuelle lors de la recharge d'une batterie à une température initiale de 10°C.

5.2 Application de l'analyse fréquentielle sur le modèle batterie

5.2.1 Mise sous forme d'état

D'après la conclusion du chapitre 4, il faut commencer par appliquer la méthodologie d'analyse fréquentielle pour savoir s'il est possible de décomposer le problème en somme de sous-problèmes.

$$SOC_{k+1} = SOC_k + \Delta t.\frac{I_{Alim}}{Q} \tag{5.6}$$

$$T^i_{c_{k+1}} = T^i_{c_k} + \Delta t.\frac{R^i(SOC_B, T^i_{c_k}).I^2_{Alim_k} - h.S.(T^i_{c_k} - T^i_{w_k})}{m_c.Cp_c} \tag{5.7}$$

$$T^i_{w_{k+1}} = \frac{m_w - \Delta t.\dot{m}_c}{m_w}.(T^i_{w_k} + \frac{\Delta t.h.S.(T^i_{c_k} - T^i_{w_k})}{m_w.Cp_w}) + T^{i-1}_{w_k}.(1 - \frac{m_w - \Delta t.\dot{m}_c}{m_w}) \tag{5.8}$$

Étant donné que la batterie est divisé en deux dans cette étude, cela implique donc deux températures de cellule T_c et d'eau sous les cellules T_w différentes. Cela porte donc le nombre de variable d'état à 5 :

$$\boldsymbol{x} = [SOC_B; T^1_c; T^2_c; T^1_w; T^2_w]^T \tag{5.9}$$

Trois variables de commande apparaisent dans les équations (5.6)-(5.8) :

$$\boldsymbol{u} = \begin{bmatrix} I_{Alim} \\ \dot{m}_c \\ T_w^0 \end{bmatrix} \tag{5.10}$$

La variable T_w^0 représente la température de l'eau à l'entrée de l'échangeur. Il est considéré que cette température peut varier instantanément sans contraintes. Les équations utilisées étant discrète en temps, la représentation d'état l'est aussi :

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}.\boldsymbol{x}_k + \boldsymbol{B}.\boldsymbol{u}_k \\ \boldsymbol{y}_{k+1} = \boldsymbol{C}.\boldsymbol{x}_k + \boldsymbol{D}.\boldsymbol{u}_k \end{cases}$$

Étant donné que le système est construit dans le domaine temporel discret, les matrices de représentation d'état doivent être adaptées [116]. Les matrices de représentation d'état sont les suivantes :

$$A = \begin{pmatrix} A_{11} & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & 0 & A_{24} & 0 \\ A_{31} & 0 & A_{33} & 0 & A_{35} \\ 0 & A_{42} & 0 & A_{44} & 0 \\ 0 & 0 & A_{53} & A_{54} & A_{55} \end{pmatrix} \tag{5.11}$$

avec $A_{11} = 1$; $A_{21} = \Delta t.\frac{I^2_{Alim_L}}{m_c.Cp_c}.\frac{\partial R^1}{\partial SOC_B}$; $A_{22} = 1 + \frac{\Delta t}{m_c.Cp_c}.(\frac{\partial R^1}{\partial T_{c_1}}.I^2_{Alim_L} - h.S)$; $A_{24} = -\frac{\Delta t.h.S}{m_c.Cp_c}$; $A_{31} = \Delta t.\frac{I^2_{Alim_L}}{m_c.Cp_c}.\frac{\partial R^2}{\partial SOC_B}$; $A_{33} = 1 + \frac{\Delta t}{m_c.Cp_c}.(\frac{\partial R^2}{\partial T_{c_2}}.I^2_{Alim_L} - h.S)$; $A_{35} = -\frac{\Delta t.h.S}{m_c.Cp_c}$; $A_{42} = \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w}.\frac{\Delta t.h.S}{m_w.Cp_w}$; $A_{44} = \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w}.(1 - \frac{\Delta t.h.S}{m_w.Cp_w})$; $A_{53} = \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w}.\frac{\Delta t.h.S}{m_w.Cp_w}$; $A_{54} = 1 - \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w}$; $A_{55} = \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w}.(1 - \frac{\Delta t.h.S}{m_w.Cp_w})$.

$$B = \begin{pmatrix} \frac{\Delta t}{Q} & -\frac{w_s}{V_{B_L}} & 0 \\ \frac{2.\Delta t.R^1.I_{Alim_L}}{m_c.Cp_c} & 0 & 0 \\ \frac{2.\Delta t.R^2.I_{Alim_L}}{m_c.Cp_c} & 0 & 0 \\ 0 & -\frac{\Delta t}{m_w}.(T^1_{w_L} + \frac{\Delta t.h.S}{m_w.Cp_w}.(T^1_{c_L} - T^1_{w_L}) + T^0_{w_L}) & 1 - \frac{m_w - \Delta t.\dot{m}_{c_L}}{m_w} \\ 0 & -\frac{\Delta t}{m_w}.(T^2_{w_L} + \frac{\Delta t.h.S}{m_w.Cp_w}.(T^2_{c_L} - T^2_{w_L}) + T^1_{w_L}) & 0 \end{pmatrix} \tag{5.12}$$

5.2.2 Résultats de l'analyse fréquentielle

Des courbes représentatives de l'analyse fréquentielle sont présentées en figure 5.4. Il est possible de constater qu'un découpage en deux problèmes se dessine. En effet, d'un côté, la commande d'intensité I_{Alim} pilote l'état de charge de la batterie (SOC_B) et les deux

températures de cellule (T_c^1 et T_c^2). De l'autre côté, le débit et la température de l'eau ($\dot{m}_c$ et T_w^0) influent sur la température des deux masses d'eau sous chacune des cellule batterie (T_w^1 et T_w^2).

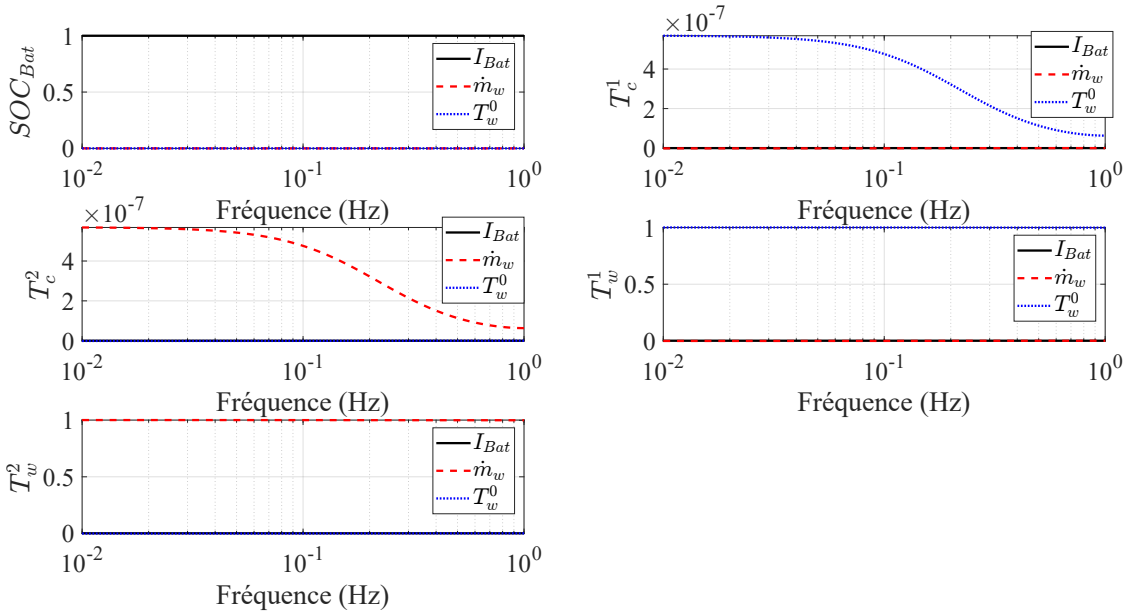

FIGURE 5.4 – Analyse *Relative Gain Array* représentative à un point de fonctionnement.

L'analyse CD^3 est alors réalisée pour savoir s'il est possible ou non de découpler totalement les problèmes. La figure 5.5 montre pour un point de linéarisation suivant : $SOC_{Bat} = 0,5$, $T_{c1} = T_{c2} = T_{w1} = T_{w2} = 10$°C et les commandes extramales et médianes, qu'il n'est pas possible de découpler totalement les deux sous-problèmes (multi-*MIMO*). Une résolution successive des deux sous-problèmes doit être mise en place pour résoudre le problème de commande optimale.

Le premier sous-problème traité sera celui qui lie l'intensité de charge à l'état de charge de la batterie et aux températures des deux modules batterie considérés. Le second sous-problème concerne la gestion de température de la batterie en pilotant le débit et la température du débit entrant dans le système de refroidissement aux deux températures de masse d'eau sous les modules batterie. Le problème initial à trois commandes et cinq états se décompose alors en un problème à une commande et trois états et un problème à deux états et deux commandes comme l'illustre la figure 5.6. L'utilisation des outils développés durant le chapitre 3 seront utilisés pour améliorer la précision du résultat obtenu, notamment pour le deuxième sous-problème qui est à deux états.

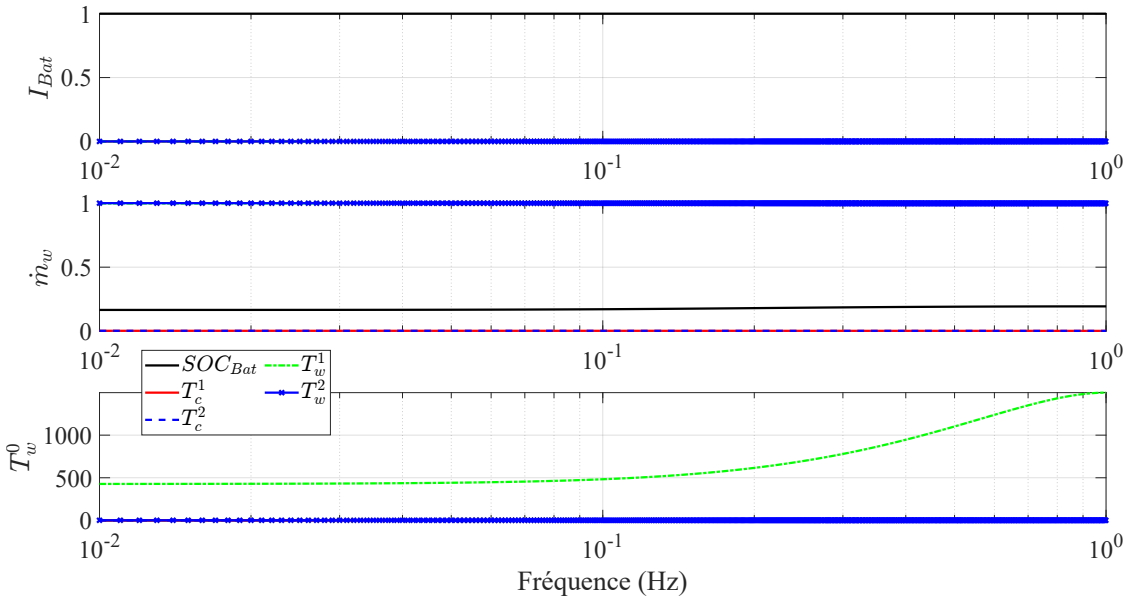

FIGURE 5.5 – Analyse *Column Diagonal Dominance Degree* représentative à un point de fonctionnement.

