# Robustness and visualization of blend's production

Jorge Antonio Aguilera Cabanas

## ▶ To cite this version:

Jorge Antonio Aguilera Cabanas. Robustness and visualization of blend's production. General Mathematics [math.GM]. Université de Grenoble, 2011. English. NNT : 2011GRENM052 . tel-00647297

**THÈSE**

Pour obtenir le grade de

**DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE**

Spécialité : **Mathématiques Appliquées**

Arrêté ministérial : 7 août 2006

Présentée par

**Jorge Aguilera**

Thèse dirigée par **Michel Mollard**
et codirigée par **Stefan Janaqi**

préparée au sein **de l'Institut Fourier, du Laboratoire LGI2P**
et de **l'École Doctorale MSTII**

# Robustesse et visualisation de production de mélanges

Thèse soutenue publiquement le **28 octobre 2011**,
devant le jury composé de :

## Remerciements

Monsieur A. Ridha Mahjoub m'a fait l'honneur d'être rapporteur de ma thèse. Pour cela et pour ses commentaires sur mon mémoire, je lui exprime ma gratitude. Je remercie également Monsieur Alexandre Dolgui d'avoir accepté d'être rapporteur de ma thèse.

Je voudrais exprimer ici ma gratitude envers Michel Mollard pour sa confiance et pour m'avoir donné l'ocassion de mener ce travail. Je tiens à remercier aussi Stefan Janaqi qui m'a recruté pour cette thèse et qui a partagé avec moi ses idées et sa connaissance.

J'adresse mes vifs remerciements à Mesdames Nadia Maïzi et Mériam Chèbre pour avoir bien voulu juger ce travail.

A mes employeurs de la Banque Centrale du Mexique, en particulier à Manuel Galan et Jorge de la Vega vont mes remerciements les plus sincères pour leur soutien.

Finalement, un grand Merci à DDelphine Moussard pour la révision du texte en français.

Ce travail est dédié à Aline Castro, ma collègue et ma femme qui est à l'origine de cette aventure.

# Resumé

Le procédé de fabrication de mélanges consiste à déterminer les proportions optimales à mélanger d'un ensemble de composants pour que le produit obtenu vérifie un ensemble de spécifications. Le problème des mélanges (PM) apparaît typiquement dans l'industrie pétrochimique où plusieurs types de composants bruts sont mélangés pour produire de l'essence et autres huiles à propriétés spécifiées.

Le système de mélange est généralement constitué de trois sous-systèmes fonctionnels : le sous-système de planification, un optimiseur en ligne et le sous-système de contrôle. Le sous-système de planification prévoit la production générale de la raffinerie et d'une façon plus spécifique, il génère la recette objective initiale pour toute opération de mélange programmée. Plusieurs sources d'incertitude (voir ci-dessous) apparaissent tout au long du procédé. L'optimiseur en ligne est alors nécessaire pour mettre à jour la recette cible qui peut devenir sous-optimale, voire irréalisable, à cause de ces perturbations dans le procédé. Le feedback du système est basé sur des mesures des propriétés du mélange actuel et celles des composants qui sont recueillies par des analyseurs en ligne. Enfin, le sous-système de contrôle est chargé de régler le taux de débit des composants afin d'atteindre la recette cible courante qui a été produite par l'optimiseur en ligne. Dans la Figure 1, nous montrons l'intégration de ces systèmes et le diagramme qui décrit un procédé de mélange typique.

Dans ce travail, nous nous concentrons sur l'analyse du système *d'Optimisation en Temps Réel* (RTO) qui est formé par l'optimiseur en ligne et le sous-système de contrôle. Une hypothèse principale du travail est que les propriétés d'un mélange sont déterminées comme une combinaison linéaire des propriétés des composants, c'est-à-dire que nous supposons que les lois de mélange sont linéaires. En général elles ne le sont pas, mais elles peuvent être approchées par des fonctions linéaires d'autres propriétés de mélange ; voir, par exemple, [34] et [23]. Nous tenons à mentionner que cette hypothèse est loin d'être une simplification excessive du problème. En fait, l'optimiseur en ligne du logiciel Anamel qui est utilisé par certaines unités de mélange au sein du groupe TOTAL (voir [13]) utilise ce modèle de mélange linéaire. En outre, nous nous concentrons sur la production d'un seul mélange à partir d'un ensemble de composants et nous considérons le cas des composants stockés dans les *"running tanks"*, c'est-à-dire le cas où les bacs des composants sont reliés directement aux procédés en amont, où les composants sont en train d'être fabriqués. Ceux-ci s'écoulent directement dans ces bacs en même temps qu'ils sont mélangées.

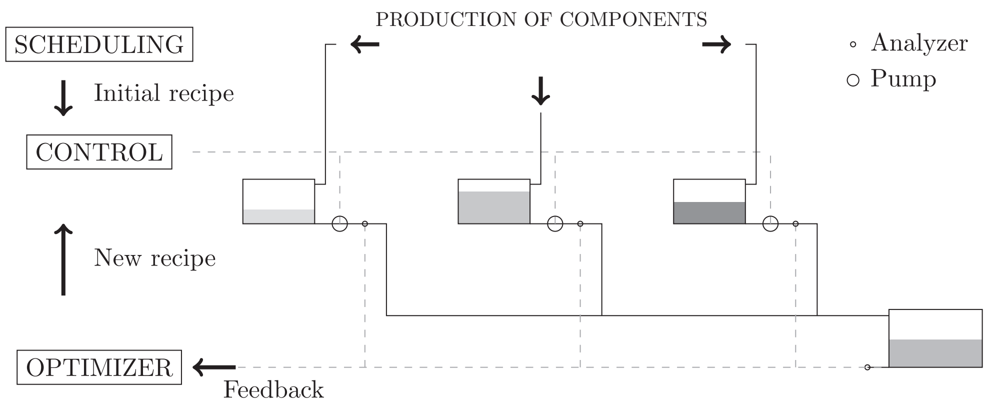


FIGURE 1: Système de mélange.

Une caractéristique importante dans le procédé de mélange est l'existence d'exigences strictes ("bornes dures") sur certaines propriétés du mélange qui sont contrôlées par des normes environnementales, réglementaires et technologiques. Si le mélange produit ne satisfait pas les exigences imposées par les "bornes soft", il y a un coût que nous encourons pour la sur-qualité (*giveaway*) du mélange. Néanmoins, si le mélange ne satisfait pas les bornes dures, alors le mélange n'est pas vendable et on doit le remélanger. Le remélange réduit la capacité de la raffinerie et il a plusieurs coûts associés, (par exemple, la location du bac, les coûts logistiques, les coûts d'inventaire,...). Donc, il faut considérer les bornes dures comme des contraintes qui doivent être satisfaites.

D'autre part, le mélange de pétrole est un procédé complexe où plusieurs facteurs inconnus et incertains ont une incidence sur les propriétés du mélange. Outre l'erreur produite par la linéarisation des lois de mélange, il existe d'autres sources d'incertitude : la précision des instruments de mesure sur les composants et les propriétés ; une connaissance incertaine des propriétés des composants en raison des variations des procédés en amont et des conditions de mélange non contrôlables tels que la température de l'air, l'humidité, etc. Ces sources d'incertitude sont typiques de tout problème RTO (cf. [47]).

Les bornes dures sur les propriétés du mélange en combinaison avec ces sources d'incertitude sont les caractéristiques idéales pour appliquer les techniques d'*Optimisation Robuste* (RO) (voir, [9]). L'optimisation robuste est une méthodologie pour faire de l'optimisation avec des données incertaines. Une caractéristique principale dans les techniques de RO est de considérer un modèle d'incertitude déterministe et basé sur des ensembles. Nous ne faisons aucune hypothèse probabiliste sur l'incertitude et la solution obtenue est optimale pour toute réalisation de l'incertitude dans un ensemble donné. Les techniques de RO semblent être ad hoc pour résoudre un problème RTO avec incertitude, où la faisabilité du problème est la principale préoccupation.

Plusieurs travaux portent sur la faisabilité et l'optimisation du PM (voir, par ex., [40, 24, 46, 41]). La faisabilité de plusieurs mélanges à produire à partir du même ensemble de composants est considérée dans [29]. Plus tard, dans [17], l'auteur est préoccupé aussi par l'incertitude des propriétés. Il développe une analyse de sensibilité de la recette lorsque les erreurs de mesure dans les propriétés des composants varient dans un ellipsoïde. Dans [38], une méthode d'optimisation en ligne est utilisée pour mettre à jour la recette et réduire la divergence, entre le modèle et le réel, causée par l'incertitude. Dans [26], une approche géométrique est présentée pour les problèmes de conception de produits et de mélange, mais en considérant seulement l'incertitude due à des imprécisions de mesure. Pour traiter le problème de l'incertitude des propriétés des composants, [43] et [39] ont proposé respectivement des modèles RTO linéaire et non-linéaire basés sur les pronostics des propriétés des composants. D'autre part, [44] a présenté un modèle de contraintes en probabilités qui est résolu par une méthode heuristique. Plus récemment, [13] introduit un algorithme de contrôle qui gère ce type d'incertitudes au moyen d'un estimateur de propriétés des composants. Un modèle non-linéaire basé sur la programmation stochastique et qui couvre différents types d'incertitude est présenté dans [48].

L'objectif principal de ce travail est d'étudier le procédé de mélange et la production de mélanges robustes d'un point de vue polyédral. Plus précisément, nous nous intéressons au développement des outils informatiques pour optimiser en ligne la production de mélanges robustes qui résistent à l'incertitude qui survient lors du procédé de mélange.

Dans le chapitre 2, nous présentons les Polytopes de Mélange inhérents au problème de mélange et l'équation de base qui sera utilisé pour introduire les données mises à jour dans la méthode RTO développée au chapitre 3. Nous modélisons différentes sources d'incertitude liées au procédé de mélange et nous proposons une méthode robuste RTO basé sur les techniques de RO. La méthode présentée est destinée à éviter le remélange et on mesure sa performance en termes de sur-qualité du mélange produit et de coût de la recette. En outre, nous analysons la faisabilité d'un mélange à fabriquer à partir d'un ensemble de composants lorsque le fond du bac d'un mélange précédent est utilisé dans la composition du nouveau mélange. Des informations essentielles pour le système de contrôle sont alors produites. Par exemple, savoir le volume minimal à couler avant d'obtenir un mélange faisable robuste réduit les interventions inutiles du système de contrôle. Celles-ci tentent de maintenir le mélange dans les spécifications en modifiant la recette produite par l'optimiseur en ligne, mais souvent elles sont fondées sur une vision limitée du problème.

Enfin, au chapitre 4, nous analysons le cas où la méthode RTO ne parvient pas à produire un mélange réalisable. Nous mesurons et produisons des visualisations de l'infaisabilité du PM et proposons une méthode pour guider le procédé de mélange vers le “meilleur“ mélange robuste.

# Polytopes de Mélange

Désignons par $n$ et $m$ le nombre de composants et de propriétés respectivement (pour l'application industrielle, $m$ et $n$ ne dépassent pas 20).

Soit $S_1$ le simplexe défini comme

$$S_1 = \{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}, \quad \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\}.$$

Nous allons représenter une *recette de mélange* (*recette*) par un vecteur $\mathbf{u} \in S_1$ tel que $u_j$ est le pourcentage du composant $j$ présent dans le mélange alors que les *propriétés du mélange* (*propriétés*) seront désignées par un vecteur $\mathbf{y} \in \mathbb{R}^m$. De cette façon, l'ensemble des *composants* sera décrit par la matrice $B$ de taille $(m, n)$, où $B_{i,j}$ est la $i^{\text{ème}}$ caractéristique physique ou chimique du composant $j$.

Lors de la fabrication d'un mélange, il est habituel d'utiliser le fond du bac d'un mélange précédent dans la production du nouveau mélange. Le procédé commence alors avec un volume $V_0$ d'un mélange précédent avec des propriétés $\mathbf{b_0} \in \mathbb{R}^m$ et continue en ajoutant progressivement un volume $v$ du nouveau mélange pour obtenir un volume du produit final $V_{total} = V_0 + v$.

Comme on l'a annoncé, dans ce travail nous supposons que les propriétés du mélange sont une combinaison linéaire (ou plus précisément, une combinaison convexe) des propriétés des composants. En vertu de ces hypothèses, pour tout $v \geq 0$ les propriétés de la recette $\mathbf{u}$ sont déterminées par l'*équation de base* définie par l'application linéaire $y : \mathbb{R}^n \to \mathbb{R}^m$ tel que :

$$\mathbf{y}(\mathbf{u}) = \alpha_v \ \mathbf{b_0} + \beta_v \ B\mathbf{u} \tag{0.1}$$

avec $\alpha_v = \frac{V_0}{V_0+v}$ et $\beta_v = \frac{v}{V_0+v}$.

Pour tout horizon de planification fixe et déterminé par le volume $(v)$, on peut obtenir la recette de coût minimum comme la solution du *Problème de* *mélange* (PM) :

$$
\begin{aligned}
& \mathit{min}\ \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} \\
& \text{s.t.} \\
& \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}} && (0.2) \\
& \underline{\mathbf{y}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}} && (0.3) \\
& \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1,\ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}. && (0.4)
\end{aligned}
$$

Ici le vecteur $\mathbf{c} \in \mathbb{R}^n$ représente les coûts des composants. Une recette $\mathbf{u}$ doit vérifier des bornes de disponibilité et hydrauliques (voir contraintes sur les composants (0.2)). De la même manière, les contraintes sur les propriétés (0.3) sont déterminées par des bornes inférieures ($\underline{\mathbf{y}}$) et supérieures ($\overline{\mathbf{y}}$). Ces bornes des propriétés sont étiquetées comme des bornes *hard* et *soft*. Les bornes hard, sont liées aux spécifications commerciales et environnementales devant être satisfaites alors que les bornes soft peuvent être violées, mais en produisant un mélange avec de la sur-qualité. Enfin, (0.4) sont les contraintes de pourcentage.

Géométriquement, l'intersection des contraintes sur les composants (0.2) avec $S_1$ détermine le polytope

$$U = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}\ ,\ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1,\ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}\}$$

qui représente l'ensemble de toutes les recettes qui peuvent être produites. D'autre part, les contraintes sur les propriétés définissent le polyèdre (car il peut être non borné) :

$$C_U = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{y}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}}\}.$$

Tous les deux sont définis dans l'*espace des composants*.

Dans l'*espace des propriétés*, on a les images de $U$ et $C_U$ par l'équation de base :

$$P = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid\ \mathbf{u} \in U\}$$

et

$$C_Y = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid \mathbf{u} \in C_U\}.$$

et le *polytope cible*

$$C = \{\mathbf{y} \in \mathbb{R}^m \mid \underline{\mathbf{y}} \leq \mathbf{y} \leq \overline{\mathbf{y}}\}.$$

Dans ce dernier polytope on distingue le *mélange cible* $\mathbf{y_t}$ qui est un mélange fourni par le département de planification ayant la sur-qualité minimale et d'autres caractéristiques souhaitables. Ce mélange sera considéré comme le mélange idéal à produire quand on ne parvient pas à générer une recette réalisable pour le PM.

À ce point nous avons identifié les polytopes sous-jacents au PM et nous observons que, pour un volume fixe $v$, une solution réalisable du PM est une recette $\mathbf{u} \in C_U \cap U$ avec des propriétés $\mathbf{y}(\mathbf{u}) \in P \cap C_Y$. De façon équivalente,

$$\text{PM est infaisable} \iff P \cap C_Y = \emptyset \iff U \cap C_U = \emptyset. \tag{0.5}$$

## Une Méthode RTO Robuste

Le système de mélange est généralement constitué de trois sous-systèmes fonctionnels : le sous-système de planification, l'optimiseur en ligne et le sous-système de contrôle. L'objectif du sous-système de planification est de planifier les opérations de raffinage pour différents horizons temporels. Dans le long terme, les objectifs de production sont déterminés en utilisant des prévisions de la demande des produits et des prix du pétrole. Dans la planification à moyen terme, ces objectifs sont révisés et ajustés par rapport aux estimations de disponibilité des composants, des réservoirs,... Enfin, dans le court terme la planification se concentre sur la production de recettes cibles pour toute opération de mélange prévue. Le volume de production, les composants à mélanger et tous les autres paramètres nécessaires au calcul de la recette cible sont déterminés à ce niveau et selon le planning général dans les raffineries. Dans ces conditions, on devrait s'attendre à avoir toujours une solution réalisable pour l'instance du PM. Toutefois, le calcul de la recette cible est la plupart du temps basé sur des prévisions et, occasionnellement, sur des mesures des propriétés des composants. Dans les deux cas, la précision du modèle est affectée : dans le premier cas, le modèle hérite des imprécisions des prédictions et dans le second cas les mesures des composants peuvent également être inexactes et retardées [13]. En fait, comme les composants sont en production en même temps qu'ils sont mélangés, l'imprécision dans le modèle induit par la connaissance défectueuse des propriétés des composants reste tout au long du procédé. Pour cela et d'autres types d'incertitudes, la recette cible peut devenir sous-optimale, voire irréalisable.

Afin de réduire la disparité entre le modèle et le système réel, l'optimiseur en ligne met à jour la recette cible, dans un cycle en boucle fermée et au moyen de mesures en ligne des propriétés des composants et du mélange actuel. La nouvelle recette servira au contrôleur à guider le procédé de mélange.

Avant de présenter le modèle RTO, on discute de la production d'un mélange à partir d'un mélange fabriqué préalablement. Si le mélange est produit à partir de zéro ou si le volume à couler est trop grand par rapport au volume du fond du bac, c'est-à-dire, si $V_0 = 0$ ou $v \to \infty$, alors l'équation de base se réduit à $\mathbf{y}(\mathbf{u}) = B\mathbf{u}$ et le polytope $P(v)$ ne dépend pas de $V_0$. D'autre part, si $V_0 \neq 0$ alors $P(v)$ est un polytope homothétique à $P(\infty)$ avec $\mathbf{b_0}$ comme centre d'homothétie. Dans l'espace des propriétés, le polytope $C$ ne dépend pas de $v$. Il existe alors un intervalle (éventuellement vide) de volumes $[V_{min}, V_{Max}]$ tel que $P(v) \cap C \neq \emptyset, \quad \forall v \in [V_{min}, V_{Max}]$. Le PM est réalisable seulement pour ces volumes (voir la Figure 2).

Le fait d'avoir les valeurs $V_{min}$ et $V_{max}$ permet de sélectionner un hori-

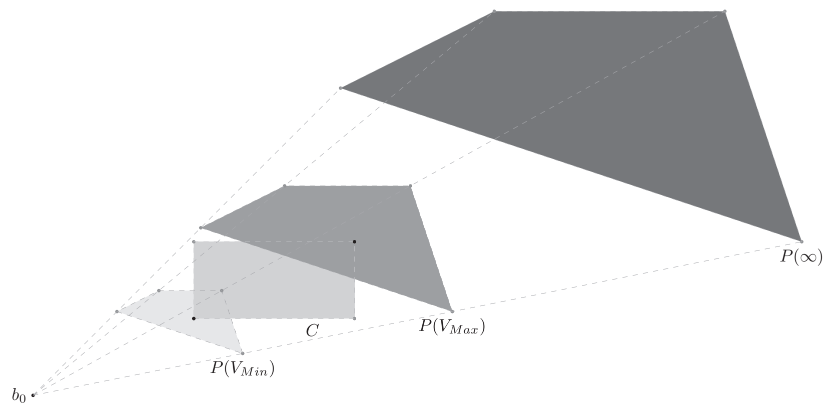


FIGURE 2: Evolution du polytope $P(v)$ avec le volume coulé.

zon de planification approprié à chaque boucle du RTO. Habituellement, le volume à produire est fixé par le département de planification, mais l'imprécision dans le modèle et l'incertitude tout au long du procédé peuvent requérir une révision constante de ces volumes. Choisir un horizon de planification avec un volume $v < V_{min}$ produit un PM irréalisable et rend nécessaire l'utilisation d'une recette alternative avec une éventuelle détérioration de la performance globale de la méthode RTO. De l'autre côté, $V_{max}$ est le volume maximum à mélanger lorsqu'on cherche un mélange qui répond aux spécifications. Après que $V_{max}$ a été coulé, $P(v)$ s'éloigne de $C$ et les propriétés du mélange se détériorent progressivement.

De la même manière, nous pouvons analyser le rôle de l'utilisation de $V_0$ dans la faisabilité du PM. En faisant $u_0 = \frac{V_0}{v}$, on peut réécrire l'équation de base d'une manière utile :

$$\mathbf{y}(\mathbf{u}) = \frac{u_0\mathbf{b_0}}{1+u_0} + \frac{B\mathbf{u}}{1+u_0}. \tag{0.6}$$

En utilisant cette équation, on peut déterminer l'intervalle $[u_{0Min}, u_{0Max}]$ pour lequel $P \cap C \neq \emptyset$ (d'une manière équivalente, $U \cap C_U \neq \emptyset$). Ces valeurs nous donnent toutes les combinaisons de volumes $(V_0, v)$ produisant un mélange réalisable.

Désignons par

$$C_{UH} = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{y}}_{\mathbf{H}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}}_{\mathbf{H}}\}$$

le polyèdre défini par les contraintes dures.

À toute boucle RTO on procède comme suit :

**Algorithme 1 Méthode RTO**

I On incorpore la nouvelle information disponible ($V_0$, $\mathbf{b_0}$, $B$ et $v$) dans l'équation de base. Si $V_0 = 0$, on résout les problèmes (0.8), (0.10) et (0.11) dans cet ordre jusqu'à obtenir une solution réalisable. Sinon, on passe à l'étape (II).

II On calcule $V_{Min}$ ($V_{Max}$) en résolvant le problème de programmation linéaire :

$$\begin{aligned} &max \quad (min) \quad u_0 \\ &\text{s.t.} \\ &\mathbf{u} \in U \cap C_U \\ &u_0 \geq 0. \end{aligned} \tag{0.7}$$

Si le problème (0.7) est irréalisable ou si $v \notin [V_{Min}, V_{Max}]$ alors on passe à l'étape (IV).

III On calcule la recette optimale :

$$\begin{aligned} &min \quad \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} \\ &\text{s.t.} \\ &\mathbf{u} \in U \cap C_U. \end{aligned} \tag{0.8}$$

IV On oublie les coûts des composants et on se concentre exclusivement sur les contraintes dures. Donc, on calcule l'intervalle $[V_{Min}, V_{Max}]$ en résolvant le problème :

$$\begin{aligned} & max \quad (min) \quad u_0 \\ & \text{s.t.} \\ & \mathbf{u} \in U \cap C_{UH} \\ & u_0 \geq 0. \end{aligned} \tag{0.9}$$

Si ce problème est irréalisable ou si $v \notin [V_{Min}, V_{Max}]$ , alors on passe à l'étape (VI).

V On calcule la recette $\mathbf{u} \in U \cap C_{UH}$ qui produit le mélange avec sur-qualité minimale.

$$\begin{aligned} & min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ & \text{s.t.} \\ & \mathbf{u} \in U \cap C_{UH}. \end{aligned} \tag{0.10}$$

VI On trouve $\mathbf{u} \in U$ pour produire un volume de mélange $v$ avec des propriétés aussi près que possible de $\mathbf{y_T}$ :

$$\begin{aligned} & min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ & \text{s.t.} \\ & \mathbf{u} \in U. \end{aligned} \tag{0.11}$$

### Incertitude dans le problème de mélange

En suivant ces étapes, la méthode RTO produit toujours une recette qui guide le processus de contrôle. Pour réduire l'imprécision du modèle produit par les incertitudes, la méthode met à jour les propriétés des composants et du mélange courant via l'équation de base. Cependant, l'actualisation du modèle peut échouer même à garantir une solution réalisable. Une des raisons principales de cet échec est l'hypothèse implicite que les données restent inchangées à l'intérieur de chaque boucle RTO. Lorsque les composants sont stockés dans des running tanks, la valeur de $B$ fluctue avec le temps parce que les composants sont issus de différents procédés présentant aussi des perturbations. De plus, et conformément à [47], l'incertitude dans tout système RTO peut être de quatre types :

1. L'incertitude du procédé : les propriétés des composants, la température, l'humidité, etc.
2. L'incertitude de mesure.

3. L'incertitude du modèle.

4. L'incertitude des marchés : les disponibilités et les prix des composants, la demande des différents mélanges, etc

Dans ce travail on fait face à l'incertitude due à la linéarisation de la loi des mélanges (incertitude du type 3) à travers de la mise à jour du modèle. On considère également les incertitudes découlant de la mesure des composants et des propriétés du mélange (incertitude du type 2) et l'incertitude causée par la connaissance imprécise des propriétés des composants (incertitude du type 1). Toutefois, d'autres types d'incertitude (e.g., l'incertitude due à des facteurs non contrôlés extérieurs au procédé comme la température ou l'humidité, l'incertitude sur les prix des composants ou l'incertitude dans les disponibilités des composants) se manifestent géométriquement d'une façon similaire et peuvent donc être traités de la même manière.

Pour fixer notre méthode RTO contre l'incertitude de mesure et la connaissance imprécise de $B$, nous utilisons les techniques d'optimisation robuste (voir [9]). L'idée consiste à calculer pour le polytope $C_U$ sa *région robuste* convexe $RC_U$ telle que tout point dans cette région résiste à l'incertitude de $B$ et aux erreurs de mesure, c'est-à-dire que tout point $\mathbf{u} \in RC_U$ est garanti de rester à l'intérieur du polytope réel pour toutes les réalisations possibles de $B$ et toute erreur de mesure, dans des limites raisonnables.

Dans la Figure 3, nous montrons les polytopes nominaux $U$ et $C_U$, le "robustope" $RC_U$ et le polytope réel $\widetilde{C}_U$. La différence entre le polytope nominal et le polytope réel $C_U$ et $\widetilde{C}_U$ est le résultat d'une différence entre la matrice nominale $B$ et la matrice réelle $\widetilde{B}$. Tout point dans $RC_U$ (par exemple la recette nominale robuste $\mathbf{u_r}$) se transforme en un point dans son voisinage (la vraie recette robuste $\widetilde{\mathbf{u}}_\mathbf{R}$) en raison des erreurs de mesure. Le robustope $RC_U$ est calculé de telle sorte que tout point de son voisinage (la région en pointillés) reste à l'intérieur du polytope réel $\widetilde{C}_U$.

Nous pouvons utiliser différents ensembles et différentes normes pour modéliser l'incertitude. Le niveau de conservatisme (combien nous voulons être protégés de l'incertitude) et la complexité du problème dépendent de ces choix. Prenons comme ensemble d'incertitude des intervalles et la norme $L_\infty$, les régions robustes obtenues sont des polytopes et la complexité du modèle est préservée au détriment d'être probablement trop conservateur (nous sommes protégés des pires déviations de toutes les valeurs de $B$ et des plus grandes erreurs de mesure survenant toutes, en même temps).

En modélisant l'incertitude de cette façon, on peut remplacer les polytopes robustes dans la méthode RTO et l'on obtient directement une nouvelle méthode RTO robuste et de même complexité. De plus, tous les outils développés pour l'analyse de l'infaisabilité du PM (voir plus bas), seront valables aussi

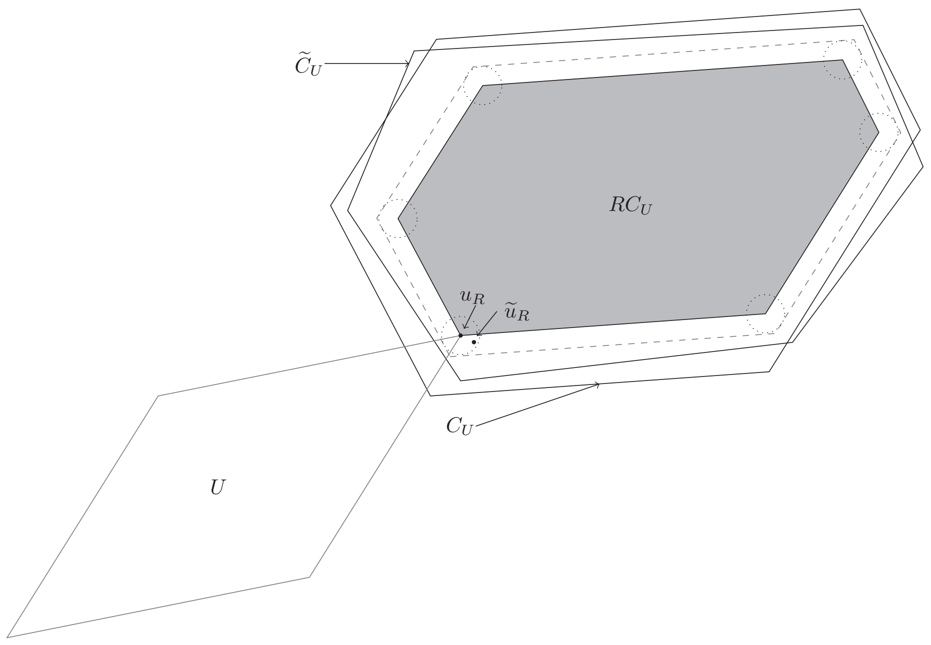


FIGURE 3: Robustope $RC_U$.

pour mesurer, visualiser et analyser la non-robustesse du PM.

En ce qui concerne le niveau de conservatisme, on peut comparer le prix de la robustesse (la différence des coûts entre la recette optimale robuste et la recette optimale nominale) avec le coût du remélange. Si le dernier est inférieur au premier, on peut ajuster les paramétres du modèle pour diminuer le niveau de conservatisme et vice versa. Le coût du remélange est difficile à estimer en pratique, alors on propose d'utiliser le prix de la robustesse comme une valeur de référence. On peut par exemple, comparer le prix de la robustesse avec une fraction du coût de remélange : le coût de la recette à produire pour corriger le mélange fabriqué au préalable et qui est hors spécifications. Il peut arriver, comme on le montre dans un exemple réel, que cette valeur soit déjà supérieure au prix de la robustesse.

## Infaisabilité du Problème de Mélange

Une instance du PM $(B, v, \mathbf{b_0}, V_0, \mathbf{c}, \underline{\mathbf{u}}, \overline{\mathbf{u}}, \underline{\mathbf{y}}, \ \overline{\mathbf{y}})$ est déterminée par les composants $B$, le volume $v$, le fond du bac $\mathbf{b_0}$ et son volume $V_0$, les coûts des composantes $\mathbf{c}$, les bornes hydrauliques et de disponibilité des composants $\underline{\mathbf{u}}$

et $\overline{\mathbf{u}}$ et par les bornes sur les propriétés $\underline{\mathbf{y}}$ et $\overline{\mathbf{y}}$. A chaque itération, la méthode RTO résout le PM afin de produire, si possible, la recette de coût minimum. Si le PM est infaisable, certaines bornes sont assouplies et un mélange faisable avec surqualité minimale est recherché. La méthode tient compte des possibles modification de l'instance du PM pour obtenir de meilleurs résultats. Plus précisément, les volumes $V_0$ et $v$ sont utilisés comme variables de contrôle dans le modèle afin de produire une recette plus économique ou faisable lorsque les valeurs originales de $V_0$ et $v$ ne le permettent pas. Dans certains cas, même avec ce degré de liberté supplémentaire, le PM reste infaisable pour les volumes requis. Alors, pour assurer la continuité de l'opération, la méthode propose de produire le mélange infaisable le plus proche du mélange cible $y_t$.

Dans ce chapitre, on s'intéresse à mesurer et visualiser l'infaisabilité du PM et à l'utilisation d'autres paramètres de l'instance du PM pour contrôler le procédé de mélange. Outre les volumes $v$ et $V_0$, la disponibilité des composants, les bornes hydrauliques ou encore les composants peuvent être utilisés pour piloter le processus de mélange. Ici, on analyse exclusivement la modification des bornes de disponibilité et hydrauliques.

## Les niveaux d'infaisabilité du problème de mélange

Définissons le polytope

$$C_{U1} = C_U \cap S_1$$

dans l'espace des composants et le polytope

$$C_{Y1} = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid \mathbf{u} \in C_{U1}\}$$

dans l'espace des propriétés.

Si le PM est infaisable pour une paire de volumes fixes $(V_0, v)$ et si l'on suppose que $U \neq \emptyset$, alors on peut avoir 3 différents niveaux d'infaisabilité qui peuvent être énoncées en termes de polytopes de mélange :

Niveau 1. $C_{U1} \neq \emptyset$ et $U \cap C_{U1} = \emptyset$.

Niveau 2. $C_U \neq \emptyset$ et $C_{U1} = \emptyset$.

Niveau 3. $C_U = \emptyset$.

Pour les trois niveaux d'infaisabilité, on peut modifier les propriétés des composants (la matrice $B$) et forcer le polytope $C_U$ à avoir une intersection non vide avec le polytope $U$ en produisant de cette manière un mélange faisable. D'autre part, comme le simplex $S_1$ contient toutes les recettes faisables, alors $C_{U1}$ représente l'ensemble de toutes les recettes ayant leurs propriétés $(C_{Y1})$

dans les spécifications. Un polytope $U$ peut atteindre, par des modifications des bornes, un point $\mathbf{u} \in C_U \setminus U$ ssi $\mathbf{u} \in S_1$, c'est-à-dire si et seulement si le PM a une infaisabilité de niveau 1.

### Mesure de l'infaisabilité et directions faisables

Pour chaque niveau d'infaisabilité, on peut définir l'*infaisabilité* du PM en termes des polytopes de mélange : l'infaisabilité de niveau 1 est définie par

$$D_U = D(U, C_{U1}) \quad \text{et} \quad D_Y = D(P, C_{Y1})$$

l'infaisabilité de niveau 2 par

$$D_{U2} = D(U, C_U) \quad \text{et} \quad D_{Y2} = D(P, C_Y)$$

et l'infaisabilité de niveau 3 par

$$D_{Y3} = D(P, C).$$

Définissons une *direction faisable* dans l'espace des composants comme une direction de translation des polytopes $U$ ou $C_U$ produisant sa collision et par conséquent la faisabilité du PM. Une direction de translation des polytopes $P$ ou $C_Y$ produisant sa collision est appelée également une direction faisable (dans l'espace des propriétés), car elle peut être vue comme l'image par l'équation de base d'une direction faisable dans l'espace des composants.

Comme on ne considère que des modifications des bornes pour transformer d'infaisable en faisable une instance du PM, dans ce qui suit, nous allons seulement développer l'analyse de l'infaisabilité de niveau 1. Dans l'espace des composants on s'intéresse au calcul de l'écart entre les polytopes $U$ et $C_{U1}$. Les deux polytopes sont décrits dans leurs formes duales. Donc, on peut calculer sa distance par la résolution du problème :

$$\begin{aligned} &min \quad \|\mathbf{t} - \mathbf{u}\| \\ &\text{s.t.} \\ &\mathbf{u} \in U, \ \mathbf{t} \in C_{U1}. \end{aligned} \tag{0.12}$$

Si $\mathbf{u}^* \in U$ et $\mathbf{t}^* \in C_{U1}$ est une solution du problème (0.12), alors $D_U = d(\mathbf{u}^*, \mathbf{t}^*) = \|\mathbf{t} - \mathbf{u}\|$ et

$$\mathbf{w_U} = \frac{\mathbf{t}^* - \mathbf{u}^*}{D_U} \tag{0.13}$$

est une direction faisable dans l'espace des composants qui exprime la direction de déplacement d'effort minimal de $U$ pour rencontrer $C_{U1}$.

On observe que $\mathbf{p}^* = \mathbf{y}(\mathbf{u}^*) \in P$ et $\mathbf{q}^* = \mathbf{y}(\mathbf{t}^*) \in C_{Y1}$. Alors,

$$\mathbf{w}_{\mathbf{Y}}^{\mathbf{U}} = \frac{\mathbf{q}^* - \mathbf{p}^*}{\|\mathbf{q}^* - \mathbf{p}^*\|}$$

est une direction faisable dans l'espace des propriétés.

