Normes des limites articulaires

Après avoir obtenu un espace de travail satisfaisant avec une qualité acceptable, nous pouvons également essayer de nous éloigner les limites articulaires. Cette fonction objectif fut mise en oeuvre pour explorer des régions de l'espace des paramètres de conception où des articulations simples avec un mouvement limité peuvent être utilisées. Il n'est pas simple de définir la distance entre l'orientation d'une articulation et ses limites pour une rotule. Nous pouvons proposer la projection du vecteur tourné sur un plan pour définir la distance à ses limites, comme représenté Figure 5.25, bien que ce ne soit pas la représentation la plus précise. Comme nous analysons les mêmes rotules, les distances n'étaient pas pondérées et la somme des distances a été utilisée comme norme de limite articulaire dans une certaine configuration (q_{pi}). La somme de q_{pi} sur l'espace de travail est utilisée comme une évaluation, tout en maximisant la distance entre les limites articulaires.

$$\|q_{pi}\| := \sum_{n=1}^{4} (l_{limit} - l_{current})_n, \quad n : no.of joints \tag{5.42}$$

$$\|q_p\| := \sum_{i=1}^{W} \|q_{pi}\| \tag{5.43}$$

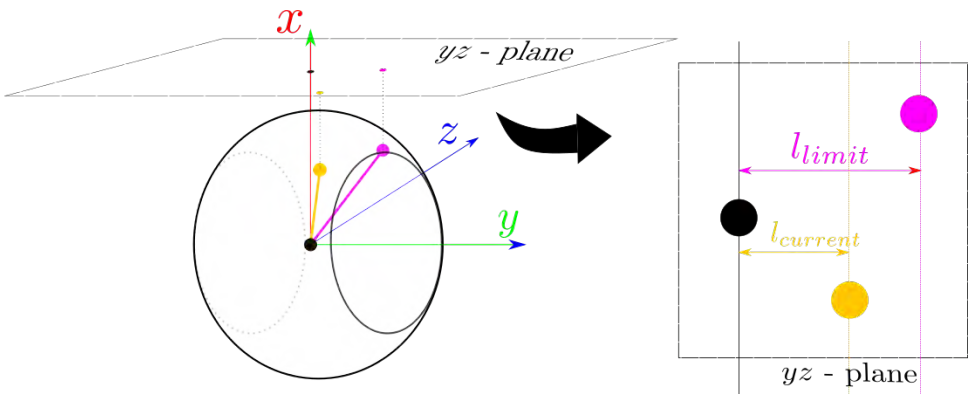


FIGURE 5.25 – Illustration de la distance de la limite articulaire.

5.3.3.2 Résumé des fonctions objectif utilisées

Ainsi, pour résumer les fonctions objectif étudiés dans cette méthodologie, nous cherchons à :

1. Maximiser l'ensemble des postures accessibles $\mathcal{F}$ dans l'espace de travail souhaité ;

2. Maximiser le conditionnement global dans l'espace souhaité ;
3. Maximiser la distance des articulations passives à leurs limites respectives.

Cela signifie mathématiquement :

$$\text{Fonctions objectif} : \max\left(\mathcal{F} \bigcap W_s\right) \tag{5.44}$$

$$\max\left(\kappa_g^{-1}\right) \tag{5.45}$$

$$\max\left(\|q_p\|\right) \tag{5.46}$$

5.3.4 Contraintes d'optimisation

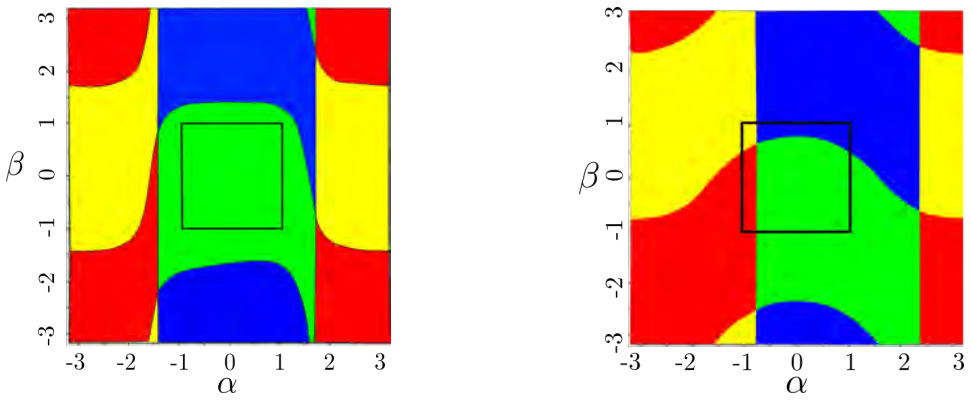
Les mécanismes parallèles ont deux caractéristiques distinctes d'une architecture en série :

1. présence d'articulations passives dont l'orientation peut être calculée, mais non explicitement contrôlée,
2. présence de plusieurs jambes - chaînes en séries connectant l'effecteur terminal à la base.

Ces deux points sont importants car ils affectent l'espace de travail du mécanisme. Ainsi, les limites articulaires passives et l'évitement de collisions internes entre les différentes jambes sont deux contraintes importantes à inclure dans notre problème d'optimisation. L'espace $\mathcal{K}$ du mécanisme est séparé par des courbes de singularité, entraînant la formation de plusieurs régions connectées nommées *aspects* [133]. Comme il n'est pas possible de générer une trajectoire contrôlable d'un *aspect* à un autre, il est important que l'espace de travail souhaité (W_s) soit inclus dans un seul *aspect*. La Figure 5.26a illustre un ensemble valide de paramètres, alors que la Figure 5.26b correspond à une architecture non utilisable pour notre application. Si la boîte dans la Figure 5.26 se compose de plusieurs couleurs, alors il est évident qu'il y a plus d'un *aspect* et qu'il n'est alors pas possible de générer des trajectoire entre toutes les configurations de l'espace de travail souhaité.

On peut se concentrer plus spécifiquement sur la gamme de l'articulation prismatique à choisir pour maximiser l'ensemble des postures accessibles dans l'espace de travail souhaité, c'est-à-dire maximiser $\mathcal{F} \bigcap W_s$. Généralement, une articulation prismatique est exprimée sous forme de contrainte avec une certaine plage minimale et maximale, et d'une contrainte sur le ratio de la longueur à l'état actionné et de la longueur par défaut :

$$\rho_{min} \leq \rho \leq \rho_{max} \tag{5.47}$$



(a) Espace de travail souhaité inclus dans une seule région connectée, ou (*aspect*).

(b) Espace de travail souhaité entrecoupé par les courbes de singularité.

FIGURE 5.26 – Les *aspects* de l'espace $\mathcal{K}$ selon différents cas.

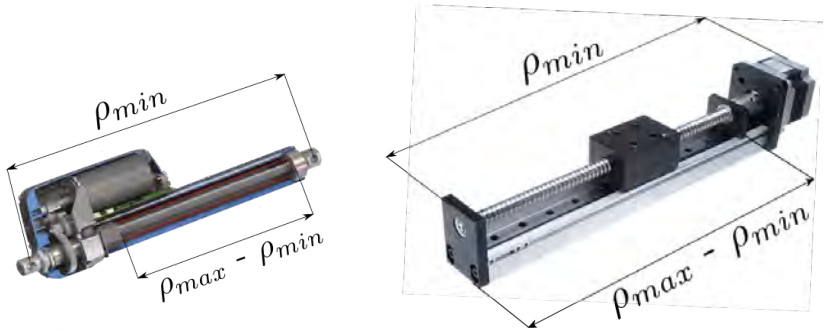


FIGURE 5.27 – Articulations prismatiques et la relation entre ρ_{min} et ρ_{max}
Source: Hanpose linear actuator HPV5 SFU1204

$$\rho_{max} \leq s\ \rho_{min}, \quad s \in [1, 2] \tag{5.48}$$

L'équation 5.48 provient de la construction physique des articulations prismatiques. Si la longueur non étendue de l'actionneur est ρ_{min}, alors il n'est pas courant pour les articulations prismatiques de s'étendre au-delà de leur longueur d'origine ($\rho_{max} < 2\rho_{min}$), comme illustré sur la Figure 5.27. La nouveauté dans l'expression de la gamme d'actionneurs dans notre travail est que nous n'avons pas de valeur statique comme limite (5.47), c'est-à-dire que nous exprimons la contrainte uniquement en fonction du ratio de course exprimé en (5.48). Cela nous permet de choisir les meilleures gammes d'actionneurs pour maximiser l'espace de travail souhaité sans imposer de contrainte sur la taille minimale ou maximale du joint prismatique. Cela est illustré par la Figure 5.28, et implémenté dans la section 5.3.6.4.