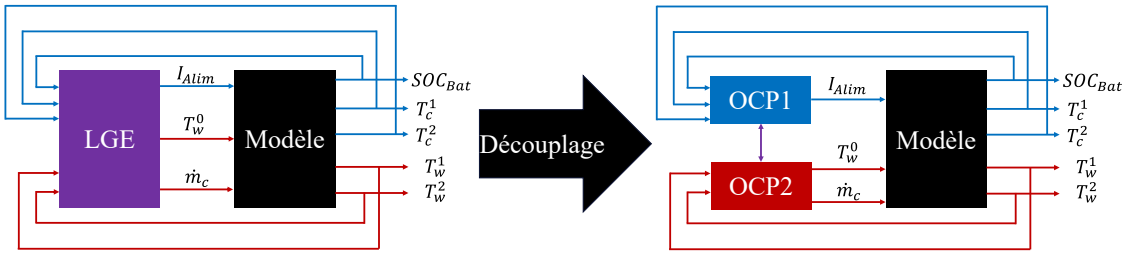

FIGURE 5.6 – Division du problème initial à cinq états et trois commandes en deux sous-problèmes dont un à trois états et une commande et l'autre à deux états et deux commandes.

5.2.3 Description du critère d'optimisation du problème de commande optimale

Le critère qui doit être minimiser dans cette étude est le temps de charge (5.13). Cela implique que l'état de charge de la batterie SOC_{Bat} doit être maximisé, ce qui revient par définition à maximiser l'intégrale du courant de la batterie I_{Bat}.

$$\begin{aligned} J &= max(\sum_{k=0}^{N-1} I_{Bat_k}.\Delta t + \Phi(x_N)) \\ J &= max(\sum_{k=0}^{N-1} I_{Alim_k} - (I_{pump_k} + I_{reg_k}).\Delta t + \Phi(x_N)) \end{aligned} \tag{5.13}$$

Ce courant de batterie est alors composée de plusieurs courant, I_{Alim} qui provient du chargeur et apporte de l'électricité à la batterie, I_{pump} qui est nécessaire à l'alimentation électrique de la pompe et I_{reg} qui provient des deux organes (*chiller* et *heater*) qui permettent de réguler la température de l'eau. Ces trois courants sont alors déterminés dans les deux sous-problèmes, ce qui donne ces deux sous-problèmes d'optimisation :

$$J_1 = max(\sum_{k=0}^{N-1} I_{Alim_k}.\Delta t + \Phi(x_N)) \tag{5.14}$$

$$J_2 = max(\sum_{k=0}^{N-1} (I_{pump_k} + I_{reg_k}).\Delta t + \Phi(x_N)) \tag{5.15}$$

Il est cependant important de garder en mémoire que c'est la somme des deux nouveaux critères J_1 et J_2 qui doit être maximisée, et qu'une sur-optimisation de l'un des deux critères peut mener à une non optimisation de critère initial J. C'est pourquoi, dans chaque programmation dynamique, l'estimation de l'impact de la variation des autres états qui sont dans l'autre programmation dynamique est réalisée.

Le maillage de l'intensité d'alimentation dans le cadre de la première programmation dynamique est qualifié de permanent et irrégulier d'après le chapitre 1. Cette irrégularité provient de la variation du courant maximal accepté par la batterie en fonction de la température des cellules et de l'état de charge de la batterie comme le montre la figure 5.7. Ne pas utiliser cette information et appliquer un maillage uniforme pour l'intensité est peu souhaitable car pour les conditions à très froides températures et haut état de charge (points en bleu sur la figure), l'intensité maximale est très faible ce qui rendrait l'utilisation de la majeure partie des commandes testées comme infaisable car la contrainte de l'intensité maximale acceptée ne serait pas validée.

5.3 Résultats

Deux conditions initiales de températures seront étudiées ici pour évaluer les performances de la nouvelle loi de commande comparé à celle déjà existante. La comparaison sera faite dans des conditions froides ($T_{ext} = -20$°C) et des conditions tempérées ($T_{ext} = 10$°C).

Pour résoudre ce problème, voici un algorigramme de la résolution proposée :

Les conditions initiales de la recharge sont les suivantes : $SOC_{Bat_0} = 0,2$, $T_{c1_0} = T_{c2_0} = T_{w1_0} = T_{w2_0} = 30$°C. Le véhicule a alors une période de trente minutes pour recharge la batterie jusqu'à la cible $SOC_{cible} = 0,8$. Dans le cas où il ne serait pas possible de recharger autant la batterie, l'état de charge final doit être le plus élevé possible. Le maillage utilisé pour résoudre le problème est le suivant :

— 11 mailles pour l'intensité normalisée entre 0 et 1.

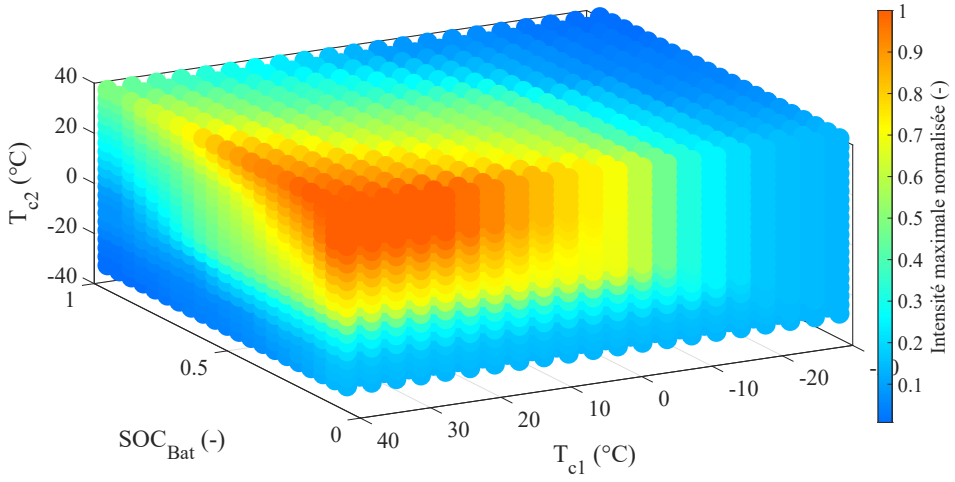

FIGURE 5.7 – Intensité maximale admissible en fonction de l'état de charge de la batterie et des deux températures de modules.

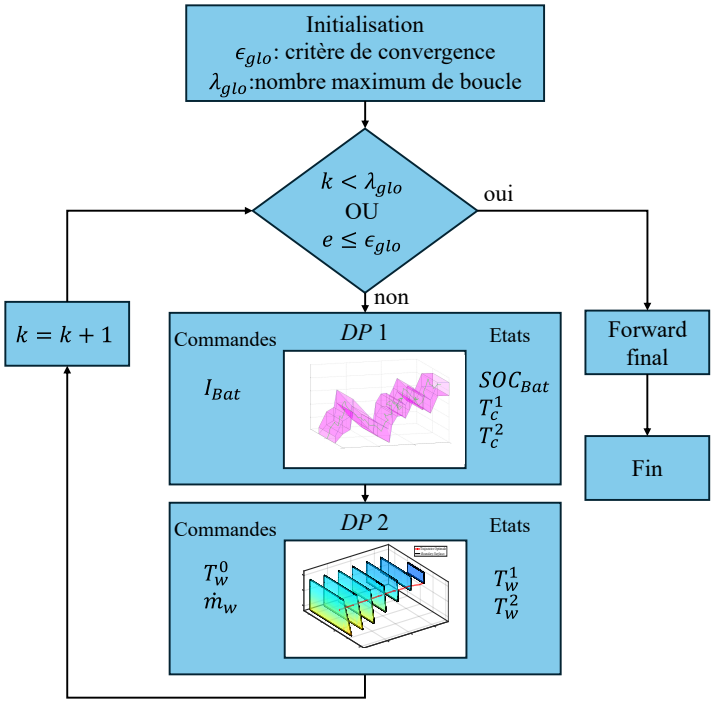

FIGURE 5.8 – Algorigramme de résolution de l'*OCP*. Le premier sous-problème est résolu par une *IDP* à une commande et trois états et le second sous-problème est résolu par une *DP* avec utilisation des *boundary surfaces* pour un problème à deux états et deux commandes

— 153 mailles pour l'état de charge entre $0,05$ et $0,81$.
— 91 mailles pour les cinq températures entre T_{ext} et 40.
— 101 mailles pour le débit entre son minimum et son maximum.

Par la suite, les températures utilisées dans la première programmation dynamique (figure 5.8), le maillage est recentré sur la trajectoire de référence avec une latitude de plus ou moins 5°C tandis que la latitude laissée à l'état de charge est de plus ou moins $0,05$. Les problèmes sont ainsi résolu en une heure environ pour une complexité de calcul de 18 Go de RAM. Avec ce maillage, il ne serait pas possible de résoudre le problème complet car la mémoire vive serait trop importante. Dans les deux sous problèmes, auncune pénalité sur les états finaux n'est appliquée $\phi(\boldsymbol{x}_N) = 0$ car dans cette étude, auncune contrainte n'est imposée sur les états finaux.

5.3.1 Conditions froides

Les conditions extérieures sont les suivantes : $T_{c1_0} = T_{c2_0} = T_{w1_0} = T_{w2_0} = -20$°C. La température de l'eau ne peut pas être inférieure à la température extérieure dans tout le circuit d'eau. Ainsi, la température T_w^0 à l'entrée de l'échangeur entre la batterie et le circuit d'eau est contrainte entre -20°C et 40°C.

Les figures 5.9 et 5.10 présentent respectivement les solutions de la résolution de l'*OCP* et de la loi de commande *rule based* de référence. Les deux lois de commande ont une courbe d'intensité d'alimentation similaires car le courant maximal est appliqué à chaque instant. Cependant, cette courbe de courant maximal varie en fonction de l'état de charge de la batterie et des températures des deux packs. C'est notamment sur la gestion des températures que les deux lois de commandes diffèrent. En effet, la loi de commande de référence applique tout le temps une puissance de chauffe maximale avec une température la plus chaude possible, la résolution de l'*OCP* indique qu'il faut certes chauffer le plus possible mais de préférence avec un débit élevé et une température légèrement plus élevée que celle déjà présente dans l'échangeur. Cela permet ainsi d'avoir un état de charge de batterie à $0,51$ au lieu de $0,46$ pour une recharge de trente minutes.

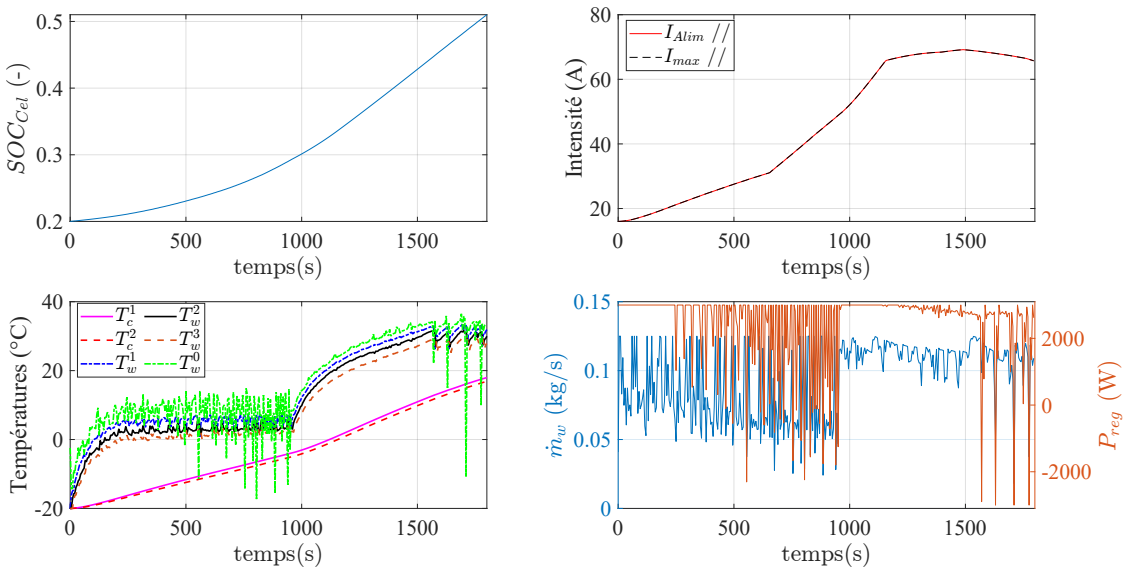

FIGURE 5.9 – Résultats par résolution de l'*OCP* pour une température extérieure de −20°C.