Pour calculer la distance entre les polytopes $P$ et $C_{Y1}$, on doit noter que pour ces polytopes nous n'avons ni leur représentation primale ni leur représentation duale. Toutefois, on peut calculer sa distance indirectement à partir des polytopes $U$ et $C_{U1}$ et de l'équation de base. En résolvant le problème

$$\begin{aligned}&min \quad \|\mathbf{y}(\mathbf{t}) - \mathbf{y}(\mathbf{u})\|\\ &\text{s.t.}\\ &\mathbf{u} \in U, \ \ \mathbf{t} \in C_{U1}\end{aligned}$$

on obtient $\mathbf{p}^* = \mathbf{y}(\mathbf{u}^*) \in P$ et $\mathbf{q}^* = \mathbf{y}(\mathbf{t}^*) \in C_{Y1}$ tel que $D_Y = d(\mathbf{p}^*, \mathbf{q}^*)$. Alors,

$$\mathbf{w}_{\mathbf{Y}} = \frac{\mathbf{q}^* - \mathbf{p}^*}{D_Y} \tag{0.14}$$

est la plus courte direction de déplacement de $P$ pour joindre $C_{Y1}$ et

$$\mathbf{w}_{\mathbf{U}}^{\mathbf{Y}} = \frac{\mathbf{t}^* - \mathbf{u}^*}{\|\mathbf{t}^* - \mathbf{u}^*\|}$$

est la direction faisable correspondante dans l'espace des composants.

### Visualisation de l'infeasibilité dans l'espace des composants

Soit $\mathbf{w_1} = \mathbf{w_U}$ et $\mathbf{w_2} = \mathbf{w}_{\mathbf{U}}^{\mathbf{Y}}$. Ces vecteurs définissent un plan 2D dans l'espace des composants dont l'équation paramétrique est

$$H_2 = \{\mathbf{u}(\mathbf{s}) \mid \mathbf{u}(\mathbf{s}) = \mathbf{u_M} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s}\}$$

avec $\mathbf{s} = (s_1, s_2)^{\mathrm{T}} \in \mathbb{R}^2$, $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2})^{\mathrm{T}}$ et $\mathbf{u_M}$ le milieu du segment $[\mathbf{u}^*, \mathbf{t}^*]$.

Désignons par $U_2$ et $C_2$ les coupes de $U$ et $C_{U1}$ avec $H_2$. Pour produire la visualisation de l'infaisabilité dans l'espace des composants, on dessine les polygones $U_2$ et $C_2$ dans le plan $H_2$ avec $\mathbf{u_M}$ pris comme centre de coordonnées et les vecteurs de $\mathbf{w_1}$ et $\mathbf{w_2}$ comme axes (voir Figure 4).

Il est important de remarquer les caractéristiques de cette visualisation.

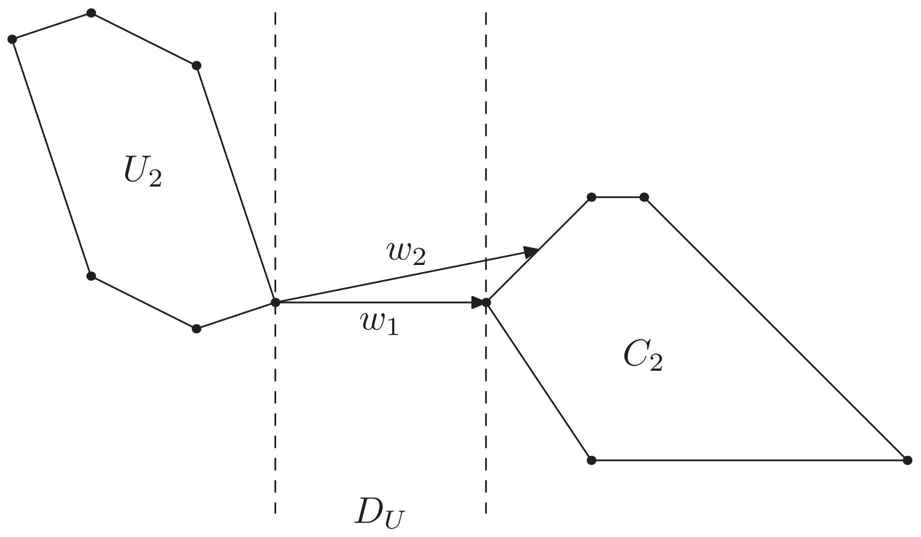


FIGURE 4: Visualisation de l'infaisabilité dans l'espace des composants.

Comme on l'obtient en coupant les polytopes $U$ et $C_{U1}$ avec un plan défini par $\mathbf{w_U}$ et $\mathbf{w_U^Y}$ qui passe par les points les plus proches dans les deux polytopes, la distance entre les polygones $U_2$ et $C_2$ est $D_U$ (la distance entre les polytopes $U$ et $C_{U1}$ dans le $n$-espace original). Pour la même raison, l'angle entre les directions $\mathbf{w_U}$ et $\mathbf{w_U^Y}$ est préservée.

## Visualisation de l'infaisabilité dans l'espace des propriétés

Considérons les directions faisables dans l'espace des propriétés $\mathbf{w_1} = \mathbf{w_Y}$ et $\mathbf{w_2} = \mathbf{w_Y^U}$ et denotons $H_2$ le 2D-plan défini par

$$H_2 = \{\mathbf{y}(\mathbf{s}) \in \mathbb{R}^2 \mid \mathbf{y}(\mathbf{s}) = \mathbf{y_M} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s}\}$$

avec $\mathbf{s} = (s_1, s_2)^T \in \mathbb{R}^2$, $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2})$ et $\mathbf{y_M}$ le milieu du segment $[\mathbf{p}^*, \mathbf{q}^*]$. Pour produire la visualisation de l'infaisabilité dans l'espace des propriétés, on peut réutiliser la procédure suivie dans l'espace des composants, à condition d'avoir la description duale des polytopes $P$ et $C_{Y1}$. Nous n'avons, ni la description primale ni la description duale de ces polytopes, mais on peut les avoir (au moins théoriquement) à partir des polytopes $U$ et $C_{U1}$ et de l'équation de base. Le problème d'énumération des sommets d'un polytope quelconque est un problème ouvert (on ne connaît pas d'algorithme polynomial pour le résoudre). Cependant, la structure des polytopes $U$ et $C_{U1}$ et leurs tailles ($m, n \leq 20$), permettent l'obtention de leurs sommes d'une manière efficace. Au contraire, les facettes de $P$ et $C_{Y1}$ ne peuvent pas toujours être obtenues de manière efficace. Parfois, nous rencontrons des polytopes $P$ et $C_{Y1}$ pour lesquels l'énumération de facettes est très coûteuse en temps de calcul. Ainsi, on modifie la procédure pour obtenir les polygones $P_2$ et $C_2$ en coupant les polytopes $P$ et $C_{Y1}$ à partir de leur forme primale.

### Devenir faisable et robuste

Dans la dernière section on a montré comment mesurer et visualiser l'infaisabilité du PM. On va voir maintenant les modifications des bornes hydrauliques et de disponibilité afin d'améliorer la qualité du mélange et de produire, si possible, une recette faisable. Plus précisément, on va transformer le polytope $U$ dans le polytope

$$U_{NEW} = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{u}} - \underline{\boldsymbol{\Delta}} \leq \mathbf{u} \leq \overline{\mathbf{u}} + \overline{\boldsymbol{\Delta}}\ ,\ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1,\ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}\}$$

en choisissant les variables élastiques appropriées $\underline{\boldsymbol{\Delta}}, \overline{\boldsymbol{\Delta}} \in \mathbb{R}^{n+}$ de façon à avoir $U_{NEW} \cap C_{U1} \neq \emptyset$.

On remarque que les bornes hydrauliques et de disponibilité peuvent être ajustées individuellement et peuvent atteindre leurs limites inférieure et supérieure : $[0, 1]$. En d'autres termes, le polytope $U_{NEW}$ peut atteindre n'importe quelle recette de $C_{U1}$ car $C_{U1} \subseteq S_1$. D'autre part, ces bornes sont fixées conformément à la planification globale et par des besoins opérationnels (par exemple, pour promouvoir l'utilisation des composants abondants ou peu coûteux, ou afin de vider le bac d'un composant). L'impact de la violation des bornes sur la performance globale de la raffinerie est difficile à mesurer. Encore plus difficile est d'estimer l'impact individuel de la violation de chaque borne. Pour cette raison, nous avons décidé de procéder comme suit : d'abord, on calcule la meilleure recette $\mathbf{u}^* \in C_{U1}$ à produire selon un critère d'optimisation et ensuite, on obtient les modifications des bornes pour avoir $\mathbf{u}^* \in U_{NEW}$.

On observe que, en utilisant les vecteurs $\mathbf{u}^-, \mathbf{u}^+ \in \mathbb{R}^{n+}$, on peut restreindre les modifications des bornes en cherchant la meilleure recette $\mathbf{u}^*$ dans $U^+ \cap C_U$ (voir Equation 0.15) au lieu de $C_{U1}$.

$$U^+ = \{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \underline{\mathbf{u}} - \mathbf{u}^- \leq \mathbf{u} \leq \overline{\mathbf{u}} + \mathbf{u}^+ \leq \mathbf{1}\ ,\ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\} \qquad (0.15)$$

En outre, dans la Section (4.7) on construit des repères concernant l'impact des bornes des composants sur le coût et la sur-qualité du mélange et sur le volume nécessaire à couler pour avoir le premier mélange faisable. Cette information pourrait être utile pour décider si l'on applique ou non les modifications des bornes.

Parmi les critères d'optimisation pour chercher la meilleure recette $\mathbf{u}^* \in C_{U1}$ on peut minimiser l'infaisaibilité (la distance entre les polytopes correspondants), minimiser les modifications des bornes ou l'on peut utiliser les mêmes critères de la méthode RTO. Une fois la recette $\mathbf{u}^*$ déterminée, on obtient les modifications optimales des bornes

$$\underline{\boldsymbol{\Delta}}^* = \max\{\mathbf{0}, \underline{\mathbf{u}} - \mathbf{u}^*\} \quad \text{et} \quad \overline{\boldsymbol{\Delta}}^* = \max\{\mathbf{0}, \mathbf{u}^* - \overline{\mathbf{u}}\}$$

qui nous permettent de produire $\mathbf{u}^*$. On désigne par

$$\mathscr{E} = \sum_{i=1}^{n} (\underline{\Delta}_i^* + \overline{\Delta}_i^*)$$

l'effort requis pour effectuer la transformation de l'instance du PM.

L'approche décrite précédemment ne considère pas l'incertitude du PM qui peut provoquer que le mélange faisable obtenu pour les données nominales soit en réalité infaisable. Cette situation s'explique par le fait que les polytopes $U_{NEW}$ et $C_{U1}$ se touchent à peine (voir Figure 5). Donc, toute petite différence entre la valeur nominale et les données réelles peut entraîner une fausse intersection. Pour remédier à cette fragilité et construire un mélange robuste, il suffit de substituer dans la procédure précédente, chaque polytopes par son robustope correspondant tels qu'ils ont été déterminés dans la Section (3.4). De cette façon, la recette $\mathbf{u}^*$ sera choisi bien à l'interieur du polytope $C_{U1}$ et résistera à toute variation des données, dans les limites pré-établies.

## Guidage du procédé de mélange

Lors du procédé de mélange, on peut vouloir obtenir un mélange faisable à partir d'un mélange infaisable ($\mathbf{b_0}$) tout en versant le moins de volume possible. Ce critère se révèle être très pratique en cas d'événements imprévus, comme le manque d'un ou plusieurs composants ou l'arrêt d'une pompe, obligeant à interrompre prématurément le procédé de mélange. En outre, il permet de vendre le mélange le plus tôt possible en réduisant ainsi les coûts d'inventaire. Pour trouver le premier mélange faisable (en termes de volume), on procède

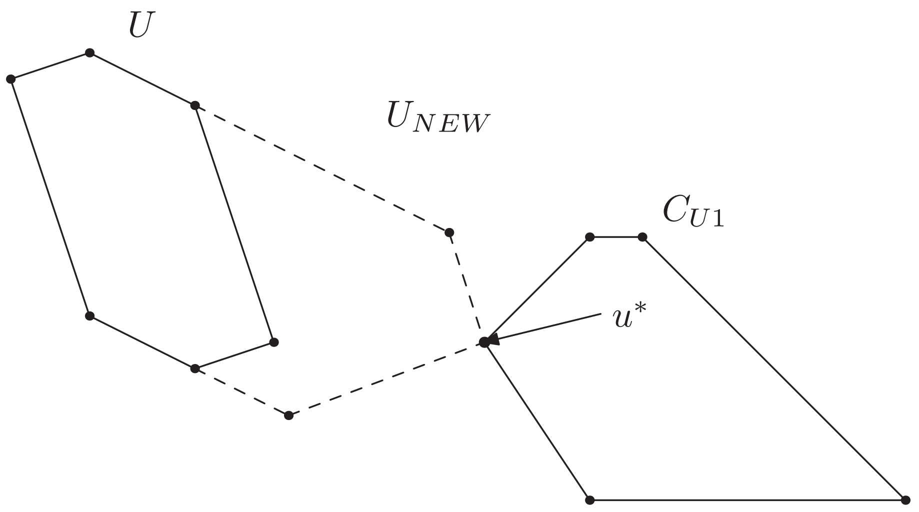


Figure 5: Transformation du polytope $U$ pour atteindre le mélange faisable $\mathbf{u}^* \in C_{U1}$.

comme suit. Tout d'abord, on résout les problèmes

$$\begin{aligned}
&max \quad (min) \quad u_0 \\
&\text{s.t.} \\
&\mathbf{u} \in C_{U1} \\
&u_0 \geq 0.
\end{aligned} \tag{0.16}$$

pour trouver $u_{0Max}$ et $u_{Min1}$ ($u_{0Min}$ et $u_{Max1}$). Puis, à partir de ces valeurs et de la relation $u_0 = V_0/v$ on obtient l'intervalle de volumes

$$[V_{Min1}, V_{Max1}]$$

pour lesquels $C_{U1} \neq \emptyset$. Les mélanges $\mathbf{y_{Min1}} = \mathbf{y}(\mathbf{u_{Min1}})$ et $\mathbf{y_{Max1}} = \mathbf{y}(\mathbf{u_{Max1}})$ sont le premier et le dernier mélange, en termes de volume, que l'on peut produire en modifiant les bornes des composants. Enfin,

$$\underline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \underline{\mathbf{u}} - \mathbf{u_{Min1}}\} \quad \text{et} \quad \overline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \mathbf{u_{Min1}} - \overline{\mathbf{u}}\}$$

sont les modifications minimales des bornes pour atteindre le mélange $\mathbf{y_{Min1}}$ au volume $V_{Min1}$. La relation entre ces volumes et l'impact des bornes des composants sur le procédé de mélange peut être mieux expliqué en renvoyant à la Figure 6.

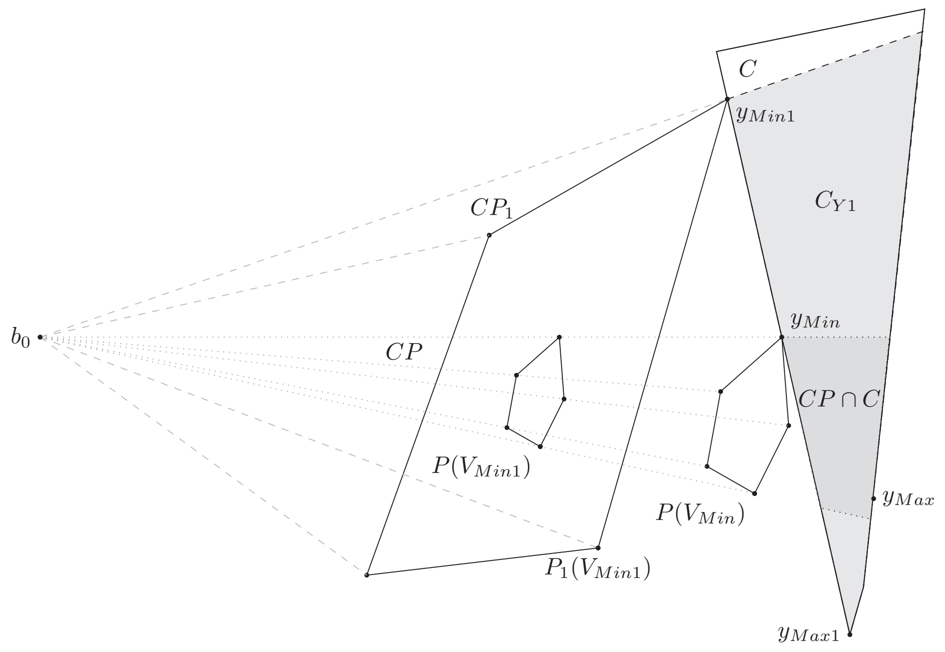


FIGURE 6: Impact de la modification des bornes des composants sur le procédé de mélange.

Dans cette figure, le polytope $P$ va de $P(0) = \mathbf{b_0}$ à $P(\infty)$ formant le cône :

$$CP = \left\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = conv\left\{V_{P(\infty)} \cup \mathbf{b_0}\right\}\right\}.$$

Les points $\mathbf{y_{Min}} \in P(V_{Min}) \cap C$ et $\mathbf{y_{Max}} \in P(V_{Max}) \cap C$ sont le premier et le dernier mélange que l'on peut produire.

D'autre part, comme $U \subseteq S_1$ alors $P(v) \subseteq P_1(v) = y(S_1), \quad \forall v \geq 0$. Par conséquent, $P_1(v)$ décrit le cône :

$$CP_1 = \left\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = conv\left\{V_{P_1(\infty)} \cup \mathbf{b_0}\right\}\right\}$$

qui comprend tous les mélanges productibles quand les bornes hydrauliques et de disponibilité ne sont pas respectées. Le premier et le dernier de ces mélanges étant dans les spécifications (dans le polytope $C$) sont les mélanges $\mathbf{y_{Min1}}$ et $\mathbf{y_{Max1}}$. L'intersection de $CP_1$ avec $C$ est le polytope $C_{Y1}$ et la visualisation dans l'espace des propriétés presentée dans la Section (4.5), c'est une coupe des polytopes $CP(v)$ et $C_{Y1}(v)$ pour un volume $v$ fixe.

Enfin, supposons que $c^*$ et $q^*$ ($c_1^*$ et $q_1^*$) sont le coût et la sur-qualité minimaux qui peuvent être atteints par un mélange dans $CP \cap C$ (dans $C_{Y1}$). Alors, comme $P(v) \subseteq P_1(v)$ pour tout $v \geq 0$, il résulte que $P \cap C \subseteq C_{Y1}$. Par conséquent, $V_{Min1} \leq V_{Min}$, $c_1^* \leq c^*$ et $q_1^* \leq q^*$. Ainsi, nous pouvons dire que les bornes des composants sont responsables de l'augmentation du coût par $(c^* - c_1^*)$, de l'augmentation de la sur-qualité par $(q^* - q_1^*)$ et du volume supplémentaire à verser (afin d'avoir le premier mélange faisable) par $(V_{Min} - V_{Min1})$. Donc, on contourne la difficulté à déterminer les coûts de modification des bornes, en fournissant les indices

$$K_C = \frac{c^* - c_1^*}{c_1^*}, \quad K_Q = \frac{q^* - q_1^*}{q_1^*}, \quad K_V = \frac{V_{Min} - V_{Min1}}{V_{Min1}}$$

.

# Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

The oil blending process consists in determining the optimal proportions to blend from a set of available feedstocks or components such that the final product obtained fulfills a set of specifications on their properties. Blending problems (BP) arise typically in the petrochemical industry where several types of raw components are blended to produce gasoline and other oils with specified properties.

The blending system is typically constituted by three functional subsystems: the scheduling subsystem, the on-line optimizer and the control subsystem. The scheduling subsystem is the one in charge of the general refinery production planning and specifically, it generates the initial target recipe for any blend operation scheduled. Several sources of uncertainty (see below) pervade the process. The on-line optimizer is then required to update the target recipe which may became sub-optimal, or even infeasible, due to these disturbances in the process. The feedback is based on measures of the blends' and components' properties gathered by on-line analyzers. Finally, the control subsystem is in charge to adjust the components flow rates in order to attain the current target recipe produced by the online optimizer. In Figure 1.1, we show the integration of these systems and the flow chart which describes a typical blending process.

In this work we concentrate on the analysis of the *Real Time Optimization* (RTO) system formed by the on-line optimizer and the control subsystem. A main assumption here is that the properties of a blend are determined as a linear combination of the components properties, that is, we suppose that properties blend linearly. In general they do not, but they can be replaced by approximated by linear functions of other blend's properties; see, e.g., [34] and [23]. We would like to mention that this assumption is by no means an over simplification of the problem. Actually, the linear blending model is on the basis of the on-line optimizer of the Anamel software which is used by some blending units within the TOTAL group (see [13]). Additionally, we focus on the production of a single blend from a set of components and we consider the case when the components are stocked in *"running" tanks*. That is, when the components' tanks are connected directly to upstream processes where the components are being produced. These ones are flowing into their tanks at the same time that they are blended.

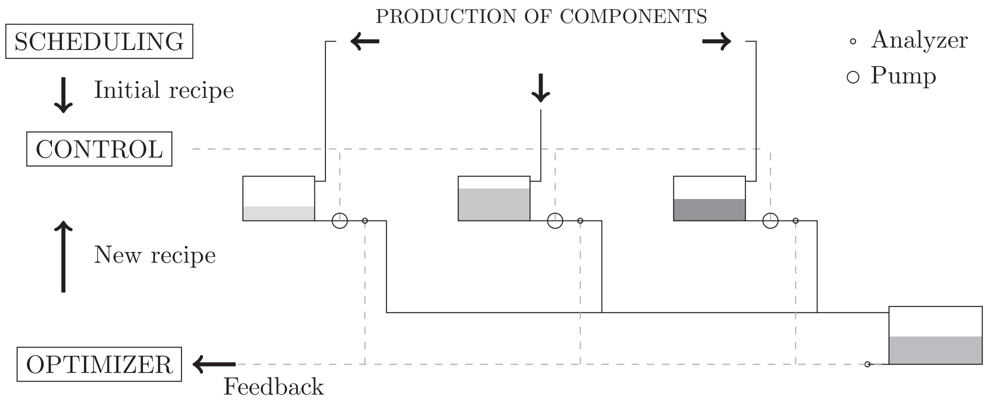


Figure 1.1: Blending system.

An important characteristic in the oil blending process is the strict requirements ("hard bounds") over some blend's properties controlled by environmental, legal and technological standards. If the blend does not satisfy these hard bounds, then it is not sellable and one must reblend it. *Reblending* reduces the refinery's capacity with several involved costs (e.g., the tank rent, the logistic and inventory costs,...). So, one should consider these blends' bounds as hard constraints which must be satisfied.

On the other hand there are "soft bounds". If the blend produced does not fulfill the requirements imposed by these bounds, then it is better than what is required and there is a cost we incur for the blend's *quality giveaway*.

The oil blending is a complex process where several unknown and uncertain factors affect the blend's properties. In addition to the plant-model error produced by the linearization of blending laws, there are other sources of uncertainty: measure errors on components and properties caused by instruments' precision; uncertain knowledge of the components' properties due to upstream process variations and uncontrolled blending conditions such as air temperature, humidity, etc. All these are typical uncertainties arising in any RTO problem (cf. [47]).

Hard constraints on the blends' properties combined to these sources of uncertainty are the ideal characteristics to apply the *Robust Optimization* (RO) techniques (see [9]). Robust Optimization is an approach to optimize under uncertainty. A main issue in the RO techniques is to consider a deterministic and set-based uncertainty model. No probabilistic assumption is made over the uncertainty and the solution obtained is optimal for any realization of the uncertainty in a given set. RO techniques seem to be ad hoc to address an uncertain RTO problem where the feasibility is the primary concern.

Several previous works (see, e.g., [40, 24, 41, 46]) deal with the BP feasibility and optimization. The feasibility of several blends to be produced from the same set of components is considered in [29]. Later, in [17] the author is concerned with the properties' uncertainty too. He develops a sensitivity analysis of the recipe when measurement errors in the components properties vary in an ellipsoid. In [38] an on-line optimization method is used to update the recipe and reduce the plant/model mismatch caused by uncertainty. In [26], a geometric approach is presented for the product and mixture design problems but only considering the uncertainty due to measurement imprecisions. To deal with components' properties uncertainty, [43] and [39] proposed a linear and non-linear blending RTO systems based on predictions of feedstocks properties whereas [44] presented a chance constraint model and an hybrid: neural networks - genetic algorithm solution. More recently, [13] introduced a linear blending control algorithm which handles this type of uncertainties via an estimator of the components' properties. A more general method for the non-linear blending model based on stochastic programming which covers various types of uncertainty is presented in [48].

This work tries to answer a number of real world questions coming from the petroleum blending industry. The answers to these questions are required to optimize and control, in real time, the production of robust blends. This explains the need for developing efficient computational tools. Another important point of this work is the generation of supportive visualizations of the objects under analysis which are intended to help the decision-making.

In Chapter 2 we present the Blending Polytopes inherent to the Blending Problem and the Basic Equation which will be used to introduce the updated data in the RTO method developed in Chapter 3. Here we analyze the feasibility of a blend to be produced from a set of components when the heel of a previous blend is used in the composition of the new blend. Critical information for the control system is then produced. For instance, knowing the minimal volume to pour before getting a robust feasible blend reduces the unnecessary control system interventions. These ones try to maintain the blend within specifications by modifying the recipe produced by the on-line optimizer but often based on a limited view of the problem. Later we model different sources of uncertainty arising in the blending process and we propose a robust RTO method based on the RO techniques. The method presented is intended to avoid reblending and we measure its performance in terms of the blend quality giveaway and the recipe's cost. We propose to compare the price of robustness (the cost difference between the robust and the nominal recipes) with the reblending cost in order to adjust the level of conservatism in the model. Finally, some numerical examples are presented.

In Chapter 4 we analyze the case when the RTO method fails to produce a feasible blend. We exploit the Blending Polytopes to measure, classify and visualize the BP infeasibility. A fine analysis of the components bounds modifications is conducted to guide the process towards the “best“ robust blend. A set of indices and visualizations provide a helpful support for the decision maker.

Finally, in the Appendix A we present a customized algorithm for computing the vertices of the blending polytopes and in the Appendix B we present some screenshots of the visualization system VISAMEL in which we have implemented some of the ideas presented in this work.

Chapter 2

# Blending Polytopes

After a set of preliminary definitions, four intrinsic polytopes of the blending problem are introduced. These polytopes constitute the core of this work because they define a natural framework for the study of the feasibility and infeasibility of the BP. The distance between the pairs of polytopes gives a measure of the BP infeasibility and determines a set of significant points and polygons allowing its visualization.

## 2.1 Preliminaries

We briefly review some of the basic definitions on polytope theory and convexity used in this work. A complete treatment of these fields can be found in [49] and [45]. For the basic notions on complexity theory we refer to [22].

Throughout this work we will denote by $\|\mathbf{x}\|_p$ the $L_p$ *norm* of a vector $\mathbf{x} \in \mathbb{R}^d$. Three important cases are the $L_1$ *norm*

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{d} |x_i|,$$

the *Euclidean norm*

$$\|\mathbf{x}\|_2 = \sum_{i=1}^{d} x_i^2$$

and the $L_\infty$ *norm*

$$\|\mathbf{x}\|_\infty = \max_{i=1,\dots,d} |x_i|.$$

For $S \subseteq \mathbb{R}^d$, the convex hull *conv(S)* of $S$ is the smallest convex set in $\mathbb{R}^d$ containing $S$. A $\mathcal{V}$*-polytope* is the convex hull of a finite set of points in some $\mathbb{R}^d$. A subset $P$ of $\mathbb{R}^d$ is called an $\mathcal{H}$*-polyhedron* if it is the intersection of a finite set of halfspaces in some $\mathbb{R}^d$. An $\mathcal{H}$*-polytope* is an $\mathcal{H}$*-polyhedron* that is bounded. A *polytope* is a set $P \subseteq \mathbb{R}^d$ which can be presented either as a $\mathcal{V}$*-polytope* or as an $\mathcal{H}$*-polytope*.

The *dimension* of a polytope is the dimension of its affine hull and a *d-polytope* is a polytope of dimension $d$ in some $\mathbb{R}^e$ $(e \geq d)$.

Let $P \subseteq \mathbb{R}^d$ be a polytope. A linear inequality $\mathbf{ax} \leq b$ is *valid* for $P$ if it is satisfied for all $\mathbf{x} \in P$. A *face* of $P$ is any set of the form

$$F = P \cap \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{ax} = b\}$$

where $\mathbf{ax} \leq b$ is a valid inequality for $P$. The *dimension* of a face of $P$ is the dimension of its affine hull: $\dim(F) = \dim(\mathrm{aff}(F))$. The faces of dimensions 0, 1, $\dim(P) - 2$, and $\dim(P) - 1$ are called *vertices*, *edges*, *ridges* and *facets* respectively. We denote the set of vertices of $P$ by $V$.

Every polytope has two representations. The *primal representation* as the convex hull of the finite set of its vertices:

$$P_{\mathcal{P}} = \{\mathbf{v} \in \mathbb{R}^d \mid \mathbf{v} = \sum_{k=1}^{s} \lambda_k v_k, \quad \sum_{k=1}^{s} \lambda_k = 1, \quad \lambda_k \geq 0, \quad k = 1, \ldots, s\}$$

and the *dual representation* as the intersection of a finite set of half-spaces:

$$P_{\mathcal{D}} = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}.$$

The question of how a polyhedron is represented has very important practical consequences. As a simple example consider the *d-cube*:

$$C_d = \{\mathbf{x} \in \mathbb{R}^d \mid -1 \leq x_i \leq 1, \quad i = 1, ..., d\}$$

and the dual representation is described by $2d$ linear inequalities, while to describe it in its primal form, $2^d$ points are required (the $d$-vectors all of whose components are 1 or -1). Thus the size of the two representations differs greatly, for large $d$. This fact has important consequences in practice when computing time is a crucial issue.

A $d$-polytope is called *simple* if each vertex is contained in exactly $d$ facets and is called *simplicial* if each facet contains exactly $d$ vertices. A vertex lying on more than $d$ facets is known as a *degenerate* vertex. In Figure 2.1 we sketch the *2-simplex*

$$S_1 = \{\mathbf{x} \in \mathbb{R}^3 \mid 0 \leq \mathbf{x} \leq 1, \ x_1 + x_2 + x_3 = 1\} = \mathrm{conv}\{(1,0,0), (0,1,0), (0,0,1)\}$$

and the *3-cube*

$$C_3 = \{\mathbf{x} \in \mathbb{R}^3 \mid -\mathbf{1} \leq \mathbf{x} \leq \mathbf{1}\} = \mathrm{conv}\left\{ \begin{array}{l} (\text{-}1,\text{-}1,\text{-}1), (\text{-}1,\text{-}1,1), (\text{-}1,1,\text{-}1), (\text{-}1,1,1), \\ (1,\text{-}1,\text{-}1), \ (1,\text{-}1,1), \ (1,1,\text{-}1), \ (1,1,1) \end{array} \right\}$$

in $\mathbb{R}^3$. Both polytopes are simple but only the simplex is simplicial.

Transforming a polytope from its primal form to its dual form and vice versa are known as the *facet enumeration* (FE) and the *vertex enumeration*

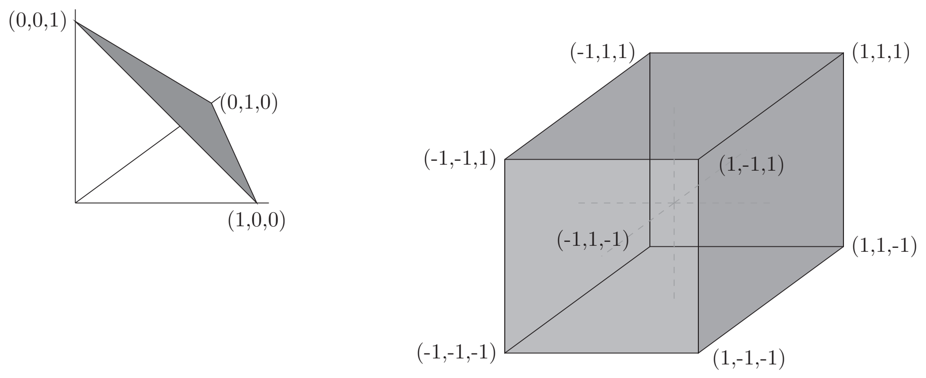


Figure 2.1: The 2-simplex and 3-cube in $\mathbb{R}^3$.

(VE) problems respectively. These problems are equivalent under polar duality: From a polytope $P$ given in its dual form, we obtain directly the primal representation of its dual ($P^{\triangle}$). Then, solving the facet enumeration problem for $P^{\triangle}$ we obtain its dual representation and from this we get the primal form of $P$.

Since the number of vertices of a $d$-polytope $P$ with $m$ facets is $\mathcal{O}(m^{\lfloor d/2 \rfloor})$ [37], then it is common to measure the efficiency of a vertex enumeration algorithm in terms of both the size of the input and the size of the output (see e.g., [42]). Under this scheme, a VE algorithm is *polynomial* if its running time is bounded by a polynomial in $d, m$ and the number of vertices $v$.

The vertex enumeration problem for polyhedra is known to be a NP-Hard problem [27] and for general polytopes the complexity is an open problem. However, for some particular types of polytopes the vertex enumeration can be made efficiently. For instance, if $P$ is a polytope simple or a *0/1-polytope*, we can find its vertices in linear time on $v$ using the reverse search method [7] and a backtracking approach [11] respectively. A *0/1-polytope* in $\mathbb{R}^d$ is one having all its vertices in $\{0, 1\}^d$.

Later we will need to compute the vertex set for the blending polytopes introduced in Section 2.3. These are not simple polytopes but their structure and their size can be exploited by a backtracking algorithm which we present in the Appendix A.

## 2.2 Distance Between Polytopes

The distance between the polytopes $A$ and $B$ is given by

$$D(A,B) = min\{\|\mathbf{x}-\mathbf{y}\| \mid \mathbf{x} \in A,\ \mathbf{y} \in B\}$$

with the most suitable norm for the context. Here we will need to compute $D(A,B)$ for the Euclidean norm and the $L_1$ norm. So, in the following we discuss the tractability of the distance problem for these norms and both representations of the polytopes $A$ and $B$.

Let's assume that $A_p$, $B_p$, $A_d$ and $B_d$ are the primal and dual forms of the polytopes $A$ and $B$. If $A$ and $B$ are given in their dual representation and the distance is measured by the Euclidean norm, then the distance problem is a convex quadratic programming problem:

$$min\{\|\mathbf{x}-\mathbf{y}\|_2^2 \text{ s.t. } \quad \mathbf{x} \in A_d,\ \mathbf{y} \in B_d\}.$$

If the $L_1$ norm is used, then $D(A,B)$ can be obtained by solving the linear programming problem

$$\begin{aligned} & min\ \mathbf{1}^{\mathrm{T}}\mathbf{s} \\ & \text{s.t.} \\ & \mathbf{s} \geq \mathbf{x}-\mathbf{y} \\ & \mathbf{s} \geq \mathbf{y}-\mathbf{x} \\ & \mathbf{x} \in A_d,\ \mathbf{y} \in B_d. \end{aligned} \tag{2.1}$$

On the other hand, if $A$ and $B$ are given in their primal form, we can proceed in a different way. By the separating hyperplane theorem (see, e.g., [45]), $A \cap B = \emptyset$ if and only if there is an hyperplane $H(\mathbf{w},b) = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} - b = 0\}$ strictly separating $A$ from $B$, i.e., such that $\mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} - b \geq 0 \quad \forall \mathbf{x} \in A$ and $\mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} - b < 0 \quad \forall \mathbf{x} \in B$. We define the *gap* of $H(\mathbf{w},b)$ as the sum of the shortest distances to $H$ from the nearest point in each polytope. In Figure 2.2, we show two hyperplanes separating the polytopes $A$ and $B$. The hyperplane of maximum gap is $H^*$ and its gap (represented as the distance between the parallel dotted lines to $H^*$) is $d(\mathbf{x_i},H^*) + d(\mathbf{y_j},H^*) = D(A,B)$.
Then, as shown in [1] and the development of support vector machines [10], finding the distance between $A$ and $B$ (using the norm $\|\cdot\|$) is equivalent to computing the separating hyperplane with the largest gap:

(i) $A \cap B = \emptyset \Leftrightarrow \exists\, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $(\mathbf{w},b) \in \mathcal{H}$.