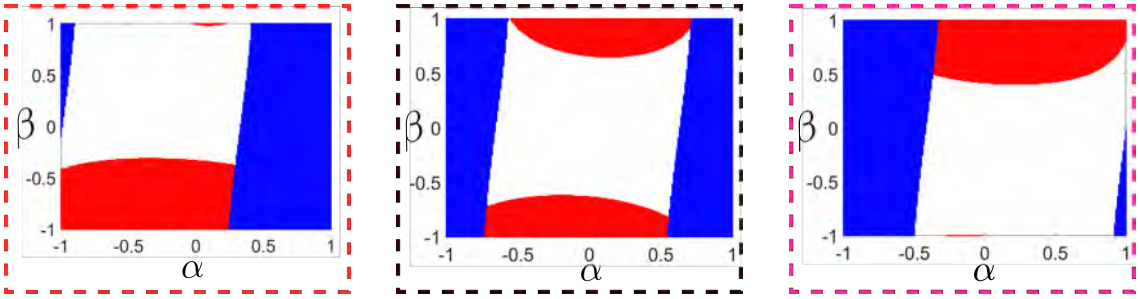


FIGURE 5.28 – Comparaison de l'espace de travail souhaité (en blanc) dans le W_s pour différentes gammes d'actionneurs. Les parties rouge et bleu représentent les violations dues aux longueurs des actionneurs des jambes 1 et 2 respectivement.

5.3.5 Variables d'optimisation

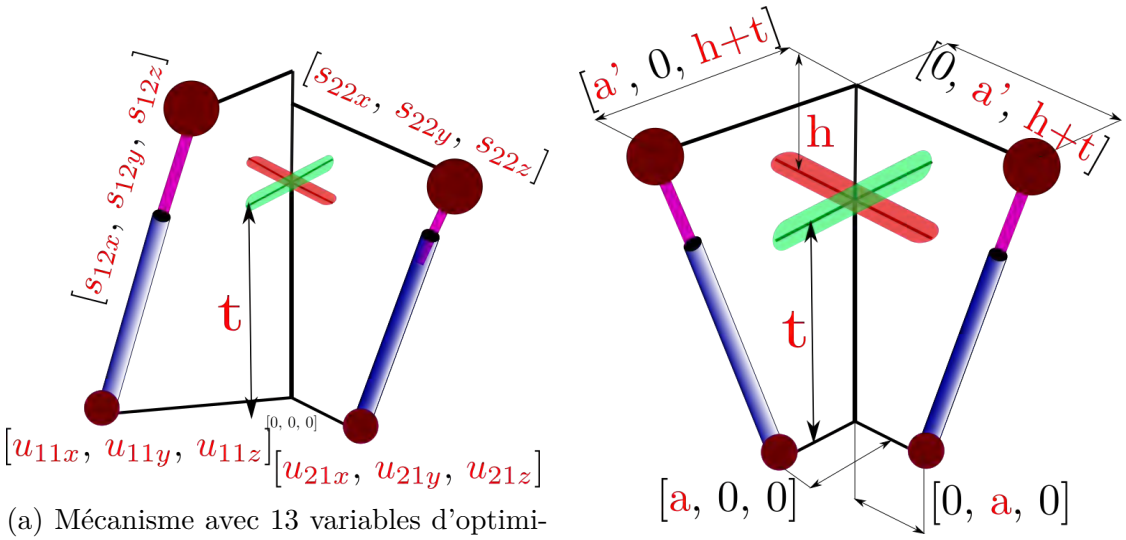
Les variables d'optimisation sont les paramètres qui sont modifiés afin d'optimiser la fonction objectif. Dans un mécanisme, la position ou l'orientation des articulations dans la chaîne cinématique sont les paramètres de conception à optimiser. Le nombre de variables forme la dimension du problème d'optimisation, qui détermine la taille de l'espace de recherche (également désigné par $\mathcal{O}$) et donc la capacité de calcul. Lors du choix des variables d'optimisation, les options suivantes doivent être prises en compte :

— utiliser une approche générale, avec 3 variables x, y, z pour chaque articulation (Figure 5.29a).

— utiliser l'intuition humaine pour fixer certaines variables afin de réduire l'espace de recherche.

Chacune de ces options présente des avantages et des inconvénients. En utilisant une approche générale pour optimiser la position des articulations, il est alors possible de trouver des résultats contre-intuitifs mais efficaces. Cela représente un coût de calcul important puisqu'il existe de nombreuses articulations dans un mécanisme parallèle. Par exemple, notre architecture présente deux jambes avec 6 DDL et une jambe avec 2 DDL. Dans un cas utilisant une combinaison de pivots pour former un cardan et une rotule, nous avons 14 articulations (6 DDL $\times$ 2 jambes + 2 DDL $\times$ 1 jambe). Il en résulte un espace d'optimisation de dimension 42, qui représente un espace très large au vu de la nature des contraintes et du temps requis pour calculer la fonction objectif pour une configuration particulière. Dans l'autre option, il est possible de réduire drastiquement la dimension de $\mathcal{O}$ en utilisant l'intuition humaine, en respectant une symétrie dans le mécanisme par exemple. Par exemple, dans la variation 2U<u>P</u>S-U, nous pouvons réduire

l'espace d'optimisation à la dimension 4, comme illustré sur la Figure 5.29b, en utilisant des liaisons de même longueur et en supposant que les jambes 1 et 2 sont identiques du point de vue de la fabrication et de l'assemblage. Cependant, le choix de fixer certaines variables risque cependant de nous faire manquer certaines configurations qui auraient été plus performantes.



(a) Mécanisme avec 13 variables d'optimisation, en rouge.

(b) Mécanisme réduit à la dimension 4 par l'intuition humaine (variables en rouge).

FIGURE 5.29 – Différentes définitions d'une même architecture, en variant le nombre des variables d'optimisation. A gauche, aucune variable n'est fixée pour la position des articulations. A droite, nous supposons des longueurs égales, une symétrie et des jambes perpendiculaires.

La Figure 5.30 montre l'espace dans lequel les cardans et les rotules peuvent se situer. L'angle de cet espace est de 200 degrés avec un minimum et maximum. Sa hauteur détermine la coordonnée selon l'axe z des articulations. La travée bleue est par rapport à la base du robot, et celle en vert par rapport au cardan de la troisième jambe passive. Lorsque la dimension est réduite à 4 comme sur la Figure 5.29b, les 2 arcs cylindriques se réduisent à 4 carrés dans 2 plans perpendiculaires.

5.3.6 Algorithme de recherche locale

L'algorithme de Nelder-Mead est un algorithme d'optimisation non linéaire [134], aussi nommé *downhill-simplex algorithm* car utilisant la notion de *simplexes* (polytope de N+1 sommets dans un espace à N dimensions). Partant initialement d'un tel simplexe, celui-ci

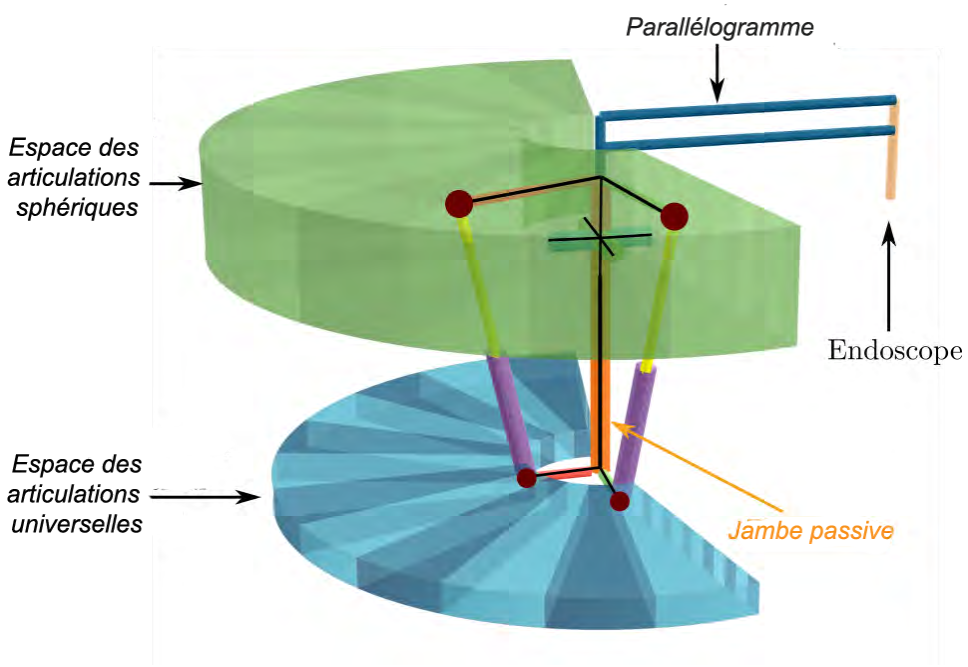


FIGURE 5.30 – Les articulations inférieures et supérieures de chaque jambe peuvent être placées dans les espaces cylindriques représentés en bleu et en vert.