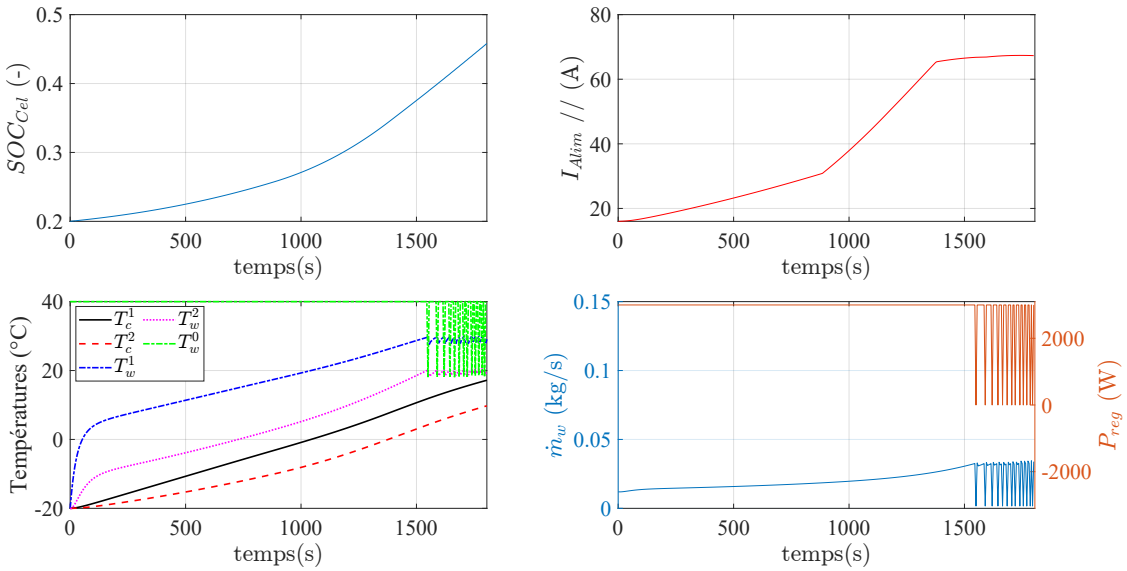

FIGURE 5.10 – Résultats de la loi de commande *rule based* de référence pour une température extérieure de −20°C.

5.3.2 Conditions tempérées

Les conditions extérieures sont les suivantes : $T_{c1_0} = T_{c2_0} = T_{w1_0} = T_{w2_0} = 10$°C. La température de l'eau ne peut pas être inférieure à la température extérieure dans tout le circuit d'eau. Ainsi, la température T_w^0 à l'entrée de l'échangeur entre la batterie et le circuit d'eau est contrainte entre 10°C et 40°C.

Les figures 5.11 et 5.12 présentent respectivement les solutions de la résolution de l'*OCP* et de la loi de commande *rule based* de référence. Les deux courbes d'intensité d'alimentation de la batterie sont également proche mais une différence importante peut être soulignée à partir de 500 secondes de charge. En effet, l'intensité déterminée par la résolution de l*OCP* reste constante un moment avant de chuter progressivement là où celle déterminée par la *rule based* diminue plus rapidement. Cette différence provient du fait que la batterie n'est plus chauffée à partir de 500 secondes dans la *rule based*. Ainsi, la batterie est totalement chargée avec la loi de commande déterminée par l'*OCP* en 1765 secondes tandis que la batterie chargée par la *rule based* n'est pas entièrement chargée ($SOC_{cel_N} = 0,79$) au bout des trente minutes de charge.

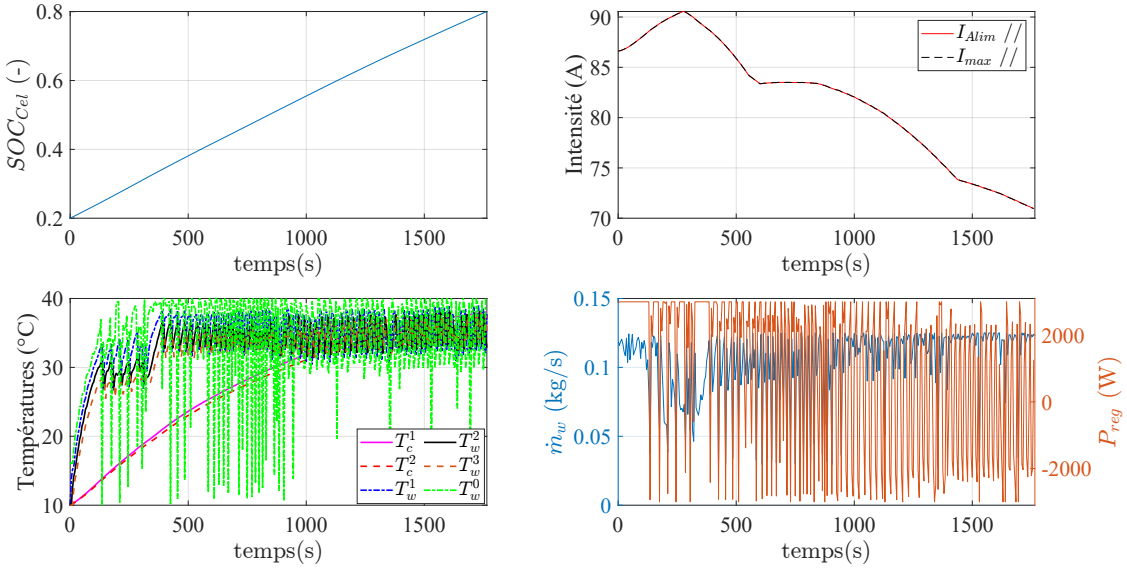

FIGURE 5.11 – Résultats par résolution de l'*OCP* pour une température extérieure de 10°C.

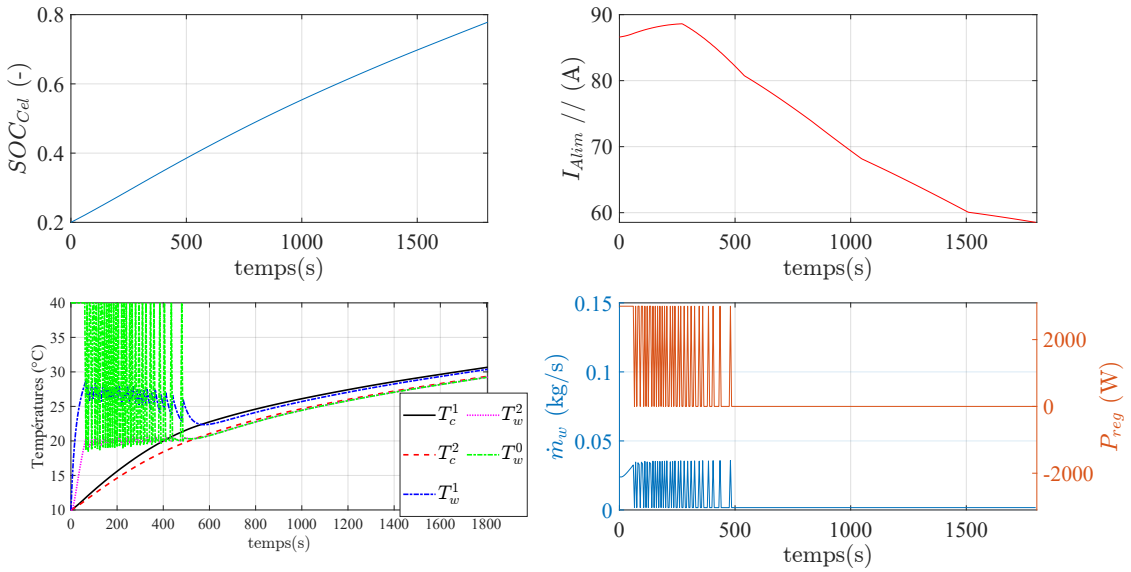

FIGURE 5.12 – Résultats de la loi de commande *rule based* de référence pour une température extérieure de 10°C.

5.4 Conclusion

Il a pu être vu au cours de ce chapitre que l'exploitation des *boundary surfaces* développées dans le chapitre 3 et de la division d'un problème en somme de sous-problèmes basée sur une analyse fréquentielle vue dans le chapitre 4 ont permis la résolution d'un problème de commande optimale à trois commandes et cinq états dans un temps de calcul raisonnable.

La résolution du premier sous-problème à trois états et une commande faite ici par une *Iterative Dynamic Programming* pourrait être améliorée par la mise en place de l'extension des *boundary surfaces* à trois états. Grâce aux travaux développés dans la thèse, il devrait être possible d'effectuer la mise en place de l'amélioration avec un temps de calcul réduit en utilisant précisément la dernière version présentée. C'est donc un premier axe de travail à mener par la suite. Un second axe de travail est l'augmentation du nombre de parties considérées pour la discrétisation de la batterie. En effet, les phénomènes observés pour $M = 2$ seront peut-être différents pour des valeurs plus grandes.

Les temps de charge présentés dans le différentes conditions initiales montrent que le temps de charge n'est pas négligeable pour le temps total de d'un long trajet. Un axe de travail pour réduire cette problématique est de préparer thermiquement la batterie avant une charge pour minimiser le temps total de trajet.

Chapitre 6

Conclusion et perspectives

Les travaux de la thèse permettent d'appliquer des méthodologies de commande optimale, en particulier la programmation dynamique, plus efficacement grâce aux développement d'outils. Ces derniers ont notamment été utilisés dans le contexte de loi de gestion de l'énergie de véhicule hybride, mais également dans un contexte de recharge batterie.

A travers le premier chapitre du manuscrit, la question du choix de la méthodologie de commande optimale a été évoquée. En effet, cette décision conditionne la modélisation du système (linéaire, convexe ou non linéaire) mais également la complexité du modèle avec le nombre de variables (commande ou état) considérées. Le choix s'est alors porté sur la programmation dynamique dont l'inconvénient majeur est l'augmentation du temps de calcul en fonction du nombre de variable impliquées.

Le second chapitre, illustre l'application de la programmation dynamique pour évaluer l'intérêt de revaloriser les pertes du moteur thermique dans une architecture hybride électrique série. L'utilisation de méthode de contrôle optimale permet dans ce cas de s'assurer que toutes les estimations de consommation déterminées sont les meilleures envisageable pour le modèle. Il a ainsi pu être montré qu'une réduction de consommation d'environ 20 % pouvait être espérée grâce à l'implémentation d'une structure de machines récupératrices basées sur des cycles de Rankine. Ces résultats ont fait l'objet d'une publication [117]. Cependant, des hypothèses fortes sur le modèle ont été faites pour pouvoir appliquer l'outil des *boundary lines* de la programmation dynamique. Pour remédier à ce problème, deux contributions ont été apportées pour supprimer ces hypothèses.

Dans le troisième chapitre, l'extension de l'outil *boundary lines* à deux états à été développée. Cette extension, appelée *boundary surfaces*, a pour effet de protéger la matrice de coût optimal des pollutions ou contaminations liées aux contraintes finales du problème de commande optimale. Cependant, les temps de calculs sont très fortement rallongés à iso-maillage car la détermination du contour à l'interface entre la surface acceptée et rejetée ainsi que les nouvelles méthodes d'interpolations ajoutent un temps de calcul important. Cependant, toujours à iso-maillage, la qualité du résultat est fortement accrue. A tel point que pour un même temps de calcul, il est préférable d'avoir un maillage moins fin et utiliser le nouvel outil plutôt que d'avoir un maillage plus fin sans l'outil. Toutefois, il n'est pas envisageable de penser à une généralisation à trois états ou plus avec cette version de l'outil. C'est pourquoi, deux autres versions de l'algorithme ont été développées. La seconde version se voit agrémentée d'une nouvelle méthode de recherche de points dans l'espace pour générer les contours. Celle-ci est bien plus rapide que celle utilisée dans la première version. La troisième et dernière

version modifie l'effet de l'information des *boundary* dans le *backward* de la programmation dynamique. La prise en compte est désormais moins précise mais plus rapide. Il est alors possible d'avoir un algorithme seulement deux fois plus lent à iso-maillage. En plus de cet avantage, la troisième version de l'algorithme peut être envisagée pour plus de deux états. La première version de l'extension à été publiée dans [118].