(ii) $A \cap B = \emptyset \Rightarrow D(A,B) = 2/min\{\|\mathbf{w}\|' \mid (\mathbf{w},b) \in \mathcal{H}\}$

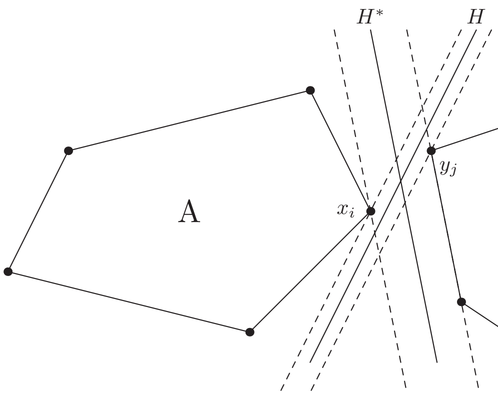


Figure 2.2: Maximum gap defined by the hyperplane H*.

where,

$$\mathcal{H} = \{(\mathbf{w}, b) \in \mathbb{R}^{d+1} \mid \mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} - b \geq 1 \ \forall \mathbf{x} \in A,\ \mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} - b \leq -1 \ \forall \mathbf{x} \in B\}$$

is the family of hyperplanes separating strictly $A$ from $B$ and $\|\cdot\|'$ is the *dual norm* to $\|\cdot\|$.

Using the Euclidean norm, $D(A, B)$ reduces to solve a quadratic problem and if the $L_1$ norm is used, the resulting problem is $min\{\|\mathbf{w}\|_\infty \mid (\mathbf{w}, b) \in \mathcal{H}\}$ which can also be stated as a linear program similar to problem (2.1) (see [16]). Finally, knowing that the convex quadratic programming problem can be solved in polynomial time [28], it follows that computing $D(A, B)$ can be done efficiently either if both polytopes are given in their primal forms or in their dual forms.

## 2.3 Blending Polytopes

Let's denote by $n$ and $m$ the number of components and properties respectively (usually, both $m$ and $n$ are lower than 20).

Let $S_1$ be the simplex defined as

$$S_1 = \{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}, \quad \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\}.$$

We will represent a *blend's recipe* (*recipe* in short) by a vector $\mathbf{u} \in S_1$ such that $u_j$ is the percentage of the component $j$ present in the blend whereas the *blend's properties* (*properties*) will be denoted by a vector $\mathbf{y} \in \mathbb{R}^m$. Then the set of *components* is described by the $m \times n$ matrix $B$ where, $B_{i,j}$ is the $i^{th}$

physical and/or chemical characteristic of the component $j$.

When blending it is customary to use the heel of a previous blend to produce a new one. The process starts then with a volume $V_0$ of a previous blend with properties $\mathbf{b_0} \in \mathbb{R}^m$ and continues adding gradually a volume $v$ of the new blend to obtain a final product's volume $V_{total} = V_0 + v$.

In this work we suppose that blend's properties are linear combination (or more precisely, a convex combination) of components' properties. Under this assumptions, for any $v \geq 0$ the blend's properties of the recipe $\mathbf{u}$ are determined by the *basic equation* defined as the linear mapping $y : \mathbb{R}^n \to \mathbb{R}^m$ such that:

$$\mathbf{y}(\mathbf{u}) = \alpha_v \ \mathbf{b_0} + \beta_v \ B\mathbf{u} \tag{2.2}$$

where, $\alpha_v = \frac{V_0}{V_0+v}$ and $\beta_v = \frac{v}{V_0+v}$.

Having poured a blend's volume $v_i$ with properties $\mathbf{b_i}$, we can use the basic equation to tune up the production of the remaining volume to blend. To do this, we just need to update $\mathbf{b_0}$ and $V_0$ and select the new volume to plan: $v \in [0, v - v_i]$. This is the kernel of the RTO method presented in Section 3.2. For this planning horizon ($v$), we can search the optimal cost recipe by solving the *Blending Problem* (BP):

$$\begin{aligned}
&min \ \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} \\
&\text{s.t.} \\
&\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}} && (2.3)\\
&\underline{\mathbf{y}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}} && (2.4)\\
&\mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1, \ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}. && (2.5)
\end{aligned}$$

Here the vector $\mathbf{c} \in \mathbb{R}^n$ represents the components' costs. A recipe $\mathbf{u}$ is subject to components availability and hydraulic bounds which impose the components constraints (2.3). In a similar way, the properties constraints (2.4) are determined by "quality" lower ($\underline{\mathbf{y}}$) and upper ($\overline{\mathbf{y}}$) bounds. These properties bounds are labeled as *hard* ($\underline{\mathbf{y}}_\mathbf{H}$, $\overline{\mathbf{y}}_\mathbf{H}$) or *soft* ($\underline{\mathbf{y}}_\mathbf{S}$, $\overline{\mathbf{y}}_\mathbf{S}$). Hard bounds are related to commercial and environmental specifications which must be satisfied whereas soft bounds can be violated but incurring into quality giveaway. Finally, (2.5) are the percentage constraints.

Geometrically, the intersection of the regions determined by the components constraints and $S_1$ defines the *recipe's polytope*:

$$U = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}} \ , \ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1, \ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}\}$$

which represents the set of all the recipes that can be produced. On the other hand, the properties constraints define the polyhedron (as it may be unbounded):

$$C_U = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{y}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}}\}.$$

Both are defined in the *components space.*

In *properties space*, the images of $U$ and $C_U$ under the basic equation are the polytopes

$$P = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid \ \mathbf{u} \in U\}$$

and

$$C_Y = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid \mathbf{u} \in C_U\}.$$

It is important to note here, that the last two polytopes are not given in a primal or dual form. Maybe, this is the reason why they were not studied previously. We show further how the basic equation and the structure of $U$ and $C_U$ are exploited in order to treat them.

Finally, the *target polytope* is defined by

$$C = \{\mathbf{y} \in \mathbb{R}^m \mid \underline{\mathbf{y}} \leq \mathbf{y} \leq \overline{\mathbf{y}}\}.$$

In this polytope we distinguish the *target blend* $\mathbf{y_T}$ which is a blend provided by the scheduling department having the minimal quality giveaway and other desirable characteristics. This blend will be considered as the ideal blend to produce when the BP fails to generate a feasible recipe.

Any point in $P$ represents the properties of a blend issued from one or more recipes in $U$ whereas $C$ depicts the set of properties within specifications. The polyhedron $C_U$ contains all the points (not necesarily recipes) in the components space that are mapped by $y$ onto properties within specifications forming the polytope $C_Y \subseteq C$.

We notice that the polytopes, $P$, $C_U$ and $C_Y$ depend on $v$. For these polytopes as for the basic equation we omit the $v$ when it is considered to be fixed.

At this point we have identified the polytopes underlying the BP and we observe that for a fixed volume $v$, a feasible solution of BP is a recipe $\mathbf{u} \in U \cap C_U$ with properties $\mathbf{y}(\mathbf{u}) \in P \cap C_Y$. Equivalently,

$$\text{BP is infeasible} \iff P \cap C_Y = \emptyset \iff U \cap C_U = \emptyset. \tag{2.6}$$

**Example 2.1 Blending Polytopes**

Let's consider the production of ($V_{total} = 10\,m^3$) of blend from two components and with only one property to control. The components are determined by the matrix $B = (4, 8)$ and their bounds are

$$\underline{\mathbf{u}} = (0, 0.3)^{\mathrm{T}} \quad \text{and} \quad \overline{\mathbf{u}} = (0.6, 1)^{\mathrm{T}}.$$

We are not using any previous blend ($V_0 = 0$) then we need to produce ($v = 10\,m^3$) of “fresh“ blend. We suppose that the blend's property must belong to the interval $[\underline{y}, \overline{y}] = [9, 11]$ and that the blend with minimal quality giveaway is $y_T = 9$.

In Figure 2.3 we present the blending polytopes in both spaces. In compo-

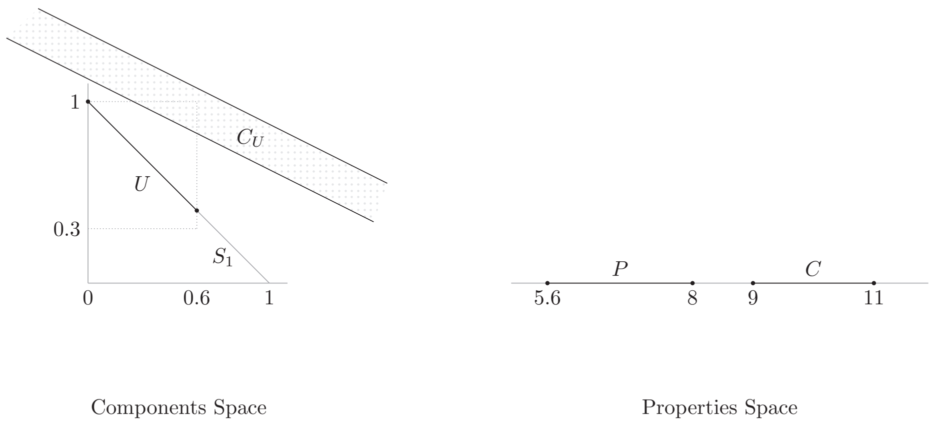


Figure 2.3: Blending polytopes in $\mathbb{R}^2$.

nents space ($\mathbb{R}^2$),we exhibit the polytope

$$U = \{(u_1, u_2) \in \mathbb{R}^2 \mid 0 \leq u_1 \leq 0.6, \quad 0.3 \leq u_2 \leq 1, \quad u_1 + u_2 = 1\}$$

and the unbounded polyhedron

$$C_U = \{(u_1, u_2) \in \mathbb{R}^2 \mid 9 \leq 4u_1 + 8u_2 \leq 11\}.$$

In properties space ($\mathbb{R}^1$), we show the polytope $P$ and the polytope $C_Y$ which coincides this time with the target polytope $C$. Here, the vertices of $U$ are the points $(0.6, 0.4)$ and $(0, 1)$ which transforms under $y$ in the vertices of $P$: 5.6 and 8. The gap between the polytopes $U$ and $C_U$ and between $P$ and $C_Y$ show us the infeasibility of this BP instance. Moreover, in this case the polytope $C_U$ does not contain any recipe as $C_U \cap S_1 = \emptyset$. Finally, we observe that using the recipe $(0, 1)$, that is, producing the blend with 100% of the second component, we obtain the closest blend to $y_T$. This infeasible blend has then the minimal quality giveaway among all the blends that can be produced.

# Chapter 3
# Robust RTO Method

The blending system is typically constituted by three functional subsystems: the scheduling subsystem, the on-line optimizer and the control subsystem. The scheduling subsystem purpose is to plan the refinery operations for different time horizons. In the long-term, production targets are determined using long-term forecasts of product demands and oil prices. In the medium-term planning, these targets are revised and adjusted regarding the estimates of components and tanks availabilities. Finally, in the short-term the planning concentrates on the production of target recipes for any blend operation scheduled. The volume to produce, the components to blend and all the other parameters needed to compute the target recipe are determined at this level and according to the general refinery planning. On these basis, one should expect the BP instance to have always a feasible solution. However, the target recipe computation is most of the times based on predictions and occasionally on measurements of the components properties. In both cases the accuracy of the model is affected: in the first case the model inherits the imprecisions of the predictions and in the second case because the components measurements can be also inaccurate and delayed [13]. Actually, as the components are in production at the same time that they are blended, therefore, the imprecision in the model induced by the defective knowledge of the components properties remains during all the process. For this and other types of uncertainties, the target recipe may became sub-optimal or even infeasible.

In order to reduce the mismatch between the model and the real system, the on-line optimizer updates the target recipe in a closed loop cycle using the on-line measures of the blend's and components' properties. The updated recipe will serve the controller to guide the blending process.

In this chapter we focus on the RTO system formed by the on-line optimizer and the control subsystem. Specifically, in Section 3.1 we analyze the blending process when the heel of a previous blend is used in the composition of the new blend. This analysis turns out to be very important for the development of the RTO method presented in Section 3.2. The RTO method is based on the basic equation and the blending polytopes, it is intended to avoid reblending and we measure its performance in terms of the blend quality give-away and the feedstocks prices. In Section 3.3, we analyze from a geometric point of view several types of uncertainty arising in the blending process and we outline how to fix the RTO method against these types of uncertainty. In

Section 3.4 we construct the corresponding uncertainty sets for these types of uncertainty using the robust optimization techniques. These sets determine the robust regions for the blending polytopes in which the new robust RTO (RRTO) method is based. Finally, in Section 3.5 we present some numerical examples to illustrate and compare some key aspects of the RTO method and its robust counterpart. The difference between the robust and the nominal recipes (the price of robustess) is compared with a component of the reblending cost. This iformation can be used to adjust the level of conservatism in the model.

## 3.1 Blending from a previous blend

If the blend is produced from scratch or the volume to pour is too big compared with the heel's volume, that is, if $V_0 = 0$ or $v \to \infty$ then the basic equation reduces to $\mathbf{y}(\mathbf{u}) = B\mathbf{u}$ and the polytope $P(v)$ does not depend on $V_0$. However, the fact of using the actual blend's properties $\mathbf{b_0}$ in the basic equation to update the model inside the RTO loop makes blending from a previous blend the regular case. Thus, we have $V_0 \neq 0$ and $P(v)$ is a polytope homothetic to $P(\infty)$ with $\mathbf{b_0}$ as center of homothecy. In properties space

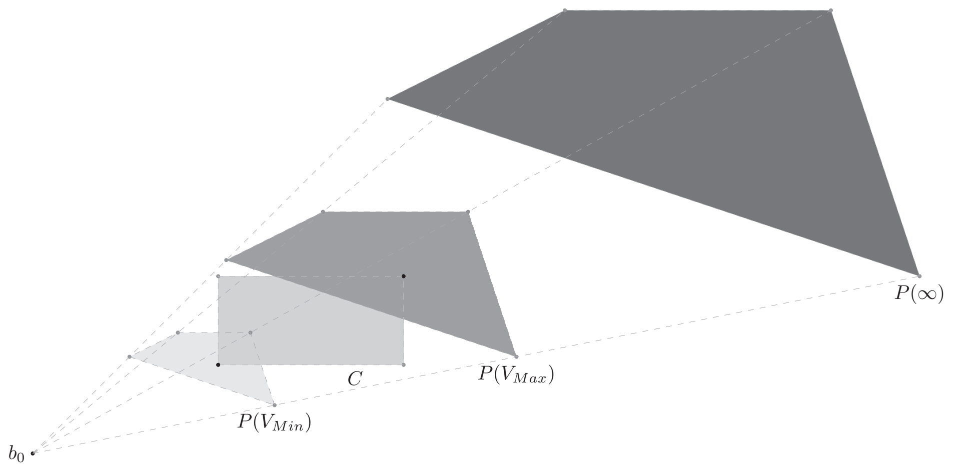


Figure 3.1: Polytope $P(v)$ with $\mathbf{b_0} \notin P(\infty)$.

the polytope $C$ doesn't depend on $v$. There exists then an interval (possibly empty) of volumes $[V_{Min}, V_{Max}]$ such that $P(v) \cap C \neq \emptyset, \quad \forall v \in [V_{Min}, V_{Max}]$.

The BP is feasible only for these volumes (see Figures 3.1 and 3.2).

Having the values of $V_{Min}$ and $V_{Max}$ helps to select the appropriate plan-

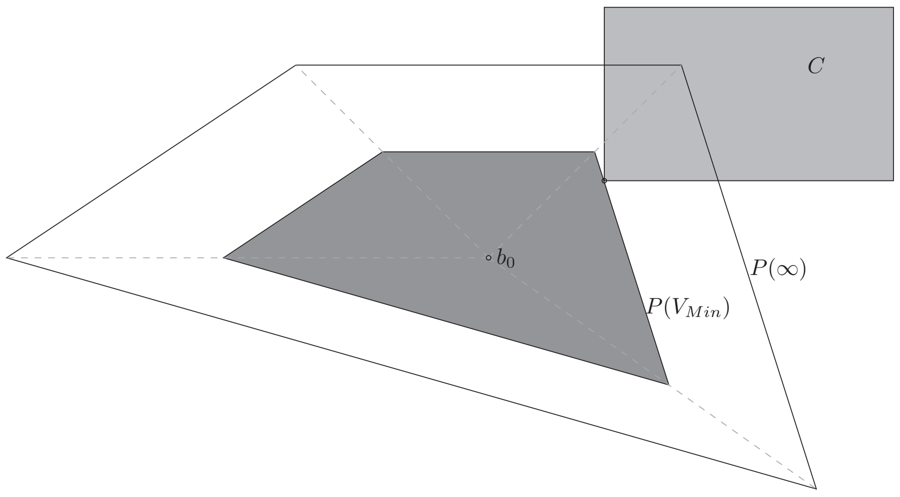


Figure 3.2: Polytope $P(v)$ with $\mathbf{b_0} \in P(\infty)$.

ning horizon at each RTO loop. Usually, the volume to produce is fixed by the scheduling department but the inaccuracy in the model and the uncertainty pervading the process may require a constant review of these volumes. Choosing a planning horizon by taking a volume $v < V_{Min}$ produces an infeasible BP and makes it necessary to generate an alternative recipe with a possible deterioration of the overall performance of the RTO method. On the other hand, $V_{Max}$ is the maximum volume to blend when looking for a blend within specifications. After $V_{Max}$ has been poured, $P(v)$ moves away from $C$ and the blend's properties deteriorate progressively.

In practice, when a blend is off-spec, the control system tries to correct it and make it to be within specifications. To do this, a small subset of components is used (the more economical or those having the properties to correct the blend faster) and often based on a limited view of the problem. A bonus of knowing $V_{Min}$ and $V_{Max}$ is that it helps the control system to optimize their interventions.

In a similar way, we can analyze the role of the use of $V_0$ in the feasibility of the BP. From the basic equation, we notice that letting fixed the volume $v$, the polytope $P$ moves from $P(\infty)$ to $\mathbf{b_0}$ as the volume $V_0$ increases. In this case, there exists also an interval (possibly empty) of volumes $[V_{0Min}, V_{0Max}]$ of the previous blend allowing us to produce a blend within specifications.

Making $u_0 = \frac{V_0}{v}$, we can rewrite the basic equation in an useful way:

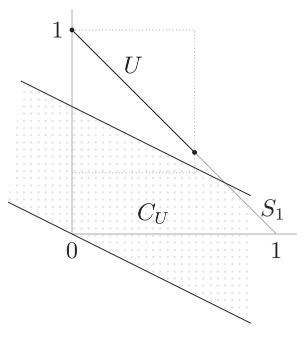
$$\mathbf{y}(\mathbf{u}) = \frac{u_0 \mathbf{b_0}}{1+u_0} + \frac{B\mathbf{u}}{1+u_0}. \tag{3.1}$$

Using this equation, we can determine the interval $[u_{0Min}, u_{0Max}]$ for which $P \cap C \neq \emptyset$ (and equivalently $U \cap C_U \neq \emptyset$). These values give us all the combinations of volumes $(V_0, v)$ producing a feasible blend.

Finally, we dedicate some attention to the cost of reblending a blend which failed to be within specifications. The reblending cost has several components, some of which are difficult to estimate. First of all, reblending affects the refinery's scheduling and particularly, the sales program. The blend off-spec may require to be transfered to a different tank and needs to be stocked until its reprocessing. This generates logistic, storage and inventory expenses. Additionally, reblending may demand the use of non standard components with the suitable properties in order to rectify the blend off-spec. Using additives or modifying the upstream processes to give the components the desirable characteristics which they lack, increase their costs. Hence, the cost difference between the optimal recipes computed when the heel's volume is used and when it is not, is also a component of the reblending cost.

### Example 3.1 Blending from a previous blend

Here, we continue the example 2.1 but considering the use of $(6\,m^3)$ of a previous blend with the property $b_0 = 15$. In this case, the polytopes $U$ and



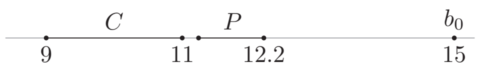


Figure 3.3: Blending from a previous blend in $\mathbb{R}^2$.

$C_Y$ do not change but the polytopes $C_U$ and $P$ do.

$$C_U = \{(u_1, u_2) \in \mathbb{R}^2 \mid 9 \leq \frac{6}{10} \times 15 + \frac{4}{10}(4u_1 + 8u_2) \leq 11\}$$

while the vertices of $P$ are $y(0.6, 0.4) = 11.24$ and $y(0, 1) = 12.2$.

In Figure 3.3 we show these polytopes and we observe that the BP for the new volumes is again infeasible. Moreover, looking the blending polytopes in the properties space we can deduce that we are taking "too much" volume from the previous blend or that we need to pour some extra volume in order to have a feasible blend. So, let's compute the extreme values of $V_0$ and $v$ for which the BP is feasible.

If we use the basic equation 3.1 to describe the polytope $C_U$ in terms of

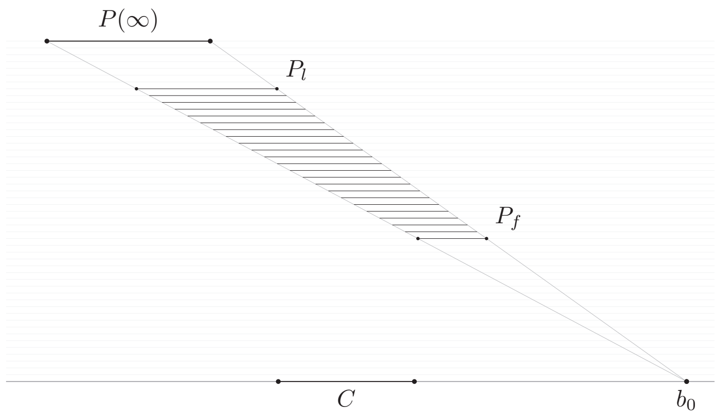


Figure 3.4: Evolution of polytope $P$ with the blending volume.

$u_0$ and $\mathbf{u}$, we have

$$\frac{9 - B\mathbf{u}}{6} \leq u_0 \leq \frac{11 - B\mathbf{u}}{4}.$$

Thus, the minimum and maximum values of $u_0$ such that $U \cap C_U \neq \emptyset$ are obtained computing $\mathbf{u_M} = \max_{\mathbf{u} \in U}\{B\mathbf{u}\}$ and $\mathbf{u_m} = \min_{\mathbf{u} \in U}\{B\mathbf{u}\}$ respectively. Here, we can find directly these values by evaluating $B\mathbf{u}$ in the vertices of $U$. Then, we get $\mathbf{u_M} = B(0, 1)^{\mathrm{T}} = 8$ and $\mathbf{u_m} = B(0.6, 0.4)^{\mathrm{T}} = 5.6$. Hence, $u_{0Min} = \max\{0, \frac{9-8}{6}\} = \frac{1}{6}$ and $u_{0Max} = \max\{0, \frac{11-5.6}{4}\} = 1.35$.

Now, by means of the relation $u_0 = \frac{V_0}{v}$ we can select a suitable pair of volumes $(V_0, v)$ producing a feasible blend. For instance, using all the available previous blend ($V_0 = 6\,m^3$), we need to produce at least $V_{Min} = 4.44\,m^3$ and no more than $V_{Max} = 36\,m^3$ in order to have a feasible blend. Otherwise, producing $v = 4\,m^3$ constraints the use of $V_0$ as follows: $V_{0Min} = 0.66\,m^3$ and $V_{0Max} = 5.4\,m^3$.

In Figure 3.4, we depict the evolution of polytope $P$ from $\mathbf{b_0}$ to $P(\infty)$. All the polytopes $P$ producing a feasible blend are in the shaded region. The first $(P_f)$ and last $(P_l)$ touching $C$, are obtained from any pair of volumes $(V_0, v)$ satisfying $\frac{V_0}{v} = u_{0Max}$ and $\frac{V_0}{v} = u_{0Min}$ respectively.

## 3.2 RTO Method

Let's denote by

$$C_{UH} = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{y}}_{\mathbf{H}} \leq \mathbf{y}(\mathbf{u}) \leq \overline{\mathbf{y}}_{\mathbf{H}}\}$$

the polyhedron defined by the hard constraints.

At any RTO loop we proceed as follows:

**Algorithm 3.1 Blending RTO Method**

I Incorporate the new available information ($V_0$, $\mathbf{b_0}$, $B$ and $v$) in the basic equation. If $V_0 = 0$, solve the problems (3.3), (3.5) and (3.6) in this order until having a feasible solution. Otherwise, go to step II.

II Compute $V_{Min}$ ($V_{Max}$) by solving the linear programming problem:

$$\begin{aligned} &max \quad (min) \quad u_0 \\ &\text{s.t.} \\ &\mathbf{u} \in U \cap C_U \\ &u_0 \geq 0. \end{aligned} \tag{3.2}$$

If Problem (3.2) is infeasible or if $v \notin [V_{Min}, V_{Max}]$ then go to step IV.

III Compute the optimal recipe:

$$\begin{aligned} &min \quad \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} \\ &\text{s.t.} \\ &\mathbf{u} \in U \cap C_U. \end{aligned} \tag{3.3}$$

IV Forget the components costs and focus only on the hard constraints. Compute the interval $[V_{Min}, V_{Max}]$ by solving the problem:

$$\begin{aligned} & max \quad (min) \quad u_0 \\ & \text{s.t.} \\ & \mathbf{u} \in U \cap C_{UH} \\ & u_0 \geq 0. \end{aligned} \tag{3.4}$$

If this problem is infeasible or if $v \notin [V_{Min}, V_{Max}]$ then go to step VI.

V Compute the recipe $\mathbf{u} \in U \cap C_{UH}$ producing the blend with minimal quality giveaway.

$$\begin{aligned} & min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ & \text{s.t.} \\ & \mathbf{u} \in U \cap C_{UH}. \end{aligned} \tag{3.5}$$

VI Find $\mathbf{u} \in U$ to produce a volume $v$ of blend with properties as near as possible to $\mathbf{y_T}$:

$$\begin{aligned} & min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ & \text{s.t.} \\ & \mathbf{u} \in U. \end{aligned} \tag{3.6}$$

Following these steps, the RTO method always produces a recipe that guides the control process (see Figure 3.5). In step I, $V_0 = 0$ and solving the Problems (3.3), (3.5) and (3.6) we obtain the feasible recipe of minimal cost, the blend with minimal quality giveaway or the closest off-spec blend to $\mathbf{y_T}$. If BP is feasible for some volume $v \in [V_{Min}, V_{Max}]$ then the method generates the feasible blend of minimal cost (step III). Otherwise, the hard constraints become the priority and the method searches a blend satisfying them while minimizing the blend's quality giveaway (step V). Finally, if there is no blend satisfying the hard constraints, the recipe producing the blend with properties as close as possible to $\mathbf{y_T}$ is proposed (step VI).

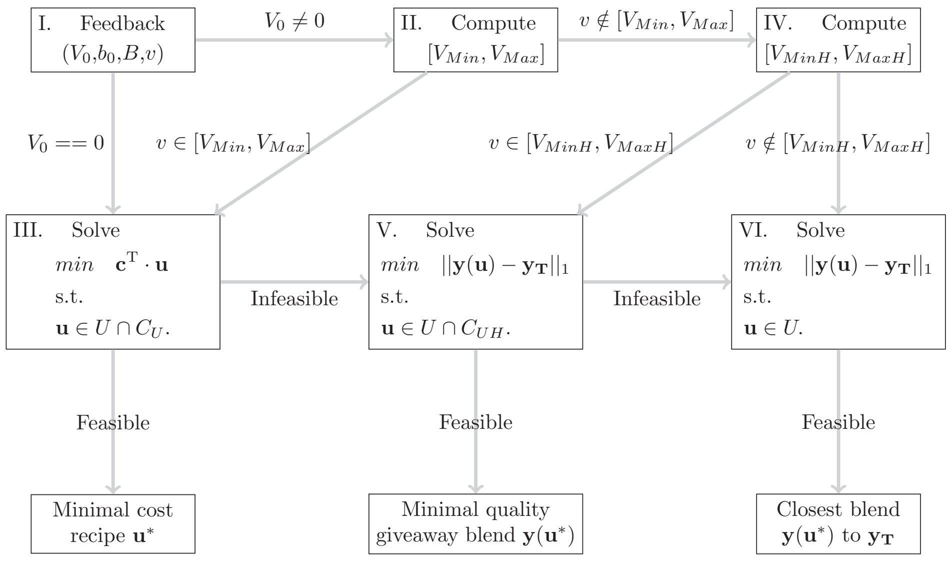
Notice that if we can let the blend's volume to be free in some interval

$$[\underline{v}, \overline{v}] \subseteq [V_{Min}, V_{Max}]$$

then solving Problem (3.3) with $v$ free and the additional constraint:

$$V_0/\overline{v} \leq u_0 \leq V_0/\underline{v}$$

we get at the same time the optimal blending horizon $v^* \in [\underline{v}, \overline{v}]$ and the optimal recipe for this volume. Similarly, in steps V and VI, we can obtain



I. Feedback ($V_0$,$b_0$,$B$,$v$)
$V_0 \neq 0$
II. Compute $[V_{Min}, V_{Max}]$
$v \notin [V_{Min}, V_{Max}]$
IV. Compute $[V_{MinH}, V_{MaxH}]$
$V_0 == 0$
$v \in [V_{Min}, V_{Max}]$
$v \in [V_{MinH}, V_{MaxH}]$
$v \notin [V_{MinH}, V_{MaxH}]$
III. Solve $min \; \mathbf{c}^T \cdot \mathbf{u}$ s.t. $\mathbf{u} \in U \cap C_U$.
Infeasible
V. Solve $min \; ||\mathbf{y}(\mathbf{u}) - \mathbf{y_T}||_1$ s.t. $\mathbf{u} \in U \cap C_{UH}$.
Infeasible
VI. Solve $min \; ||\mathbf{y}(\mathbf{u}) - \mathbf{y_T}||_1$ s.t. $\mathbf{u} \in U$.
Feasible
Feasible
Feasible
Minimal cost recipe $\mathbf{u}^*$
Minimal quality giveaway blend $\mathbf{y}(\mathbf{u}^*)$
Closest blend $\mathbf{y}(\mathbf{u}^*)$ to $\mathbf{y_T}$

Figure 3.5: RTO method.

the optimal volume $v^* \in [\underline{v}, \overline{v}]$ and the blend with minimal quality giveaway in that range. To do this, we perform a dichotomic search over $v$ solving at each step (with $v$ fixed) the Problems (3.5) and (3.6) respectively.

We finish this section by stressing the possible application of the RTO method to determine the appropriate heel's volume to use at the beginning of the blending process. In our analysis, $V_0$ is considered as fixed but we can solve the Problem 3.3 plus the constraints

$$\begin{aligned} &u_{0Min} \leq u_0 \leq u_{0Max} \\ &V_0 \geq 0, \quad v \geq 0 \end{aligned} \tag{3.7}$$

in order to obtain the cheapest recipe with $u_0^* \in [u_{0Min}, u_{0Max}]$. Then we can find (by means of the relation $u_0 = \frac{V_0}{v}$) a suitable pair of volumes $(V_0, v)$.

## 3.3 BP Uncertainty

In the previous section we proposed a blend RTO method based on the Blending Polytopes and its evolution with the blended volume. Looking to reduce the model deviations produced by some uncertainties, the method uses blend's and components' properties updates to feedback the model via the basic equation. However, model updating may fail to guarantee even a feasible solution. A main reason of this failure is the implicit assumption that data remains unchanged inside each RTO loop.

When blending from running tanks, $B's$ values fluctuate with time because the components are issued from different processes presenting also perturbations. To address this problem, [39] proposed a blending RTO method which updates the model with predictions of the components' properties. Although this method improves the model accuracy, it continues to be non robust as it depends on the quality of the predictions. Moreover, the uncertainty in the blending process affects other than $B's$ values as we will see below.

In accordance with [47], uncertainty in any RTO system may be of four types:

1. Process uncertainty: components properties, external factors (temperature, humidity, ...).
2. Measurement uncertainty.
3. Model uncertainty.
4. Market uncertainty: components availabilities and prices, blends demands, etc.

In this work we address the model uncertainty (type 3) through the RTO model update schema. We consider also the uncertainties arising from components measurement and blend's properties measurement (type 2) and the uncertainty caused by the imprecise knowledge of the components properties (type 1).

Measurement and components properties uncertainties manifest geometrically in different ways: For the first type, the real recipe (the real percentages of each component used in the blend) and its properties may differ from the nominal ones. So, the real recipe and its properties are located in neighborhoods of the nominal recipe and its properties respectively. For the second type, the real matrix $\widetilde{B}$ (the real components' properties) differs from the nominal matrix $B$. Hence the real polytope $\widetilde{C}_U$ is different from the nominal polytope $C_U$. In both cases, when a nominal feasible recipe $\mathbf{u} \in U \cap C_U$ is computed, the real recipe may fall outside the polytopes $U$ and $C_U$ whereas the real blend's properties may be outside the polytopes $P$ and $C$. Then, the real recipe results to be infeasible.

An intuitive idea to fix our RTO method against measure and $B's$ uncertainty follows from the previous geometric information. The idea consists in computing for the polytope $C_U$ its convex *robust region* $RC_U$ such that any point in this region resists to $B's$ uncertainty and to measurement errors.

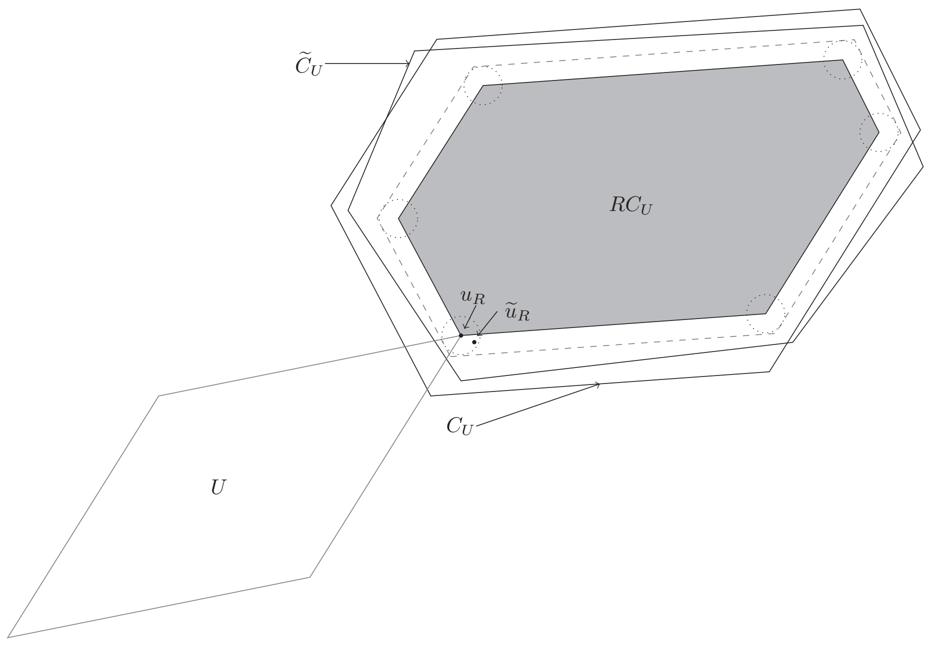


Figure 3.6: Robustope $RC_U$.