subit des transformations simples au cours des itérations : il se déforme, se déplace et se réduit progressivement jusqu'à ce que ses sommets se rapprochent d'un point où la fonction est localement minimale. Dans cette section, nous discuterons, après la présentation de cet algorithme, de sa mise en oeuvre dans l'optimisation mécanique.

5.3.6.1 L'algorithme de Nelder-Mead (NM)

Pour un $\mathcal{O}$ de dimension n, nous avons besoin d'un simplexe d'au moins n+1 points dans $\mathcal{O}$, afin d'éviter une *convergence prématurée*. Dans notre cas, nous commençons avec un simplexe de n+1 points $(\mathbf{v_0}, \mathbf{v_1}, ...\mathbf{v_n})$ tel que la fonction objectif évaluée du sommet i^{th} ait une valeur meilleure ou égale à celle du sommet $(i+1)^{th}$. Un point moyen $(\mathbf{v_m})$ est calculé en excluant le pire point.

$$\mathbf{v_m} := \frac{\sum_{i=0}^{n-1} \mathbf{v_i}}{n} \tag{5.49}$$

L'algorithme d'optimisation compare ensuite le point moyen et recherche de meilleurs points par des opérations géométriques appelées (*i*) réflexion, (*ii*) expansion, (*iii*) contraction et (*iv*) rétrécissement. Ces opérations peuvent être expliquées comme suit :

— Réflexion :

$$\mathbf{v_{reflect}} := \mathbf{v_m} + r\,(\mathbf{v_m} - \mathbf{v_n}), \quad r := \text{coefficient de réflexion}\,(r > 0) \tag{5.50}$$

— Expansion :

$$\mathbf{v_{expand}} := \mathbf{v_m} + e\,(\mathbf{v_{reflect}} - \mathbf{v_m}), \quad e := \text{coefficient d'expansion}\,(e > 1) \tag{5.51}$$

— Contraction externe :

$$\mathbf{v_{oc}} := \mathbf{v_m} + k\,(\mathbf{v_m} - \mathbf{v_n}), \quad k := \text{Coefficient de contraction}\,(0 < k < r) \tag{5.52}$$

— Contraction interne :

$$\mathbf{v_{ic}} := \mathbf{v_m} - k\,(\mathbf{v_m} - \mathbf{v_n}), \quad k := \text{Coefficient de contraction} \tag{5.53}$$

— Rétrécissement :

$$\forall\, i \in [1, n] \quad \mathbf{v_i} = s\,.\mathbf{v_i}, \quad s := \text{Facteur de rétrécissement}\,(0 < s < 1) \tag{5.54}$$

Le nouveau point ($\mathbf{v_{new}}$) introduit dans le simplexe dépend de l'évaluation de $\mathbf{v_{reflect}}$, $\mathbf{v_{expand}}$, $\mathbf{v_{oc}}$ et $\mathbf{v_{ic}}$. L'opération est poursuivie jusqu'à ce que le critère d'arrêt soit atteint. Le simplexe s'arrête s'il rétrécie en dessous d'une certaine valeur ϵ_1 et l'évaluation de chaque sommet du simplexe rétréci varie en fonction du seuil maximal ϵ_2. Le critère d'arrêt est présenté dans l'algorithme 1. La procédure complète pour un départ de l'algorithme de NM est donnée dans l'algorithme 2. Un exemple des opérations dans un $\mathcal{O}$ à 2 dimensions est illustré sur la Figure 5.31 pour présenter la nature géométrique de la recherche de $\mathcal{O}$ dans l'algorithme de NM.

5.3.6.2 Exécution

Nous allons voir un exemple d'optimisation du mécanisme 2UPS-U avec 4 variables à optimiser, comme représenté sur la Figure 5.29b. Nous mentionnerons la méthode pour optimiser d'autres mécanismes avec l'algorithme. Comme discuté dans la section 5.3.3, nous voulons calculer une évaluation du mécanisme recouvrant les informations relatives à l'espace de travail réalisable, $\mathcal{F} \bigcap W_s$, ainsi que la performance globale, κ_g^{-1}, ou la

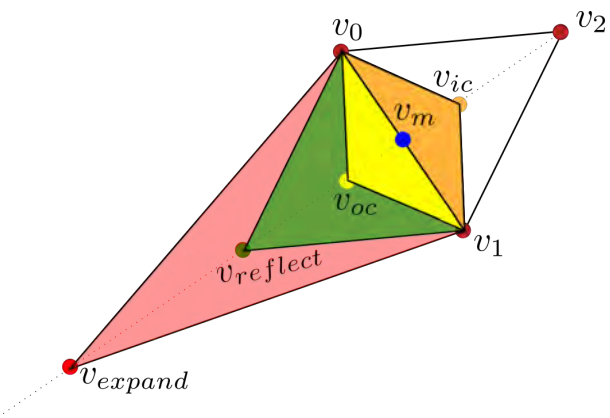


FIGURE 5.31 – Exemple d'une opération sur un simplexe dans un $\mathcal{O}$ à 2 dimensions.

norme des limites articulaires passives $\|q_p\|$. Comme nous savons que κ_g^{-1} et $\|q_p\|$ sont des objectifs contradictoires, nous pouvons définir différentes fonctions de récompenses pour chacun d'eux.

5.3.6.3 Stratégies de récompenses

Les stratégies de récompenses se réfèrent aux différentes méthodes permettant d'évaluer d'une fonction objectif particulière. Par exemple, pour optimiser la qualité d'une contrainte, nous pouvons biaiser la récompense vers le centre. Il en résulte des mécanismes ayant un très bon mouvement dans et autour du centre de l'espace de travail souhaité (W_s). Diverses stratégies peuvent être utilisées pour évaluer l'objectif. Nous discrétisons W_s en divisant la plage de chaque degré de liberté en 201 points (2/201 division des radians) ; il en résulte 40401 points équidistants, pour calculer l'évaluation nécessaire à chaque configuration de l'espace de travail.

Pour un ensemble donné de variables d'optimisation $[a, a', h, t]$ à partir de la Figure 5.29b, nous souhaitons savoir quelle est la qualité du mécanisme, E, résolvant le modèle géométrique inverse (MGI) du mécanisme dans chaque configuration, $(\alpha, \beta) \in W_s$, nous obtenons l'espace de configuration ($\mathcal{D}$) complet. Cela nous fournit toutes les valeurs d'articulation passive et la longueur actionnée pour les deux jambes. La même fonction est aussi utilisée pour dériver $\mathbf{J}$ et son déterminant. Comme nous vérifions également les collisions entre les actionneurs, une fonction, notée f($\mathbf{v}$) dans l'algorithme 3, permet d'inclure un contrôle de collision. Les actionneurs considérés pour le mécanisme ont un diamètre de tige plus grand en bas qu'en haut, comme illustré Figure 5.32. Ainsi, l'actionneur a été divisé en 5 points et la distance entre ces points discrétisés a été calculée.

Algorithm 1: Critère d'arrêt dans l'algorithme de NM [92].