Le quatrième chapitre porte sur une approche fréquentielle du problème de contrôle optimal. L'objectif est de décomposer le problème initial en une succession de sous-problèmes afin de réduire la complexité de calcul de la programmation dynamique dans ce manuscrit mais son application n'est pas seulement limitée à la programmation dynamique. Pour mener l'analyse fréquentielle, deux outils sont utilisés : le *Relavtive Gain Array* qui détermine l'importance de chaque commande pour toutes les sorties ; le *Column Diagonal Dominance Degree* qui évalue le taux d'interaction d'une commande sur les sorties qui ne lui sont pas associées. Trois résultats sont possible à la suite de l'analyse : non/partiellement/totalement dissociable. Dans le premier cas, il faut considérer le problème dans son ensemble pour résoudre le problème, tandis que dans les deux autres cas, il est possible de décomposer le problème. Dans les deux cas ; la résolution des problèmes est successive et itérative jusqu'à atteindre la convergence du résultat. Résoudre des problèmes de contrôle optimal avec ce découplage fréquentiel apporte une sous optimalité qu'il faut maîtriser et minimiser. Ce dernier point s'effectue par une allocation des ressources de calcul facilité par la taille des problèmes qui sont plus raisonnables. Ainsi des résultats d'une bonne qualité peuvent être obtenus dans un temps plus que raisonnable. Cela s'explique par le fait que les résultats de la programamtion dynamique sont dépendant du maillage et le découplage permet d'avoir un maillage plus fin. Ce travail a été publié [60].

Il se dégage une certaine imbrication des deux contributions présentées dans les chapitres trois et quatre. Étude fréquentielle du problème pour décomposer le mieux possible le problème initial, puis utiliser des outils d'amélioration type *boundaries* en fonction du nombre d'états concernés par le sous-problème.

C'est sur ce principe qu'est construit le cinquième et dernier chapitre qui condense dans un seul problème tous les apports de la thèse. Le problème considéré est celui de la charge rapide de batterie d'un véhicule électrique. L'objectif est dans un temps limité d'effectuer une recharge complète ou de stocker le plus d'énergie possible dans la batterie.

Plusieurs perspectives de travail s'ouvrent à la fin de celui-ci. Premièrement, les contributions pourraient être appliquées au modèle, et ses complexifications, du second chapitre. Cela permettrait aux résultats d'être plus proches de ce qui serait obtenu par un prototype ou banc expérimental.

Deuxièmement, l'étude des maillage hétérogène dans le chapitre trois permet de mettre en lumière son intérêt, mais il serait intéressant de prédire à l'avance la répartition des ressources de calcul à appliquer au maillage.

Troisièmement, l'application de l'enxtension des *boundaries* à trois états ou plus sur un modèle amélioré de celui du chapitre 2 ou d'un nouveau modèle serait intéressant.

Annexe A

Effet du nombre de points N_{max} dans la construction du contour des *boundary surfaces.*

La détermination du nombre de points maximal à sélectionner pour définir le contour a fait l'objet d'une étude de sensibilité. La table A.1 présente les résultats obtenus pour la première version de l'amélioration pour un maillage uniforme homogène permanant de 21 mailles par variables. Uniquement le résultat des conditions initiales particulières est regardé.

Table A.1 – Résultats obtenus avec une Programmation Dynamique avec *boundary surfaces* première génération (Gen.1) pour le problème avec contraintes finales.

N_{max} (-)	Coût (MJ)	SOC_{Bat_N} (-)	T_{Mel_N} (°C)	RAM (Go)	Temps de calcul (s)			
					BdS	*BW*	*FW*	Total
25	64,75	0,9752	41	0,05	170,6	179,4	7	357
50	42,86	0,723	40,2	0,05	337,6	387,9	0,7	726,2
100	38,42	0,6997	38,7	0,06	922,5	630,2	0,8	1553,5
150	38,11	0,6975	38,6	0,06	1084,5	649,2	0,7	1734,5
200	38,13	0,6963	38,6	0,06	1574,6	768,7	0,7	2344

L'étude montre que la qualité des résultats augmente avec le nombre de points maximal pour générer le contour à l'interface entre le domaine faisable et infaisable. Cependant, le temps de calcul augmente également avec la qualité car le nombre de calcul à faire est proportionnel aux nombres de points dans le contour aussi bien lors de la génération des contours à l'interface entre le domaine faisable et infaisable que lors de son exploitation dans les interpolations dans le *backward*. C'est pourquoi le choix $N_{max} = 100$ a été fait comme compromis entre qualité des résultats et temps de calcul.

Annexe B

Coût au Bilan Batterie Nul en fonction du temps de calcul des contributions.

Les figures 3.14, 3.17, 3.23 et 4.19 sont présentés ici pour le coût à bilan batterie nul. Les figures présentées dans cette annexe sont très proches de celles citées car les résultats aux conditions initiales spécifiques et à bilan batterie nul sont proches numériquement et l'évolution des résultats en fonction du maillage sont similaires.

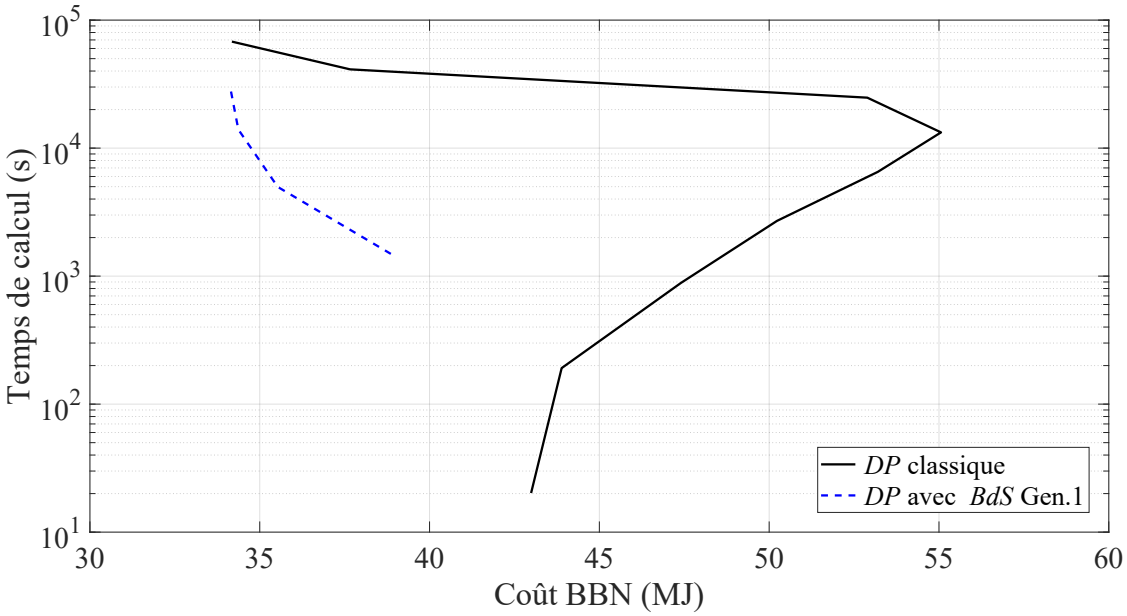

Figure B.1 – Temps de calcul en fonction du coût à bilan batterie nul pour le problème avec contraintes finales (3.19) de la génération 1 des *boundary surfaces* et de la programmation dynamique classique.

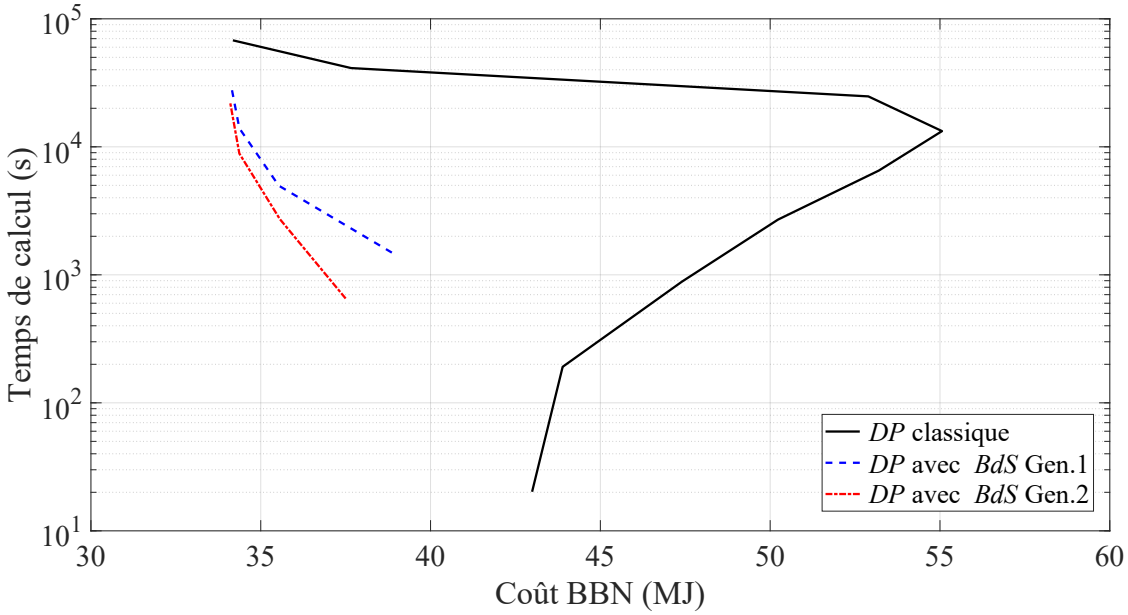

FIGURE B.2 – Temps de calcul en fonction du coût à bilan batterie nul pour le problème avec contraintes finales (3.19) des générations 1 et 2 des *boundary surfaces* et de la programmation dynamique classique.

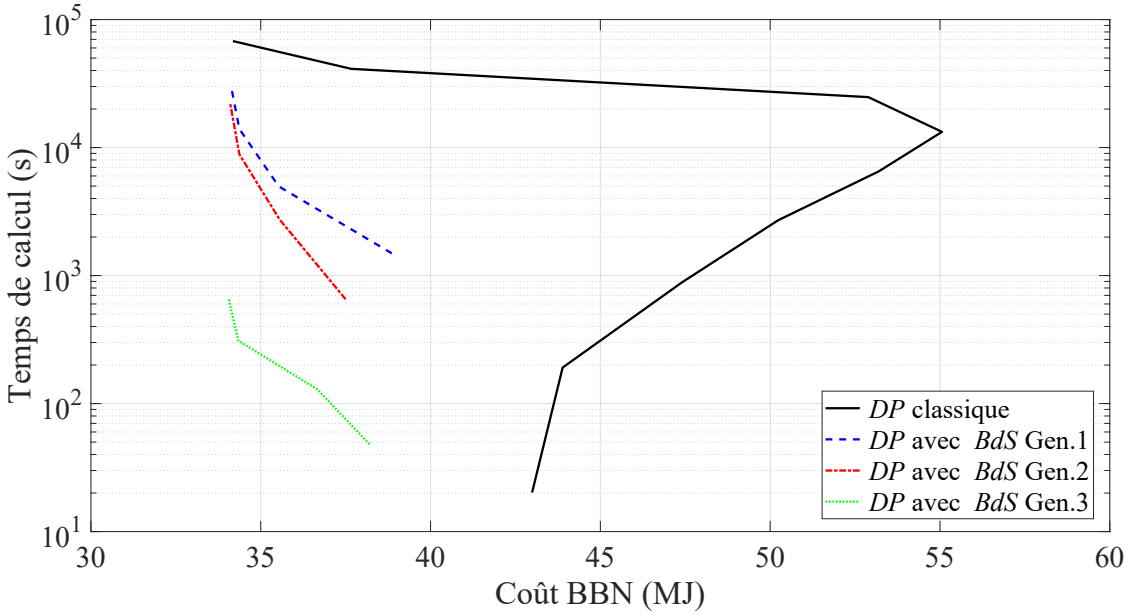

FIGURE B.3 – Temps de calcul en fonction du coût à bilan batterie nul pour le problème avec contraintes finales (3.19) des générations 1 à 3 des *boundary surfaces* et de la programmation dynamique classique.