That is, any point $\mathbf{u} \in RC_U$ is guaranteed to remain inside the real polytope for all possible realizations of $B$ and any measurement error, within some reasonable limits.

In Figure 3.6, we show the nominal polytopes $U$ and $C_U$, the "robustope" $RC_U$ and the real polytope $\widetilde{C}_U$. The difference between the nominal and real polytopes $C_U$ and $\widetilde{C}_U$ results from a difference between the nominal matrix $B$ and the real matrix $\widetilde{B}$. Any point in $RC_U$ (e.g. the nominal robust recipe $\mathbf{u_R}$) transforms in a point in its neighborhood (the real robust recipe $\widetilde{\mathbf{u}}_\mathbf{R}$) because of measurement errors. The robustope $RC_U$ is computed such that any point in its neighborhood (the dotted region) remains inside the real polytope $\widetilde{C}_U$. Here we are not computing the robust region of polytope $U$ because the components constraints are considered as soft constraints. The real recipe $\widetilde{\mathbf{u}}_\mathbf{R}$ is usable even if it falls slightly outside the polytope $U$.

In this way, replacing the polytope $C_U$ by $RC_U$ in the RTO method presented in Section 3.2, we have a new robust RTO method. To develop this idea, first we need to model and measure the uncertainty we would like to be protected of and then to compute the appropriate robust region of $C_U$. This development follows the robust optimization theory (see [9]).

## 3.4 Robust Regions and Robust RTO

Let $\mathbf{u}$ be a nominal recipe and $\mathbf{y}(\mathbf{u})$ its properties. To model the components and properties measurement uncertainties, we suppose that the real recipe $\widetilde{\mathbf{u}}$ lies in the ball $S(\mathbf{u}, \delta_u)$ of radius $\delta_u$ and center $\mathbf{u}$ whereas the real blend's properties $\widetilde{\mathbf{y}}$ lies in the ball

$$S(\mathbf{y}(\mathbf{u}), \boldsymbol{\delta_y}) = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} \in [\mathbf{y}(\mathbf{u}) - \boldsymbol{\delta_y}, \mathbf{y}(\mathbf{u}) + \boldsymbol{\delta_y}]\}$$

with center $\mathbf{y}(\mathbf{u})$. Finally, to model matrix $B$ uncertainty, we use interval sets. That is, we suppose that for each $i = 1, \dots, m$ and $j = 1, \dots, n$, the real value $\widetilde{B}_{i,j}$ is comprised in the interval

$$\mathscr{I}_{i,j} = [B_{i,j} - \varepsilon^-_{i,j}, B_{i,j} + \varepsilon^+_{i,j}]$$

around its nominal value $B_{i,j}$ for some $\varepsilon^-_{i,j}, \varepsilon^+_{i,j} \in \mathbb{R}^+$.

Here we can use different sets and any norm to model the uncertainty. The level of conservatism (how much we want to be protected of uncertainty) and the problem complexity depend on these selections. Taking interval sets and the norm $L_\infty$, the robust regions obtained are polytopes and the complexity in the model is preserved at the expense of being probably too conservative (we are protected from the "worst" deviations of all $B's$ values and from the biggest measurement errors occurring all at the same time). Now we proceed to construct the robust region of polytope $C_U$.

Any point $\mathbf{u} \in C_U$ is robust regarding $B$ uncertainty iff

$$\underline{\mathbf{y}} \leq \widetilde{\mathbf{y}}(\mathbf{u}) \leq \overline{\mathbf{y}}.$$

That is, iff for any $\widetilde{B}$ such that $\widetilde{B}_{i,j} \in \mathscr{I}_{i,j}$,

$$\underline{y}_i \leq \alpha_v\, b_{0_i} + \beta_v\, \widetilde{B}_i \cdot \mathbf{u} \leq \overline{y}_i, \quad \forall i = 1, \dots, m \tag{3.8}$$

holds. Notice that any $\widetilde{B}_i \in \mathscr{I}_i$ can be expressed parametrically as

$$\widetilde{B}_i(\mathbf{z}) = B_i + \mathbf{z}^{-\mathrm{T}} Q^-_i + \mathbf{z}^{+\mathrm{T}} Q^+_i \tag{3.9}$$

with $Q^-_i = diag(\varepsilon^-_{i,1}, \dots, \varepsilon^-_{i,n})$, $Q^+_i = diag(\varepsilon^+_{i,1}, \dots, \varepsilon^+_{i,n})$, $\mathbf{z}^- = \min\{\mathbf{0}, \mathbf{z}\}$ and $\mathbf{z}^+ = \max\{\mathbf{0}, \mathbf{z}\}$ for some $\mathbf{z} \in \mathbb{R}^n$ such that $\|\mathbf{z}\|_\infty \leq 1$. Hence, replacing (3.9) in (3.8) we have that $\mathbf{u} \in C_U$ is $B$-robust iff

$$\underline{y}_i - \beta_v(\mathbf{z}^{-\mathrm{T}} Q^-_i + \mathbf{z}^{+\mathrm{T}} Q^+_i) \cdot \mathbf{u} \leq y_i(\mathbf{u}) \leq \overline{y}_i - \beta_v(\mathbf{z}^{-\mathrm{T}} Q^-_i + \mathbf{z}^{+\mathrm{T}} Q^+_i) \cdot \mathbf{u},\ \forall i = 1, \dots, m$$

is valid for all $\mathbf{z} \in \mathbb{R}^n$ such that $\|\mathbf{z}\|_\infty \leq 1$. Now, as

$$\min_{\mathbf{z}\in\mathbb{R}^n:\|\mathbf{z}\|_\infty\leq 1} \{\mathbf{z}^{-\mathrm{T}} Q_i^- + \mathbf{z}^{+\mathrm{T}} Q_i^+\} = -\boldsymbol{\varepsilon}_{\boldsymbol{i}}^-, \quad \forall i = 1, \ldots, m$$

and

$$\max_{\mathbf{z}\in\mathbb{R}^n:\|\mathbf{z}\|_\infty\leq 1} \{\mathbf{z}^{-\mathrm{T}} Q_i^- + \mathbf{z}^{+\mathrm{T}} Q_i^+\} = \boldsymbol{\varepsilon}_{\boldsymbol{i}}^+, \quad \forall i = 1, \ldots, m,$$

then we have that $\mathbf{u} \in C_U$ is $B$-robust iff

$$\underline{y}_i + \beta_v \boldsymbol{\varepsilon}_{\boldsymbol{i}}^- \cdot \mathbf{u} \leq y_i(\mathbf{u}) \leq \overline{y}_i - \beta_v \boldsymbol{\varepsilon}_{\boldsymbol{i}}^+ \cdot \mathbf{u}, \quad \forall i = 1, \ldots, m. \tag{3.10}$$

From now on we denote

$$\gamma_i^-(\mathbf{u}) = \beta_v \boldsymbol{\varepsilon}_{\boldsymbol{i}}^- \cdot \mathbf{u} \quad \text{and} \quad \gamma_i^+(\mathbf{u}) = \beta_v \boldsymbol{\varepsilon}_{\boldsymbol{i}}^+ \cdot \mathbf{u}, \quad \forall i = 1, \ldots, m.$$

In addition to $B$ uncertainty, $\mathbf{u} \in C_U$ resists also to components' measure uncertainty if and only if any point in $S(\mathbf{u}, \delta_u)$ satisfies (3.10). That is, iff

$$\underline{y}_i + \gamma_i^-(\mathbf{u}) \leq y_i(\mathbf{u}+\mathbf{z}) \leq \overline{y}_i - \gamma_i^+(\mathbf{u}), \quad \forall i = 1, \ldots, m,$$

holds for all $\mathbf{z} \in \mathbb{R}^n$ such that $\|\mathbf{z}\|_2 \leq \delta_u$. Then, noticing that

$$\min_{\mathbf{z}\in\mathbb{R}^n:\|\mathbf{z}\|_2\leq \delta_u} \{B_i \cdot \mathbf{z}\} = -\delta_u \|B_i\|_2, \quad \forall i = 1, \ldots, m$$

and

$$\max_{\mathbf{z}\in\mathbb{R}^n:\|\mathbf{z}\|_2\leq \delta_u} \{B_i \cdot \mathbf{z}\} = \delta_u \|B_i\|_2, \quad \forall i = 1, \ldots, m$$

we have that (3.4) is true iff

$$\underline{y}_i + \gamma_i^-(\mathbf{u}) + \delta_i \leq y_i(\mathbf{u}) \leq \overline{y}_i - \gamma_i^+(\mathbf{u}) - \delta_i, \quad \forall i = 1, \ldots, m \tag{3.11}$$

with $\delta_i = \beta_v \delta_u \|B_i\|_2$.

Equivalently, any recipe $\mathbf{u} \in C_U$ is robust regarding $B$ uncertainty and properties' measure uncertainty iff

$$\underline{y}_i + \gamma_i^-(\mathbf{u}) \leq y_i(\mathbf{u}) + Z_i \leq \overline{y}_i - \gamma_i^+(\mathbf{u}), \quad \forall i = 1, \ldots, m \tag{3.12}$$

is valid for all $\mathbf{Z} \in \mathbb{R}^m$ such that $|Z_i| \leq \delta_{yi}$. As previously, computing the minimum and maximum on $\mathbf{Z}$, (3.12) holds iff

$$\underline{y}_i + \gamma_i^-(\mathbf{u}) + \delta_{yi} \leq y_i(\mathbf{u}) \leq \overline{y}_i - \gamma_i^+(\mathbf{u}) - \delta_{yi}, \quad \forall i = 1, \ldots, m \tag{3.13}$$

Finally, letting $\Delta_i = \max\{\delta_i, \delta_{yi}\}$ we obtain the *robustope* $RC_U$ of polytope $C_U$:

$$RC_U = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{y}_i + \gamma_i^-(\mathbf{u}) + \Delta_i \leq y_i(\mathbf{u}) \leq \overline{y}_i - \gamma_i^+(\mathbf{u}) - \Delta_i, \quad \forall i = 1, \ldots, m\}.$$

Any $\mathbf{u} \in RC_U$ resists to $B$ uncertainty and to components and properties measurement uncertainty.

To summarize, let's consider a nominal feasible recipe $\mathbf{u} \in U \cap C_U$. If we suppose that $\widetilde{\mathbf{u}} \in S(\mathbf{u}, \delta_u)$ and $\widetilde{\mathbf{y}} \in S(\mathbf{y}(\mathbf{u}), \boldsymbol{\delta_y})$ for some $\delta_u \in \mathbb{R}^+$ and $\boldsymbol{\delta_y} \in \mathbb{R}^{m+}$ and if $\widetilde{B}_{i,j} \in \mathscr{I}_{i,j}$ for some $\varepsilon_{i,j}^-, \varepsilon_{i,j}^+ \in \mathbb{R}^+$, $\forall i = 1, \ldots, m, \quad \forall j = 1, \ldots, n$, then the real recipe $\widetilde{\mathbf{u}}$ will satisfy the real hard constraints: $\widetilde{\mathbf{u}} \in \widetilde{C}_U$.

The RTO method proposed in Section 3.2 transforms then into a robust RTO method by a simple substitution of $C_U$ with $RC_U$. This robust RTO method depends completely on the robustope $RC_U$ and to obtain it we only need to determine the values of $\varepsilon^-$, $\varepsilon^+$, $\delta_u$ and $\boldsymbol{\delta_y}$. It's worthwhile to mention that while $\delta_u$ and $\boldsymbol{\delta_y}$ are fixed values independent of the RTO loop's length, the parameters $\boldsymbol{\varepsilon}^-$ and $\varepsilon^+$ depend on it. As fluctuations on $B$ may accumulate over time, the longest the loop's length is, the biggest these fluctuations can be.

We notice that when $v$ and $V_0$ are considered as free variables in the RTO method, the value of $\beta_v$ is not determined. In this case, we need to use the identity $\beta_v = \frac{1}{1+u_0}$ and the Equations (3.11) and (3.13) to describe the polytope $RC_U$.

In the previous discussion, we limit the analysis to measurement and components properties uncertainties. However, other types of uncertainty manifest geometrically in the same way and thus can be treated identically. For instance, when the uncertainty is due to uncontrolled factors like temperature or humidity, the real blend's properties are located in a ball around the nominal blend's properties. We can include these uncertainties by taking $\boldsymbol{\delta_y}$ as the maximum radius of these balls. To model the uncertainty in the components prices, we can transform the optimization problem to one with certain objective function and such that uncertainty appears as a constraint which may be considered as a row of the matrix $B$. We can proceed similarly to deal with the uncertainty in the components availabilities. In this case the robust region $RU$ of polytope $U$ needs to be determined.

**Example 3.2 Computing a robust blend**

In the example 3.1 we determine the interval $[u_{0Min}, u_{0Max}] = [\frac{1}{6}, 1.35]$ from which we can compute all the couples $(V_0, v)$ producing a feasible blend. Let's

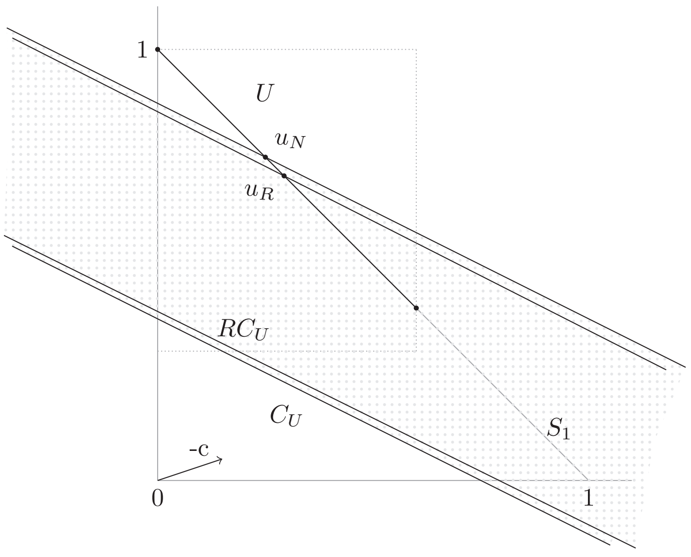


Figure 3.7: Computing a robust blend in $\mathbb{R}^2$.

suppose that we want to empty the container of the previous blend ($V_0 = 6\,m^3$) producing no more than $v = 6\,m^3$ of the new blend and let's assume that the cost of the component $u_1$ is three times the cost of component $u_2$ ($c = (3, 1)$). The resulting blending polytopes for these volumes and the optimal nominal recipe $u_N = (0.25, 0.75)$ with cost $c_N^* = 1.5$ are shown in Figure 3.7.

If we suppose that components and properties measurement errors are bounded by $\delta_u = 0.01$ and $\boldsymbol{\delta_y} = 0.01$ and that $B$ can vary in the interval $[B^-, B^+] = [[3.9, 7.9], [4.1, 8.1]]$, then the recipe $u_N$ risks to be infeasible. To generate a robust blend which resists to these uncertainties, we compute $\gamma_1^-(\mathbf{u}) = \gamma_1^+(\mathbf{u}) = 0.05u_1 + 0.05u_2$ and $\Delta_1 = \delta_1 = 0.0447$ in order to obtain the robustope

$$RC_U = \{(u_1, u_2) \in \mathbb{R}^2 \mid 0.7922 \leq u_1 + 2.02u_2, \quad u_1 + 1.97u_2 \leq 1.6855\}.$$

The optimal robust recipe is $\mathbf{u_R} = (0.2933, 0.7067)$ and has a cost of $c_R^* = 1.59$. In this case, producing the robust recipe is 5.7% more expensive than producing the nominal one.

## 3.5 Example

In this section we present an example to illustrate and compare some key aspects of the RTO method and its robust counterpart. The $BP(1)$ consists in producing a fixed volume $V_{total} = 5000\,m^3$ of blend from 8 components and $V_0 = 2000\,m^3$ of the heel's volume from a previous blend. Each component and the previous blend has 7 properties to be controlled during the process and they are represented by the $7 \times 8$ matrix $B$ and vector $\mathbf{b_0}$ respectively. Vectors $\underline{\mathbf{y}}$, $\overline{\mathbf{y}}$ and $\underline{\mathbf{u}}$, $\overline{\mathbf{u}}$ stand for the properties and components bounds respectively while vector $\mathbf{c}$ denotes the components costs.

$$B = \begin{pmatrix} 36.0000 & 36.0000 & 32.0000 & 42.0000 & 16.0000 & 31.0000 & 35.0000 & 46.0000 \\ 0.0434 & 0.0434 & 0.0262 & 0.0750 & 0.0750 & 0.1367 & 0.0567 & 0.5483 \\ 630.0000 & 620.0000 & 600.0000 & 580.0000 & 620.0000 & 600.0000 & 540.0000 & 450.0000 \\ 32.7706 & 32.7706 & 32.7706 & 16.9807 & 16.9807 & 37.7157 & 24.0770 & 8.2596 \\ 937.9460 & 937.9460 & 636.6170 & 199.0570 & 199.0570 & 170.4740 & 1381.9000 & 2.8034 \\ 0.8000 & 0.1000 & 0.0500 & 0.0400 & 1.5000 & 2.5000 & 0.0500 & 0.0050 \\ 50.0000 & 49.0000 & 50.0000 & 55.0000 & 25.0000 & 39.0000 & 41.0000 & 45.0000 \end{pmatrix}$$

$$[\mathbf{b_0}, \underline{\mathbf{y}}, \overline{\mathbf{y}}] = \begin{pmatrix} 30.0000 & 30.0000 & 46.0000 \\ 1.2327 & 0.0262 & 1.2327 \\ 640.0000 & 540.0000 & 640.0000 \\ 35.1902 & 22.8734 & 35.1902 \\ 1381.9000 & 3.2734 & 432.0940 \\ 10.2000 & 0.0000 & 10.0000 \\ 40.0000 & 40.0000 & 55.0000 \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} 87.6 & 87.6 & 87.2 & 86.0 & 83.8 & 78.5 & 87.6 & 117.1 \end{pmatrix}^T$$

$$\underline{\mathbf{u}} = \begin{pmatrix} 0.1428 & 0.0819 & 0.0352 & 0.0881 & 0.0338 & 0.0100 & 0.0352 & 0.0100 \end{pmatrix}^T$$

$$\overline{\mathbf{u}} = \begin{pmatrix} 0.4225 & 0.1930 & 0.0704 & 0.1690 & 0.2000 & 0.2000 & 0.3592 & 0.2000 \end{pmatrix}^T$$

In order to produce a robust recipe, we assume that components and properties measurement errors are bounded by $\delta_u = 0.01$ and

$$\boldsymbol{\delta_y} = \begin{pmatrix} 0.1200 & 0.0003 & 4.5000 & 0.0826 & 0.0280 & 0.000049 & 0.2000 \end{pmatrix}^T$$

respectively. Regarding $B$ uncertainty, we dispose of $B^-$ and $B^+$ the absolute lower and upper limits of matrix $B$. So, we define $\tau^- = B - B^-$, $\tau^+ = B^+ - B$ and $T = B^+ - B^-$. We suppose that $B's$ real values are comprised in the intervals $[B - \varepsilon^-, B + \varepsilon^+]$ with $\varepsilon^- = \min\{\tau^-, \theta T\}$ and $\varepsilon^+ = \min\{\tau^+, \theta T\}$ for some $\theta \in [0, 1]$. As we stated previously, the values of $\delta_u$ and $\boldsymbol{\delta_y}$ are fixed during the process whereas $\theta$ is directly related to the RTO loops' length. We take $\theta = 0.01$ for the 2-hours length RTO loops.

Solving Problem (3.2) we get

$$[u_{0MinR}, u_{0MaxR}] = [0, 0.0362].$$

The fact of having $u_{0MinR} = 0$ has a special meaning that we are going to exploit in this example. As $u_0 = \frac{V_0}{V}$, then taking $V_0 = 0$ (producing the blend from scratch) there exists a robust recipe for any volume $v > 0$. For the moment, let's consider $V_0 = 2000\,m^3$. This gives the robust feasible volumes interval:

$$[V_{MinR}, V_{MaxR}] = [55304, \infty).$$

The corresponding interval for the nominal case is:

$$[V_{Min}, V_{Max}] = [34578, \infty).$$

This means that we need to produce at least $34578\,m^3$ ($55304\,m^3$) in order to get a (robust) blend within specifications which uses completely $V_0$. For the problem with only the hard constraints, we obtain similar intervals.

If we decide to produce $V_{total} = 5000\,m^3$ using $V_0 = 2000\,m^3$, we solve Problem (3.6) and we obtain a recipe with cost 90.72 per $m^3$ but producing a blend out of specifications. Instead of this, we can compute the biggest heel's volume allowing us to produce $5000\,m^3$ of robust blend: From relations $u_{0MaxR} = \frac{V_0}{V}$ and $V_{total} = V_0 + v$ we obtain

$$V_{0MaxR} = \frac{V_{total} \times u_{0MaxR}}{1 + u_{0MaxR}} = 174.5\,m^3.$$

Next, fixing $V_0 = 174.5\,m^3$ and $v = 4825.5\,m^3$ we solve Problem (3.3) to obtain the optimal robust recipe:

$$\mathbf{u}^*_{\mathbf{R}} = \begin{pmatrix} 0.1428 & 0.0819 & 0.0352 & 0.1049 & 0.2000 & 0.2000 & 0.0352 & 0.2000 \end{pmatrix}^T$$

with cost $c^*_R = 90.72$. Incidentally, this is the same recipe that produces the blend out of specifications!

On the other hand, the optimal nominal recipe:

$$\mathbf{u}^*_{\mathbf{N}} = \begin{pmatrix} 0.1428 & 0.0819 & 0.0478 & 0.1690 & 0.2000 & 0.2000 & 0.0352 & 0.1233 \end{pmatrix}^T$$

has cost $c^*_N = 88.35$. Therefore, producing a robust recipe induces a cost increase of 2.68% over the nominal recipe's cost. This is the price of robustness. However, taking $V_0 = 0$ (no reblending), the robust recipe cost is $c^*_{R0} = 87.70$ and the nominal recipe cost is $c^*_{N0} = 87.04$. Hence, the price of robustness is only 0.76%. Moreover, we observe that both, the nominal and the robust costs have decreased. The difference $c_N^* - c_{N0}^*$ is a component of the reblending cost.

From these results we are interested in comparing the price of robustness with a component of the reblending cost (the cost difference between the recipes obtained when the heel's volume is used and when it is not). We notice that this comparison is biased against the price of robustness as the components for the blending process are selected, obviously, according to the properties of the previous blend. Thus, trying to produce a blend from the same components without using the previous blend, may result more expensive or even infeasible. In a fair comparison, we should select the best available components to produce the blend from scratch. We do not have the elements to do this, but in order to provide a comparison as fair as possible, the price of robustness is obtained by taking $V_0 = 0$ (no reblending involved) and the reblending cost from the nominal recipes (no robustness involved).

To compute the price of robustness we conduct a blending simulation over 36 RTO loops of 2-hours length. In Table (3.1) we show the average recipe's cost over the 36 periods and the percentage cost increase ($\Delta_c$) from the nominal recipe to the robust recipe for different $\theta$ values and taking $V_0 = 0$.

Then we generate the optimal nominal recipes when $V_0 = 168.44$ ($V_{0Max}$

| $\theta$ | $c_R^*$ | $c_N^*$ | $\Delta_c(\%)$ |
|---|---|---|---|
| 0.01 | 87.3851 | 87.1097 | 0.32 |
| 0.02 | 87.3542 | 87.0763 | 0.32 |
| 0.03 | 87.3847 | 87.1036 | 0.32 |
| 0.04 | 87.3632 | 87.0795 | 0.32 |
| 0.05 | 87.4058 | 87.1194 | 0.33 |
| 0.06 | 87.4134 | 87.1249 | 0.33 |
| 0.07 | 87.4232 | 87.1309 | 0.33 |
| 0.08 | 87.4258 | 87.1306 | 0.34 |
| 0.09 | 87.4297 | 87.1307 | 0.34 |
| 0.10 | 87.4077 | 87.1062 | 0.35 |

Table 3.1: $BP(1)$: Relative price of robustness.

for $\theta = 0.1$) and $V_0 = 0$ are used in the production of $5000\,m^3$ of blend. The costs of these recipes are 88.29 and 87.04 respectively. We observe that for this case, the nominal recipe's cost with reblending is $(88.29 - 87.04)/87.04 = 1.43\%$ greater than the nominal recipe's cost without reblending whereas the relative price of robustness is of only 0.35% for a significant $\theta$ value of 0.1.

In Tables (3.2) to (3.5), we show the relative price of robustness $\Delta_c$ for a set of BP instances considering all of them the use of a previous blend. In the following, we comment these results.

First, we notice that the price of robustness for the $BP(4)$ is zero for all the $\theta$ values considered. This is explained by recalling that the feasible region of the robust BP is $U \cap RC_U \subseteq U \cap C_U$. Thus, if the optimal robust solution to the BP is a vertex of $U$, then it is also the optimal solution for the nominal BP. For $BP(2)$ and $BP(4)$, the price of robustness is less than 1% whereas for $BP(3)$ it reaches 4.53% for the highest level of conservatism.

On the other hand, fixing $V_0 = 0$, the only feasible instance is $BP(5)$. For the BP instances 2 to 4, it is impossible to produce the blend from scratch using the same components that were used to reblend. The optimal recipe's cost for $BP(5)$ is 86.9480 which is 0.14% more expensive than the optimal nominal blend using the heel's volume.

As we have already noticed, using the same set of components to produce the blend from scratch and from a previous blend is not the appropiate way to compute the reblending cost. However, considering that the comparison scheme is biased against the price of robustness and that we are only comparing a component of the reblending cost, we think that the results obtained from these comparisons comfort the idea that reblending might be much more expensive than producing a robust blend even if the uncertainty is modeled by interval sets, which may be too conservative.

Finally, we highlight the importance of computing the reblending cost using the best available components to produce independently the blend from scratch and using heel's volume. This value can be used to adjust the level of conservatism in the model via the parameters $\delta_u$, $\delta_y$ and $\theta$. In the absence of this value, we can use the price of robustness as a benchmark for the reblending cost. The price of robustness is the highest value to pay for reblending.

After these considerations, we can conclude that for $BP(1)$ and $BP(4)$, the cost of producing the robust blend is lower than the reblending cost. For the BP instances 2, 3 and 5 we should produce the robust recipe unless its reblending costs are lower than 0.77%, 4.53% and 0.1% respectively.

| $\theta$ | $c_R^*$ | $c_N^*$ | $\Delta_c(\%)$ |
|---|---|---|---|
| 0.01 | 86.0184 | 85.8682 | 0.17 |
| 0.02 | 86.0567 | 85.8682 | 0.22 |
| 0.03 | 86.1158 | 85.8682 | 0.29 |
| 0.04 | 86.1749 | 85.8682 | 0.36 |
| 0.05 | 86.2342 | 85.8682 | 0.43 |
| 0.06 | 86.2935 | 85.8682 | 0.50 |
| 0.07 | 86.3529 | 85.8682 | 0.56 |
| 0.08 | 86.4124 | 85.8682 | 0.63 |
| 0.09 | 86.4720 | 85.8682 | 0.70 |
| 0.10 | 86.5317 | 85.8682 | 0.77 |

Table 3.2: $BP(2)$: Relative price of robustness.

| $\theta$ | $c_R^*$ | $c_N^*$ | $\Delta_c(\%)$ |
|---|---|---|---|
| 0.01 | 84.8437 | 82.2032 | 3.21 |
| 0.02 | 84.9591 | 82.2032 | 3.35 |
| 0.03 | 85.0749 | 82.2032 | 3.49 |
| 0.04 | 85.1911 | 82.2032 | 3.63 |
| 0.05 | 85.3079 | 82.2032 | 3.78 |
| 0.06 | 85.4251 | 82.2032 | 3.92 |
| 0.07 | 85.5427 | 82.2032 | 4.06 |
| 0.08 | 85.6609 | 82.2032 | 4.21 |
| 0.09 | 85.7916 | 82.2032 | 4.37 |
| 0.1 | 85.9239 | 82.2032 | 4.53 |

Table 3.3: $BP(3)$: Relative price of robustness.

| $\theta$ | $c_R^*$ | $c_N^*$ | $\Delta_c(\%)$ |
|---|---|---|---|
| 0.01 | 87.2632 | 87.2632 | 0 |
| 0.02 | 87.2632 | 87.2632 | 0 |
| 0.03 | 87.2632 | 87.2632 | 0 |
| 0.04 | 87.2632 | 87.2632 | 0 |
| 0.05 | 87.2632 | 87.2632 | 0 |
| 0.06 | 87.2632 | 87.2632 | 0 |
| 0.07 | 87.2632 | 87.2632 | 0 |
| 0.08 | 87.2632 | 87.2632 | 0 |
| 0.09 | 87.2632 | 87.2632 | 0 |
| 0.1 | 87.2632 | 87.2632 | 0 |

Table 3.4: $BP(4)$: Relative price of robustness.

| $\theta$ | $c_R^*$ | $c_N^*$ | $\Delta_c(\%)$ |
|---|---|---|---|
| 0.01 | 86.8841 | 86.8267 | 0.07 |
| 0.02 | 86.8872 | 86.8267 | 0.07 |
| 0.03 | 86.8903 | 86.8267 | 0.07 |
| 0.04 | 86.8933 | 86.8267 | 0.08 |
| 0.05 | 86.8964 | 86.8267 | 0.08 |
| 0.06 | 86.8995 | 86.8267 | 0.08 |
| 0.07 | 86.9025 | 86.8267 | 0.09 |
| 0.08 | 86.9067 | 86.8267 | 0.09 |
| 0.09 | 86.9113 | 86.8267 | 0.1 |
| 0.1 | 86.9159 | 86.8267 | 0.1 |

Table 3.5: $BP(5)$: Relative price of robustness.

Chapter 4

# Blending Problem Infeasibility

A BP instance $(B, v, \mathbf{b_0}, V_0, c, \underline{\mathbf{u}}, \overline{\mathbf{u}}, \underline{\mathbf{y}},\ \overline{\mathbf{y}})$ is determined by the components $B$, the volume $v$, the heel $\mathbf{b_0}$ and its volume $V_0$, the components cost $\mathbf{c}$, the components' availability and hydraulic bounds $\underline{\mathbf{u}}$ and $\overline{\mathbf{u}}$ and by the properties bounds $\underline{\mathbf{y}}$ and $\overline{\mathbf{y}}$. At each loop, the RTO method presented in Section 3.2 solves the BP in order to produce, if possible, the minimal cost recipe. If the BP is infeasible, some bounds are relaxed and a feasible blend with minimal quality giveaway is searched. The method considers the possible modification of the BP instance to achieve better results. Specifically, the volumes $V_0$ and $v$ are used as control variables in the model in order to produce a more economical recipe or a feasible one when the original values of $V_0$ and $v$ do not allow it. In some cases, even with this extra degree of freedom, the BP remains infeasible for the required volumes $V_0$ and $v$. Then, in order to assure the continuity of the operation, the method proposes to produce the closest infeasible blend to the "ideal" target blend $y_T$.

In this chapter, we are interested in measuring and visualizing the BP infeasibility and in using other parameters of the BP instance to control the blending process. Besides the volumes $v$ and $V_0$, the components' availability, the hydraulic capacities or even the components itself can be used to drive the Blending Process. Here, we consider only components' availability and hydraulic bounds modifications.

The organization of this chapter is as follows: First, we review the main features of the infeasibility analysis of linear systems in Section 4.1. Then, in Section 4.2 we classify the BP infeasibility and we state the necessary conditions under which bounds modifications can produce a feasible BP instance. In Section 4.3 we measure the BP infeasibility in terms of the blending polytopes and we highlight the existence of two vectors related to the infeasible BP. These vectors will be exploited in Sections 4.4 and 4.5 to construct 2D-visualizations of the BP infeasibility. Later, in Section 4.6 we design customized procedures to modify the components' bounds and transform an infeasible BP instance into a feasible one according with the optimization criteria used in the RTO method. We observe also that replacing the Blending Polytopes by their corresponding robustopes these procedures can be directly applied to produce a robust blend. In Section 4.7 we use the bounds' modifications procedure to drive the blending process towards the blend within specifications requiring to pour the minimum blends' volume. This analysis leads us to investigate the impact of the components bounds on the blending process. We propose then a set of indices to measure the impact of the components bounds on the blends' qualities and we accompany these measures with a supportive visualization. An example is given in Section 4.8.

## 4.1 Measuring and Analyzing the BP Infeasibility

In the industrial applications, when a linear programming problem results to be infeasible, the decision maker faces the problem of finding a way to transform the infeasible model into a feasible one. The infeasibility of the problem can be due to a bad formulation (e.g., a model not reflecting the real problem or a model not well-posed for its numerical solution by a certain method). Most linear solvers scale the constraints (see below) in a preprocessing stage to improve the characteristics of the model and make it best suited for its solution. However, the infeasibility can be also the result of a set of incompatible requirements in the model which, if seen individually, are completely pertinent. The detection of these incompatibilities and the study of the required modifications to repair the problem instance is known as *Infeasibility Analysis* (see ([14]) for a recent review of this subject). The potential risk of inconsistencies and the complexity to detect them increases with the problem size. The linear programming models arising in industrial applications today are increasingly complex and can have up to millions of constraints. For this reason, it is becoming more necessary to have automated methods to detect and handle the infeasibility.

A different reason for developing these methods, which particularly concern us, is the widespread use of linear solvers in the kernel of a RTO system. In this situation, the optimizer should propose, in real time, a solution that guarantees the continuity of the operation even if the problem is infeasible. A method to identify the causes of infeasibility and to propose how to repair the model will be very useful to drive the process towards a tailored feasible solution.

The most usual method to measure the infeasibility of a point $\mathbf{x}$ violating an individual constraint $A_i\mathbf{x} \leq b_i$ is the difference $A_i\mathbf{x} - b_i$. This method extends to measure the infeasibility of a point over a set of constraints in two main ways:

- The sum of the infeasibilities over all the constraints.

- The number of violated constraints.

In practice, a constraint is considered as violated if the violation exceeds a tolerance $\tau$. That is, if $A_i\mathbf{x} - b_i > \tau$. This test is very easy to compute but it is very sensitive to scaling factors as we can see in the following example. Let's consider the constraint $x \leq 1$ evaluated at $x = 1 + \frac{\tau}{2}$. Numerically, this constraint is considered as satisfied. However, the same constraint scaled by 10 ($10x \leq 10$) and evaluated at the same point produces a violation of the same order than the scaling factor: $10 + 10 \times \frac{\tau}{2} - 10 - \tau = 4\tau$.

To avoid this problem, one can scale the constraints to ensure all the coefficients to have approximately the same magnitude before measuring the infeasibility. A different approach, exempt from scaling issues, consist in measuring the infeasibility as the Euclidean distance between $\mathbf{x}$ and the closest point in the feasible region defined by the set of constraints. In Section 4.2 we will use this idea to measure the infeasibility of the whole BP.

Among the tools developed to detect and repair the infeasibility there are three main approaches ([14]):

- Identification of an *Irreducible Infeasible Subset* (IIS) of constraints.
- Identification of a *Maximum Feasible Subset* (Max FS) of constraints.
- *Best Repair* proposal of the model.