Result: Boolean for stopping condition
sorted simplex $\{\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_{n-1}}, \mathbf{v_n}\}$;
evaluations $\{e_0, e_1, e_2, ..., e_{n-1}, e_n\}$;
maximum iteration = *max_iter*;
iteration count = iter;
$l_{ij} = \|\mathbf{v_j} - \mathbf{v_i}\|$;
$e_{ij} = |e_i$ - $e_j|$;
size = max(l_{ij});
eval = max(e_{ij});
if $size \leq \epsilon_1$ && $eval \leq \epsilon_2$ **then**
  stop = 1
else
  stop = 0
end
if $iter \geq max\_iter$ **then**
  stop = 1
else
  stop = 0
end

Sachant que la tige du vérin est plus petite que le rayon extérieur de l'actionneur, nous réduisons ainsi la contrainte pour la distance entre les points appartenant à la tige intérieur. Par exemple, la distance utilisée comme contrainte de collision pour les 4 points inférieurs dans la Figure 5.32 est différente de celle pour le $5^{ème}$ point, puisqu'il appartient au piston interne plus petit en taille. Cet ajustement de la contrainte de collision, bien qu'apparemment petit, a un impact important sur l'évaluation, selon la définition des stratégies de récompense.

5.3.6.3.a Stratégies de récompenses : espace de travail souhaité

Explorant la possibilité d'avoir un espace de travail large avec le mécanisme considéré, il a été décidé de voir d'abord les résultats pour maximiser l'espace de travail souhaité uniquement. Cela est réalisé avec la stratégie de récompense binaire. Chaque configuration est récompensée soit de 1, soit de 0, en fonction du point respectant les limites articulaires passives et les limites de l'actionneur uniquement, comme illustré sur la Figure 5.33. Les contraintes pour les points non singuliers et les collisions sont traitées plus strictement. Si la moindre configuration dans W_s est singulière ou ne respecte pas les contraintes

Algorithm 2: Départ unique de l'algorithme d'optimisation de Nelder-Mead [92].

Result: Local minimum evaluation and the optimized parameters

initial sorted simplex $\{\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_{n-1}}, \mathbf{v_n}\}$;
evaluations $\{e_0, e_1, e_2, ..., e_{n-1}, e_n\}$;
while *stop == 0* **do**
    calculate $\mathbf{v_m}$, $\mathbf{v_{reflect}}$ and $e_{reflect}$;
    if $(e_n < e_{reflect} < e_0)$ **then**
        $\mathbf{v_n} = \mathbf{v_{reflect}}$;
    else if $(e_0 < e_{reflect})$ **then**
        if $(e_{reflect} < e_{expand})$ **then**
            $\mathbf{v_n} = \mathbf{v_{expand}}$;
        else
            $\mathbf{v_n} = \mathbf{v_{reflect}}$;
        end
    else if $(e_n < e_{reflect} < e_{n-1})$ **then**
        if $(e_{oc} > e_{reflect})$ **then**
            $\mathbf{v_n} = \mathbf{v_{oc}}$;
        else
            $\forall\, i \in [1, n] \quad \mathbf{v_i} = s.\mathbf{v_i}$;
        end
    else if $(e_{reflect} > e_n)$ **then**
        if $(e_{ic} > e_{reflect})$ **then**
            $\mathbf{v_n} = \mathbf{v_{ic}}$;
        else
            $\forall\, i \in [1, n] \quad \mathbf{v_i} = s.\mathbf{v_i}$;
        end
    sort the simplex;
    if $\mathbf{v_{0new}} > \mathbf{v_0}$ **then**
        iter = 0
    else
        iter = iter + 1
    end
    Update stop from Algorithm 1
end
return $\mathbf{v_0}, e_0$

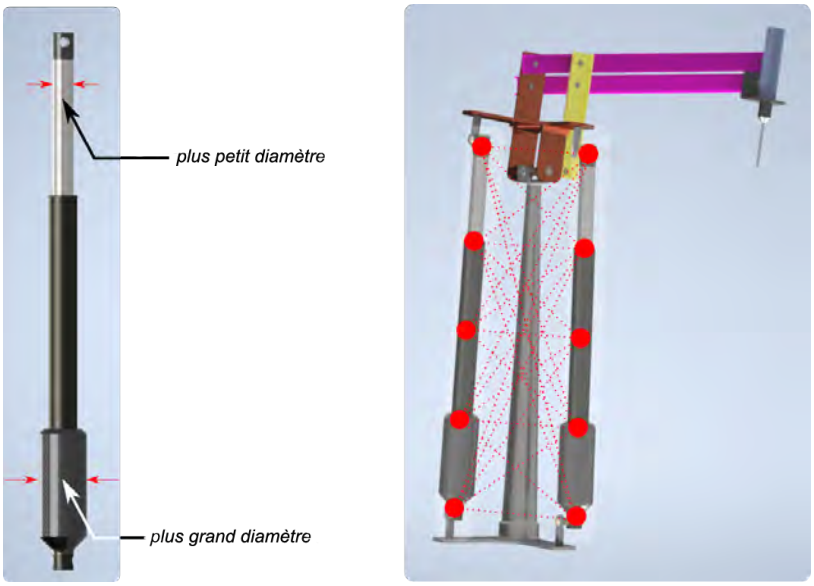


FIGURE 5.32 – Actionneur et les points discrétisés pour le calcul de collision (à droite).

de collision, alors l'évaluation reçoit une pénalité importante. Cela assure que, quel que soit la grandeur de l'espace de travail, la solution est disqualifiée si elle contient une configuration singulière ou une collision. Nous pouvons aussi biaiser la récompense pour obtenir de meilleures performances cinématiques dans et autour du centre de l'espace de travail souhaité, comme représenté sur la Figure 5.34.

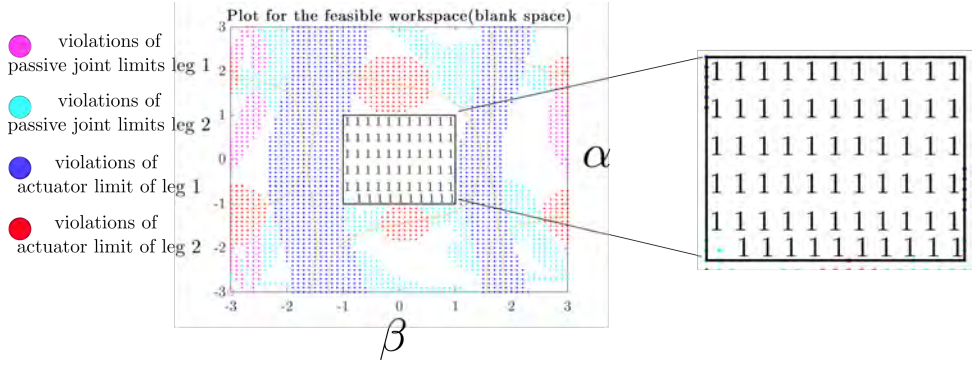


FIGURE 5.33 – Illustration de la récompense binaire pour l'espace de travail souhaité.

5.3.6.3.b Stratégies de récompenses : performance minimale

Nous souhaitons que le mécanisme ait la capacité à se déplacer dans toutes les directions avec une agilité équivalente, dans toutes les configurations de l'espace de travail

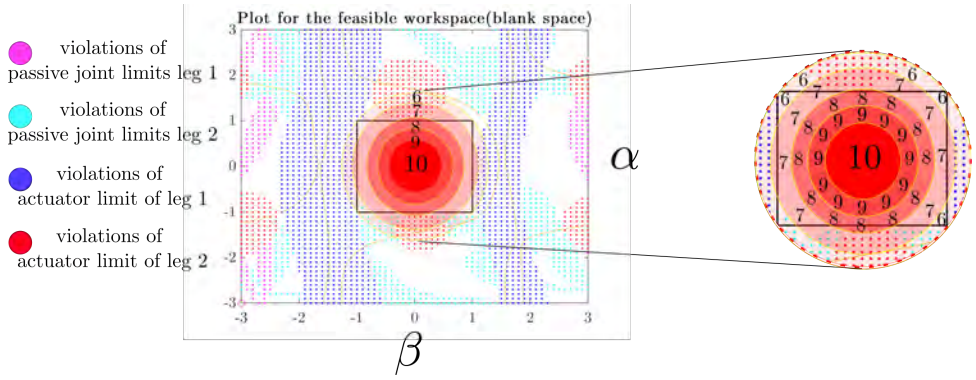


FIGURE 5.34 – Illustration de la récompense biaisée pour l'espace de travail souhaité (les points autour du centre reçoivent une pondération plus élevée).

souhaité. Cet objectif est difficile à atteindre, car nous savons que l'espace de travail est majoritairement limité par les courbes de singularité. Au fur et à mesure que nous nous rapprochons de configurations singulières, les performances du mécanisme diminuent. Comme le nombre de conditionnement global, κ^{-1}, ou le facteur d'amplification de vitesse dépendent des valeurs singulières de $\mathbf{J}$, il devient difficile d'avoir une dextérité au voisinage des configurations singulières, comme vu en (5.35) et (5.36).