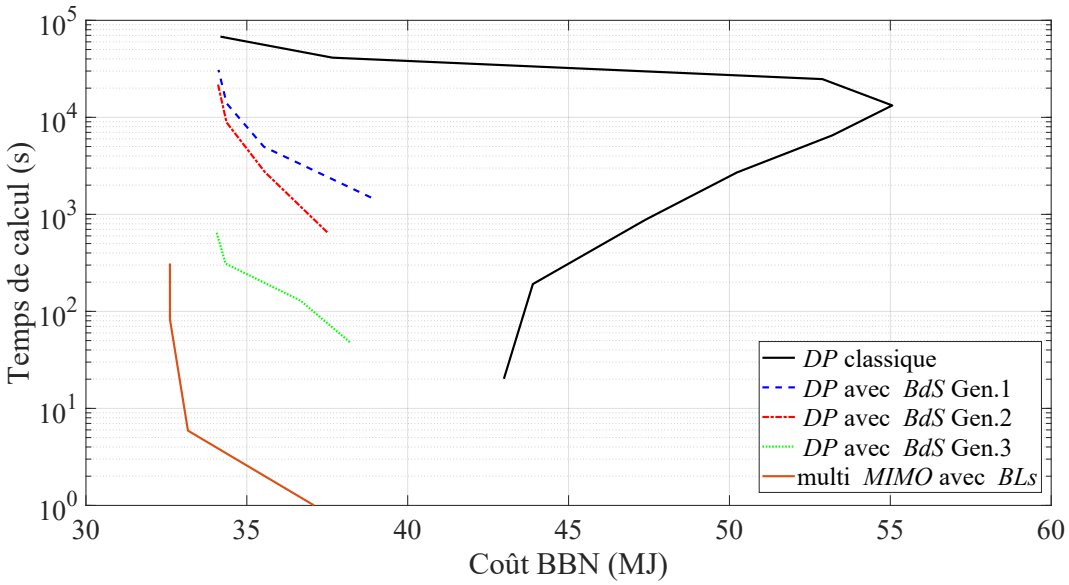

FIGURE B.4 – Temps de calcul en fonction du coût à bilan batterie nul pour le problème avec contraintes finales (3.19) des générations 1 à 3 des *boundary surfaces*, de la programmation dynamique classique et de la résolution basée sur l'analyse fréquentielle (multi *MIMO*).

Annexe C

Complément de résultat à l'analyse fréquentielle du modèle véhicule hybride avce dynamique de température

Ces analyses RGA et CD^3 illustrent une variation de chacune des trois autres variables ($\dot{m}_c$, SOC_{Bat} et T_{Mel}) entre les valeurs indiquées dans les titres. La linéarisation des autres variables est effectuée au même points que ceux données dans le chapitre 4. La commande de couple C_{Mel} est linéarisée à 30 N.m.

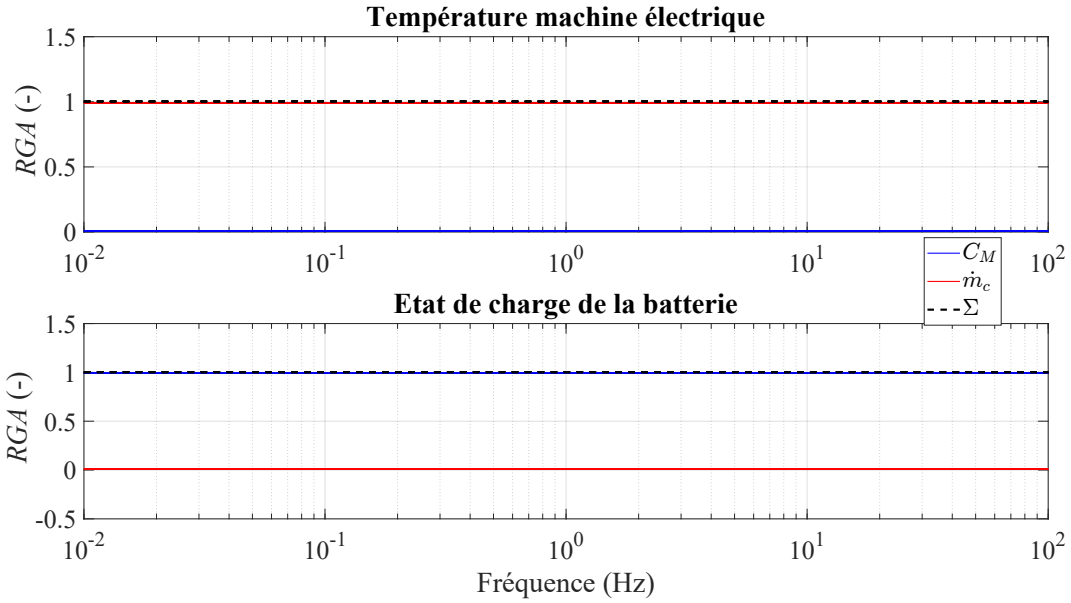

Figure C.1 – AnalyseRGA pour la variation de $\dot{m}_c$ entre 0 et 0,05.

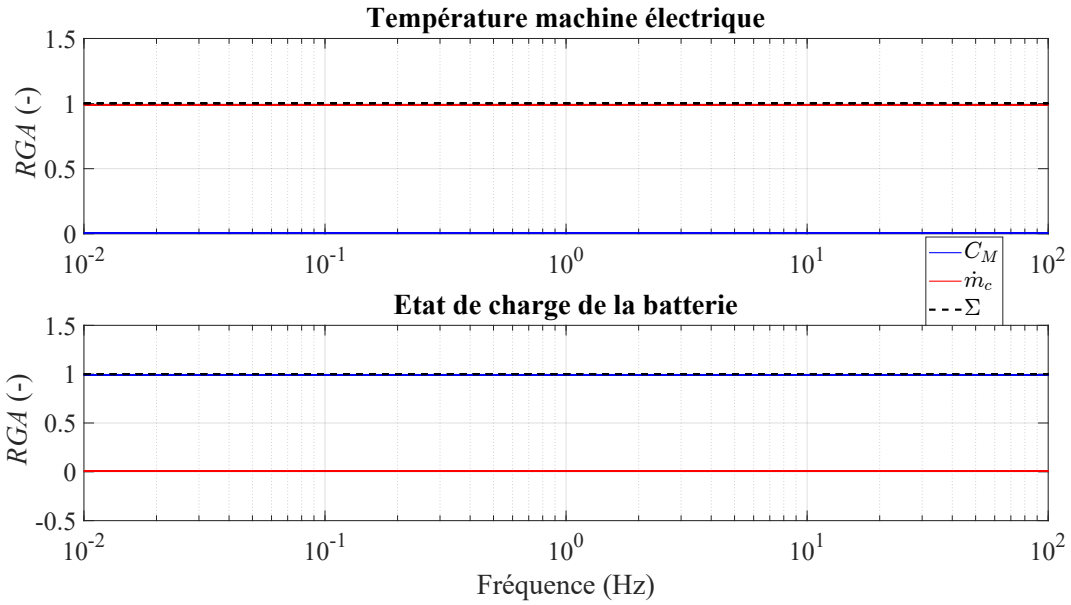

FIGURE C.2 – Analyse RGA pour la variation de SOC_{Bat} entre 0 et 1.

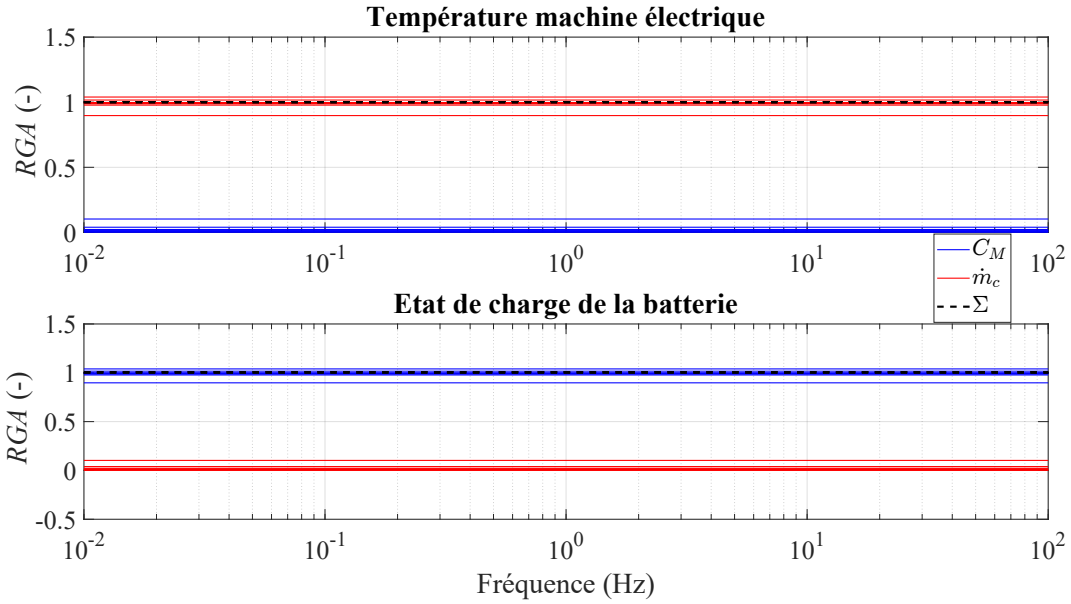

FIGURE C.3 – Analyse RGA pour la variation de T_{Mel} entre 20 et 60.

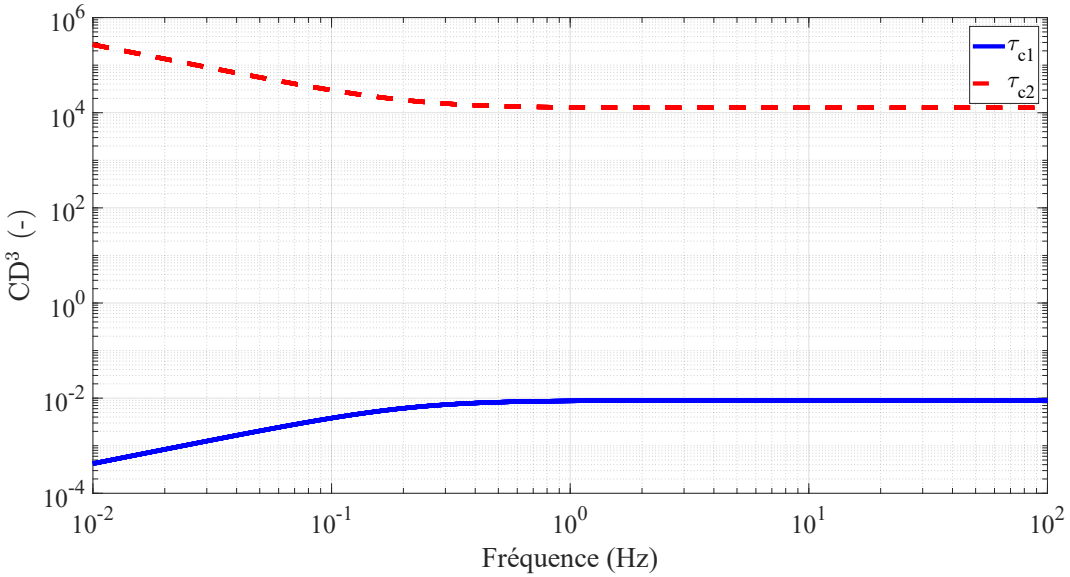

FIGURE C.4 – Analyse CD^3 pour la variation de $\dot{m}_c$ entre 0 et 0,05.