An IIS has the property of being infeasible but becomes feasible if any single constraint is removed. Detection of IIS allows the decision maker to concentrate in small subsets of inconsistencies which are easier to understand. Notice that an infeasible model can have several IIS's, then repairing an IIS may not repair the whole system. A brute-force method to repair the whole system consists in detecting and repairing iteratively an IIS until feasibility. The identification of a Max FS of constraints can also be viewed as the detection of the minimum number of constraints whose removal leaves a feasible system. This is the *minimum unsatisfied linear relation* problem (MIN ULR) which is known to be NP-Hard (see [12]). Both approaches, IIS and Max FS detection have a similar aim, to reduce the scope of the problem and let the decision maker to use its expertise to repair the model. This intervention turn them in semi-automated methods. An empirical analysis of three methods (including IIS) to identify and isolate the infeasibility in a Blending Problem was conducted in 1992 ([25]). The conclusions of the study points out the superiority of the IIS method over the others, but it also highlights the difficulty to release useful information from the isolation of the infeasibility.

The best repair proposal approach differs, from the later two, in this sense. It focuses on determining algorithmically the best modification to the problem' constraints with respect to a predefined criterion. Among the methods of this type there are those that only modify the right hand side vector (RHS Methods) and those looking the feasibility of the problem by modifying all the coefficients of the constraints. The last problem, much more difficult than the former one is usually addressed by minimizing the norm of a correction matrix $H$ (see [4]). Somme common choices for the matrix norm are $L_1$ and $L_\infty$ and the $\infty$-norm : $\|H\|_\infty = \max_i \sum_j |h_{ij}|$. In ([4]) the Frobenius norm is used. The optimal matrix correction (according to some criterion) gives a relatively similar feasible system of constraints, although it is not always clear how to use this information. Here, we will focus on the RHS Methods.

## RHS Methods

In [33], the authors distinguish four different criteria to transform the infeasible system $A\mathbf{x} \leq \mathbf{b}$ onto a feasible one using only RHS's modifications:

*The Smallest Changes Model.* This model seeks to modify the minimum number of RHS terms (equivalent to the MIN ULR problem).

*The Smallest Penalty Model.* If the penalty cost $f_i$ of changing the RHS $b_i$ is fixed for all $i$ (does not depend on the modification's magnitude), this method seeks to change the set of $b_i$'s making the problem feasible while minimizing the associated sum of penalties.

*The Smallest Variable Cost Model.* If the penalty cost depends on the amount of each change, this model seeks to minimize the total variable cost of all the changes to the $b_i$'s which makes the problem feasible. This can be written as the following linear program:

$$\begin{aligned}
&min \quad \sum_{i=1}^{m} \left(c_i^+ e_i^+ + c_i^- e_i^-\right) + \sum_{i=m+1}^{m+p} c_i^- e_i^- \\
&s.t. \\
&\qquad A_i\mathbf{x} + e_i^- - e_i^+ = b_i, \qquad i = 1, \dots, m \\
&\qquad A_i\mathbf{x} + e_i^- \geq b_i, \qquad i = m+1, \dots, m+p \\
&\qquad e_i^-, e_i^+ \geq 0, \qquad i = 1, \dots, m+p
\end{aligned}$$

where $c_i^+$ ($c_i^-$) denote the non-negative cost per unit increase (decrease) in the value of $b_i$. The variables $e_i^+$ and $e_i^-$ are called *elastic variables* as they let the constraints to "stretch" to make the feasible region nonempty. Notice that for the $b_i$'s associated to the inequality constraints, we only need the elastic variables increasing the feasible region. So, we let $e_i^+ = 0$ for $i = m+1, \dots, m+p$. If $(\mathbf{x}^*, e_i^{-*}, e_i^{+*})$ is an optimal solution of this problen, then $b_i' = b_i - e_i^- + e_i^+$ for all $i$ is the optimal modification of $b$ under this model and $\mathbf{x}^*$ is a feasible solution of the modified problem.

*The Smallest Variable Cost Model with Bounds.* This is the same as the previous model, except that bounds are imposed on the changes. Then, letting $u_i^+ \geq 0$ ($u_i^- \geq 0$) be the maximum possible increase (decrease) on $b_i$ for all $i$, we have:

$$\begin{array}{lrl} min & \multicolumn{2}{l}{\sum_{i=1}^{m} \left(c_i^+ e_i^+ + c_i^- e_i^-\right) + \sum_{i=m+1}^{m+p} c_i^- e_i^-} \\ s.t. & & \\ & A_i\mathbf{x} + e_i^- - e_i^+ = b_i, & i = 1, \dots, m \\ & A_i\mathbf{x} + e_i^- \geq b_i, & i = m+1, \dots, m+p \\ & 0 \leq e_i^- \leq u_i^-, \quad 0 \leq e_i^+ \leq u_i^+, & i = 1, \dots, m+p. \end{array}$$

If some $b_i$ cannot be increased, decreased or modified at all, we fix $u_i^+ = 0$, $u_i^- = 0$ or $u_i^+ = u_i^- = 0$.

Other different measures of the required adjustment to the RHS can be considered, e.g., $\min\|(\mathbf{b}-A\mathbf{x})^+\|_1$ or $\min \sum_{i=1}^m [(b_i - A_i\mathbf{x})^+]^2$ with $(\cdot)^+ = \max\{0, (\cdot)\}$. All these methods have different desirable characteristics that distinguish them (robustness, ease of calculation, etc.). However, the "best" way to repair an infeasible problem depends on the particular situation.

## 4.2 Infeasibility Levels

Let's define the polytope

$$C_{U1} = C_U \cap S_1$$

in components space and the polytope

$$C_{Y1} = \{\mathbf{y}(\mathbf{u}) \in \mathbb{R}^m \mid \mathbf{u} \in C_{U1}\}$$

in properties space.

If the BP results to be infeasible for a fixed pair of volumes $(V_0, v)$ and assuming that $U \neq \emptyset$, then one may have 3 different levels of infeasibility which can be stated in terms of the blending polytopes:

Level 1. $C_{U1} \neq \emptyset$ and $U \cap C_{U1} = \emptyset$.

Level 2. $C_U \neq \emptyset$ and $C_{U1} = \emptyset$.

Level 3. $C_U = \emptyset$.

For the three levels of infeasibility, we can modify the components properties (the matrix $B$) and force the polytope $C_U$ to have a non empty intersection with the polytope $U$ producing in this way a feasible blend. On the other hand, as the simplex $S_1$ contains all the possible recipes, then $C_{U1}$ represents the set of all the recipes having their properties $(C_{Y1})$ within specifications. A polytope $U$ can reach, by bounds modifications, a point $\mathbf{u} \in C_U \setminus U$ iff $\mathbf{u} \in S_1$. That is, iff the BP has infeasibility of level 1.

In Figure 4.1 we show the three levels of infeasibility for a BP instance

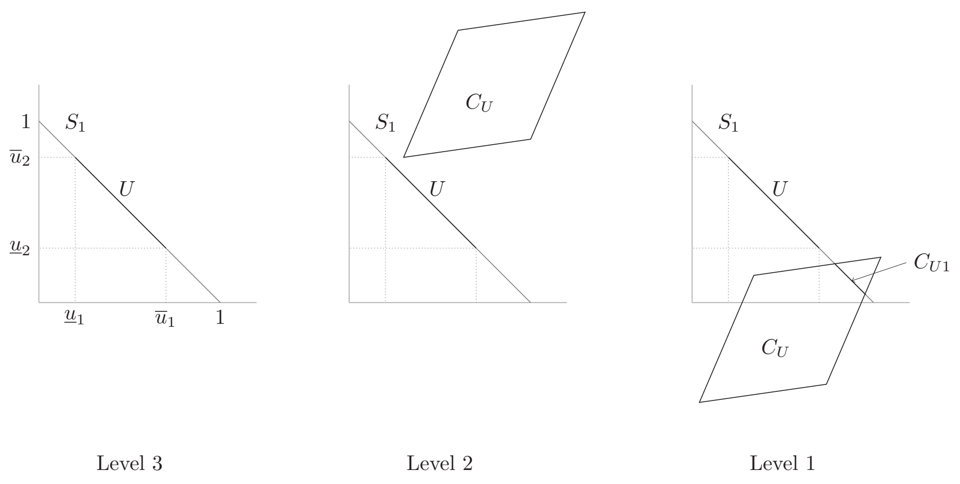


Figure 4.1: Infeasibility levels.

having two components and two properties. When the infeasibility is of level 1, we observe that modifying $\overline{u}_1$ and $\underline{u}_2$ the polytope $U$ expands and we can reach any recipe $\mathbf{u} \in C_{U1}$. In contrast, having infeasibility of levels 2 and 3 we need to change the components $(B)$ in order to get a non empty $C_U$ touching the polytope $U$. In this case, it is the polytope $C_U$ which is modified while $U$ stays unchanged.

For each infeasibility level, the infeasibility of a recipe $\mathbf{u} \in U$ can be measured in both spaces as the Euclidean distance between $\mathbf{u}$ and the closest point in the corresponding polytope. In the components space, for levels 1 and 2, as the distance between $\mathbf{u}$ and the closest point in $C_{U1}$ and $C_U$ respectively. In the properties space, for the three levels of infeasibility, as the distance between $\mathbf{y}(\mathbf{u})$ and the closest point in $C_{Y1}$, $C_Y$ and $C$.

In a natural way, we can define then the *BP infeasibility* in terms of the blending polytopes: the BP infeasibility of level 1 is defined by

$$D_U = D(U, C_{U1}) \quad \text{and} \quad D_Y = D(P, C_{Y1})$$

the infeasibility of level 2 by

$$D_{U2} = D(U, C_U) \quad \text{and} \quad D_{Y2} = D(P, C_Y)$$

and the infeasibility of level 3 by

$$D_{Y3} = D(P, C).$$

These distances give us relevant information for the analysis of infeasibility, although of different kind: In the components space, $D_U$ and $D_{U2}$ are estimates of the required "effort" to obtain a feasible BP instance by hydraulic and availability bounds modifications and by components modifications respectively. These distances are only an estimate because, as we will see later, the required effort to transform the instance depends on the type of transformation applied and on the way to measure the effort. In the properties space, $D_Y$, $D_{Y2}$ and $D_{Y3}$ measure the blend's quality losse that can be saved by components bounds modifications and/or by components modifications.

## 4.3 Gap Computation and Feasible Directions

Let's define a *feasible direction* in component's space as a translation direction of polytopes $U$ or $C_U$ producing its collision and therefore the BP feasibility. A translation direction of polytopes $P$ or $C_Y$ producing their collision is called also a feasible direction (in properties space) as it can be seen as the image under the basic equation of a feasible direction in components space.

As we have stated early, here we are considering only bounds modifications to transform an infeasible BP instance into a feasible one. In the following, we will only develop the analysis for the infeasibility of level 1. However, we notice that we can measure and construct the visualizations (when they exist) for the infeasibility of levels 2 and 3. We will highlight the differences in the procedure explained below.

In components space we are interested in computing the gap between polytopes $U$ and $C_{U1}$. Both polytopes are described in their dual form. So, as discussed in Section 2.2, we can compute this distance by solving:

$$\begin{aligned} &min \quad \|\mathbf{t} - \mathbf{u}\|_2^2 \\ &\text{s.t.} \\ &\mathbf{u} \in U, \ \mathbf{t} \in C_{U1}. \end{aligned} \tag{4.1}$$

If $\mathbf{u}^* \in U$ and $\mathbf{t}^* \in C_{U1}$ is a solution to problem 4.1, then $D_U = d(\mathbf{u}^*, \mathbf{t}^*)$ and

$$\mathbf{w_U} = \frac{\mathbf{t}^* - \mathbf{u}^*}{D_U} \tag{4.2}$$

is a feasible direction in components space which expresses the minimal effort moving direction of $U$ in order to meet $C_{U1}$.

We observe that $\mathbf{p}^* = \mathbf{y}(\mathbf{u}^*) \in P$ and $\mathbf{q}^* = \mathbf{y}(\mathbf{t}^*) \in C_{Y1}$. Thus,

$$\mathbf{w}_{\mathbf{Y}}^{\mathbf{U}} = \frac{\mathbf{q}^* - \mathbf{p}^*}{\|\mathbf{q}^* - \mathbf{p}^*\|_2}$$

is a feasible direction in properties space.

To compute the distance between polytopes $P$ and $C_{Y1}$, we should notice that for these polytopes we don't have neither the primal nor the dual representations. However, we can compute their distance indirectly from polytopes $U$ and $C_{U1}$ and the basic equation. Solving the problem

$$\begin{aligned}&min \quad \|\mathbf{y}(\mathbf{t}) - \mathbf{y}(\mathbf{u})\|_1\\ &\text{s.t.}\\ &\mathbf{u} \in U,\ \ \mathbf{t} \in C_{U1}\end{aligned}$$

we obtain $\mathbf{p}^* = \mathbf{y}(\mathbf{u}^*) \in P$ and $\mathbf{q}^* = \mathbf{y}(\mathbf{t}^*) \in C_{Y1}$ such that $D_Y = d(\mathbf{p}^*, \mathbf{q}^*)$. Then

$$\mathbf{w}_{\mathbf{Y}} = \frac{\mathbf{q}^* - \mathbf{p}^*}{D_Y} \tag{4.3}$$

is the shortest moving direction of $P$ in order to meet $C_{Y1}$ and

$$\mathbf{w}_{\mathbf{U}}^{\mathbf{Y}} = \frac{\mathbf{t}^* - \mathbf{u}^*}{\|\mathbf{t}^* - \mathbf{u}^*\|_2}$$

is the corresponding feasible direction in components space.

The distances $D_{U2}$ and $D_{Y2}$ can be obtained in the same way: The polytope $C_{U2}$ is given in its dual form and $C_{Y2}$ is the image of $C_{U2}$ under the basic equation. In contrast, the polytope $C$ has no preimage when the infeasibility is of level 3. Hence, $D_{Y3}$ should be computed directly in the properties space. The vertices of $C$ are easy to obtain and in Section 4.5, we will show how to get those ones of $P$. Then, computing the separating hyperplane of largest gap between the polytopes $P$ and $C$ in their primal form (see Section 2.2), we have $D_{Y3}$.

In this section we highlight the existence of the BP inherent feasible directions:

(i) $\mathbf{w}_{\mathbf{U}}$ and $\mathbf{w}_{\mathbf{U}}^{\mathbf{Y}}$ in components space and

(ii) $\mathbf{w}_{\mathbf{Y}}$ and $\mathbf{w}_{\mathbf{Y}}^{\mathbf{U}}$ in properties space.

Next we are going to exploit these directions to obtain 2D-visualizations of the BP infeasability in each space.

## 4.4 Infeasibility Visualization in Components Space

Let $\mathbf{w_1} = \mathbf{w_U}$ and $\mathbf{w_2} = \mathbf{w_U^Y}$. These vectors define a 2D-plane in the components space whose parametric equation is

$$H_2 = \{\mathbf{u(s)} \mid \mathbf{u(s)} = \mathbf{u_M} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s}\}$$

with $\mathbf{s} = (s_1, s_2)^{\mathrm{T}} \in \mathbb{R}^2$, $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2})^{\mathrm{T}}$ and $\mathbf{u_M}$ the midpoint of the segment $[\mathbf{u}^*, \mathbf{t}^*]$ (see Equation (4.2)).

Let's denote by $U_2$ and $C_2$ the cuts of $U$ and $C_{U1}$ with $H_2$. The polygone $U_2$ is then described by the set of points $\mathbf{u(s)} \in H_2$ verifying:

$$\mathbf{1}^{\mathrm{T}} \cdot \mathbf{u(s)} = 1 \tag{4.4}$$

$$\underline{\mathbf{u}} \leq \mathbf{u(s)} \leq \overline{\mathbf{u}}. \tag{4.5}$$

Equation (4.4) is satisfied by all $\mathbf{u(s)} \in H_2$ since $\mathbf{u_M} \in S_1$ and $\mathbf{1}^{\mathrm{T}} \cdot \mathbf{w} = (0, 0)$. Thus the vertex set of $U_2$ can be calculated "explicitly" because it is the intersection of the set of lines in $H_2$ described by (4.5). In the same manner, we can find the vertices of the polygone

$$C_2 = \{\mathbf{u(s)} \in H_2 \mid \underline{\mathbf{y}} \leq \mathbf{y}(\mathbf{u(s)}) \leq \overline{\mathbf{y}},\ \mathbf{u(s)} \in S_1\}$$

as the intersection of a set of lines in $H_2$. We know that

$$\mathbf{1}^{\mathrm{T}} \cdot \mathbf{u(s)} = 1, \quad \forall \mathbf{u(s)} \in H_2,$$

therefore $C_2$ is completely characterized by the set of points $\mathbf{u(s)} \in H_2$ such that:

$$\mathbf{0} \leq \mathbf{u(s)} \leq \mathbf{1} \tag{4.6}$$

$$\underline{\mathbf{y}} \leq \alpha_v\ \mathbf{b_0} + \beta_v\ B\mathbf{u(s)} \leq \overline{\mathbf{y}}. \tag{4.7}$$

Finally, replacing $\mathbf{u(s)}$ in (4.6) and (4.7) we get the set of lines in $H_2$ whose intersection determines the vertex set of $C_2$:

$$-B\mathbf{u_M} \leq \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s} \leq \mathbf{1} - B\mathbf{u_M}$$

$$\frac{\underline{\mathbf{y}} - \alpha_v \mathbf{b_0}}{\beta_v} - B\mathbf{u_M} \leq (Bw)^{\mathrm{T}} \cdot \mathbf{s} \leq \frac{\overline{\mathbf{y}} - \alpha_v \mathbf{b_0}}{\beta_v} - B\mathbf{u_M}.$$

To obtain the infeasibility visualization in components space (see Figure 4.2), we simply draw these polygones in the plane $H_2$ with $\mathbf{u_M}$ taken as the center of coordinates and vectors $\mathbf{w_1}$ and $\mathbf{w_2}$ as the axis. Notice that these axis are not orthogonal in general.

It is important to remark the characteristics of this visualization. As we

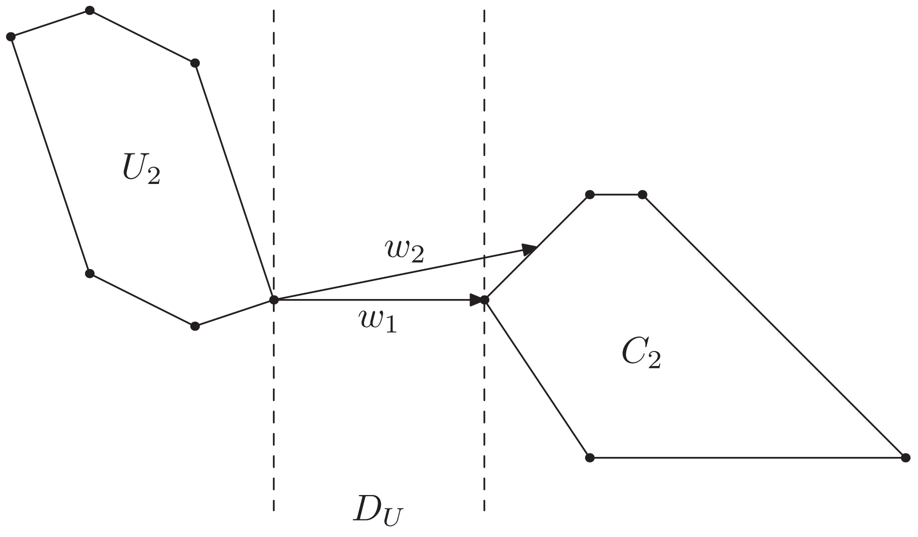


Figure 4.2: Infeasibility visualization in components space.

obtain it by cutting polytopes $U$ and $C_{U1}$ with a plane defined by $\mathbf{w_U}$ and $\mathbf{w_U^Y}$ which passes through the nearest points in both polytopes, the distance between the polygones $U_2$ and $C_2$ is $D_U$. The distance between the polytopes $U$ and $C_{U1}$ in the original $n$-space. By the same reason, the angle between directions $\mathbf{w_U}$ and $\mathbf{w_U^Y}$ is preserved.

## 4.5 Infeasibility Visualization in Properties Space

Following the same lines of the preceding section, let's consider the feasible directions in the properties space $\mathbf{w_1} = \mathbf{w_Y}$ and $\mathbf{w_2} = \mathbf{w_Y^U}$ and let's denote by $H_2$ the 2D-plane defined by

$$H_2 = \{\mathbf{y}(\mathbf{s}) \in \mathbb{R}^2 \mid \mathbf{y}(\mathbf{s}) = \mathbf{y_M} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s}\}$$

where $\mathbf{s} = (s_1, s_2)^T \in \mathbb{R}^2$, $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2})$ and $\mathbf{y_M}$ is the midpoint of the segment $[\mathbf{p}^*, \mathbf{q}^*]$ (see Equation (4.3)). We can reuse the same development of the previous section to obtain the polygons $P_2$ and $C_2$ by cutting $P$ and $C_{Y1}$ with $H_2$ on condition to have the dual description of $P$ and $C_{Y1}$. We don't have neither the primal nor the dual description of these polytopes but they can be obtained (at least theoretically) as follows.

Let $V_U$ and $V_P$ be the vertex sets of polytopes $U$ and $P$. First, we observe that $V_P$ is contained in the image of $V_U$ under the basic equation as stated in the next

**Property 4.5.1.** *If $V_P$ and $V_U$ are the vertex sets of polytopes $P$ and $U$ respectively then $V_P \subseteq \{\mathbf{y}(\mathbf{u}) \mid \mathbf{u} \in V_U\}$.*

**Proof.** Let $\mathbf{y}(\mathbf{u}) \in V_P$ and suppose $\mathbf{u} \notin V_U$. Then $\mathbf{u}$ is a convex combination of $V_U$ with at least two coefficients strictly positive:

$$\mathbf{u} = \sum_{u_i \in V_U} \lambda_i u_i$$

with

$$\sum_{u_i \in V_U} \lambda_i = 1, \quad \lambda_i \geq 0 \quad \text{and} \quad \lambda_1, \lambda_2 > 0.$$

Applying the basic equation to $\mathbf{u}$ we have

$$\mathbf{y}(\mathbf{u}) = \alpha_v\ \mathbf{b_0} + \beta_v\ B\mathbf{u} = \sum_{u_i \in V_U} \lambda_i(\alpha_v\ \mathbf{b_0} + \beta_v\ B\mathbf{u_i})$$

with

$$\sum_{u_i \in V_U} \lambda_i = 1, \quad \lambda_i \geq 0 \quad \text{and} \quad \lambda_1, \lambda_2 > 0.$$

So, $\mathbf{y}(\mathbf{u})$ is a convex combination of vectors in $P$ which contradicts our assumption. □

Then, we obtain the primal and dual representations of $P$ from the dual of $U$ in three steps:

(I) Compute $V_U = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_N}\}$ and then $V = \{\mathbf{v_i} \mid \mathbf{v_i} = \mathbf{y}(\mathbf{u_i})\ ,\ i = 1, \ldots, N\}$.

(II) Delete from $V$ iteratively each $\mathbf{v_i}$ which can be expressed as a convex combination of $V \setminus \mathbf{v_i}$ to obtain $V_P$.

(III) Solve the facet enumeration problem for the set $V_P$.

We can proceed in the same way to construct the primal and dual forms of $C_{Y1}$.

In Section 2.1 we state that vertex and facet enumeration are difficult problems for general polytopes. If we want to reuse the last section procedure to produce the infeasibility visualization in properties space, we need to solve both problems. In practice, the vertex enumeration for polytopes $U$ and $C_{U1}$

can be done efficiently. We can partially explain this situation by noticing that polytope $U$ has a particularly simple structure that can be exploited by a customized algorithm. It is the intersection of a parallelepiped with the simplex $S_1$ and all its vertices are either vertices of the parallepiped or they lie in the interior of its edges (see Appendix A).

On the contrary, the facets of $P$ and $C_{Y1}$ can not always be obtained efficiently. Occasionally, we encounter polytopes $P$ and $C_{Y1}$ for which the facet enumeration is very time consuming (several minutes). For the practical application this computing times are unacceptable, thus we need to modify the procedure to generate the desired visualization in the properties space avoiding the dual construction. Specifically, in the following algorithm we produce the polygons $P_2$ and $C_2$ by cutting $P$ and $C_{Y1}$ with $H_2$ but having $P$ and $C_{Y1}$ in their primal form.

**Algorithm 4.1 2D-Cut Primal Form**

**Start**
Compute the extreme points of $P \cap H_2$ in the directions $\mathbf{w_1}$ and $\mathbf{w_2}$ by solving the problem:

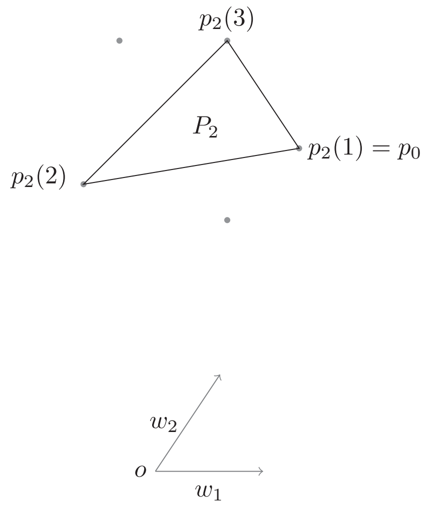
$$\begin{aligned}
&max \quad \mathbf{d}^{\mathrm{T}} \cdot \mathbf{s} \\
&\text{s.t.} \\
&\mathbf{y_M} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s} = \sum_{p_i \in V_P} \lambda_i \mathbf{p_i} \\
&\sum_{p_i \in V_P} \lambda_i = 1 \\
&\lambda_i \geq 0, \quad i = 1, \ldots, |V_P|
\end{aligned} \tag{4.8}$$

for $\mathbf{d}^{\mathrm{T}} = [1, 0]$, $\mathbf{d}^{\mathrm{T}} = [0, -1]$, $\mathbf{d}^{\mathrm{T}} = [-1, 0]$ and $\mathbf{d}^{\mathrm{T}} = [0, 1]$ (see Figure 4.3a).
Let $P_2 = \{\mathbf{p_2(1)}, \ldots, \mathbf{p_2(k)}\}$ be the polygon obtained with at least one vertex ($\mathbf{p}^*$ is the optimal solution of Problem (4.8) for $\mathbf{d}^{\mathrm{T}} = [1, 0]$).
If $P_2$ has more than one vertex, then we set $i = 1$ and we try to expand $P_2$ iteratively in the next step. Else, we go to **End**.

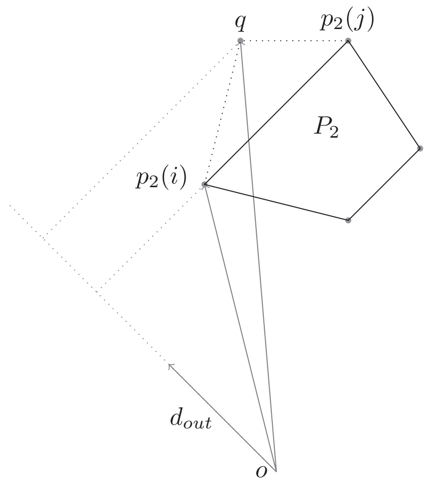
**Expand**
We look for a new vertex in the outer normal direction $\mathbf{d_{out}}$ to the edge $(\mathbf{p_2(i)}, \mathbf{p_2(j)})$, where $j = (i+1) \bmod (|V_{P_2}|)$. This is done by solving Problem (4.8) with $\mathbf{d} = \mathbf{d_{out}}$. Let $\mathbf{q}^*$ be the optimal solution for this problem.
If $\mathbf{d}^{\mathrm{T}} \cdot \mathbf{p_2(i)} < \mathbf{d}^{\mathrm{T}} \cdot \mathbf{q}^*$ then $\mathbf{q}^*$ is a new vertex in $P_2$ (see Figure 4.3b). We insert $\mathbf{q}^*$ in the vertex list at position $i+1$.

Otherwise, we set $i = i + 1$.
If $i < |V_{P_2}|$, we return to **Expand**.

**End**



(a): Start.



(b): Expand.

Figure 4.3: Algorithm 2D-Cut from primal form.

We proceed similarly to obtain the polygon $C_2$ and finally produce the infeasible visualization in the properties space (see Figure 4.4). This visualization also preserves the same important characteristics as in components space: The distance $D_Y$ between polygones $P_2$ and $C_2$ which is the same as the distance between $P$ and $C_{Y1}$ and the angle between directions $\mathbf{w_Y}$ and $\mathbf{w_Y^U}$.

We finish this section remarking that we can use the algorithm (4.1) to produce the visualizations for the infeasibility of levels 2 and 3. For the level 2, $D_{U2}$ and $D_{Y2}$ determine 4 feasible directions (2 in each space) that define the 2-D plans with which to cut the pair of polytopes ($U$, $C_U$) in the components space and ($P$, $C_Y$) in the properties space. For the infeasibility of level 3, we have only the feasible direction generated by the closest points in $P$ and $C$ (the polytope $C_U1$ is empty). Then, a second direction is choosen “conveniently“ in order to get an infeasibility visualization in the properties space.

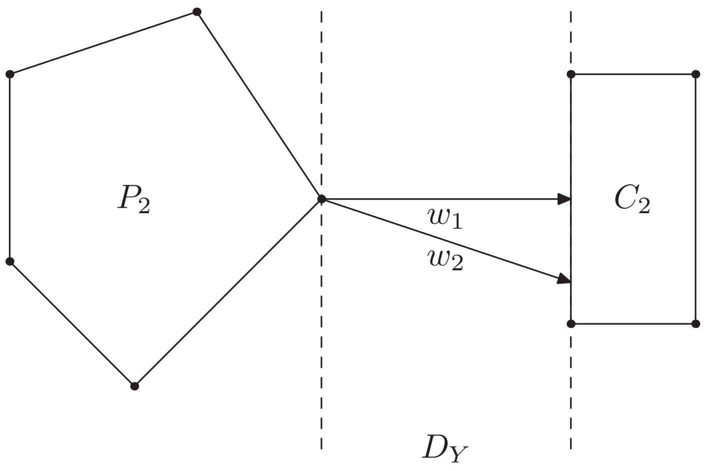


Figure 4.4: Infeasibility visualization in properties space.

## 4.6 Becoming Feasible and Robust

In the previous sections we showed how to measure and visualize the BP infeasibility. Let's now consider how to modify the hydraulic and availability bounds in order to improve the blend's quality and to produce, if possible, a feasible recipe. Specifically, we are going to transform the polytope $U$ into the polytope:

$$U_{NEW} = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{u}} - \underline{\boldsymbol{\Delta}} \leq \mathbf{u} \leq \overline{\mathbf{u}} + \overline{\boldsymbol{\Delta}}\ ,\ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1,\ \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}\}$$

by choosing the appropriate elastic variables $\underline{\boldsymbol{\Delta}}, \overline{\boldsymbol{\Delta}} \in \mathbb{R}^{n+}$ and looking to have $U_{NEW} \cap C_{U1} \neq \emptyset$.

We notice that hydraulic and availability bounds can be adjusted individually and can reach their lower and upper limits: $[0,1]$. In other words, the polytope $U_{NEW}$ can attain any recipe in $C_{U1}$ as $C_{U1} \subseteq S_1$. On the other hand, these bounds are settled in accordance with the global refinery planning and by operational requirements (e.g., to promote the use of plentiful or inexpensive components, or in order to empty one component's tank). The impact of the bounds violation on the overall refinery's performance is difficult to measure. Even harder is to estimate the individual impact of each bound violation. For this reason, we decide to proceed as follows: First we compute the best recipe $\mathbf{u}^* \in C_{U1}$ to produce according to some optimization criterion and then we obtain the bounds' modifications to have $\mathbf{u}^* \in U_{NEW}$ (see Figure 4.5).

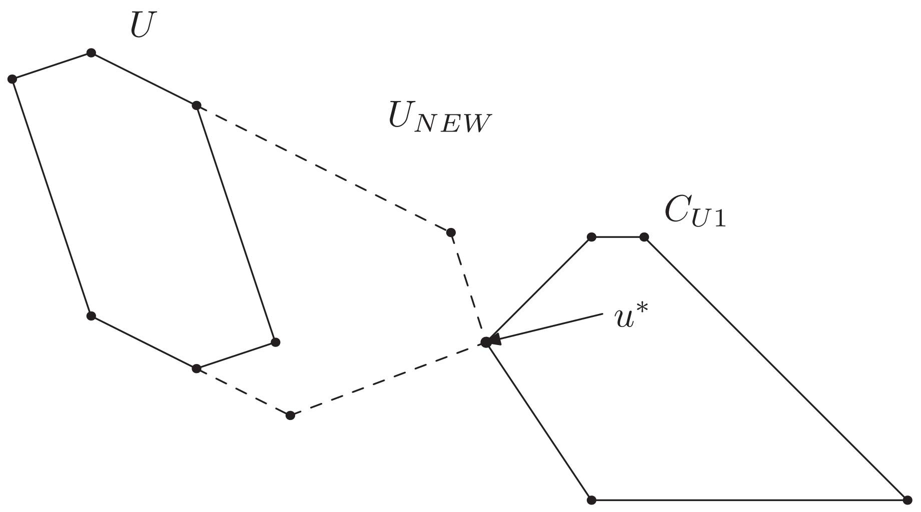


Figure 4.5: Transforming $U$ onto $U_{NEW}$ to reach the feasible blend $\mathbf{u}^*$.

We observe that choosing the vectors $\mathbf{u}^-, \mathbf{u}^+ \in \mathbb{R}^{n+}$, we can constrain the bounds modifications by searching the best recipe $\mathbf{u}^*$ in $U^+ \cap C_U$ (see Equation 4.9) instead of $C_{U1}$.

$$U^+ = \{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \underline{\mathbf{u}} - \mathbf{u}^- \leq \mathbf{u} \leq \overline{\mathbf{u}} + \mathbf{u}^+ \leq \mathbf{1} \ , \ \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\} \tag{4.9}$$

In particular, fixing $u_i^- = 0$ or $u_i^+ = 0$ we forbid the modification of the lower or upper bound for the $i^{th}$ component. Aditionally, in Section 4.7 we construct some benchmarks of the impact of the components bounds on the blends' cost, on the quality giveaway and on the blending volume. These benchmarks might be useful to decide whether or not to apply the bounds' modifications.

Some particular optimization criteria to search the best recipe are the minimization of the infeasibilities of level 1 in both spaces and the minimization of the bounds' modifications. This last one is the smallest variable cost model (see Section 4.1) with all the costs having the value 1.

Another natural choice is to select $\mathbf{u}^* \in C_{U1}$ using the same optimization criteria applied in the RTO method: we start searching a feasible recipe of minimal cost by solving the problem

$$\begin{aligned} &min \quad \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} \\ &\text{s.t.} \\ &\mathbf{u} \in C_{U1}. \end{aligned} \tag{4.10}$$

If the problem 4.10 is infeasible ($C_{U1} = \emptyset$), then we search the blend with minimal quality giveaway but considering exclusively the hard constraints:

$$\begin{aligned} &min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ &\text{s.t.} \\ &\mathbf{u} \in C_{U1H}. \end{aligned} \tag{4.11}$$

Finally, if the problem 4.11 is infeasible ($C_{U1H} = \emptyset$), then we can not transform $U$ into $U_{NEW}$ to reach a feasible recipe but solving

$$\begin{aligned} &min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 \\ &\text{s.t.} \\ &\mathbf{u} \in S_1 \end{aligned} \tag{4.12}$$

we obtain the recipe with properties as close as possible to $\mathbf{y_T}$.

Once the recipe $\mathbf{u}^*$ is determined, we get the optimal bounds' modifications

$$\underline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \underline{\mathbf{u}} - \mathbf{u}^*\} \quad \text{and} \quad \overline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \mathbf{u}^* - \overline{\mathbf{u}}\}$$

allowing us to produce $\mathbf{u}^*$. We denote by

$$\mathscr{E} = \sum_{i=1}^{n} (\underline{\Delta}_i^* + \overline{\Delta}_i^*) \tag{4.13}$$

the required effort to perform the BP instance transformation.