Ainsi, une contrainte est nécessaire pour définir une valeur minimum de l'index de performance dans l'espace de travail. Cela nous permet de stopper le mécanisme s'il s'éloigne de de sa zone de travail définie. Nous limitons la performance avec des plages acceptables dans 64% du w_s.

5.3.6.4 Choix de la gamme d'actionneurs

Il faut choisir une gamme d'actionneurs, pour que les points jugés réalisables (mathématiquement) respectent également les longueurs des actionneurs et leurs limites physiques. Les contraintes des actionneurs sont présentées dans (5.48) et sur la Figure 5.27. Pour le mettre en oeuvre, nous re-vérifions les points valides enregistrés de W_s pour les différents supports de la gamme d'actionneurs. Ainsi, lors de la maximisation de la performance globale, nous nous assurons que le support choisi a au moins 90% de W_s. Par exemple, si un support particulier est choisi et présente la plus grande performance globale, mais n'a pas au moins 90% des 40401 points discrétisés, alors nous ignorons la performance et choisissons un support avec un maximum de points valides, comme détaillé dans l'algorithme 4.

Algorithm 3: Algorithme pour l'optimisation de l'espace de travail [92].

Result: evaluation e for **v** and corresponding range of actuators, **rho_range**

input $\rightarrow$ **v**;
x_i is the i^{th} variable of the output space;
for $x_1 = x_{1min} : interval : x_{1max}$ **do**
    ...
    for $x_n = x_{nmin} : interval : x_{nmax}$ **do**
        f(**v**) := function that solves inverse kinematic model, collision distance and κ^{-1};
        [det(**J**), passive_joints, ρ_1, ρ_2, κ^{-1}, collision distance] = f(**v**);
        if $|det(\mathbf{J})| \leq \epsilon_3$ **then**
            e = -∞;
            break;
        else
            $reward = 1$
        end
        if $\|\mathbf{q_{passive}}\| \geq$ **limits then**
            $reward = 0$
        else
            $reward = 1$
        end
        if *collision_distance* $\geq$ *max_collision* **then**
            $reward = 0$
        else
            $reward = 1$
        end
        if *reward == 1* **then**
            e = e + reward;
            valid_points[i] = [ρ_1, ρ_2, reward]
        $\rho_{\mathbf{vec}}[i]$ = [ρ_1, ρ_2];
    end
    ...
end
Implement the algorithm 4;
return e, **rho_range**

Algorithm 4: Algorithme pour le choix de la meilleure gamme d'actionneurs [92].

Result: evaluation e and corresponding range of actuators, **rho_range**
input → **valid_points** from algorithm 3;
rho_range = [minimum($\rho_{\mathbf{vec}}$), maximum($\rho_{\mathbf{vec}}$)] = [ρ_{min}, ρ_{max}];
Checking for feasible set of the actuators;
if *stroke* × **rho_range***[1]* ≥ **rho_range***[2]* **then**
  for *rho_lower* = ρ_{min} *: steps :* ρ_{max} **do**
    for *n = 1 : length(***valid_points***)* **do**
      e = 0;
      if ρ_1, ρ_2 ≤ *rho_lower* || ρ_1, ρ_2 ≥ *stroke* × *rho_lower* **then**
        feasible_points[j] = **valid_points**[n];
        j = j+1;
        e = e + **valid_points**[n, 3];
    end
    eval_vector[k]= [e, ρ_{min}, j];
    k = k + 1;
  end
  [e_1, $index_1$] = maximum(eval_vector, 1);
  e = $eval\_vector[index_1][1]$;
  ρ_{min} = **eval_vector**$[index_1][2]$;
  rho_range = [ρ_{min}, stroke.ρ_{min}];
else
  e = maximum($valid\_points$[3]);
  rho_range = [minimum($\rho_{\mathbf{vec}}$), maximum($\rho_{\mathbf{vec}}$)]
end
return e, **rho_range**

5.3.6.5 Avantages et inconvénients de l'algorithme NM

L'algorithme de Nelder-Mead est assez simple pour modéliser un problème d'optimisation pour la conception de mécanismes. Cet avantage nous permet d'avoir une méthodologie générale pour optimiser n'importe quel mécanisme parallèle. Comme il s'agit d'un algorithme non linéaire, nous pouvons introduire des fonctions objectif complexes difficiles à formaliser. Un exemple serait le conditionnement global, κ_g^{-1}, défini en section 5.3.4. De plus, les contraintes peuvent être construites de manière modulaire nous permettant d'expérimenter différentes contraintes à n'importe quel stade du développement.

Un autre avantage en lien avec la conception mécanique est sa méthode de recherche géométrique. La base de l'espace d'optimisation dans l'algorithme de NM sont les va-

riables d'optimisation elles-mêmes. Il est logique d'utiliser cette méthode, les paramètres de conception étant choisis en fonction de la combinaison des paramètres du simplexe précédent, plutôt que des méthodes complexes telles que les chromosomes dans les GA qui peuvent choisir une proposition sans explication géométrique. Il est également intéressant de pouvoir régler les paramètres d'exploration, les coefficients de réflexion, d'expansion, de contraction et de rétrécissement, avec l'intuition humaine et quelques connaissances préalables sur l'importance de différents paramètres.

Bien qu'il soit adapté à notre application, il existe aussi quelques inconvénients lorsque l'on utilise cet algorithme. Selon certaines hypothèses, l'algorithme de NM a une preuve de convergence jusqu'à la dimension 2 [135], mais n'aurait aucune preuve de convergence au-delà de l'optimisation bidimensionnelle. Si la mise en oeuvre est incorrecte, les simplexes peuvent avoir tendance à dégénérer, convergeant ainsi vers une solution non stationnaire [136]. La convergence dépend beaucoup de la taille initiale du simplexe et l'effet des coefficients a été étudié par Wang et al [137].

En dépit de ces lacunes, l'algorithme de NM est utile dans notre cas, car l'objectif de satisfaire à toutes les contraintes et d'obtenir ensuite une performance du mécanisme acceptable. Cela a été mis en oeuvre avec succès par le passé [122, 123]. Différentes variantes ont également été proposées pour contourner la convergence sur un minimum local [138] permettant à l'algorithme de converger vers d'autres solutions optimales.

Pour obtenir de meilleurs résultats, nous complétons la recherche locale de l'algorithme de NM avec quelques variantes et avec une technique de recherche globale de l'espace d'optimisation, dans la prochaine section.

5.3.7 Algorithme de recherche globale

L'algorithme de NM peut être associé à d'autres méthodes de recherche globale, comme les points à faible discrépance *(low-discrepancy points)* [139], les algorithmes génétiques [120] et l'optimisation de Powell [121]. Nous avons étudié un algorithme de NM multi-start avec des points de faible discrépance pour explorer un espace d'optimisation global. Dans cette méthode, nous exécutons l'algorithme de NM avec différents simplexes initiaux. Il est très important d'avoir des simplexes distribués uniformément sur l'espace d'optimisation afin d'explorer le maximum de surface de l'espace d'optimisation.

5.3.7.1 Simplexes initiaux pour le multi-start

Une façon simple d'obtenir un ensemble d'échantillonnage $\mathcal{O}_M \subset \mathcal{O}$ est le *Monte Carlo sampling* avec une distribution uniforme [140]). Malheureusement, les points obtenus ont tendance à former des clusters [141], particulièrement dans les contextes à haute dimension, qui nuisent à l'uniformité de la discrétisation. Un meilleur choix consiste à avoir les M points de la discrétisation $\mathcal{O}_M$ de $\mathcal{O}$ uniformément étendu. En particulier, il est souhaitable que les points soient suffisamment proches les uns des autres. Dans cette optique, on peut utiliser certaines techniques d'échantillonnage déterministes [142, 33, 143, 144].