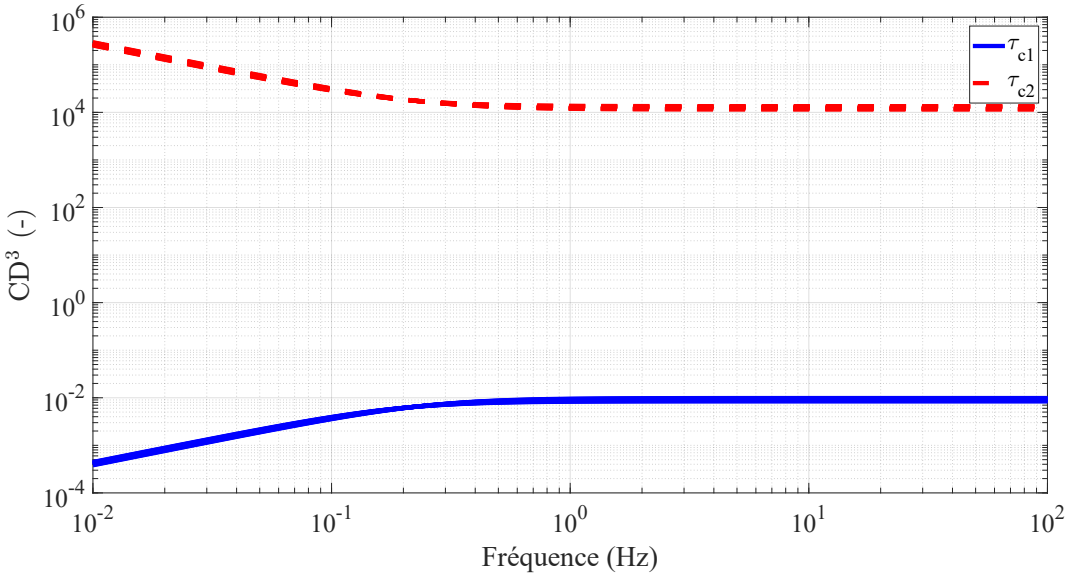

FIGURE C.5 – Analyse CD^3 pour la variation de SOC_{Bat} entre 0 et 1.

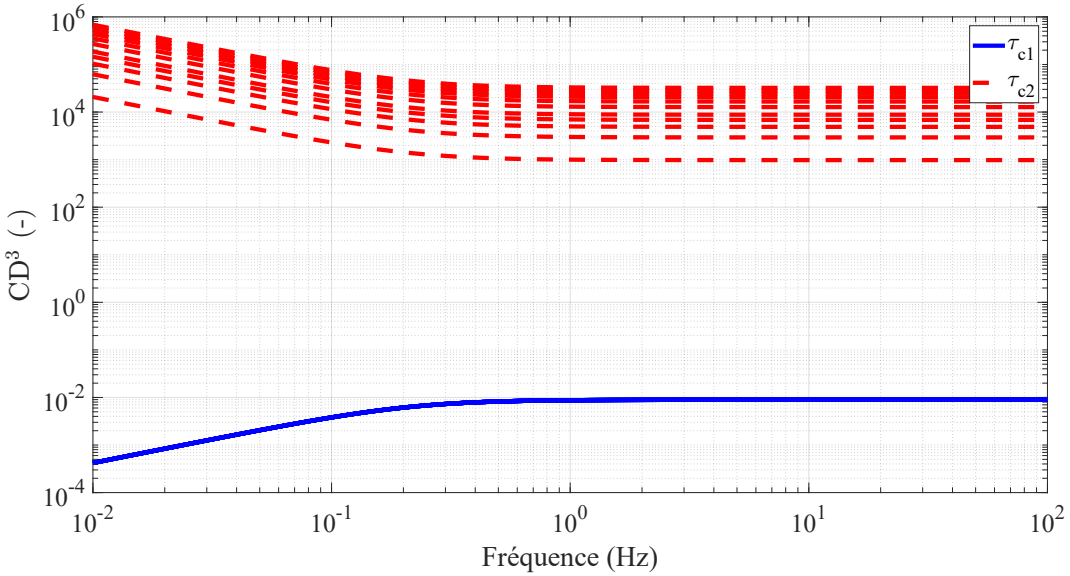

FIGURE C.6 – Analyse CD^3 pour la variation de T_{Mel} entre 20 et 60.

Annexe D

Travail sur le maillage de la solution *FIFO* pour obtenir des trajectoires similaires à la solution multi *MIMO*

Les trajectoires obtenues lors de la résolution par multi *MIMO* sont très différentes de celles obtenues lors de la résolution *FIFO*. La figure D.1 est obtenue par un maillage homogène permanant à 101 mailles par variables avec les *boundary surfaces*. Pour l'obtenir, une réduction de la plage de la température de la machine électrique a été effectué. Dans ce calcul, T_{Mel} est compris entre 35 et 40 °C pour augmenter le nombre de mailles par degré Celsius. Un coût pour la trajectoire spécifique de 33,15 MJ est obtenu pour un état de charge final de la batterie de 0,6441 et une température finale de 37°C. La qualité n'est pas améliorée pour autant, mais c'est à cause du maillage dans le cas *FIFO* qu'il n'est pas possible de retrouver les mêmes trajectoires de T_{Mel} car il n'est pas assez fin.

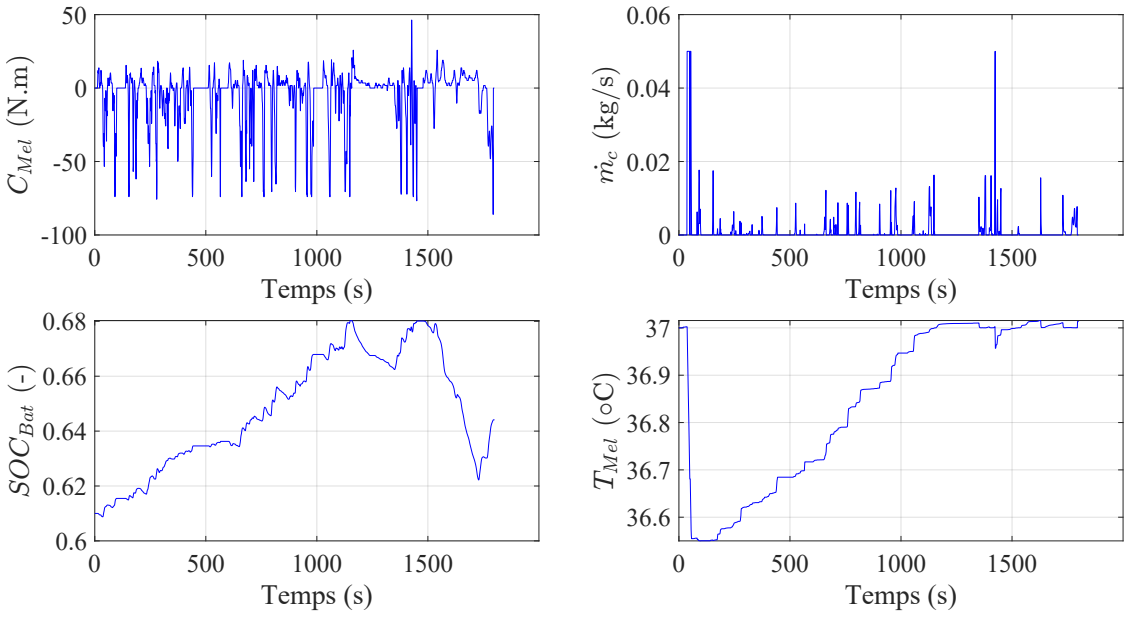

Figure D.1 – Trajectoire de la programmation dynamique avec *boundary surfaces* avec un maillages de 101 mailles par variables avec une réduction de la plage possible de la température T_{Mel}.

Bibliographie

[1] Emmanuel Trélat. *Contrôle optimal : théorie & applications*. T. 36. Vuibert Paris, 2005 (cf. p. 6, 31, 32).

[2] Yuhan Huang et al. « Eco-driving technology for sustainable road transport : A review ». In : *Renewable and Sustainable Energy Reviews* 93 (2018), p. 596-609 (cf. p. 6).

[3] Ludovic Horrein. « Gestion d'énergie décomposée d'un véhicule hybride intégrant les aspects thermiques via la représentation énergétique macroscopique ». Thèse de doct. Lille 1, 2015 (cf. p. 6, 47).

[4] Simona Onori, Lorenzo Serrao et Giorgio Rizzoni. *Hybrid electric vehicles : Energy management strategies*. T. 13. Springer, 2016 (cf. p. 6).

[5] Nicolo Robuschi et al. « Minimum-fuel energy management of a hybrid electric vehicle via iterative linear programming ». In : *IEEE Transactions on Vehicular Technology* 69.12 (2020), p. 14575-14587 (cf. p. 6).

[6] Alessandro Locatello et al. « Time-optimal control of electric race cars under thermal constraints ». In : *2021 European Control Conference (ECC)*. IEEE. 2021, p. 905-912 (cf. p. 6).

[7] Felicitas Mensing et al. « Eco-driving : An economic or ecologic driving style ? » In : *Transportation Research Part C : Emerging Technologies* 38 (2014), p. 110-121 (cf. p. 6).

[8] Djamaleddine Maamria et al. « Computation of eco-driving cycles for Hybrid Electric Vehicles : Comparative analysis ». In : *Control Engineering Practice* 71 (2018), p. 44-52 (cf. p. 6).

[9] Wissam Dib et al. « Optimal energy management for an electric vehicle in eco-driving applications ». In : *Control Engineering Practice* 29 (2014), p. 299-307 (cf. p. 6).

[10] Edwin Solano Araque et al. « Energy analysis of eco-driving maneuvers on electric vehicles ». In : *IFAC-PapersOnLine* 51.31 (2018), p. 195-200 (cf. p. 6).

[11] Saratou Souley et al. « Optimization of the travel time of an electric vehicle with consideration of the recharging terminals ». In : *IFAC-PapersOnLine* 54.2 (2021), p. 121-126 (cf. p. 6).

[12] Pierre Michel et al. « Optimizing fuel consumption and pollutant emissions of gasoline-HEV with catalytic converter ». In : *Control Engineering Practice* 61 (2017), p. 198-205 (cf. p. 6).

[13] Antoine Simon et al. « Gasoline-HEV equivalent consumption and pollutant minimization strategy ». In : *2015 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE. 2015, p. 1-6 (cf. p. 6).

[14] Bruno Jeanneret et al. « Optimal Control for Cleaner Hybrid Vehicles : A Backward Approach ». In : *Applied Sciences* 12.2 (2022), p. 578 (cf. p. 6).

[15] Giovanni De Nunzio, Ibtihel Ben Gharbia et Antonio Sciarretta. « A Time-and Energy-Optimal Routing Strategy for Electric Vehicles with Charging Constraints ». In : *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, p. 1-8 (cf. p. 6).

[16] LA Wulf Ribelles et al. « Towards a Long Trip Management Strategy for Electric Vehicles with Uncertain Traveling Conditions ». In : *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE. 2023, p. 2008-2013 (cf. p. 6, 15).

[17] Michel Minoux. *Programmation mathématique. Théorie et algorithmes*. Lavoisier, 2008 (cf. p. 7).

[18] Richard Bellman. « The theory of dynamic programming ». In : *Bulletin of the American Mathematical Society* 60.6 (1954), p. 503-515 (cf. p. 8, 14).

[19] Dimitri Bertsekas. *Dynamic programming and optimal control : Volume I*. T. 1. Athena scientific, 2012 (cf. p. 8, 17).

[20] Felicitas Mensing, Rochdi Trigui et Eric Bideaux. « Vehicle trajectory optimization for application in ECO-driving ». In : *2011 IEEE vehicle power and propulsion conference*. IEEE. 2011, p. 1-6 (cf. p. 8).

[21] Lev Semenovitch Pontriaguine et al. *Théorie mathématique des processus optimaux*. éditions Mir, 1974 (cf. p. 9).