The approach described in this section does not consider the BP uncertainty which may cause the feasible blend obtained for the nominal data to be in reality infeasible. This situation is reflected by the fact that polytopes $U_{NEW}$ and $C_{U1}$ are just touching (see Figure 4.5). So, even a little difference between the nominal and the real data may result in a false intersection. In order to overcome this fragility and construct a robust blend, we only need to substitute in the previous procedure the polytopes by their corresponding robustopes as determined in Section 3.4. In this way, the recipe $\mathbf{u}^*$ will be selected deeply inside the polytope $C_{U1}$ and will resist to any variation in the data, within the pre-established limits. In Figure 4.6 we illustrate the result obtained from this procedure. In this case, the components' bounds are also considered as hard constraints which must be satisfied. So, after selecting the robust recipe $\mathbf{u}^* \in RC_{U1}$ we transform the robustope $RU$ onto $RU_{NEW}$ in order to have $\mathbf{u}^* \in RU_{NEW} \cap RC_{U1}$.

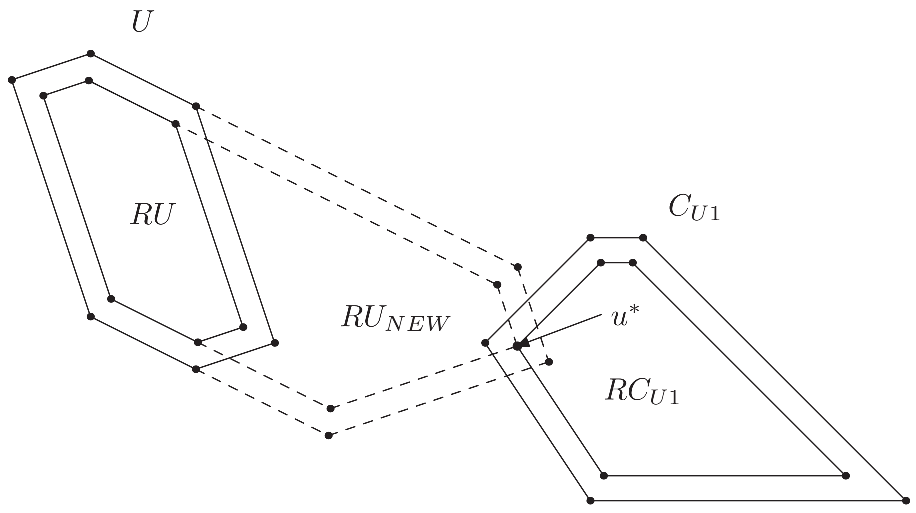


Figure 4.6: Transforming $RU$ onto $RU_{NEW}$ to reach the robust blend $\mathbf{u}^*$.

## 4.7 Overseeing the Blending Process

The bound's modification procedure of the previous section was developed to fix the BP infeasibility in accordance to the optimization criteria of the RTO method. Here, we present other meaningful objective toward which to drive the Blending Process.

When blending, one may be interested in getting from the current infeasible blend ($\mathbf{b_0}$) a feasible one while pouring the least possible blend's volume. This criterion turns out to be very practical in case of unforeseen events, like a component shortage or a pump halt, requiring to interrupt prematurely the blending process. Moreover, it allows to sell the blend as soon as possible thus reducing the inventory costs. Next we describe the procedure to compute the first reachable blend (in terms of volume) within specifications. First, solving the problems

$$\begin{aligned} & max \quad (min) \quad u_0 \\ & \text{s.t.} \\ & \mathbf{u} \in C_{U1} \\ & u_0 \geq 0. \end{aligned} \tag{4.14}$$

we obtain $u_{0Max}$ and $u_{Min1}$ ($u_{0Min}$ and $u_{Max1}$). Then, from these values and the relation $u_0 = V_0/v$ we get the volumes interval

$$[V_{Min1}, V_{Max1}]$$

for which $C_{U1} \neq \emptyset$. The blends $\mathbf{y_{Min1}} = \mathbf{y}(\mathbf{u_{Min1}})$ and $\mathbf{y_{Max1}} = \mathbf{y}(\mathbf{u_{Max1}})$ are the first and last blends, in terms of volume, that can be produced by means of "some" bounds' modifications. Finally,

$$\underline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \underline{\mathbf{u}} - \mathbf{u_{Min1}}\} \quad \text{and} \quad \overline{\mathbf{\Delta}}^* = \max\{\mathbf{0}, \mathbf{u_{Min1}} - \overline{\mathbf{u}}\}$$

are the minimal bounds modifications to reach the blend $\mathbf{y_{Min1}}$ at volume $V_{Min1}$.

Regarding robustness, we only need to replace the polytope $C_{U1}$ by its robustope $RC_{U1}$ in Problem (4.14) in order to recover the first and last producible robust blends when no bounds are considered.

The procedure to obtain $[V_{Min1}, V_{Max1}]$ is the same we used in the RTO method to compute the feasible volumes interval $[V_{Min}, V_{Max}]$ but taking $U \cap S_1$ instead of $U \cap C_U$. The relation between these volumes and the impact of the components bounds on the blending process can be better explained by referring to Figure 4.7. In this figure, the polytope $P$ runs from $P(0) = \mathbf{b_0}$ to $P(\infty)$ forming the cone

$$CP = \left\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = conv\left\{V_{P(\infty)} \cup \mathbf{b_0}\right\}\right\}.$$

The points $\mathbf{y_{Min}} \in P(V_{Min}) \cap C$ and $\mathbf{y_{Max}} \in P(V_{Max}) \cap C$ are the first and last feasible blends that can be produced.

On the other hand, as $U \subseteq S_1$ then $P(v) \subseteq P_1(v) = y(S_1), \quad \forall v \geq 0$. Therefore, $P_1(v)$ describes the cone

$$CP_1 = \left\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = conv\left\{V_{P_1(\infty)} \cup \mathbf{b_0}\right\}\right\}$$

which consists of all the producible blends when the hydraulic and availability bounds are disregarded. The first and last of these blends being within specifications (in polytope $C$) are $\mathbf{y_{Min1}}$ and $\mathbf{y_{Max1}}$. The intersection of $CP_1$ with $C$ is the polytope $C_{Y1}$ and the visualization in properties space presented in

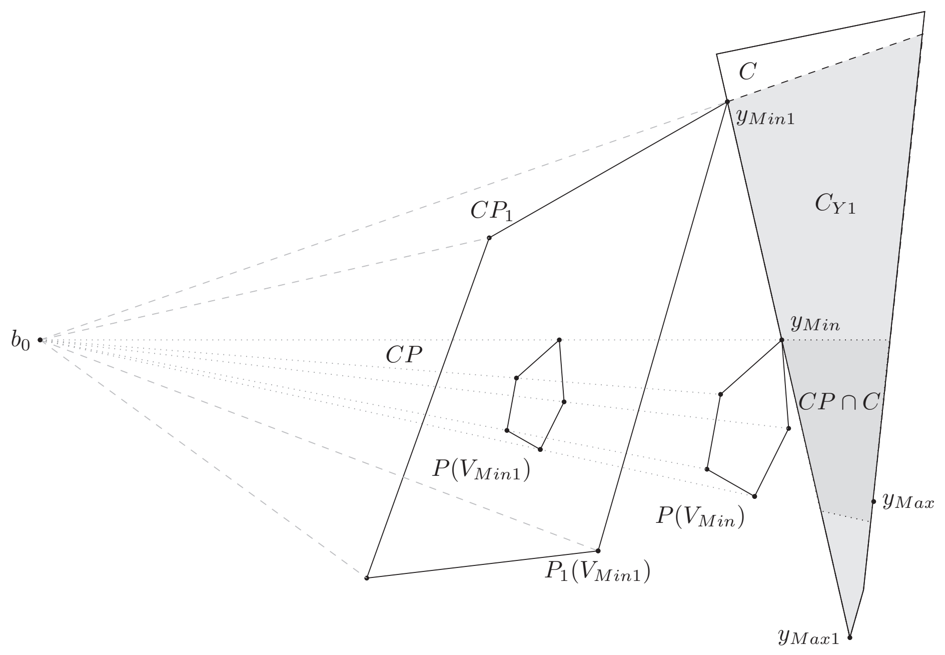


Figure 4.7: Impact of the components' bounds on the blending process.

Section 4.5, is a cut of the polytopes $CP(v)$ and $C_{Y1}(v)$ for a fixed volume $v$.

From the basic equation (Equation 2.2), we have that the initial blend $\mathbf{b_0}$ dilutes in the new blend as the poured volume increases. Therefore, the possible modifications to $\mathbf{b_0}$ provided by the new blend reduce with time. This traduces in a reduction of the movement speed of polytopes $P$ and $P_1$ as they move away from $\mathbf{b_0}$. For the same reason, the only variation in this figure when we use a different initial volume $V_0$, is the movement speed of $P$ and $P_1$: reducing $V_0$ speeds up the movement of both polytopes as we need to pour a smaller amount of blend's volume in order to modify $\mathbf{b_0}$.

Finally, let's suppose that $c^*$ and $q^*$ ($c_1^*$ and $q_1^*$) are the minimal cost and minimal quality giveaway that can be achieved by any blend in $CP \cap C$ (in $C_{Y1}$). Hence, as $P(v) \subseteq P_1(v)$ for all $v \geq 0$, it follows that $P \cap C \subseteq C_{Y1}$. Consequently, $V_{Min1} \leq V_{Min}$, $c_1^* \leq c^*$ and $q_1^* \leq q^*$. Thus, we can say that the components bounds have an impact of $(c^* - c_1^*)$ in the blend's cost, of $(q^* - q_1^*)$ in the blend's quality giveaway and of $(V_{Min} - V_{Min1})$ in the required volume to pour in order to have the first feasible blend.

It is worth to mention that we can generate this supportive visualization at any time in the blending process. All that we need to do is to compute the vertex sets $VC$, $VP(\infty)$ and $VP_1(\infty)$. Then, adding the point $\mathbf{b_0}$ to $VP$ and to $VP_1$, we have the primal description of $CP$ and $CP_1$. Next taking the directions $\mathbf{w_1} = \mathbf{y_{Min}} - \mathbf{b_0}$ and $\mathbf{w_2} = \mathbf{y_{Min1}} - \mathbf{b_0}$ and by means of the algorithm (4.1), we obtain the cuts of $C$, $CP$ and $CP_1$ with the 2D-plane $H_2$ defined by

$$H_2 = \{y(s) \mid \mathbf{y(s)} = \mathbf{b_0} + \mathbf{w}^{\mathrm{T}} \cdot \mathbf{s}\}$$

where $\mathbf{s} = (s_1, s_2)^T \in \mathbb{R}^2$ and $\mathbf{w} = (\mathbf{w_1}, \mathbf{w_2})$.

In the blending practice, it is common to monitor graphically the properties of the blend during its production (see Figure 4.8). These graphs offer at a glance useful information for the control department. For instance, one can

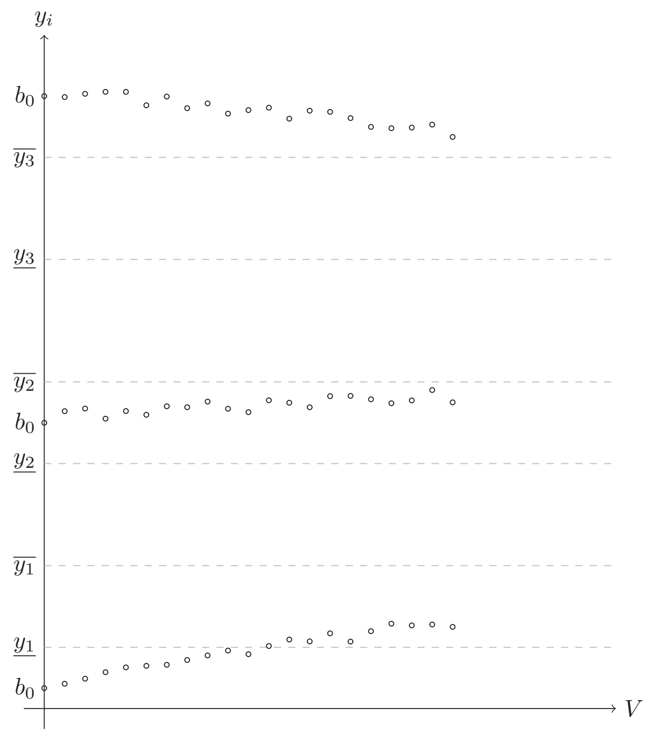


Figure 4.8: Monitoring the blend's properties.

distinguish instantly which properties are out of specifications and have a visual estimation of the bounds violations. Then, following the data trend and having a good knowledge of the components in use, the control staff can deduce the required adjustment to the recipe in order to reduce these violations.

In this paragraph, we propose an improvement for these graphs inspired by Figure 4.7. The enhancement consists in adding at the end of the observed data, a suitable cut of the blending cones $CP$ and $CP_1$ and of the target polytope $C$. Let's suppose that $\mathbf{b_0}$ is the actual blend and $\mathbf{y}^* \in P \cap C$ is the optimal blend to produce. Then, cutting $CP$, $CP_1$ and $C$ with the plane defined by vectors $\mathbf{w1} = \mathbf{y}^* - \mathbf{b_0}$ and $\mathbf{w2} = \mathbf{e_i}$ (the $i^{th}$ canonical vector in $\mathbb{R}^m$) and passing through $\mathbf{b_0}$, we obtain the desired cut (see Figure 4.9). In this visualization, we distinguish some new information elements: The

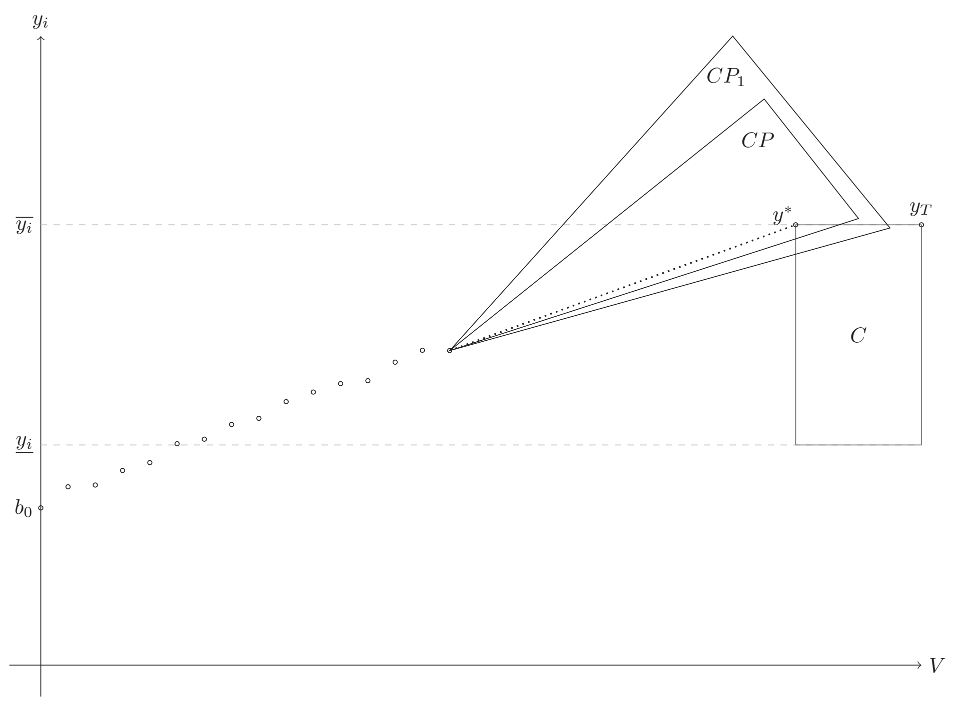


Figure 4.9: Monitoring the blend's properties with new information elements.

dotted line between $\mathbf{b_0}$ and $\mathbf{y}^*$ represents the "nominal path" of the $i^{th}$ blend's property i.e., the path to be followed by the $i^{th}$ blend's property in absence of uncertainty. The cone $CP$ shows the limitations imposed by the components bounds on the range of attainable values for the property $i$ and the cone $CP_1$ shows all the possible values of property $i$ if the components bounds are disregarded. The relative location of the target blend with respect to the blending polytopes (here, we observe that $y_T$ is out of reach even with bounds modifications) and finally, the location of $y^*$ in $C$ and $CP_1$ give us some information about the non-robustness of $y^*$ and the capacity to correct it. Here, the blend $y^*$ is located on the boundary of $C$ and is prone to become infeasible, however, with bounds modifications a more robust blend can be produced.

Now we explain how to produce some benchmarks of the impact of the components bounds on the blends' cost, on the quality giveaway and on the blending volume. These benchmarks might be useful to decide whether or not to apply the bounds' modifications.

First we construct a cost-related benchmark $K_C$. To do this, let's consider the Problem (3.3) with the additional constraints (3.7):

$$\begin{aligned} &min \quad \mathbf{c}^{\mathrm{T}} \cdot \mathbf{u} && (4.15)\\ &\text{s.t.}\\ &\mathbf{u} \in U \cap C_U && (4.16)\\ &u_{0Min} \leq u_0 \leq u_{0Max}. && (4.17) \end{aligned}$$

Let $\mathbf{u}^*$ be the optimal solution and $c^*$ its cost. This is the cheapest recipe we can produce while satisfying the components bounds. Then, replacing the constraints 4.16 and 4.17 by

$$\begin{aligned} &\mathbf{u} \in C_{U1}\\ &u_{0Min1} \leq u_0 \leq u_{0Max1} \end{aligned}$$

we ignore the components' bounds and we get the optimal recipe $\mathbf{u}_\mathbf{1}^*$ with cost $c_1^*$. The ratio

$$K_C = \frac{c^* - c_1^*}{c_1^*}$$

measures the relative cost increase produced by the components bounds.

Similarly, by means of a dichotomic search over $v$ and solving the problem

$$\begin{aligned} &min \quad \|\mathbf{y}(\mathbf{u}) - \mathbf{y_T}\|_1 && (4.18)\\ &\text{s.t.}\\ &\mathbf{u} \in U \end{aligned}$$

with $v \in [V_{Min}, V_{Max}]$ fixed at each step, we get the blend in $P \cap C$ with minimal quality giveaway $q^*$. Then, after replacing $U$ by $S_1$, and $[V_{Min}, V_{Max}]$ by $[V_{Min1}, V_{Max1}]$ we obtain the blend in $C_{Y1}$ with minimal quality giveaway $q_1^*$. In this case, the ratio

$$K_Q = \frac{q^* - q_1^*}{q_1^*}$$

measures the relative quality giveaway increase produced by the components bounds.

Finally, the quotient $K_V = (V_{Min} - V_{Min1})/V_{Min1}$ measures the impact of hydraulic and availability bounds on the blending volume required to produce the first feasible blend.

We highlight the possibility to compute these benchmarks in order to know the impact of a single bound or a set of bounds on the three blend's qualities: cost, giveaway and volume. To do this, we simply fix the corresponding variables $u_i^-$ and/or $u_i^+$ to zero for all the components bounds under analysis and we let the other bounds to be variable.

## 4.8 Example

Let's consider the BP instance $BP(1) = (B, v, \mathbf{b_0}, V_0, \underline{\mathbf{u}}, \overline{\mathbf{u}}, \underline{\mathbf{y}}, \overline{\mathbf{y}}, \mathbf{c})$ presented in the example of Section 3.5.

The next discussion can be followed in Figure 4.10 which was produced by cutting the blending cones and the target polytope with the plane determined by the vectors $\mathbf{y_{Min}} - \mathbf{b_0}$ and $\mathbf{y_{Min1}} - \mathbf{b_0}$ passing through $\mathbf{b_0}$, (see 4.7). In this visualization, we observe how polytopes $P$ and $P_1$ evolve in time as the blending volume increases. From $\mathbf{b_0} = P(0) = P_1(0)$ until $P(\infty)$ and $P_1(\infty)$ they form the cones $CP$ and $CP_1$ respectively.

As we have seen in Section 3.5, the $BP(1)$ is infeasible if we want to

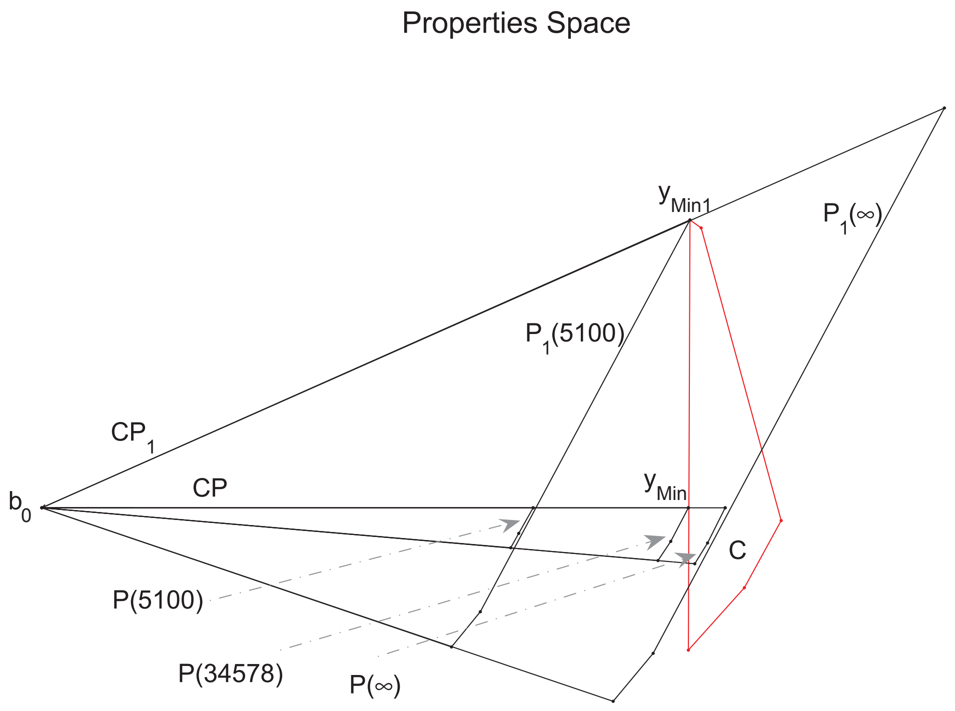


Figure 4.10: $BP(1)$: Impact of the components' bounds on the blending process.

produce a volume $V_{total} = 5000\,m^3$ of blend and using $V_0 = 2000\,m^3$ of the previous blend. The infeasible blend $\mathbf{y_1}$ producing the minimal quality giveaway was then computed. The cost, the quality giveaway and the required effort (see Equation 4.13) to reach $y_1$ are grouped in its *quality vector*:

$$\mathbf{Q}(\mathbf{y_1}) = [90.7162 \quad 422.9910 \quad 0]^{\mathrm{T}}.$$

Then we compute the interval $[u_{0Min}, u_{0Max}]$ for which a feasible blend exist. From these values and the relation $u_0 = V_0/v$, we can compute all the couples $(V_0, v)$ producing a blend within specifications. In particular, a minimum volume $v = 34578\,m^3$ needs to be produced if we want to use completely the heel's volume. All this is reflected in Figure 4.10, where the blend $\mathbf{y_{Min}} \in P(34578) \cap C$ is the first feasible blend on the path followed by the polytope $P$.

From the material presented in this chapter, we have more information about the $BP(1)$ infeasibility. The problem has infeasibility of level 3 as the polytope $C_U(3000)$ is empty. This means that for $v = 3000\,m^3$ and $V_0 = 2000\,m^3$, bounds modifications do not suffice to attain a feasible blend. For these volumes, the best we can do is to extend the polytope $U$ to reach the recipe $\mathbf{u_2} \in S_1$ with minimal quality giveaway but remaining infeasible. To do this, we solve the Problem (4.12) getting the blend $\mathbf{y_2}$ with quality vector

$$\mathbf{Q}(\mathbf{y_2}) = [111.0754 \quad 156.1711 \quad 1.0614]^{\mathrm{T}}.$$

Otherwise, solving the Problem (4.14) we get $(u_{0Max1}, u_{0Min1})$ and from them, taking $V_0 = 2000\,m^3$ we compute the range of volumes

$$[V_{Min1}, V_{Max1}] = [5100, \infty)$$

from which bounds modifications, including the null modification, can produce a feasible blend. In Figure 4.10, the blend $\mathbf{y_{Min1}} \in P1(5100) \cap C$ is the first reachable blend by bounds modifications. For any volume less than 5100, the cone $CP_1$ does not intersect $C$.

For $v = 5100$, the problem continues to be infeasible but of level 1. The polytopes $U$ and $C_{U1}$ are at distance $D_U = 0.5404$ whereas the distance between $P$ and $C_{Y1}$ is $D_Y = 234$. Taking $\mathbf{w_1} = \mathbf{w_U}$ and $\mathbf{w_2} = \mathbf{w_U^Y}$ we cut the polytopes $U$ and $C_{U1}$ to get the infeasibility visualization in components space (see Figure 4.11). The distance between the polytopes $U$ and $C_{U1}$ living in $\mathbb{R}^8$ is exactly the gap depicted between the cuts of $U$ and $C$. The last one consists only of the point $\mathbf{t}^*$.

In properties space, the visualization (see Figure 4.12) is generated using the vectors $\mathbf{w_1} = \mathbf{w_Y}$ and $\mathbf{w_2} = \mathbf{w_Y^U}$. Here again, the cut of $C_{Y1}$ consists only of the point $\mathbf{q}^*$. The gap separating the doted lines is the distance between

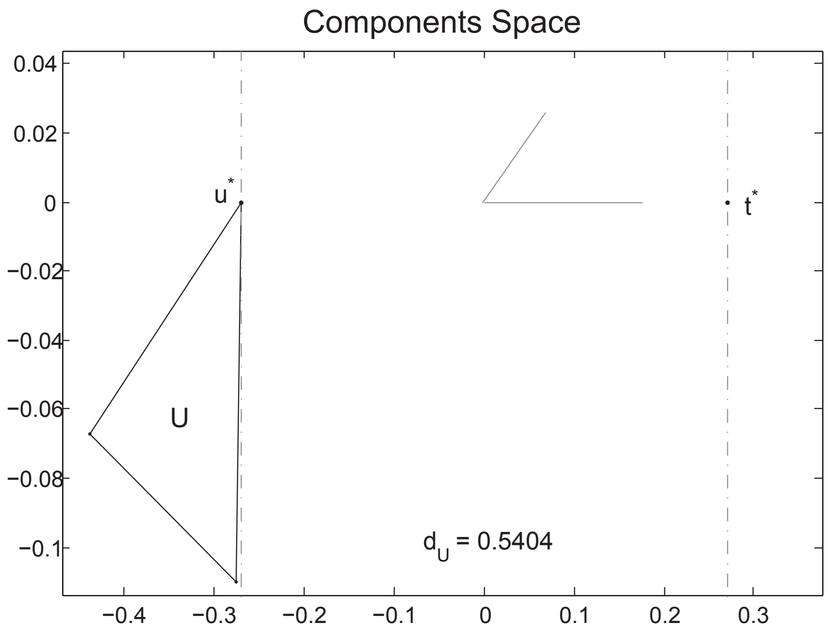


Figure 4.11: $BP(1)$: Infeasibility visualization in components space.

the polytopes $P$ and $C_{Y1}$ living in $\mathbb{R}^7$.

Having infeasibility of level 1, we can compute the minimal cost recipe

$$\mathbf{u_3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0.3389 & 0 & 0.6611 \end{pmatrix}^{\mathrm{T}}$$

in $C_{U1}(5100)$. This recipe can be attained by the polytope $U$ applying the following bounds' modifications:

$$(\underline{\boldsymbol{\Delta}}, \overline{\boldsymbol{\Delta}}) = \begin{pmatrix} 0.1428 & 0 \\ 0.0819 & 0 \\ 0.0352 & 0 \\ 0.0881 & 0 \\ 0.0338 & 0 \\ 0 & 0.1389 \\ 0.0352 & 0 \\ 0 & 0.4611 \end{pmatrix}.$$

The quality vector of this blend is

$$\mathbf{Q(y_3)} = [104.0023, 17.2782, 1.017].$$

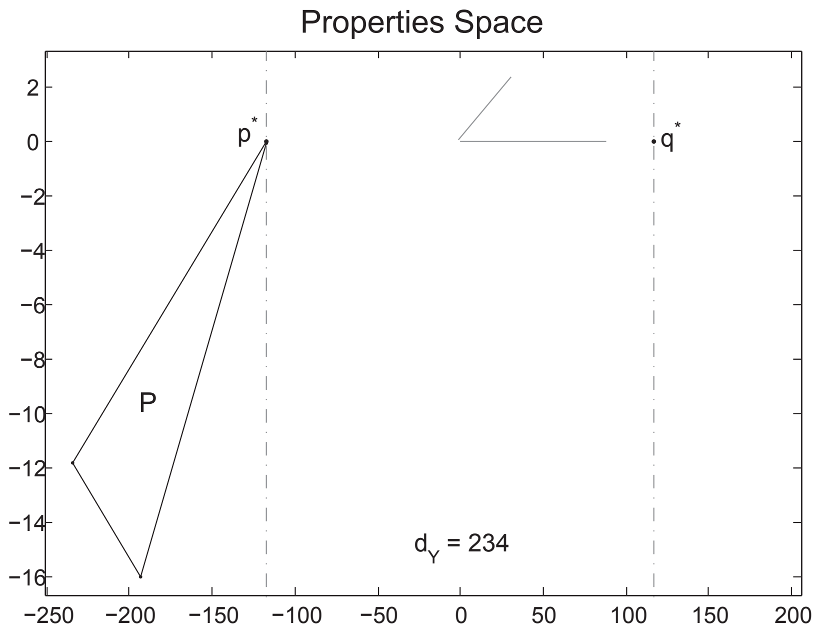


Figure 4.12: $BP(1)$: Infeasibility visualization in properties space.

and the visualization of this transformation is presented in Figure 4.13. Here we are cutting the polytopes $U$, $U_{NEW}$ and $C_{U1}$ using $\mathbf{w_1} = \mathbf{w_U}$ and $\mathbf{w_2} = \mathbf{u_3} - \mathbf{u_M}$. In this way, we get in the same cut the cheapest recipe $\mathbf{u_3}$ and the closest one $\mathbf{t}^*$. This explains why the cuts of $U$ and $C_{U1}$ are different from the ones of the Figure 4.11. Here, the cut of $U$ consists only of the point $\mathbf{u}^*$ whereas the cut of $C$ is the segment $[\mathbf{t}^*, \mathbf{u_3}]$.

Now we proceed to measure the impact of the components' bounds on the blending process. In Figure 4.10, we can see at a glance how the components bounds impose limitations not only on the size of the feasible region $CP \cap C$ but also to the required time (that is the volume to pour) to reach a feasible blend. Numerically, the impact of the components bounds on the optimal blend's cost, on the quality giveaway and on the required volume to pour before getting the first feasible blend are:

$$K_C = 9.71\% \quad K_Q = 234.38\% \quad K_V = 578.10\%$$

These indices tell us that components' bounds are responsible of an increase of almost 10% in the blend's cost, of 234% in the blend's quality giveaway and of 578% in the blending volume required to produce the first blend within

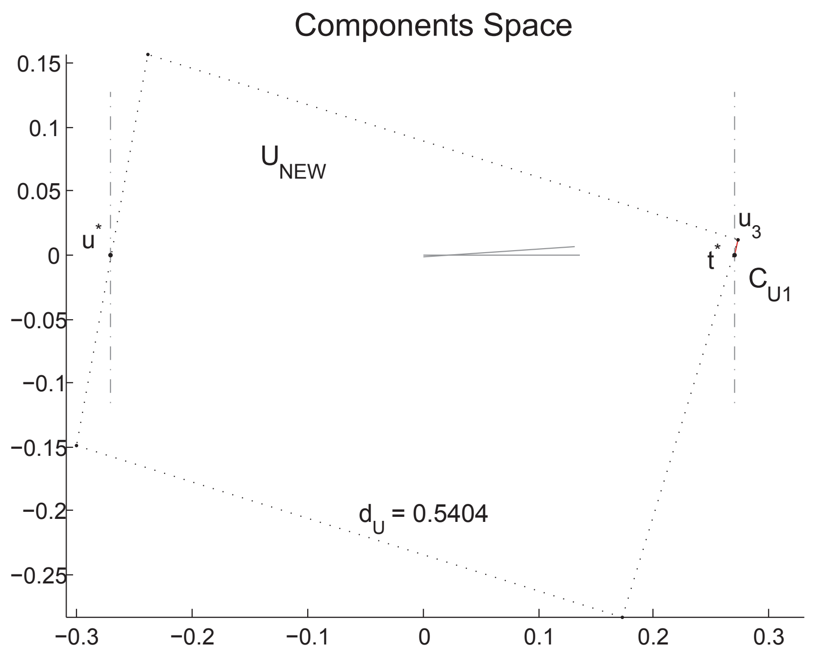


Figure 4.13: Bounds modifications of polytope $U$ to attain the cheapest recipe $u_3 \in C_{U1}$.

specifications.

Now, we measure the impact of each bound on the blend's qualities. That is, we compare the optimal blend's qualities for the $BP(1)$ when all the components bounds are disregarded versus the $BP(1)$ when only one bound at a time is active. These indices are showed (in percentage) in Table 4.1. There we can read, for instance, that the upper bound 8 has no incidence on the blend's cost, and that it increases the quality giveaway in 107% and the minimal blending time in 26%. This does not mean that allowing the modification of $\underline{u}_8$ will reduce the quality giveaway and the blending time by these quantities. What this really means is that $\underline{u}_8$ prevents to reduce by more than half the quality giveaway and by 26% the blending time, even if all the other bounds are free to be modified.

Regarding the uncertainty in the process, let's suppose that we want to be protected from measurement and $B$'s uncertainty within the following limits:

$$\boldsymbol{\delta_y} = \begin{pmatrix} 0.1200 & 0.0003 & 4.5000 & 0.0826 & 0.0280 & 0.000049 & 0.2000 \end{pmatrix}^T$$

| Component | LB | | | UB | | |
|---|---|---|---|---|---|---|
| | $K_C$ | $K_Q$ | $K_T$ | $K_C$ | $K_Q$ | $K_T$ |
| 1 | 1 | 32 | 48 | 0 | 0 | 0 |
| 2 | 1 | 16 | 23 | 0 | 0 | 0 |
| 3 | 0 | 4 | 5 | 0 | 0 | 0 |
| 4 | 0 | 25 | 4 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 5 | 9 | 32 |
| 7 | 0 | 0 | 16 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 107 | 26 |

Table 4.1: $BP(1)$: Individual bounds impact on the blend's qualities.

$\delta_u = 0.01$ and $\theta = 0.01$ as we did at the beginning of the example in Section 3.5. Then we compute the robust volumes intervals:

$$[V_{MinR}, V_{MaxR}] = [55304, \infty)$$

and

$$[V_{Min1R}, V_{Max1R}] = [5784, \infty).$$

This means that we need to pour a surplus of $784\,m^3$ of blend in order to be in range of getting by bounds modifications the first robust blend. Let's produce $5784\,m^3$ of blend and let

$$\mathbf{u_6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0.4793 & 0 & 0.5207 \end{pmatrix}^{\mathrm{T}}$$

and

$$\mathbf{u_7} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0.6014 & 0 & 0.3986 \end{pmatrix}^{\mathrm{T}}$$

be the minimal cost of robust and nominal recipes that can be reached by bounds modifications. The quality vectors of these blends are:

| Blend | Cost | Quality Giveaway | Effort |
|---|---|---|---|
| $\mathbf{y_6}$ | 98.5824 | 45.4120 | 1.017 |
| $\mathbf{y_7}$ | 93.8729 | 44.3127 | 1.017 |

From these vectors we observe that the cost of the robust blend $\mathbf{y_6}$ is 5.01 % higher than the cost of the nominal blend $\mathbf{y_7}$. In contrast, we realize that the cost of the nominal blend $\mathbf{y_3}$ (for $v = 5100$) is 10.79 % higher than the cost of $\mathbf{y_7}$. Thus, producing the blend's surplus of $784\,m^3$ we can reach a robust and cheaper blend than $\mathbf{y_3}$.