5.3.7.2 Dispersion et discrépance

Soit $O_M \subset \mathcal{O}$ un ensemble M de points d'échantillonage o_i, $i = 1, \ldots, M$, et définissons la *dispersion* [141] de O_M comme

$$\theta(O_M) := \max_{o \in \mathcal{O}} \left(\min_{\tilde{o} \in O_M} \|o - \tilde{o}\| \right) \tag{5.55}$$

A partir de cette définition, il est clair que la dispersion quantifie "avec quelle uniformité" les points sont répartis dans l'espace. Il s'agit d'une mesure de l'uniformité de la distribution des points de O_M dans $\mathcal{O}$. Schématiquement, une valeur faible de $\theta(O_M)$ garantie que les points de O_M sont distribués sur $\mathcal{O}$ de façon uniforme, c'est-à-dire sans laisser de régions de l'espace "sous-échantillonnées" et en laissant les points suffisamment rapprochés les uns des autres.

Les meilleures propriétés de dispersion, dans le sens décrit ci-dessus, appartiennent à des suites *déterministes* appelées suites à discrépance faible, couramment utilisées en statistiques par exemple. Le concept de discrépance a été étudié [141, 145] pour mesurer l'uniformité d'un ensemble de points dans un domaine compact. Nous allons ainsi discuter des résultats se référant au n-cube (hypercube n-dimensionnel) $I^n := [0, 1]^n$, mais d'autres ensembles compacts peuvent être envisagés en effectuant des transformations appropriées.

Soit O_M un ensemble de M points dans I^n. Définissons $\mathcal{P}$ comme la famille de tous les sous-intervalles $\mathcal{I}$ de la forme $\times_{i=1}^{n} [o_{1,i}, o_{2,i}]$, où $o_{1,i}, o_{2,i} \in [0, 1]$, $o_{1,i} < o_{2,i}$, et soit $c(\mathcal{I}, O_M)$ la *fonction de comptage*, qui compte le nombre de points de O_M dans $\mathcal{I}$ (c'est-à-dire $c(\mathcal{I}, O_M)$ est le nombre de points de O_M appartenant à $\mathcal{I}$). La *discrépance* de O_M

est défini comme [145]

$$D(O_M) := \sup_{\mathcal{I}\in\mathcal{P}} \left| \frac{c(\mathcal{I}, O_M)}{M} - \lambda(\mathcal{I}) \right| ,$$

où $\lambda(\mathcal{I})$ est la mesure de Lebesgue de $\mathcal{I}$.
La *discrépance* est strictement reliée à la dispersion. En particulier, il peut être montré [145] que, pour tout ensemble de M points dans I^n, les inégalités

$$(b_n M)^{-1/n} \leq \theta(O_M) \leq \sqrt{n} D(O_M)^{1/n}$$

sont conservées, où b_n est le volume (ou, plus précisément, la mesure de Lebesgue) de la boule euclidienne de dimension n. Ainsi, un ensemble à faible discrépance est également un ensemble à faible dispersion.

5.3.7.3 Suites à discrépance faible

Les suites dont la *discrépance* satisfait la relation suivante

$$D(\{O_M\}) \leq O(M^{-1}(\log M)^n) \tag{5.56}$$

sont nommées *suites à discrépance faible*. Leur construction varie d'une méthode à l'autre. Dans [145], un repère commun pour la construction de certaines séquences, nommées (t, n)-*suites*, qui satisfont de manière déterministe la condition (5.56), a été présentée. D'autres exemples existent, comme la *good lattice points sequence*, la suite de *Halton*, et la suite de *Hammersley* [145, 141]. L'utilisation des *suites à discrépance faible* est à la base de la *méthode de quasi-Monte-Carlo* [145], initialement introduite comme une alternative efficace à la méthode de Monte-Carlo, méthodes de calcul numérique des intégrales.

Pour résumer, sur la base de la discussion précédente, une procédure efficace pour générer uniformément des ensembles de points déterministes dispersés consiste à prendre des portions finies de suites à discrépance faible. Ils atteignent *de manière déterministe* un taux de convergence pour la dispersion de l'ordre de

$$O\left(M^{-1/2n}\right) \leq \theta\left(S_M\right) \leq O\left(\sqrt{2n} M^{-1/2n}\right)$$

La Figure 5.35 illustre la comparaison entre un échantillonage d'un hypercube de dimen-

sion 2 par une séquence de 500 points indépendants et identiquement distribués selon la distribution uniforme, et un échantillonage du même cube obtenu par une suite à faible discrépance (la suite *Sobol* [146] dans ce cas). On peut voir clairement comment l'espace est mieux couvert par la suite à faible discrépance, et la présence de larges espaces entre les points dans le premier cas.

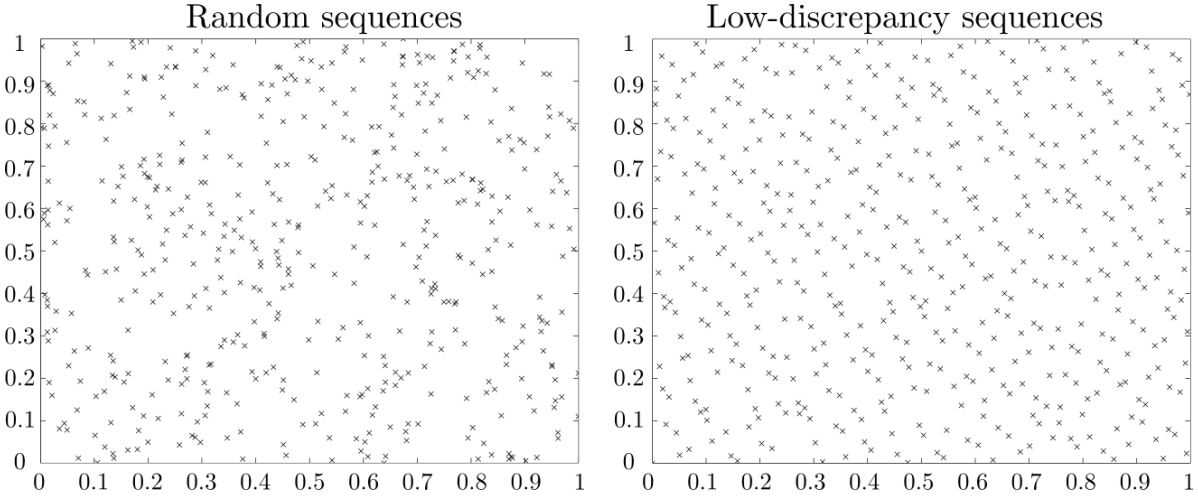


FIGURE 5.35 – Comparaison entre un échantillonage aléatoire et un échantillonage à faible dicrépance d'un hypercube de dimension 2 [33]

5.3.8 Recherches grossière et fine

Lors de l'exécution normale de l'algorithme de Nelder-Mead, l'itération s'arrête soit lorsqu'un simplexe a rétréci à une taille souhaitable avec des évaluations presque identiques, ou si ce même point a été rencontré pour les itérations maximales autorisées prédéfinies voir l'algorithme 1). Afin de diminuer le temps nécessaire à obtenir une convergence locale et pour nous permettre d'explorer plus de simplexes, nous avons adapté une méthodologie inspirée du tournage grossier et fin en machinerie. Lorsque nous voulons éliminer le plus rapidement possible un excédent de matière sur une pièce, nous augmentons la vitesse et ne nous concentrons pas sur la finition de l'ouvrage ; ensuite, lorsque nous sommes proches des dimensions désirées, la vitesse est diminuée et l'accent est mis sur la finition.

De la même manière, l'idée est d'initialiser des simplexes en multi-start et de réaliser une recherche grossière d'un optima local. Ensuite, nous collectons les optima locaux de tous les simplexes utilisés, puis implémentons des critères d'arrêt plus stricts sur quelques optima locaux sélectionnés leur permettant de converger vers un point stationnaire avec

une qualité plus fine. Fondamentalement, nous rejetons les optima locaux qui ne promettent pas une bonne évaluation, même après une recherche plus longue, réduisant ainsi considérablement le temps de mise en œuvre. Par ailleurs, comme nous avons déjà un sommet optimisé comme simplexe initial, nous pouvons construire le reste des sommets selon notre choix, contrôlant ainsi la taille du simplexe initial. L'une de ces implémentations est détaillée dans l'algorithme 5, où la condition d'incrémentation de l'itération est modifiée. La condition selon laquelle la nouvelle évaluation trouvée est meilleure que la précédente, est uniquement si elle dépasse l'évaluation précédente de 1%.