[22] Sergey M Aseev et Arkadii V Kryazhimskii. « The Pontryagin maximum principle and optimal economic growth problems ». In : *Proceedings of the Steklov institute of mathematics* 257.1 (2007), p. 1-255 (cf. p. 9).

[23] Gustavo R Gonçalves Da Silva et Mircea Lazar. « Long hauling eco-driving : heavy-duty trucks operational modes control with integrated road slope preview ». In : *2022 European Control Conference (ECC)*. IEEE. 2022, p. 1752-1758 (cf. p. 9).

[24] Emmanuel Vinot et Rochdi Trigui. « Optimal energy management of HEVs with hybrid storage system ». In : *Energy Conversion and Management* 76 (2013), p. 437-452 (cf. p. 9).

[25] Edward D Tate et Stephen P Boyd. « Finding ultimate limits of performance for hybrid electric vehicles ». In : *SAE transactions* (2000), p. 2437-2448 (cf. p. 9).

[26] T.C.J Romijn et al. « A distributed optimization approach for complete vehicle energy management ». In : *IEEE Transactions on Control Systems Technology* 27.3 (2018), p. 964-980 (cf. p. 9).

[27] Sigurd Skogestad et Ian Postlethwaite. *Multivariable feedback control : analysis and design*. john Wiley & sons, 2005 (cf. p. 9, 114, 115).

[28] Jean-Christophe Poudou et Lionel Thomas. *Optimisation pour l'Analyse Economique et les Sciences de Gestion*. Rapp. tech. 2011 (cf. p. 9).

[29] JAJ Hall. « Towards a practical parallelisation of the simplex method ». In : *Computational Management Science* 7 (2010), p. 139-170 (cf. p. 9).

[30] Joel Andersson et Moritz Diehl. « A general-purpose software framework for dynamic optimization (een algemene softwareomgeving voor dynamische optimalisatie) ». In : (2013) (cf. p. 9).

[31] Said Belhadj, Karim Belmokhtar et Kaci Ghedamsi. « Improvement of energy management control strategy of fuel cell hybrid electric vehicles based on artificial intelligence techniques ». In : *J. Eur. Syst. Autom* 52 (2019), p. 541-550 (cf. p. 9).

[32] Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019 (cf. p. 9).

[33] Driss Laraqui et al. « A Systemic Approach for Hybrid Energy Management Strategy Based on a Deep Neural Network ». In : *2023 IEEE Vehicle Power and Propulsion Conference (VPPC)*. 2023, p. 1-6 (cf. p. 9).

[34] Oludare Isaac Abiodun et al. « State-of-the-art in artificial neural network applications : A survey ». In : *Heliyon* 4.11 (2018) (cf. p. 9).

[35] Thomas Bartz-Beielstein et al. « Evolutionary algorithms ». In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 4.3 (2014), p. 178-195 (cf. p. 10).

[36] Zheng Chen et al. « Energy management of a power-split plug-in hybrid electric vehicle based on genetic algorithm and quadratic programming ». In : *Journal of Power Sources* 248 (2014), p. 416-426 (cf. p. 10).

[37] Nicolas Barnier et Pascal Brisset. « Optimisation par algorithme génétique sous contraintes ». In : *Revue des Sciences et Technologies de l'Information-Série TSI : Technique et Science Informatiques* 18.1 (1999), pp-1 (cf. p. 10).

[38] Zbigniew Michalewicz, Cezary Z Janikow et Jacek B Krawczyk. « A modified genetic algorithm for optimal control problems ». In : *Computers & Mathematics with Applications* 23.12 (1992), p. 83-94 (cf. p. 10).

[39] Thomas Vallée et Murat Yildizoğlu. « Présentation des algorithmes génétiques et de leurs applications en économie ». In : *Revue d'économie politique* (2004), p. 711-745 (cf. p. 10).

[40] Nicoleta Stroe et al. « Predictive control framework for HEV : Energy management and free-wheeling analysis ». In : *IEEE Transactions on Intelligent Vehicles* 4.2 (2019), p. 220-231 (cf. p. 10).

[41] Gabriele Pozzato et al. « Economic MPC for online least costly energy management of hybrid electric vehicles ». In : *Control Engineering Practice* 102 (2020), p. 104534 (cf. p. 10).

[42] Hoseinali Borhan et al. « MPC-based energy management of a power-split hybrid electric vehicle ». In : *IEEE Transactions on Control Systems Technology* 20.3 (2011), p. 593-603 (cf. p. 10).

[43] JC Flores Paredes, GP Padilla Cazar et MCF Donkers. « A shrinking horizon approach to eco-driving for electric city buses : Implementation and experimental results ». In : *IFAC-PapersOnLine* 52.5 (2019), p. 556-561 (cf. p. 10).

[44] Nizar Touzi. *Optimisation dynamique*. Cours de l'école nationale de la statistique et de l'administration économique. 2002 (cf. p. 14).

[45] Tom W Archibald, KIM McKinnon et LC Thomas. « An aggregate stochastic dynamic programming model of multireservoir systems ». In : *Water Resources Research* 33.2 (1997), p. 333-340 (cf. p. 15).

[46] Lucile Marescot et al. « Complex decisions made simple : a primer on stochastic dynamic programming ». In : *Methods in Ecology and Evolution* 4.9 (2013), p. 872-884 (cf. p. 15).

[47] Sheldon M Ross. *Introduction to stochastic dynamic programming*. Academic press, 2014 (cf. p. 15).

[48] Glen Field et Yury Stepanenko. « Iterative dynamic programming : an approach to minimum energy trajectory planning for robotic manipulators ». In : *Proceedings of IEEE international conference on robotics and automation*. T. 3. IEEE. 1996, p. 2755-2760 (cf. p. 15).

[49] Rein Luus. *Iterative dynamic programming*. Chapman et Hall/CRC, 2019 (cf. p. 15).

[50] Pierre Michel. « Gestion d'énergie d'un véhicule hybride électrique-essence équipé d'un catalyseur par minimisation conjointe consommation-pollution : étude et validation expérimentale ». Thèse de doct. Orléans, 2015 (cf. p. 16, 39, 43).

[51] Dimitri P Bertsekas. « Approximate dynamic programming ». In : (2008) (cf. p. 16).

[52] Warren B Powell. « What you should know about approximate dynamic programming ». In : *Naval Research Logistics (NRL)* 56.3 (2009), p. 239-249 (cf. p. 16).

[53] Martijn RK Mes et Arturo Pérez Rivera. *Approximate dynamic programming by practical examples*. Springer, 2017 (cf. p. 16).

[54] Warren B Powell. « Perspectives of approximate dynamic programming ». In : *Annals of Operations Research* 241 (2016), p. 319-356 (cf. p. 16).

[55] Olle Sundström, Daniel Ambühl et Lino Guzzella. « On implementation of dynamic programming for optimal control problems with final state constraints ». In : *Oil & Gas Science and Technology–Revue de l'Institut Français du Pétrole* 65.1 (2010), p. 91-102 (cf. p. 16, 64).

[56] Marcelino Sanchez Pantoja. « Hybrid Vehicle Design and Control ». Thèse de doct. Université Polytechnique Hauts-de-France ; Institut national des sciences appliquées Hauts-de-France, 2020 (cf. p. 17).

[57] Olle Sundstrom et Lino Guzzella. « A generic dynamic programming Matlab function ». In : *2009 IEEE control applications,(CCA) & intelligent control,(ISIC)*. IEEE. 2009, p. 1625-1630 (cf. p. 17, 64).

[58] Lino Guzzella, Antonio Sciarretta et al. *Vehicle propulsion systems*. T. 1. Springer, 2007 (cf. p. 17, 40, 43).

[59] Maxime Debert. « Stratégies optimales multi-critères, prédictives, temps réel de gestion des flux d'énergie thermique et électrique dans un véhicule hybride ». Thèse de doct. Université d'Orléans, 2011 (cf. p. 17, 32, 39).

[60] W. Cottin et al. « On the use of Frequency Analysis tools to Minimize Numerical Complexity of Optimal Control Problem ». In : *IFAC-PapersOnLine* (2022). 18th IFAC Workshop on Control Applications of Optimization (CAO2022) (cf. p. 17, 148).

[61] Olivier Rukundo et Hanqiang Cao. « Nearest neighbor value interpolation ». In : *International Journal of Advanced Computer Science and Applications* 3.4 (2012) (cf. p. 20).

[62] Sky McKinley et Megan Levine. « Cubic spline interpolation ». In : *College of the Redwoods* 45.1 (1998), p. 1049-1060 (cf. p. 20).

[63] Frederick N Fritsch et Ralph E Carlson. « Monotone piecewise cubic interpolation ». In : *SIAM Journal on Numerical Analysis* 17.2 (1980), p. 238-246 (cf. p. 20).

[64] KH Schweizerhof et P Wriggers. « Consistent linearization for path following methods in nonlinear FE analysis ». In : *Computer Methods in Applied Mechanics and Engineering* 59.3 (1986), p. 261-279 (cf. p. 24).

[65] Pascal Heid et Thomas Wihler. « Adaptive iterative linearization Galerkin methods for nonlinear problems ». In : *Mathematics of computation* 89.326 (2020), p. 2707-2734 (cf. p. 24).

[66] Souad Hadj-Said et al. « Convex optimization for energy management of parallel hybrid electric vehicles ». In : *IFAC-PapersOnLine* 49.11 (2016), p. 271-276 (cf. p. 31).

[67] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987 (cf. p. 31).

[68] Grégory Rousseau. « Véhicule hybride et commande optimale ». Thèse de doct. École Nationale Supérieure des Mines de Paris, 2008 (cf. p. 34).

[69] *Futur Energétique 2050, RTE*. `https://assets.rte-france.com/prod/public/2021-10/Futurs-Energetiques-2050-principaux-resultats_0.pdf`. Accédé : Septembre 2023. 2021 (cf. p. 39).

[70] LA Wulf Ribelles et al. « Development of Analytical Eco-Driving Cycles for Electric Vehicles ». In : *2021 European Control Conference (ECC)*. IEEE. 2021, p. 1359-1366 (cf. p. 39).

[71] Antoine Simon. « Optimisation énergétique de chaînes de traction hybrides essence et Diesel sous contrainte de polluants : Étude et validation expérimentale ». Thèse de doct. Orléans, 2018 (cf. p. 39, 40).

[72] Jeremy Neubauer et Eric Wood. « The impact of range anxiety and home, workplace, and public charging infrastructure on simulated battery electric vehicle lifetime utility ». In : *Journal of power sources* 257 (2014), p. 12-20 (cf. p. 39).

[73] Nadine Rauh, Thomas Franke et Josef F Krems. « Understanding the impact of electric vehicle driving experience on range anxiety ». In : *Human factors* 57.1 (2015), p. 177-187 (cf. p. 39).

[74] Anatole Desreveaux et al. « Impact of the velocity profile on energy consumption of electric vehicles ». In : *IEEE transactions on Vehicular Technology* 68.12 (2019), p. 11420-11426 (cf. p. 40).

[75] Pascal Higelin, Alain Charlet et Yann Chamaillard. « Thermodynamic simulation of a hybrid pneumatic-combustion engine concept ». In : *International Journal of Thermodynamics* 5.1 (2001), p. 1-11 (cf. p. 40).

[76] Jayesh Khatri et Lucien Koopmans. *Water Injection System Application in a Mild Hybrid Powertrain*. Rapp. tech. SAE Technical Paper, 2020 (cf. p. 40).

[77] Willem Gabriel Le Roux, Tunde Bello-Ochende et Josua P Meyer. « A review on the thermodynamic optimisation and modelling of the solar thermal Brayton cycle ». In : *Renewable and sustainable energy reviews* 28 (2013), p. 677-690 (cf. p. 41).

[78] OM Ibrahim et SA Klein. « High-power multi-stage Rankine cycles ». In : (1995) (cf. p. 41).

[79] Massimiliano Esposito et al. « Efficiency at maximum power of low-dissipation Carnot engines ». In : *Physical review letters* 105.15 (2010), p. 150603 (cf. p. 41).