In Figure 4.14 we show the infeasibility visualization in components space for $v = 5784$. This time we are cutting the polytopes $U$, $U_{NEW}$, $C_{U1}$ and

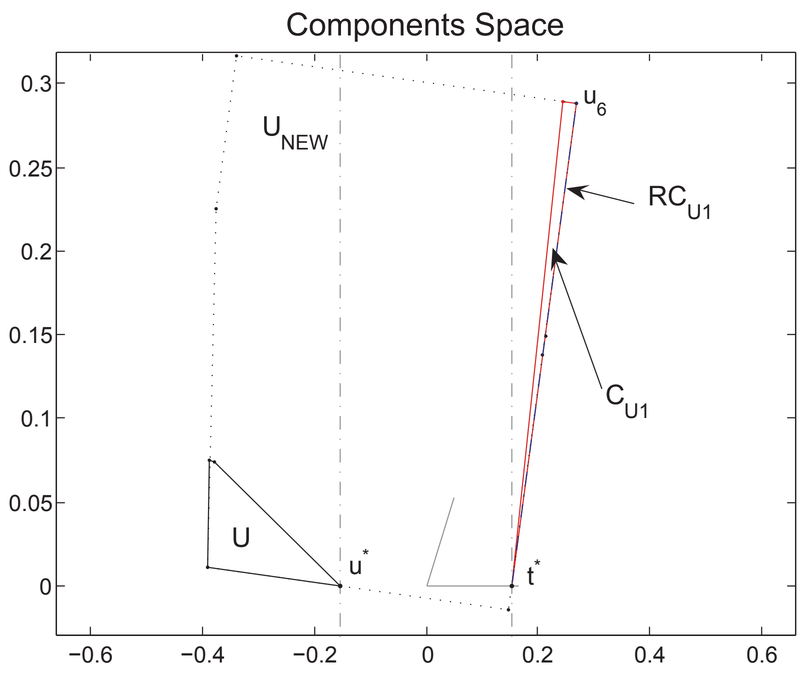


Figure 4.14: Transforming polytope $U$ to reach the robust recipe $u_6$.

$RC_{U1}$ taking the vectors $\mathbf{w_1} = \mathbf{w_U}$ and $\mathbf{w_2} = \mathbf{u_6} - \mathbf{u_M}$. The cut of $RC_{U1}$ being the line segment $[\mathbf{t}^*, \mathbf{u_6}]$. Notice that the robust recipe $\mathbf{u_6}$ is located on the boundary of the polytope $C_{U1}$. Contrary to the intuition, this can happen because the boundary where $\mathbf{u_6}$ is located is a boundary of $S_1$: recall that $C_{U1} = C_U \cap S_1$ and observe that $\mathbf{u_6}$ is the strict convex combination of two vertices of $S_1$. Thus, not being $\mathbf{u_6}$ on the boundary of $C_U$, it can be robust to properties measurement and to $B$'s uncertainty. Moreover, as it is impossible to produce a recipe outside $S_1$, then $\mathbf{u_6}$ is robust also to components measurement uncertainty.

Now, in Figures 4.15 to 4.21 we show the visualizations to monitor the blend's properties of $BP(1)$. Here, we are tracking the blend

$$\mathbf{y}^* = \begin{pmatrix} 33.2649 & 0.2305 & 579.2266 & 24.3628 & 432.0940 & 1.4379 & 41.8209 \end{pmatrix}^{\mathrm{T}}$$

which is the first feasible blend that can be produced. We notice that at the beginning of the process $\mathbf{b_0}$ has 5 of 7 properties within specifications and that the property 5 will be the last property to reach its specifications. In this example, the cuts of $CP$ are very thin and are confused with the nominal path of $\mathbf{y}^*$. We observe also that the target blend is out of reach even with bounds modifications ($\mathbf{y_T} \notin CP_1$). Finally, regarding the robustness of $\mathbf{y}^*$, it seems that the properties at most risk are the properties 5 and 6.

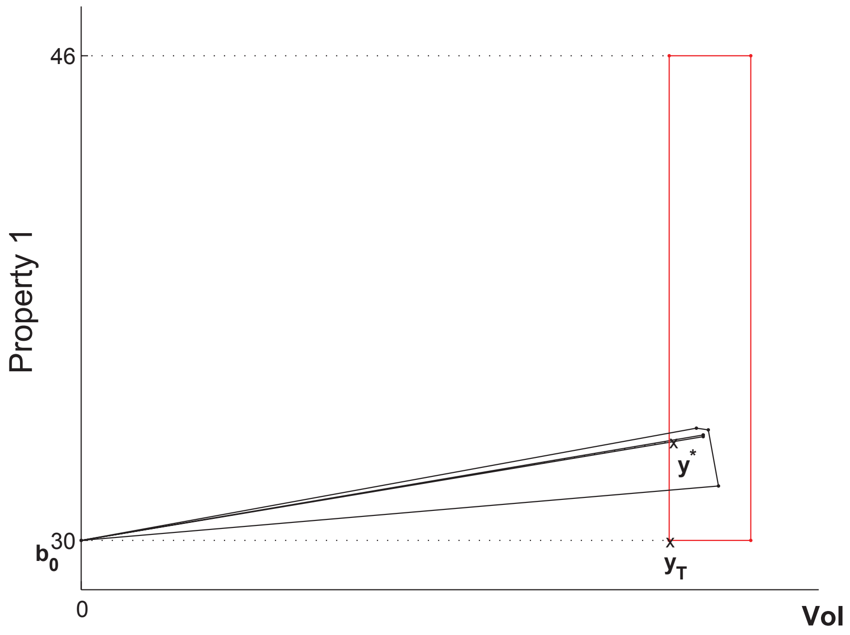


Figure 4.15: $BP(1)$: Monitoring the property 1.

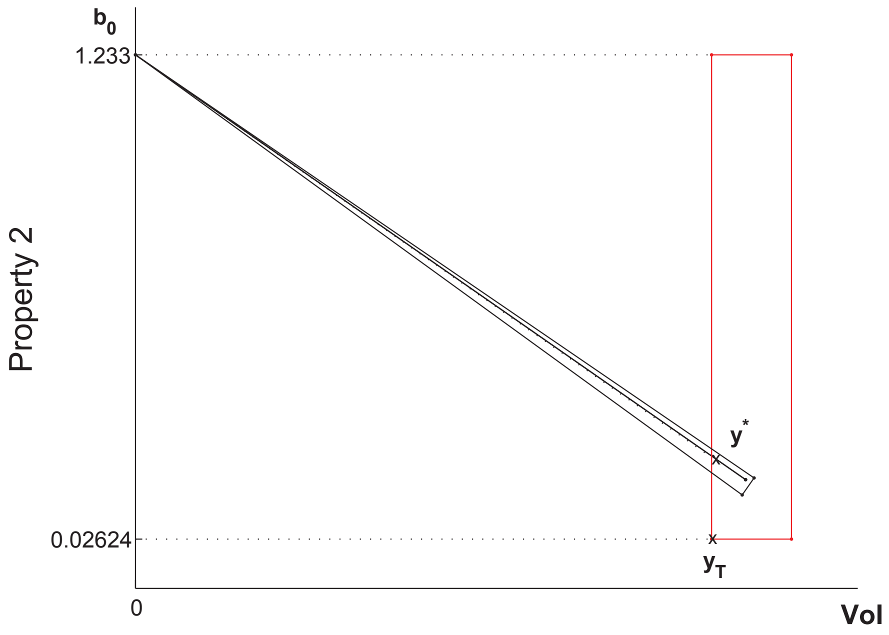


Figure 4.16: $BP(1)$: Monitoring the property 2.

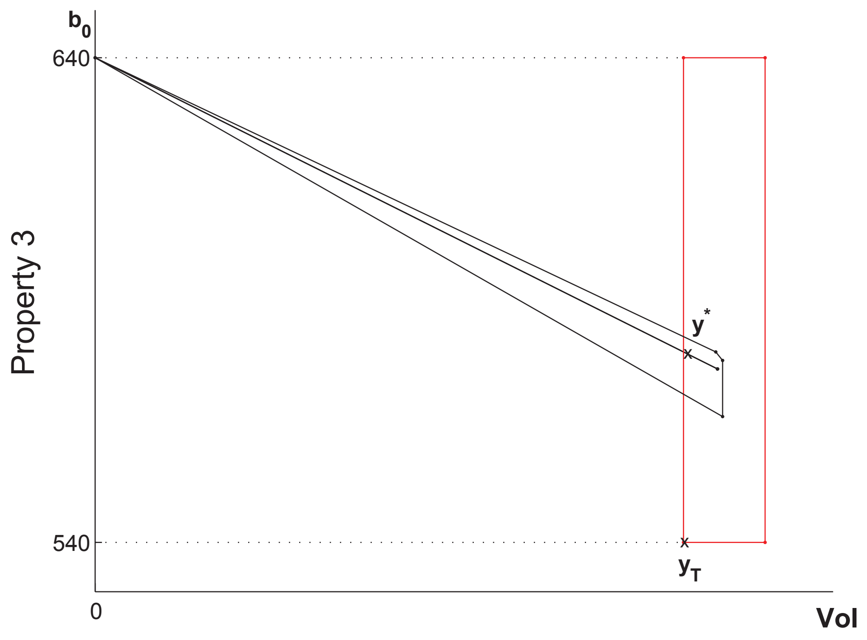


Figure 4.17: $BP(1)$: Monitoring the property 3.

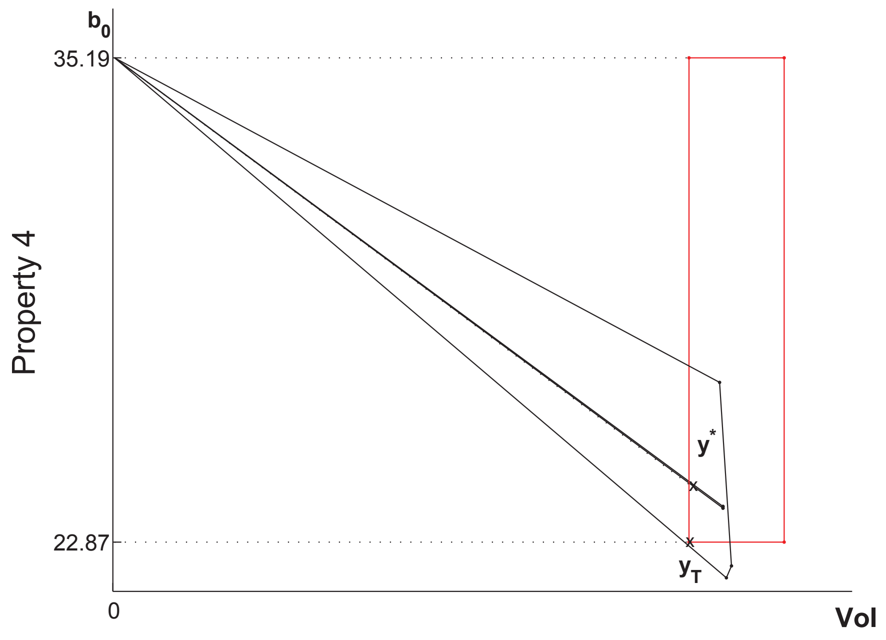


Figure 4.18: $BP(1)$: Monitoring the property 4.

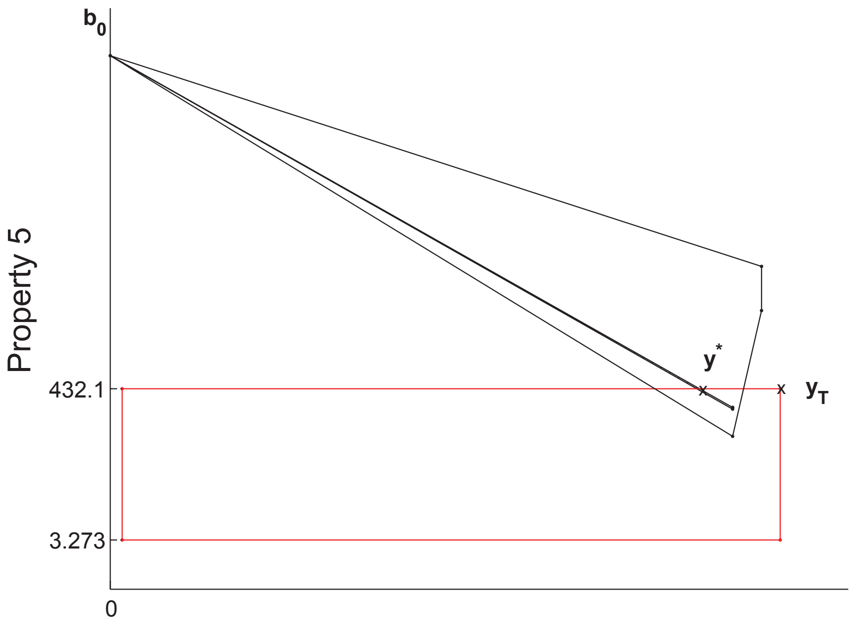


Figure 4.19: $BP(1)$: Monitoring the property 5.

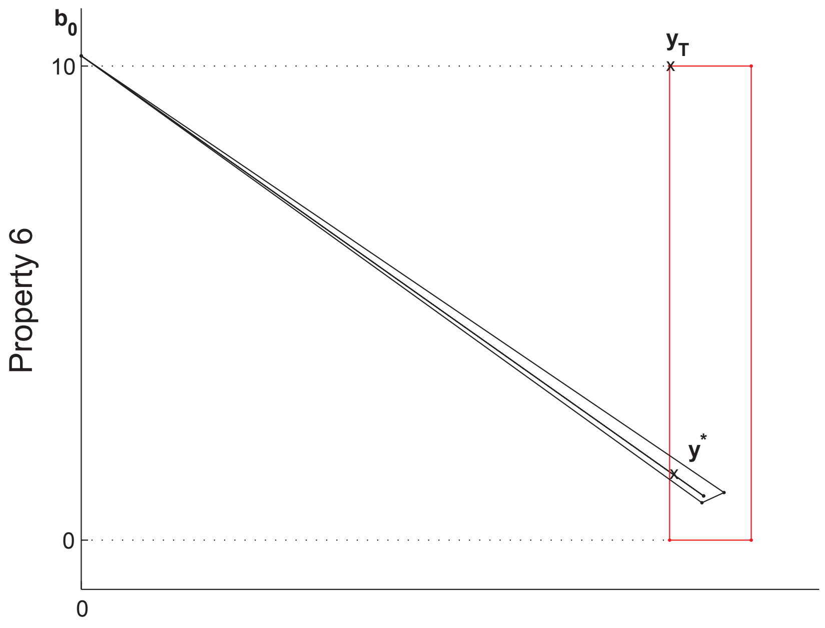


Figure 4.20: $BP(1)$: Monitoring the property 6.

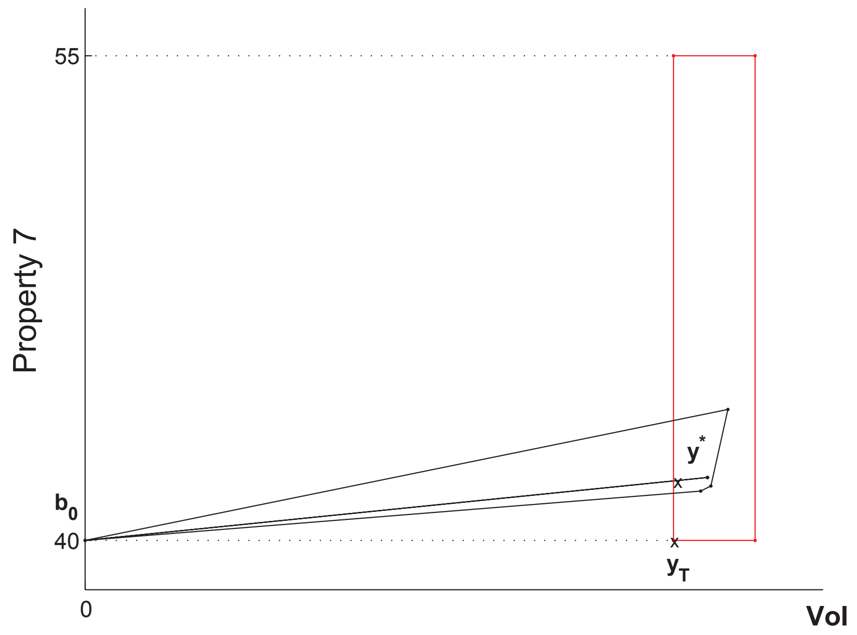


Figure 4.21: $BP(1)$: Monitoring the property 7.

# Chapter 5
# Conclusions

In this work, we introduced the Blending Polytopes inherent to the Blending Problem. Based on these polytopes we proposed a real time optimization (RTO) method for the linear oil blending process. This method is intended to avoid reblending while minimizing the blend's cost and the quality giveaway.

A main characteristic of our RTO method is the integration of the case when the blend has to be produced using the heel of a previous blend. This feature provides meaningful information for the control system, for instance, to determine the appropriate heel's volume to use in the blend or to have an estimate of the required volume to pour before getting the first blend within specifications.

Several sources of uncertainty pervade the blending process. We identified and modeled some different types of them following the RO techniques. Using interval sets, the robust regions are polytopes and a straightforward robust version of the RTO method is obtained.

Reblending a previous blend which failed to be within specifications has several costs associated: The tank rent, the logistic and inventory costs… Another one is the recipe's cost increase for using the heel's volume of the previous blend. Here we compared "this" component of the reblending cost with the price of robustness to stress the convenience of producing robust recipes and prevent reblending. The example presented and the limited set of other simulations that we conducted are not conclusive and further studies are needed before considering the implementation of our method in the refinery blending circuit. However, we are only comparing a component of the reblending cost (the recipe's cost increase) and the results obtained comfort the idea that reblending might be much more expensive than producing a robust blend. Then, using a more conservative technique like RO may improve the global performance of the blending process. The recipe's cost rises by taking the robust recipe in place of the nominal one but the reblending expenses cancel.

In the second part of this work, we exploited the Blending Polytopes to develop a fine analysis of the BP infeasibility. We use them to classify, measure and visualize the BP infeasibility and to generate a component's bounds modifications procedure to drive the blending process towards the "best" robust blend. A set of indices measuring the impact of the component's bounds on the recipe's cost, on the blend's quality giveaway and on the blending time required to produce the first feasible recipe have been proposed. These indices might be useful to decide whether or not to apply the bounds' modifications. We accompany these measures with supportive visualizations for the decision maker.

Appendix A

# Vertex Enumeration

In this section we are interested in computing the vertex set for the blending polytopes $U$ and $C_{U1}$. For completeness, in Section A.1 we rewrite some definitions and we present a brief survey of the existing methods to solve the vertex enumeration problem. Then, in Section A.2 we review the algorithm *zerOne* presented in [11] which enumerates efficiently the vertices of a class of polytopes which are combinatorially equivalent to the $d$-cube. The blending polytopes $U$ and $C_{U1}$ are contained in the $d$-cube but we don't know if they belong to this class of polytopes. However, the algorithm zerOne provides insight into the characteristics of the blending polytopes and we exploit this information in the vertex enumeration algorithm presented in Section A.3.

## A.1 State of the art

For $S \subseteq \mathbb{R}^d$, the convex hull *conv(S)* of $S$ is the smallest convex set in $\mathbb{R}^d$ containing $S$. A $\mathcal{V}$*-polytope* is the convex hull of a finite set of points in some $\mathbb{R}^d$. A subset $P$ of $\mathbb{R}^d$ is called an $\mathcal{H}$*-polyhedron* if it is the intersection of a finite set of halfspaces in some $\mathbb{R}^d$. An $\mathcal{H}$*-polytope* is an $\mathcal{H}$*-polyhedron* that is bounded. A *polytope* is a set $P \subseteq \mathbb{R}^d$ which can be presented either as a $\mathcal{V}$*-polytope* or as an $\mathcal{H}$*-polytope.*

Every polytope has then two representations. The *primal representation* as the convex hull of the finite set of its vertices:

$$P_{\mathcal{P}} = \{\mathbf{v} \in \mathbb{R}^d \mid \mathbf{v} = \sum_{k=1}^{s} \lambda_k v_k, \quad \sum_{k=1}^{s} \lambda_k = 1, \quad \lambda_k \geq 0, \quad k = 1, \ldots, s\} \tag{A.1}$$

and the *dual representation* as the intersection of a finite set of half-spaces:

$$P_{\mathcal{D}} = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \leq \mathbf{b}\}. \tag{A.2}$$

The transformation A.1 to A.2 is known as the *vertex enumeration* (VE) problem and from A.2 to A.1 is known as the *facet enumeration* (FE) problem.

The *dimension* of a polytope is the dimension of its affine hull and a $d$-*polytope* is a polytope of dimension $d$ in some $\mathbb{R}^e$ ($e \geq d$).

A $d$-polytope is called *simple* if each vertex is contained in exactly $d$ facets and is called *simplicial* if each facet contains exactly $d$ vertices. A vertex lying on more than $d$ facets is known as a *degenerate* vertex.

Since the number of vertices of a $d$-polytope $P$ with $m$ facets is $\mathcal{O}(m^{\lfloor d/2 \rfloor})$ [37], then it is common to measure the efficiency of a vertex enumeration algorithm in terms of both the size of the input and the size of the output (see e.g., [42]). Under this scheme, a VE algorithm is *polynomial* if its running time is bounded by a polynomial in $d, m$ and the number of vertices $v$.

In the literature several works have been devoted to the VE problem. See, e.g., [31, 32, 8, 18, 19, 36, 16, 6] and all the references therein. The VE problem for polyhedra is known to be a NP-Hard problem [27] and for polytopes no efficient algorithm is known.

For the general case, there are two main classes of algorithms for solving VE: the pivoting methods and the insertion methods. Apparently, the first published method to solve VE is the insertion method [32], now known as the *double description method.* Te DD method involves the insertion of constraints iterativelly. At each step, a vertex list is updated: the old vertices which become infeasibles in the new system are deleted whereas the vertices generated by the new constraint are inserted. To detect the new vertices at each step, the DD method and its variants must determine whether two former vertices $x_i$ and $x_j$ are adjacent. The complexity of this test depends strongly on the number of present vertices at each step and thus on the insertion order of the constraints. In [18], an example is presented to show that the worst-case complexity of the insertion methods is not bounded by a polynomial function on $m, n$ and $v$. However, assuming that the number of present vertices at intermediate steps remains in $\mathcal{O}(v)$, the whole complexity of the DD method is in $\mathcal{O}(m^2 v^3)$ (see [21] for efficient implementations).

The *pivoting methods*, inspired by the simplex method, exploit the relation between the polytope's vertices and the *Basic Feasible Solutions* (see, e.g., [16]). For simple polytopes this relation is one to one whereas for the degenerate case, a degenerate vertex is related to several BFS. These methods start producing an arbitrary vertex and continue iteratively generating all the neighbor BFS produced by a pivoting step. Each neighbor BFS determines a vertex which might have been discovered at a previous step. Thus, the method must include a test to prevent duplicated vertices to be output. In the worst case a polytope may have a unique vertex and up to $\binom{m}{d}$ BFS. Thus, the pivoting methods are not efficient for degenerated polytopes.

For some special classes of polytopes, efficient algorithms have been proposed. For example, if $P$ is a polytope simple, [18] presented the first linear-time algorithm (on $v$) to solve the VE problem. The algorithm performs a depth first search on the *graph* of $P$ and uses a *balanced tree* (see, e.g., [3]) to store the polytope vertices. Later, the *reverse search method* was presented in [7]. This method enumerates the vertices of $P$ in time $\mathcal{O}(ndv)$ and has the advantage of using a memory space which is bounded by a polynomial in the input size. For polytopes arising from networks, [35] gives an algorithm in $\mathcal{O}(v^2)$ and for polythedra associated with dual LI(2) systems (systems having at most two nonzero coefficients per constraint), [2] presents a pivoting algoritm running in $\mathcal{O}(nv)$. For polytopes having all its vertices in $\{0,1\}^d$, a backtracking approach is presented in [11] which runs in polynomial time on the input size and in linear time on the number of vertices.

Available codes implementing some of these methods can be found in [20, 5, 15, 30].

## A.2 ZerOne Algorithm

Let $P$ be a polytope described in its dual form and let's denote by $\mathbf{P}^{\square}$ the class of polytopes which are *combinatorially equivalent* to a $d$-cube of appropriate dimension. That is, it exists a bijection between the faces of the $d$-cube and those ones of $P^{\square} \in \mathbf{P}^{\square}$ that preserves the inclusion relation. Finally, let $\Pi$ the class of polytopes which can be obtained taking the convex hull of a set of vertices from a polytope $P^{\square} \in \mathbf{P}^{\square}$. In Figure A.1 we depict the construction in dimension 3 of a polytope $P$ in this class.

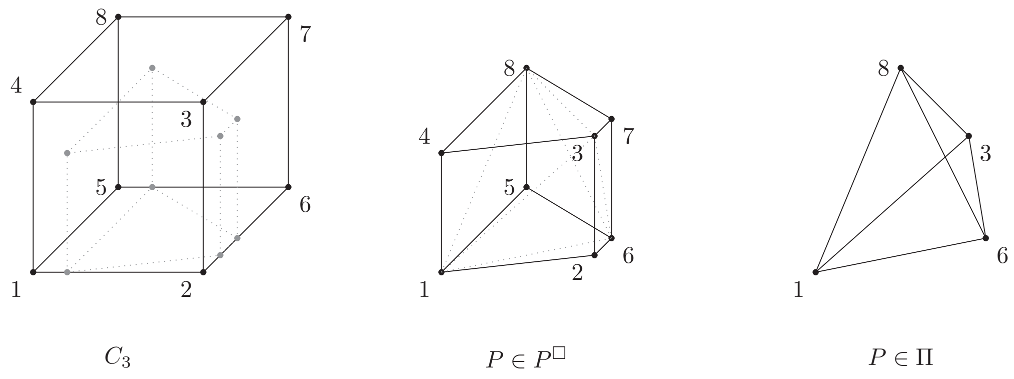


Figure A.1: Obtaining a polytope in class $\Pi$ from the $3-\textit{cube}$ .

In [11] the authors presented a backtracking algorithm to compute the vertex set of a polytope $P$ in class $\Pi$. The algorithm assumes the knowledge of the dual description of $P^{\square} \in \mathbf{P}^{\square}$ and runs in $\mathcal{O}(v \times n \times T_{LP})$ where $T_{LP}$ is the complexity of a feasibility check.

We don't know if the blending polytopes belong to the class $\Pi$ (see below). However, we continue to present this algorithm as it is the basis for the vertex enumeration algorithm that we introduce in Section A.3.

Similar to the dual description of the $d$-cube, we can describe any polytope $P^{\square} \in \mathbf{P}^{\square}$ as

$$P^{\square} = \{\mathbf{x} \in \mathbb{R}^n \mid A^L\mathbf{x} \leq b^L, \quad A^U\mathbf{x} \leq b^U\}$$

where $A^L, A^U \in \mathbb{R}^{n\times n}$, and each vertex of $P^{\square}$ is uniquely determined by $n$ inequalities satisfied with equality.

Geometrically, we have $n$ pairs of inequalities that represent opposite facets of $P^{\square}$ and every vertex $v$ of $P^{\square}$ is determined by a *partition* $\mathscr{L} \cup \mathscr{U}$ of the indices $\{1, \ldots, n\}$ for which $A^L_{\mathscr{L}} = b^L_{\mathscr{L}}$ and $A^U_{\mathscr{U}} = b^U_{\mathscr{U}}$.

Given a polytope $P \in \Pi$, we add the redundant description of its *boundary polytope* $P^{\square}$. The algorithm 1 generates recursively all the partitions $\mathscr{L} \cup \mathscr{U}$ of the rows' indices using the constraints $A^L_{\mathscr{L}}\mathbf{x} = b^L_{\mathscr{L}}$ and $A^U_{\mathscr{L}}\mathbf{x} = b^U_{\mathscr{U}}$ to locate each vertex of $P$ whereas its feasibility is tested using $A\mathbf{x} \leq \mathbf{b}$.

For a set of rows' indices $\mathscr{L}$, $\mathscr{U}$ , let's define the polytope

$$\mathscr{P}(\mathscr{L}, \mathscr{U}) = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}, \quad A^L_{\mathscr{L}}\mathbf{x} = b^L_{\mathscr{L}}, \quad A^U_{\mathscr{U}}\mathbf{x} = b^U_{\mathscr{U}}\}.$$

Then, the algorithm can be described as follows:

**Algorithm 1** Vertex enumeration for polytopes in class $\Pi$

```
1: function $enum(\mathscr{L}, \mathscr{U})$ {initial call with $\mathscr{L} = \mathscr{U} = \emptyset$}
2: if $\mathscr{P}(\mathscr{L}, \mathscr{U}) \neq \emptyset$ then
3:     $last = \max\{\{0\}, \mathscr{L}, \mathscr{U}\}$
4:     if $last == n$ then
5:         output($x$) {the vertex defined by $\mathscr{L}$ and $\mathscr{U}$}
6:     else {$last < n$}
7:         $enum(\mathscr{L} \cup \{last + 1\}, \mathscr{U})$
8:         $enum(\mathscr{L}, \mathscr{U} \cup \{last + 1\})$
9:     end if
10: end if
11: end {enum}
```

The complexity of the algorithm 1 ($\mathcal{O}(nvT_{LP})$) follows from the fact that each vertex $v \in V$ is generated only once and requires at most $n$ feasibility tests. In order to exemplify the algorithm, let's consider the recipe's polytope

$$U = \{\mathbf{u} \in \mathbb{R}^3 \mid 0.2 \leq u_1 \leq 1,\ 0 \leq u_2 \leq 1,\ 0 \leq u_3 \leq 1,\ u_1 + u_2 + u_3 = 1\}$$

which is depicted in the Figure A.2. In this case, we notice that the polytope

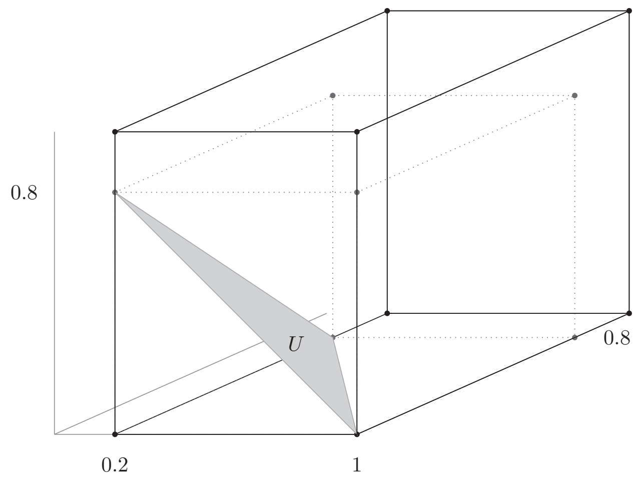


Figure A.2: Recipe's polytope and its boundary polytope $P^{\square}$.

$U$ belongs to the class $\Pi$ as it can be obtained from the polytope

$$U^{\square} = \{\mathbf{u} \in \mathbb{R}^3 \mid 0.2 \leq u_1 \leq 1,\ 0 \leq u_2 \leq 0.8,\ 0 \leq u_3 \leq 0.8,\ u_1 + u_2 + u_3 = 1\}$$

in class $\mathbf{P}^{\square}$ (showed in dotted lines). Then, using both representations $U$ and

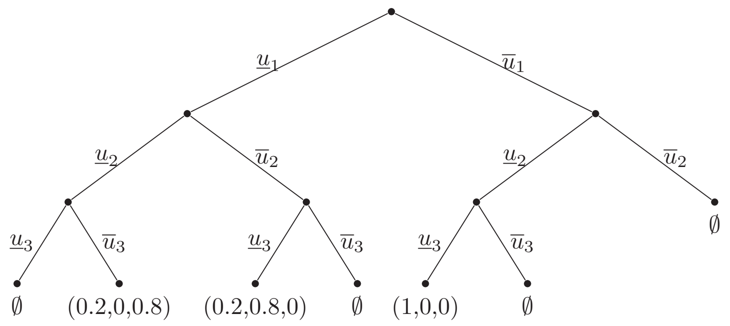


Figure A.3: Search tree generated by algorithm 1.

$U^{\square}$ in algorithm 1, we obtain the search tree outlined in Figure A.3 and the vertex set

$$V_U = \{(0.2, 0, 0.8), (0.2, 0.8, 0), (1, 0, 0)\}.$$

For this example, we were able to obtain the boundary polytope $U^\square \in \mathbf{P}^\square$ of the recipe's polytope $U$ by shrinking all its redundant bounds. Adding $U^\square$ to the descrption of $U$, all the vertices in $V_U$ become degenerate. Each vertex is determined by three bounds and all of them satisfy the equality $u_1 + u_2 + u_3 = 1$. Thus, we can ignore the equality constraint for the location of the vertices and use it only for the feasibility test.

This procedure does not work in general and we don't know if the blending polytopes belong or not to the class $\Pi$. In the Figure A.4 we show the recipe's polytope

$$U_2 = \{\mathbf{u} \in \mathbb{R}^3 \mid 0 \le u_1 \le 0.8,\ 0 \le u_2 \le 1,\ 0 \le u_3 \le 0.8,\ u_1 + u_2 + u_3 = 1\}.$$

In this case, the constraints $0 \le u_1 \le 0.8$, $0 \le u_2 \le 1$, $0 \le u_3 \le 0.8$ form a polytope in $\mathbf{P}^\square$ but $U_2$ can not be obtained as a convex combination of a subset of its vertices. The polytope $U_2$ has four non-degenerate vertices,

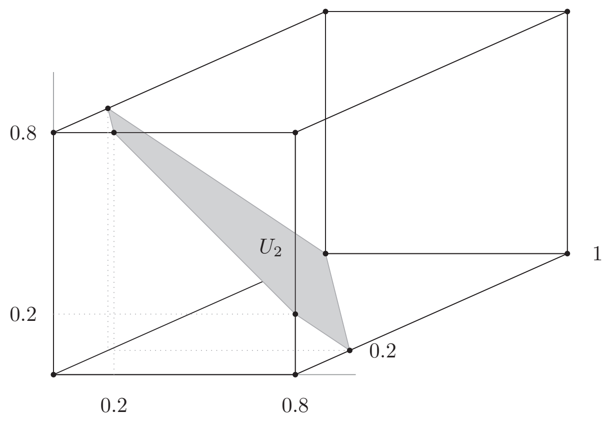


Figure A.4: Recipe's polytope $U$ without a known boundary polytope.

$(0.2, 0, 0.8)$, $(0, 0.2, 0.8)$, $(0.8, 0, 0.2)$ and $(0.8, 0.2, 0)$ which require the use of the equality constraint for their location.

In conclusion, as we don't have a method to produce $U^\square$ (if this exists), then we can not apply the zerOne algorithm to the recipe's polytope. In the next section we present a very fast VE algorithm for the recipe's polytope which explores the rows partitions and exploits the fact that the feasibility test involves only one addition. Before presenting the algorithm, we discuss some issues regarding the number of degenerate vertices of the recipe's polytope.

**Property A.2.1.** *A recipe's polytope $U$ in $\mathbb{R}^n$ can have up to $\binom{n}{\lfloor n/2 \rfloor}$ degenerate vertices.*

**Proof.** The number of degenerate vertices of $U$ equals the number of vertices of the $n$-parallepiped

$$L = \{\mathbf{u} \in \mathbb{R}^n \mid \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}\}$$

lying in the simplex

$$S_1 = \{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}, \quad \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\}.$$

So, let's consider the parallepiped $L_{n,k}$ defined by $\overline{\mathbf{u}} = s/k$ and $\underline{\mathbf{u}} = (1 - s)/(n-k)$ for some $0.5 < s < 1$ and $1 < k < n-1$.