Algorithm 5: Implémentation des critères de recherche locaux grossiers et fins [92].

```
   Result: Optimised point
 1 input : Initial set of simplexes from the algorithm 2;
 2 For coarse search;
 3 max_iter = 3 n;
 4 margin = 1.05 ... (suggesting 5% increment);
 5 For fine search;
 6 max_iter = 10 n;
 7    margin = 1;
 8 max_evaluation = e_max;
 9 stop = 0 ; while stop = 0 do
10 |  Perform algorithm 2 except for last step of checking stop from algorithm 1;
11 |  Perform algorithm 3 with finer intervals;
12 |  if e_new > margin v then
13 |  |  iter = 0
14 |  else
15 |  |  iter = iter + 1
16 |  end
17 |  if iter > max_iter then
18 |  |  return stop = 1;
19 |  if e_new >0.8 e_max then
20 |  |  return stop = 1;
21 end
22 return v_best from the algorithm 2
```

5.3.9 Résumé de l'optimisation

Dans cette partie, nous avons présenté plusieurs fonctions objectif pouvant être utilisées pour obtenir un résultat optimisé. Différentes contraintes ont été discutées, comme

les limites articulaires passives, les collisions internes ou la condition de non-singularité. Nous avons également discuté de la méthodologie adéquate pour choisir la meilleure course d'actionneur pour maximiser l'espace de travail souhaité. Ensuite, plusieurs stratégies de récompenses ont été présentées, ainsi que leur pertinence selon les situations.

L'algorithme de Nelder-Mead a été détaillé pour mettre en lumière la nature géométrique de la recherche de l'espace d'optimisation et donc sa pertinence dans l'optimisation de la conception. Ensuite, pour différentes contraintes et stratégies de récompenses, des algorithmes ont été étudiés et programmés sous Matlab. Dans la dernière partie, nous avons présenté une nouvelle implémentation pour changer la nature de la recherche dans l'algorithme Nelder-Mead, dans l'objectif d'une recherche globale plus rapide et efficace de l'espace d'optimisation. La Figure 5.36 montre l'organigramme pour la méthodologie d'optimisation mise en œuvre. La figure illustre également les différents choix disponibles et les différentes décisions nécessaires pour une définition complète du problème d'optimisation. La Figure 5.37 montre le processus complet, avec les simplexes multi-start ainsi que l'ordre de la recherche grossière et de la recherche fine dans l'algorithme de Nelder-Mead en départ unique.

5.3.10 Résultats de l'optimisation

5.3.10.1 Comment interpréter les résultats

Lorsque le mécanisme a été configuré avec 13 paramètres pour le cas 2U<u>P</u>S-U, le tableau suivant a été utilisé pour définir un point dans l'espace d'optimisation correspondant, $\mathcal{O}$, et correspond au mécanisme illustré sur la Figure 5.38.

$$[a_1, \phi_1, h_1, b_1, \psi_1, h_3, a_2, \phi_2, h_2, b_2, \psi_2, h_4, t]$$

Les limites de ces paramètres sont :
$a_1, a_2 \in [0.25, 1.5]$, $\phi_1, \phi_2, \psi_1, \psi_2 \in [-100°, 100°]$, $b_1, b_2 \in [0.25, 2]$, $h_1, h_4 \in [-0.1, 0.1]$ et $h_2, h_3 \in [-0.5, 0.5]$.

Le point optimisé était représenté avec un diagramme schématique. La ligne rouge en pointillés correspond à l'actionneur de la première jambe, tandis que la ligne noire en pointillés correspond à l'actionneur de la deuxième jambe. Tous les schémas sont affichés dans la position par défaut de $\alpha = 0$ et $\beta = 0$. Un exemple de code de tracé avec des paramètres $[1, -1, 0, 1, -0.8, 0.5, 0.9, 0.2, 0.5, 1, 1, -0.2, 3.5]$ est représenté sur la Figure 5.39.

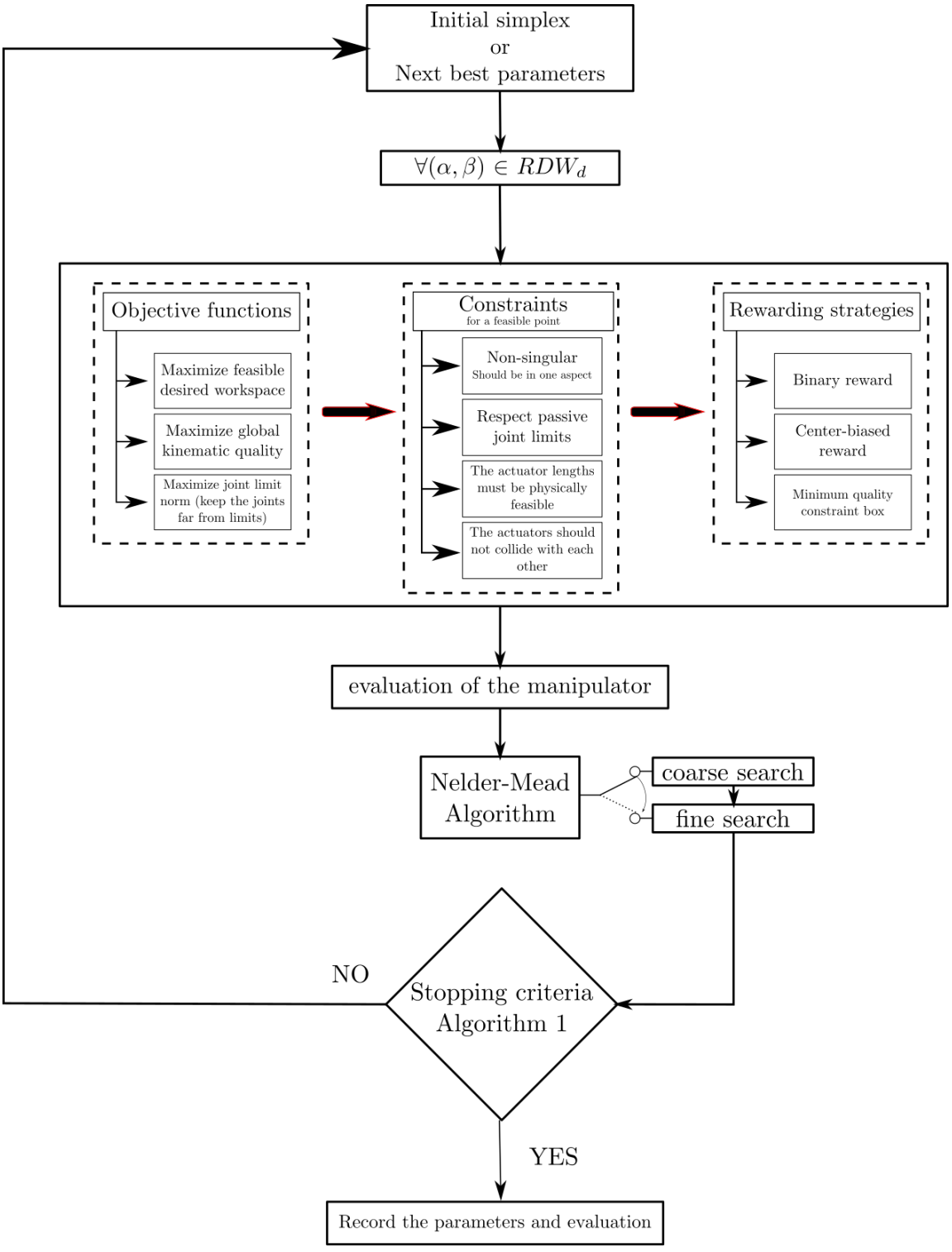


FIGURE 5.36 – Organigramme pour un démarrage unique de la méthodologie d'optimisation.

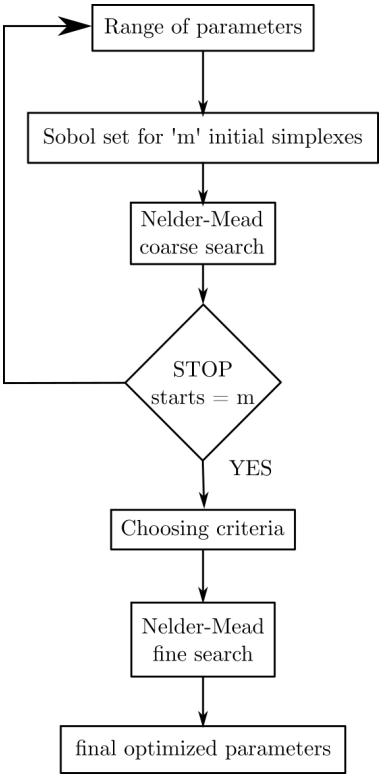


FIGURE 5.37 – Flowchart pour la méthodologie d'optimisation complète.