[80] Y Ust, B Sahin et Aykut Safa. « The effects of cycle temperature and cycle pressure ratios on the performance of an irreversible Otto cycle ». In : *Acta Physica Polonica A* 120.3 (2011), p. 412-416 (cf. p. 41).

[81] Jian Sun et Wenhua Li. « Operation optimization of an organic Rankine cycle (ORC) heat recovery power plant ». In : *Applied Thermal Engineering* 31.11-12 (2011), p. 2032-2041 (cf. p. 42).

[82] Quentin Danel et al. « Waste heat recovery applied to a tractor engine ». In : *Energy Procedia* 74 (2015), p. 331-343 (cf. p. 42).

[83] Arash Nemati et al. « A comparative thermodynamic analysis of ORC and Kalina cycles for waste heat recovery : A case study for CGAM cogeneration system ». In : *Case Studies in Thermal Engineering* 9 (2017), p. 1-13 (cf. p. 42).

[84] Xinxin Zhang, Maogang He et Ying Zhang. « A review of research on the Kalina cycle ». In : *Renewable and sustainable energy reviews* 16.7 (2012), p. 5309-5318 (cf. p. 42).

[85] Wenlu Zhou et al. « Review on the battery model and SOC estimation method ». In : *Processes* 9.9 (2021), p. 1685 (cf. p. 43, 45).

[86] Charbel Mansour et al. « Waste heat recovery from engine coolant on mild hybrid vehicle using organic Rankine cycle ». In : *Proceedings of the Institution of Mechanical Engineers, Part D : Journal of Automobile Engineering* 233.10 (2019), p. 2502-2517 (cf. p. 45).

[87] Bin Chen et al. « A combination of electric supercharger and Miller Cycle in a gasoline engine to improve thermal efficiency without performance degradation ». In : *Case Studies in Thermal Engineering* 14 (2019), p. 100429 (cf. p. 46).

[88] Chih Wu, Paul V Puzinauskas et Jung S Tsai. « Performance analysis and optimization of a supercharged Miller cycle Otto engine ». In : *Applied Thermal Engineering* 23.5 (2003), p. 511-521 (cf. p. 46).

[89] Daniel Champier et al. « Study of a TE (thermoelectric) generator incorporated in a multifunction wood stove ». In : *Energy* 36.3 (2011), p. 1518-1526 (cf. p. 47).

[90] Ken-ichi Uchida et al. « Thermoelectric generation based on spin Seebeck effects ». In : *Proceedings of the IEEE* 104.10 (2016), p. 1946-1973 (cf. p. 47).

[91] Paul Boldrin et Nigel P Brandon. « Progress and outlook for solid oxide fuel cells for transportation applications ». In : *Nature Catalysis* 2.7 (2019), p. 571-577 (cf. p. 47).

[92] Charbel Mansour et al. « Assessing additional fuel consumption from cabin thermal comfort and auxiliary needs on the worldwide harmonized light vehicles test cycle ». In : *Transportation research part D : Transport and Environment* 62 (2018), p. 139-151 (cf. p. 48).

[93] William Morrow Kays et Alexander Louis London. *Compact heat exchangers*. McGraw-Hill, New York, NY, 1984 (cf. p. 48, 53).

[94] Huijuan Chen, D Yogi Goswami et Elias K Stefanakos. « A review of thermodynamic cycles and working fluids for the conversion of low-grade heat ». In : *Renewable and sustainable energy reviews* 14.9 (2010), p. 3059-3067 (cf. p. 49).

[95] Mansoor Ahmad. *Experimental assessment of droplet impact erosion of low-pressure steam turbine blades*. Shaker, 2009 (cf. p. 49).

[96] Eric W Lemmon, Marcia L Huber et Mark O McLinden. « NIST reference fluid thermodynamic and transport properties–REFPROP ». In : *NIST standard reference database* 23.2002 (2002), p. v7 (cf. p. 51).

[97] PK Nag. « Engineering thermodynamics ». In : *New Delhi* (2008) (cf. p. 51).

[98] Charbel Mansour et al. « Assessing additional fuel consumption from cabin thermal comfort and auxiliary needs on the worldwide harmonized light vehicles test cycle ». In : *Transportation Research Part D : Transport and Environment* 62 (2018), p. 139-151 (cf. p. 60).

[99] Aaron Isenstadt, John German et Mihai Dorobantu. « Naturally aspirated gasoline engines and cylinder deactivation ». In : *International Council on Clean Transportation Working Paper* 12 (2016), p. 2016 (cf. p. 63).

[100] Nic Lutsey et al. « Efficiency technology and cost assessment for US 2025–2030 light-duty vehicles ». In : *International Council on Clean Transportation (ICCT), Washington, USA, White Paper, 22nd March* (2017) (cf. p. 63).

[101] Johannes Liebl et al. « The thermoelectric generator from BMW is making use of waste heat ». In : *MTZ worldwide* 70.4 (2009), p. 4-11 (cf. p. 64).

[102] Jos van Schijndel et al. « Dynamic programming for integrated emission management in diesel engines ». In : *IFAC Proceedings Volumes* 47.3 (2014), p. 11860-11865 (cf. p. 73).

[103] Herbert Edelsbrunner. « Alpha shapes-a survey ». In : *Tessellations in the Sciences : Virtues, Techniques and Applications of Geometric Tilings*. 2011 (cf. p. 83).

[104] Boris Delaunay et al. « Sur la sphere vide ». In : *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), p. 1-2 (cf. p. 84).

[105] Isaac Amidror. « Scattered data interpolation methods for electronic imaging systems : a survey ». In : *Journal of electronic imaging* 11.2 (2002), p. 157-176 (cf. p. 88).

[106] Edgar Bristol. « On a new measure of interaction for multivariable process control ». In : *IEEE transactions on automatic control* 11.1 (1966), p. 133-134 (cf. p. 114).

[107] Jean-Pierre Corriou. *Commande des procédés*. Tec & Doc Lavoisier, 1996 (cf. p. 114).

[108] Sigurd Skogestad et Manfred Morari. « Control configuration selection for distillation columns ». In : *AIChE Journal* 33.10 (1987), p. 1620-1635 (cf. p. 114).

[109] Morten Hovd et Sigurd Skogestad. « Simple frequency-dependent tools for control system analysis, structure selection and design ». In : *Automatica* 28.5 (1992), p. 989-996 (cf. p. 114).

[110] Gene Golub et William Kahan. « Calculating the singular values and pseudo-inverse of a matrix ». In : *Journal of the Society for Industrial and Applied Mathematics, Series B : Numerical Analysis* 2.2 (1965), p. 205-224 (cf. p. 114).

[111] Julien Marques. « Plate-forme d'aide à l'éco-conception de systèmes multiphysiques : démarche énergétique pour la validation et la réduction de modèles ». Thèse de doct. Université d'Orléans, 2010 (cf. p. 115, 116).

[112] Paul Heckbert. « Fourier transforms and the fast Fourier transform (FFT) algorithm ». In : *Computer Graphics* 2.1995 (1995), p. 15-463 (cf. p. 115).

[113] Dominique Nelson-Gruel. « Extension de la commande CRONE multivariable aux systèmes non carrés : application à un système d'air de moteurs essence et diesel ». Thèse de doct. Bordeaux 1, 2009 (cf. p. 118).

[114] Matthew Shirk et Jeffrey Wishart. *Effects of electric vehicle fast charging on battery life and vehicle performance*. Rapp. tech. Idaho National Lab.(INL), Idaho Falls, ID (United States), 2015 (cf. p. 133).

[115] Marco Bernagozzi et al. « Heat pipes in battery thermal management systems for electric vehicles : A critical review ». In : *Applied Thermal Engineering* 219 (2023), p. 119495 (cf. p. 134).

[116] Olivier BACHELIER. « Cours d'Automatique ». In : () (cf. p. 138).

[117] Willy Cottin et al. *Fuel Consumption Potential Gains of Rankine Thermal Power Recovery for Series Hybrid Electric Vehicles*. Rapp. tech. SAE Technical Paper, 2023 (cf. p. 147).

[118] Willy Cottin et al. « From boundary lines to boundary surfaces for dynamic programming with final state constraints ». In : *IFAC-PapersOnLine* (2023). 22nd IFAC World Congress (IFAC2023) (cf. p. 148).

Willy COTTIN

Contributions à la commande optimale par Programmation Dynamique

Application à la gestion de l'énergie de l'automobile électrifiée

Résumé :

Les contextes environnementaux et sociétaux exigent de l'industrie automobile une réduction significative des émissions de gaz à effet de serre et de polluants. L'option de l'électrification partielle ou totale des chaînes de tractions des véhicules est choisie pour répondre aux attentes. L'électrification partielle offre la possibilité de répartir la demande de puissance entre la partie thermique et électrique de la chaîne de traction des véhicules hybrides pour réduire la consommation de carburant de ces derniers. La résolution d'un Problème de Commande Optimale permet de déterminer la loi de commande qui minimise le critère souhaité. Il est ainsi possible de dimensionner et de contrôler des nouvelles chaînes de tractions complexes et d'évaluer avec équité les nouvelles technologies s'intégrant à la chaîne de traction du véhicule. Différentes méthodes existent pour résoudre les problèmes de commande optimale. Celle étudiée et utilisée dans ce manuscrit est la Programmation Dynamique dont l'inconvénient majeur est le temps de calcul. Les deux contributions majeures de la thèse visent à réduire le temps de calcul nécessaire à cette méthode pour fournir des résultats de qualité. La première, nommée *boundary surfaces*, généralise pour deux états une optimisation de l'algorithme disponible jusqu'ici que pour un seul état. L'amélioration de la qualité des résultats est significative, notamment pour les maillages peu raffinés. La seconde contribution est une méthodologie pour réduire la complexité du problème en le divisant en une somme de sous-problèmes basé sur des outils d'analyse fréquentielle. Ces deux contributions sont illustrées séparément dans un premier temps sur un modèle véhicule hybride électrique parallèle et sont appliquées ensemble sur le problème de recharge rapide d'une batterie automobile. En plus de ces apports méthodologiques, ce manuscrit présente l'étude de l'intégration d'une structure de machines récupératrices valorisant les pertes du moteur thermique d'un véhicule hybride électrique série.

Mots clés : Problème de Commande Optimale, Programmation Dynamique, Loi de Gestion de l'Energie, Véhicule électrifié, Analyse fréquentielle, *Boundary surfaces*, Machines récupératrices de puissance, Charge rapide

Contributions to optimal control by Dynamic Programming

Application to energy management of electrified vehicles

Summary:

Environmental and societal contexts require the automotive industry to significantly reduce greenhouse gas and pollutant emissions. The option of partial or total electrification of the vehicles' powertrains is chosen to meet expectations. Partial electrification offers the possibility of distributing the power demand between the thermal and electric parts of the powertrain of hybrid vehicles to reduce the vehicles' fuel consumption. Solving an Optimal Control Problem allows to determine the control law that minimizes the desired criterion. This makes it possible to design and test complex new powertrains and to fairly evaluate new technologies that are integrated into the vehicle's powertrain. Different methods exist to solve optimal control problems. The one studied and used in this manuscript is Dynamic Programming, the major disadvantage of which is the computation time. The two major contributions of the thesis aim to reduce the computational time required for this method to provide quality results. The first, called boundary surfaces, generalizes to two states an optimization of the algorithm that was previously available only for one state. The improvement in the quality of the results is significant, especially for poorly refined meshes. The second contribution is a methodology to reduce the complexity of the problem by dividing it into a sum of sub-problems based on frequency analysis tools. These two contributions are illustrated separately first on a parallel hybrid-electric vehicle model and are applied together on the problem of fast charging of a car battery. In addition to these methodological contributions, this manuscript presents the study of the integration of a Waste Heat Recovery Systems valuing the losses of the engine of a series hybrid electric vehicle.

Keywords: Optimal Control Problem, Dynamic Programming, Energy Management Strategy, Electrified vehicles, Frequency analysis, Boundary surfaces, Waste Heat Recovery Systems, Fast charge

Laboratoire PRISME, 8 rue Léonard de Vinci, 45072 Orléans Cedex 2