The number of vertices in $L_{n,k}$ determined by $k$ upper bounds and $n-k$ lower bounds is $\binom{n}{k}$ and by construction all of them are in $S_1$. Then, as $\binom{n}{k}$ reaches its maximum when $k = \lfloor n/2 \rfloor$, the recipe's polytope $U$ can have up to $\binom{n}{\lfloor n/2 \rfloor}$ degenerate vertices. □

These pathological polytopes have $\binom{n}{\lfloor n/2 \rfloor}$ vertices all of them degenerate and each vertex is determined by exactly $\binom{n}{n-1} = n$ partitions $\mathscr{L} \cup \mathscr{U}$ of $n-1$ rows' indices.

## A.3 Vertex Enumeration of Blending Polytopes

In this section we introduce a customized VE algorithm for the recipe's polytope $U$. The algorithm is inspired from the zerOne method and exploits the structure of the recipe's polytope. As each vertex $\mathbf{u} \in U$ is determined by $n$ bounds or by $n-1$ bounds and the percentage constraint ($\mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1$), then $u$ is either a vertex of the parallepiped $L$ or it lies in the interior of one of its edges. Thus, we need to search the vertices of $U$ in the edges of $L$ in an efficient manner. The second property that we use is the simplicity of the feasibility test: a partition $p \in \mathscr{L} \cup \mathscr{U}$ is feasible iff the coordinates of the vertex $v(p)$ determined by $p$ sum one.

Let's denote by $\mathbf{s} = \overline{\mathbf{u}} - \underline{\mathbf{u}}$ the slack between the components bounds and by $\mathbf{e_i}$ the $i$-th canonical vector in $\mathbb{R}^n$. For convenience, we denote a partition $p \in \mathscr{L} \cup \mathscr{U}$ as a natural number whose binary representation has the $i$-th bit set to 1 iff $i \in \mathscr{U}$. The set of $2^n$ partitions encode the vertices of $L$ and the function *enumU* presented in the Algorithm 2 conducts a *lexicographic depth first search* on them in order to detect those which are also vertices of $U$.

The function *enumU* is called with three parameters. The partition $p$ to be analyzed, the most significant active bit of $p$ (which corresponds with the biggest index of the upper bounds used in $p$) and the sum of the bounds determining the partition $p$: $sum_p = \sum_{i=1}^{n} p_i \underline{\mathbf{u}}_i + (1 - p_i)\overline{\mathbf{u}}_i$. Initially we call *enumU* with $p = 0$, $last = 0$ and $sum_0 = \sum_{i=1}^{n} \underline{\mathbf{u}}_i$. That is, the method starts the dfs analyzing the vertex of $L$ determined by all the lower bounds. We suppose that $sum_0 < 1$ otherwise $L$ has no intersection with $S_1$ and $U = \emptyset$ or $\underline{\mathbf{u}}$ is the only vertex of $U$.

In general, each partition $p$ being analyzed satisfies $sum_p < 1$. The analysis of $p$ consists in scanning all the neighbor partitions $q$ such that $p <_{Lex} q$. If $sum_q < 1$ then we proceed to analyze $q$. If $sum_q = 1$ then $q$ determines a (degenerate) vertex of $U$ and if $sum_q > 1$ then the edge $(p, q)$ of $L$ contains the vertex $\mathbf{v}(p) + (1 - sum_p)\,\mathbf{e_k}$ of $U$, where $k$ is the only different coordinate between $p$ and $q$.

**Algorithm 2** Vertex enumeration for the recipe's polytopes

```
1:  function $enumU(p, last, sum_p)$
2:  for $i = last + 1$ to $n$ do
3:      $q = p + 2^{i-1}$
4:      $sum_q = sum_p + s_i$
5:      if $sum_q < 1$ then
6:          $enumU(q, i, sum_q)$
7:      else
8:          if $sum_q > 1$ then
9:              output$(\mathbf{v}(p) + (1 - sum_p)\,\mathbf{e_i})$
10:         else
11:             output$(\mathbf{v}(q))$
12:         end if
13:     end if
14: end for
15: for $i = 1$ to $last - 1$ do
16:     $q = p + 2^{i-1}$
17:     $sum_q = sum_p + s_i$
18:     if $(p(i) == 0)$ and $(sum_q > 1)$ then
19:         output$(\mathbf{v}(p) + (1 - sum_p)\,\mathbf{e_i})$
20:     end if
21: end for
22: end {enumU}
```

To show the validity of the method, we notice that any vertex of $U$ is either the extreme point $q$ or an interior point of an edge $(p, q)$ of $L$ where $sum_p < 1 \leq sum_q$. The algorithm analyzes all the partitions $p$ such that $sum_p < 1$ and scans exactly once all the edges $(p, q)$ with $sum_q \geq 1$. Therefore, any vertex of $U$ is enumerated by the algorithm. Finally, a partition $q$ such that $sum_q = 1$ is detected only once (line 11 of the algorithm) because of the use of the lexicographic order to scan the partitions. In this way we avoid the duplication of any degenerate vertex.

In the worst case, the complexity of the algorithm is exponential in the input size. For example, the algorithm requires to analyze all the edges of $L$ ($n2^{n-1}$) in order to find the only vertex of the polytope

$$U = \{\mathbf{u} \in \mathbb{R}^n \mid 0 \leq \mathbf{u} \leq 1/n \;,\; \mathbf{1}^{\mathrm{T}} \cdot \mathbf{u} = 1\}. \tag{A.3}$$

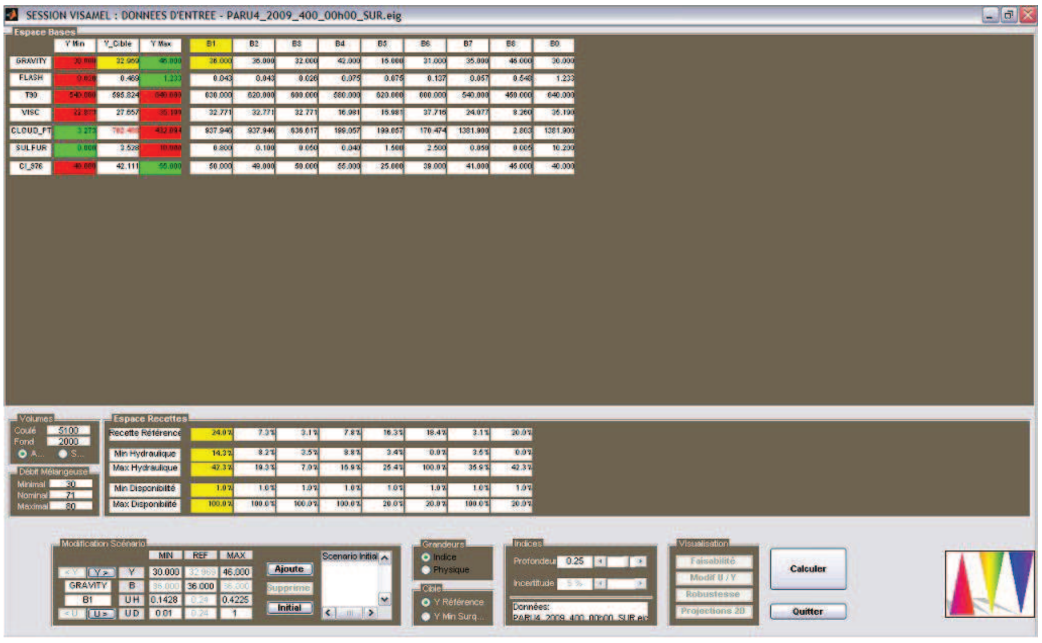
In spite of this, the algorithm has a very good behavior in comparison with two of the best general methods. For $n = 20$, which is the maximum number of components used in practice, the algorithm finds the $184,756$ vertices of the highly degenerate polytope $L_{20,10}$ described at the end of the previous section in 58 seconds. In contrast, the reverse search method ([5]) finds these vertices in more than 29 minutes and the double description method ([20]) takes more than 19 minutes to enumerate the $48,620$ vertices of $L_{18,9}$.

All the previous discussion and the algorithm 2 concern only the recipe's polytope $U$. However, we need to solve also the VE for the polytope $C_{U1}$. Unfortunately, the structure of this polytope depends completely on the components matrix $B$ from which we don't have any information.

In order to enumerate the vertices of $C_{U1}$, we use a backtracking algorithm similar to the algorithm 1. This algorithm generates all the partitions of the indices of $n-1$ rows and it tests its feasibility solving the simplex phase 1 in order to establish if the partition defines a vertex or not. If feasible, the test provides a vertex which is determined by the partition and by the equality constraint $\sum_i^n u_i = 1$. However, by using this approach, we need to deal with the redundant partitions producing the same degenerate vertex. Theoretically, the algorithm can behave badly, nevertheless it performs well in practice. (It solves the VE for any polytope $C_{U1}$ with $n \leq 20$ in less than 1 minute).

# Appendix B
# VISAMEL

In this section we present some screenshots of the decision support system VISAMEL (Blends Visualization) developed for the TOTAL group. Some of the ideas presented in this work have been already implemented in this application while others are being evaluated for their future inclusion.



SESSION VISAMEL : DONNEES D'ENTREE - PARU4_2009_400_00h00_SUR.eig

Espace Bases

| | Y Min | Y_Cible | Y Max | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GRAVITY | 30.000 | 32.969 | 46.000 | 36.000 | 36.000 | 32.000 | 42.000 | 16.000 | 31.000 | 35.000 | 46.000 | 30.000 |
| FLASH | 0.026 | 0.469 | 1.233 | 0.043 | 0.043 | 0.026 | 0.075 | 0.075 | 0.137 | 0.057 | 0.546 | 1.233 |
| T90 | 540.000 | 595.824 | 640.000 | 630.000 | 620.000 | 600.000 | 580.000 | 620.000 | 600.000 | 540.000 | 460.000 | 640.000 |
| VISC | 22.877 | 27.657 | 35.190 | 32.771 | 32.771 | 32.771 | 16.981 | 16.981 | 37.716 | 24.077 | 8.260 | 36.190 |
| CLOUD_PT | 3.273 | 782.405 | 432.894 | 937.946 | 937.946 | 636.017 | 199.057 | 199.057 | 170.474 | 1381.900 | 2.803 | 1381.900 |
| SULFUR | 0.000 | 3.528 | 10.000 | 0.800 | 0.100 | 0.050 | 0.040 | 1.500 | 2.500 | 0.050 | 0.005 | 10.200 |
| CI_976 | 40.000 | 42.111 | 55.000 | 50.000 | 49.000 | 50.000 | 55.000 | 25.000 | 39.000 | 41.000 | 46.000 | 40.000 |

Volumes

| | |
|---|---|
| Coulé | 5100 |
| Fond | 2000 |

[● A...] ○ S...

Débit Mélangeuse

| | |
|---|---|
| Minimal | 30 |
| Nominal | 71 |
| Maximal | 80 |

Espace Recettes

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Recette Référence | 24.0 % | 7.3 % | 3.1 % | 7.8 % | 16.3 % | 18.4 % | 3.1 % | 20.0 % |
| Min Hydraulique | 14.3 % | 8.2 % | 3.5 % | 8.8 % | 3.4 % | 0.0 % | 3.6 % | 0.0 % |
| Max Hydraulique | 42.3 % | 19.3 % | 7.0 % | 16.9 % | 25.4 % | 100.0 % | 35.9 % | 42.3 % |
| Min Disponibilité | 1.0 % | 1.0 % | 1.0 % | 1.0 % | 1.0 % | 1.0 % | 1.0 % | 1.0 % |
| Max Disponibilité | 100.0 % | 100.0 % | 100.0 % | 100.0 % | 20.0 % | 20.0 % | 100.0 % | 20.0 % |

Modification Scénario

| | | MIN | REF | MAX |
|---|---|---|---|---|
| < Y / Y > | Y | 30.000 | 32.969 | 46.000 |
| GRAVITY | B | 36.000 | 36.000 | 36.000 |
| B1 | U H | 0.1428 | 0.24 | 0.4225 |
| < U / U > | U D | 0.01 | 0.24 | 1 |

Scenario Initial

Grandeurs: [● Indice] ○ Physique

Cible: [● Y Référence] ○ Y Min Surq...

Indices

Profondeur: 0.25

Incertitude: 5 %

Données:
PARU4_2009_400_00h00_SUR.eig

Figure B.1: VISAMEL main screen.

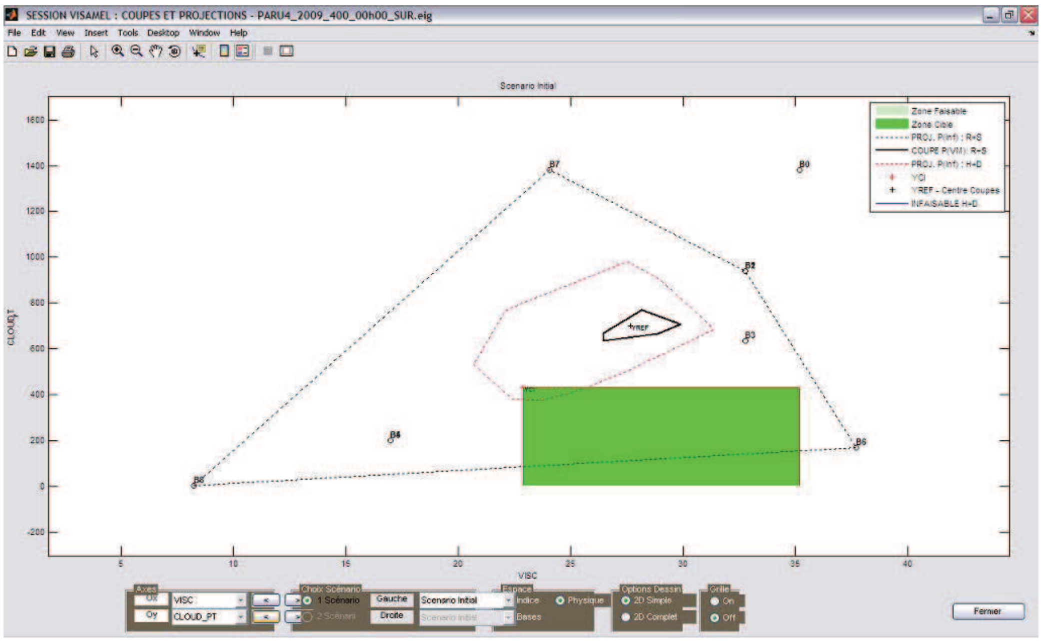
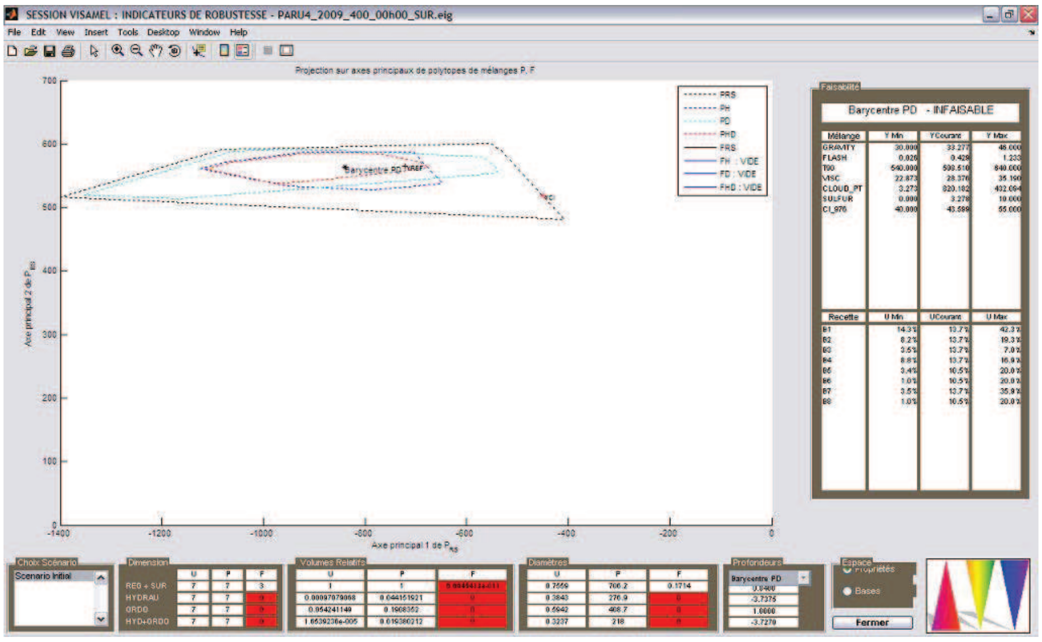


Figure B.2: Projection of the polytope $P$ and the target region.


SESSION VISAMEL : INDICATEURS DE ROBUSTESSE - PARU4_2009_400_00h00_SUR.eig
File Edit View Insert Tools Desktop Window Help
Projection sur axes principaux de polytopes de mélanges P, F
PRS
PH
PD
PHD
FRS
FH : VIDE
FD : VIDE
FHD : VIDE
Barycentre PD
YREF
YCI
Axe principal 2 de P_RS
Axe principal 1 de P_RS
Fermer

Faisabilité

Barycentre PD - INFAISABLE

| Mélange | Y Min | YCourant | Y Max |
|---|---|---|---|
| GRAVITY | 30.000 | 33.277 | 46.000 |
| FLASH | 0.020 | 0.428 | 1.233 |
| T90 | 540.000 | 509.510 | 640.000 |
| VISC | 22.873 | 28.376 | 35.190 |
| CLOUD_PT | 3.273 | 820.182 | 432.094 |
| SULFUR | 0.000 | 3.278 | 10.000 |
| CI_976 | 40.000 | 43.599 | 55.000 |

| Recette | U Min | UCourant | U Max |
|---|---|---|---|
| B1 | 14.3% | 13.7% | 42.3% |
| B2 | 8.2% | 13.7% | 19.3% |
| B3 | 3.5% | 13.7% | 7.0% |
| B4 | 8.8% | 13.7% | 16.9% |
| B5 | 3.4% | 10.5% | 20.0% |
| B6 | 1.0% | 10.5% | 20.0% |
| B7 | 3.5% | 13.7% | 35.9% |
| B8 | 1.0% | 10.5% | 20.0% |

Choix Scénario: Scenario Initial

Dimension

| | U | P | F |
|---|---|---|---|
| REG + SUR | 7 | 7 | 3 |
| HYDRAU | 7 | 7 | 0 |
| ORDO | 7 | 7 | 0 |
| HYD+ORDO | 7 | 7 | 0 |

Volumes Relatifs

| U | P | F |
|---|---|---|
| 1 | 1 | 0.00454134e-011 |
| 0.00097079068 | 0.044151921 | 0 |
| 0.054241149 | 0.1908352 | 0 |
| 1.6539236e-005 | 0.019380212 | 0 |

Diamètres

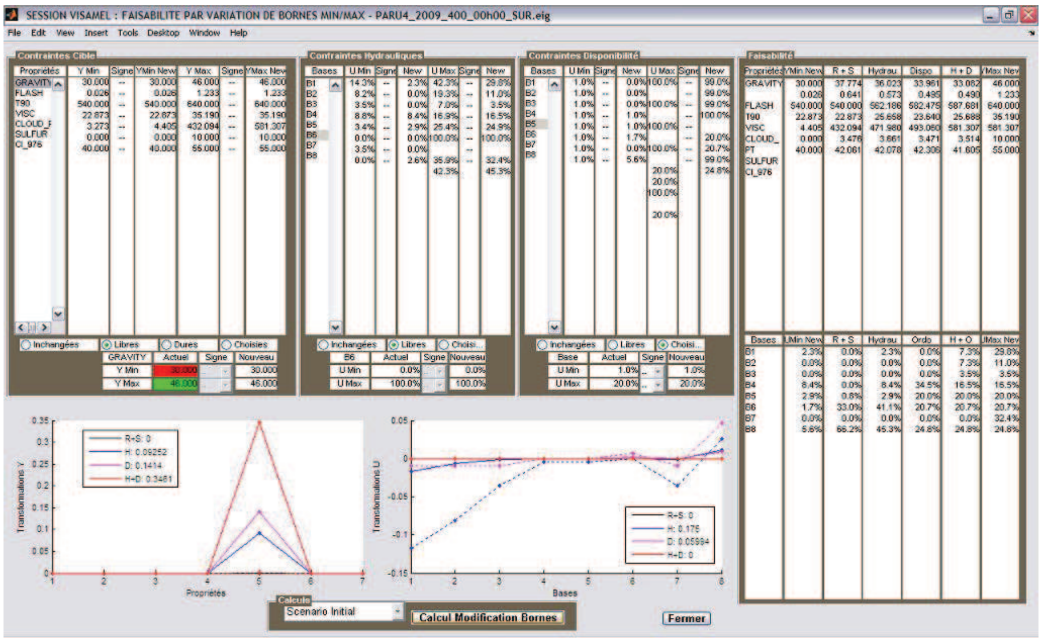
| U | P | F |
|---|---|---|
| 0.7559 | 706.2 | 0.1714 |
| 0.3843 | 276.9 | 0 |
| 0.5942 | 408.7 | 0 |
| 0.3237 | 218 | 0 |

Profondeurs: Barycentre PD

| |
|---|
| 0.0400 |
| -3.7375 |
| 1.0000 |
| -3.7270 |

Espace: Propriétés, Bases

Figure B.3: Some geometric measures of robustness.



SESSION VISAMEL : FAISABILITE PAR VARIATION DE BORNES MIN/MAX - PARU4_2009_400_00h00_SUR.eig

Contraintes Cible

| Propriétés | Y Min | Signe | YMin New | Y Max | Signe | YMax New |
|---|---|---|---|---|---|---|
| GRAVITY | 30.000 | -- | 30.000 | 46.000 | -- | 46.000 |
| FLASH | 0.026 | -- | 0.026 | 1.233 | -- | 1.233 |
| T90 | 540.000 | -- | 540.000 | 640.000 | -- | 640.000 |
| VISC | 22.873 | -- | 22.873 | 35.190 | -- | 35.190 |
| CLOUD_F | 3.273 | -- | 4.405 | 432.094 | -- | 581.307 |
| SULFUR | 0.000 | -- | 0.000 | 10.000 | -- | 10.000 |
| CI_976 | 40.000 | -- | 40.000 | 55.000 | -- | 55.000 |

Inchangées [✓ Libres] Dures Choisies

| GRAVITY | Actuel | Signe | Nouveau |
|---|---|---|---|
| Y Min | 30.000 | | 30.000 |
| Y Max | 46.000 | | 46.000 |

Contraintes Hydrauliques

| Bases | U Min | Signe | New | U Max | Signe | New |
|---|---|---|---|---|---|---|
| B1 | 14.3% | -- | 2.3% | 42.3% | -- | 29.8% |
| B2 | 8.2% | -- | 0.0% | 19.3% | -- | 11.0% |
| B3 | 3.5% | -- | 0.0% | 7.0% | -- | 3.5% |
| B4 | 8.8% | -- | 8.4% | 16.9% | -- | 16.5% |
| B5 | 3.4% | -- | 2.9% | 25.4% | -- | 24.9% |
| B6 | 0.0% | -- | 0.0% | 100.0% | -- | 100.0% |
| B7 | 3.5% | -- | 0.0% | | -- | |
| B8 | 0.0% | -- | 2.6% | 35.9% | -- | 32.4% |
| | | | | 42.3% | | 45.3% |

Inchangées [✓ Libres] Choisi...

| B6 | Actuel | Signe | Nouveau |
|---|---|---|---|
| U Min | 0.0% | | 0.0% |
| U Max | 100.0% | | 100.0% |

Contraintes Disponibilité

| Bases | U Min | Signe | New | U Max | Signe | New |
|---|---|---|---|---|---|---|
| B1 | 1.0% | -- | 0.0% | 100.0% | -- | 99.0% |
| B2 | 1.0% | -- | 0.0% | | -- | 99.0% |
| B3 | 1.0% | -- | 0.0% | 100.0% | -- | 99.0% |
| B4 | 1.0% | -- | 1.0% | | -- | 100.0% |
| B5 | 1.0% | -- | 1.0% | 100.0% | -- | |
| B6 | 1.0% | -- | 1.7% | | -- | 20.0% |
| B7 | 1.0% | -- | 0.0% | 100.0% | -- | 20.7% |
| B8 | 1.0% | -- | 5.6% | | -- | 99.0% |
| | | | | 20.0% | | 24.8% |
| | | | | 20.0% | | |
| | | | | 100.0% | | |
| | | | | | | |
| | | | | 20.0% | | |

Inchangées Libres [✓ Choisi...]

| Base | Actuel | Signe | Nouveau |
|---|---|---|---|
| U Min | 1.0% | -- | 1.0% |
| U Max | 20.0% | -- | 20.0% |

Faisabilité

| Propriétés | YMin New | R + S | Hydrau | Dispo | H + D | YMax New |
|---|---|---|---|---|---|---|
| GRAVITY | 30.000 | 37.774 | 36.023 | 33.961 | 33.082 | 46.000 |
| | 0.026 | 0.641 | 0.573 | 0.495 | 0.490 | 1.233 |
| FLASH | 540.000 | 540.000 | 562.186 | 582.475 | 587.681 | 640.000 |
| T90 | 22.873 | 22.873 | 25.658 | 23.640 | 25.688 | 35.190 |
| VISC | 4.405 | 432.094 | 471.980 | 493.060 | 581.307 | 581.307 |
| CLOUD_ | 0.000 | 3.476 | 3.661 | 3.471 | 3.514 | 10.000 |
| PT | 40.000 | 42.061 | 42.078 | 42.306 | 41.605 | 55.000 |
| SULFUR | | | | | | |
| CI_976 | | | | | | |

| Bases | UMin New | R + S | Hydrau | Ordo | H + O | UMax New |
|---|---|---|---|---|---|---|
| B1 | 2.3% | 0.0% | 2.3% | 0.0% | 7.3% | 29.8% |
| B2 | 0.0% | 0.0% | 0.0% | 0.0% | 7.3% | 11.0% |
| B3 | 0.0% | 0.0% | 0.0% | 0.0% | 3.5% | 3.5% |
| B4 | 8.4% | 0.0% | 8.4% | 34.5% | 16.5% | 16.5% |
| B5 | 2.9% | 0.8% | 2.9% | 20.0% | 20.0% | 20.0% |
| B6 | 1.7% | 33.0% | 41.1% | 20.7% | 20.7% | 20.7% |
| B7 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 32.4% |
| B8 | 5.6% | 66.2% | 45.3% | 24.8% | 24.8% | 24.8% |

Figure B.4: Analysis of bounds modifications.

# Bibliography

[1] *Computation of certain measures of proximity between convex polytopes: a complexity viewpoint*, 1992. 8

[2] S. D. Abdullahi, M. E. Dyer, and L. G. Proll. Listing Vertices of Simple Polyhedra Associated with Dual LI(2) Systems. page 221. 2003. 73

[3] A. V. Aho and J. E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1974. 72

[4] P. Amaral, J. Júdice, and H. D. Sherali. A reformulation-linearization-convexification algorithm for optimal correction of an inconsistent system of linear constraints. *Comput. Oper. Res.*, 35(5):1494–1509, May 2008. 38

[5] D. Avis. *lrs home page*. School of Computer Science, Montreal, Canada, 1993. 73, 79

[6] D. Avis and D. Bremner. How Good are Convex Hull Algorithms? *Computational Geometry: Theory and Applications*, 7:265–301, 1995. 72

[7] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8(3):295–313, December 1992. 7, 73

[8] M. L. Balinski. *An algorithm for finding all vertices of convex polyhedral sets*. PhD thesis, Princeton, NJ, USA, 1959. 72

[9] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization (Princeton Series in Applied Mathematics)*. Princeton University Press, August 2009. iv, xii, 2, 24

[10] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. 8

[11] M. Bussieck and M. Lübbecke. The vertex set of a 0/1-polytope is strongly P-enumerable. *Computational Geometry*, 11(2):103–109, October 1998. 7, 71, 73

[12] N. Chakravarti. Some results concerning post-infeasibility analysis. *European Journal of Operational Research*, 73(1):139–143, February 1994. 37

[13] M. Chèbre, Y. Creff, and N. Petit. Feedback control and optimization for the production of commercial fuels by blending. *Journal of Process Control*, 20(4):441–451, April 2010. iii, v, viii, 1, 3, 15

[14] J. W. Chinneck. *Feasibility and Infeasibility in Optimization:: Algorithms and Computational Methods (International Series in Operations Research & Management Science)*. Springer, 1 edition, December 2007. 36, 37

[15] T. Christof and A. Löbel. *PORTA: Polyhedron representation transformation algorithm*, 1997. 73

[16] V. Chvátal. *Linear Programming (Series of Books in the Mathematical Sciences)*. W. H. Freeman, September 1983. 9, 72

[17] O. Daoudi. *Zonotopes et zonoïdes : études et applications aux processus de la séparation.* PhD thesis, Université Joseph-Fourier - Grenoble I, October 1995. v, 3

[18] M. E. Dyer. The Complexity of Vertex Enumeration Methods. *Mathematics of Operations Research*, 8(3):381–402, 1983. 72

[19] M. E. Dyer and L. G. Proll. An algorithm for determining all extreme points of a convex polytope. *Mathematical Programming*, 12(1):81–96, December 1977. 72

[20] K. Fukuda. *cdd/cdd+ Reference Manual*. Institute for Operations Research, Zurich, Switzerland, 1995. 73, 79

[21] K. Fukuda and A. Prodon. Double description method revisited. pages 91–111. 1996. 72

[22] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, first edition edition, January 1979. 5

[23] J. H. Gary, G. E. Handwerk, and M. J. Kaiser. *Petroleum Refining: Technology and Economics, Fifth Edition*. CRC Press, 5 edition, March 2007. iii, 1

[24] D. Girard and P. Valentin. *Zonotopes and Mixtures management*. International Series of Numerical Mathematics. Birkhäuser Verlag, 1989. v, 3

[25] H. J. Greenberg. An empirical analysis of infeasibility diagnosis for instances of linear programming blending models. *IMA Journal of Management Mathematics*, 4(2):163–210, January 1992. 37

[26] E. Hendrix. Finding robust solutions for product design problems. *European Journal of Operational Research*, 92(1):28–36, July 1996. v, 3

[27] L. Khachiyan, E. Boros, K. Borys, K. M. Elbassioni, and V. Gurvich. Generating All Vertices of a Polyhedron Is Hard. *Discrete & Computational Geometry*, 39(1-3):174–190, 2008. 7, 72

[28] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan. Polynomial Solvability of Convex Quadratic Programming. *Doklady Akademiia Nauk SSSR*, 248, 1979. 9

[29] B. Lacolle and P. Valentin. Modélisation géométrique de la faisabilité de plusieurs mélanges. *Modélisation mathématique et analyse numérique*, 27(3):313–348, 1993. v, 3

[30] M. Lübbecke. *zerOne*, 1999. 73

[31] T. H. Mattheiss. An Algorithm for Determining Irrelevant Constraints and all Vertices in Systems of Linear Inequalities. *Operations Research*, 21(1):247–260, 1973. 72

[32] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. *The double description method, in Contributions to the Theory of Games*, volume II, pages 51–73. 1953. 72

[33] K. G. Murty, S. N. Kabadi, and R. Chandrasekaran. Infeasibility Analysis for Linear Systems, a survey. *Arabian Journal of Science and Technology*, 25(1C):3–18, 2000. 38

[34] N. Odeh. *Modélisation mathématique des propriétés de mélanges : B-splines et optimisation avec conditions de forme.* PhD thesis, Université Joseph-Fourier - Grenoble I, March 1990. iii, 1

[35] J. S. Provan. Efficient enumeration of the vertices of polyhedra associated with network LP's. *Mathematical Programming*, 63(1):47–64, January 1994. 73

[36] R. Seidel. Output-size sensitive algorithms for constructive problems in computational geometry. Technical report, Cornell University, Ithaca, NY, USA, September 1986. 72

[37] R. Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Comput. Geom. Theory Appl.*, 5(2):115–116, September 1995. 7, 72

[38] Y. Serpemen, F. W. Wenzel, and A. Hubel. Blending technology key to making new gasolines. *Oil and Gas Journal*, 89(11), 1991. v, 3

[39] A. Singh. Model-based real-time optimization of automotive gasoline blending operations. *Journal of Process Control*, 10(1):43–58, February 2000. v, 3, 23

[40] K. H. Slaoui. *Application des techniques mathématiques à la gestion des mélanges : histosplines et optimisation.* PhD thesis, Institut National Polytechnique de Grenoble - INPG, June 1986. v, 3

[41] N. Szafran. *Zonoèdres : de la géométrie algorithmique à la théorie de la séparation.* PhD thesis, Université Joseph-Fourier - Grenoble I, October 1991. v, 3

[42] L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. 7, 72

[43] P. J. Vermeer, C. C. Pedersen, W. M. Canney, and J. S. Ayala. Blend-control system all but eliminates reblends for Canadian refiner. *Oil and Gas Journal*, 95(30):74+, 1997. v, 3

[44] W. Wang, Z. Li, Q. Zhang, and Y. Li. On-line optimization model design of gasoline blending system under parametric uncertainty. In *Proceedings of the 15th Mediterranean Conference on Control & Automation*, 2007. v, 3

[45] R. Webster. *Convexity (Oxford Science Publications).* Oxford University Press, USA, 1995. 5, 8

[46] J. White and F. Hall. Gasoline blending optimization cuts use of expensive components. *Oil and Gas Journal*, 90(45):81–84, 1992. v, 3

[47] Y. Zhang. Results analysis for trust constrained real-time optimization. *Journal of Process Control*, 11(3):329–341, June 2001. iv, xi, 2, 23

[48] Y. Zhang, D. Monder, and J. Fraser Forbes. Real-time optimization under parametric uncertainty: a probability constrained approach. *Journal of Process Control*, 12(3):373–389, April 2002. v, 3

[49] G. M. Ziegler. *Lectures on Polytopes (Graduate Texts in Mathematics).* Springer, 2001. 5

## Robustesse et visualisation de production de mélanges

**Résumé :** Le procédé de fabrication de mélanges (PM) consiste à déterminer les proportions optimales à mélanger d'un ensemble de composants de façon que le produit obtenu satisfasse un ensemble de spécifications sur leurs propriétés. Deux caractéristiques importantes du problème de mélange sont les bornes dures sur les propriétés du mélange et l'incertitude répandue dans le procédé. Dans ce travail, on propose une méthode pour la production de mélanges robustes en temps réel qui minimise le coût de la recette et la surqualité du mélange. La méthode est basée sur les techniques de l'Optimisation Robuste et sur l'hypothèse que les lois des mélange sont linéaires. On exploite les polytopes sous-jacents pour mesurer, visualiser et caractériser l'infaisabilité du PM et on analyse la modification des bornes sur les composants pour guider le procédé vers le "meilleur" mélange robuste. On propose un ensemble d'indicateurs et de visualisations en vue d'offrir une aide à la décision.
**Mots clés : Problème de mélanges, Optimisation Robuste, Polyhèdres, Infaisabilité, Visualisation**

## Robustness and visualization of blends production

**Abstract:** The oil blending process (BP) consists in determining the optimal proportions to blend from a set of available components such that the final product fulfills a set of specifications on their properties. Two important characteristics of the blending problem are the hard bounds on the blend's properties and the uncertainty pervading the process. In this work, a real-time optimization method is proposed for producing robust blends while minimizing the blend quality giveaway and the recipe's cost. The method is based on the Robust Optimization techniques and under the assumption that the components properties blend linearly. The blending intrinsic polytopes are exploited in order to measure, visualize and characterize the infeasibility of the BP. A fine analysis of the components bounds modifications is conducted to guide the process towards the "best" robust blend. A set of indices and visualizations provide a helpful support for the decision maker.
**Keywords: Blending Problem, Robust Optimization, Polyhedra, Infeasibility Analysis, Visualization**