L'évaluation de ces paramètres est réalisée en vérifiant les contraintes implémentées dans le problème d'optimisation. L'espace opérationnel est visualisé pour comprendre les zones où les contraintes sont violées. Avec les violations de contraintes, il est également utile de visualiser la performance locale du mécanisme. Cela nous permet de comprendre graphiquement la qualité de comportement du mécanisme dans l'espace de travail souhaité. Pour ces raisons, nous visualisons la performance locale grâce à une carte de chaleur *(heat map)* où la valeur minimale (la plus sombre) correspond aux régions de singularité, tandis que les régions les plus claires correspondent à une configuration isotrope.

5.3.10.2 Différentes fonctions objectif

Les deux fonctions objectif utilisées sont le nombre de conditionnement global et la norme des limites articulaires. Jusqu'à présent, seule une optimisation mono objectif a été réalisée, et les deux fonctions objectif ont été implémentées individuellement. La Figure 5.41 est le résultat de la maximisation du nombre de conditionnement et de la maximisation de l'espace de travail souhaité. Le cadre noir de ± 1 radian est la limite de

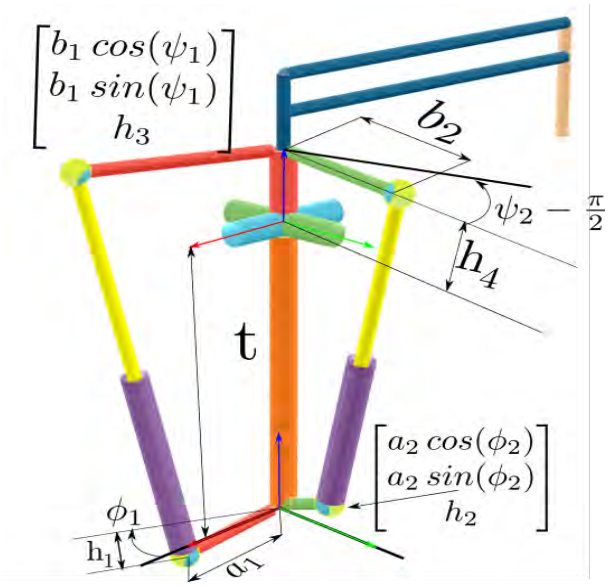


FIGURE 5.38 – Paramètres à optimiser du mécanisme 2UPS-U.

l'espace de travail souhaité. Si l'on maximise la norme des limites articulaires passives, il est intéressant de noter que le conditionnement est très mauvais, suggérant que la norme des limites articulaires et l'indice de performance cinématique sont des fonctions objectif contradictoires. En effet, dans ce cas, même s'il dispose d'un grand espace de travail souhaité, le mécanisme fonctionne mal d'un point de vue cinématique.

5.3.10.3 Effet du changement de contraintes

5.3.10.3.a Meilleure gamme d'actionneurs

La méthode utilisée avec l'algorithme de Nelder-Mead nous permet de calculer la meilleure gamme d'actionneurs pour maximiser l'espace de travail souhaité. Pour l'illustrer, nous lançons deux instances de l'algorithme avec un simplexe initial identique. Les résultats obtenus en utilisant la plus grande puis la meilleure gamme d'actionneurs sont représentés sur les Figures 5.42 et 5.43.

Pour l'exemple de la Figure 5.42, nous avons mis en place une contrainte telle que la plus grande plage d'actionneurs possible soit choisie pour l'évaluation. La Figure 5.43 montre les résultats lorsque la plage d'actionneurs optimisée est choisie pour l'évaluation. Cet exemple met en évidence le fait que la plus grande plage d'actionneurs n'est pas toujours la meilleure plage d'actionneurs. Les deux méthodes d'optimisation ayant été initiées avec des simplexes identiques, nous pouvons également conclure que la manière

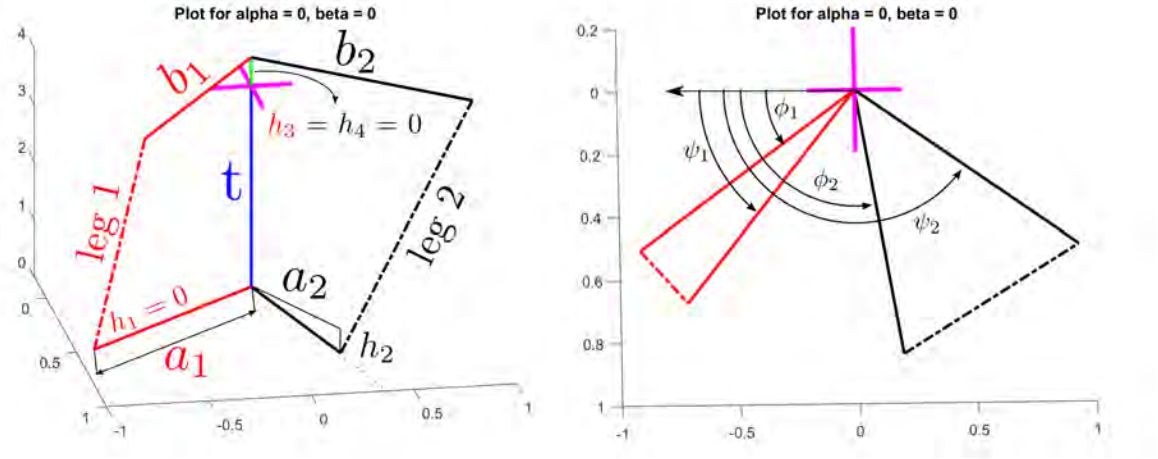


FIGURE 5.39 – Exemple de diagramme schématique du mécanisme 2UPS-U.

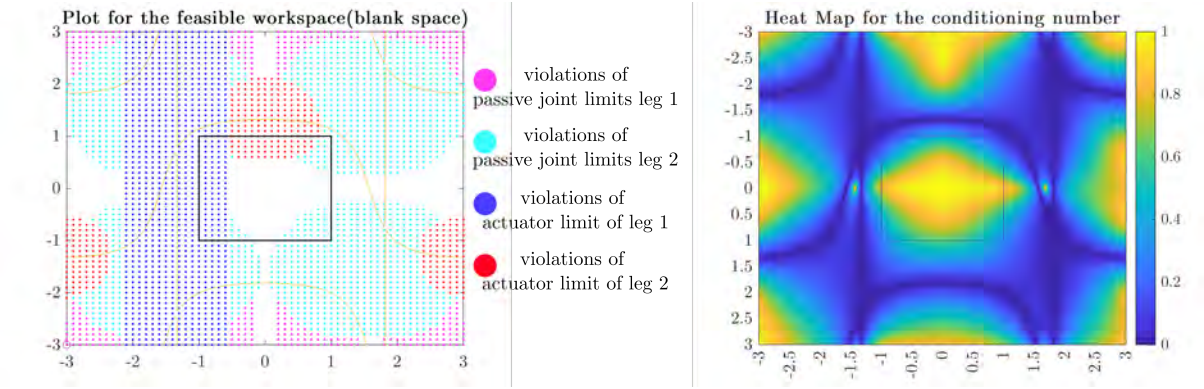


FIGURE 5.40 – Exemple de l'espace opérationnel avec violation de contraintes (à gauche) et heat map pour l'index de performance (à droite).

dont les contraintes sont mises en œuvre a un impact important sur les résultats obtenus.

5.3.10.3.b Contraintes de collision

Une autre contrainte importante à prendre en compte est la contrainte de collision. Lors de la mise en oeuvre d'une conception censée être valide selon l'intuition humaine, nous avons trouvé des limites articulaires respectées pour un large espace de travail et une bonne performance cinématique globale ; pourtant, l'algorithme ne proposait jamais de solution vers la zone attendue. Avec une intervention manuelle et en forçant l'algorithme à se rapprocher de la zone, il a été observé que les actionneurs entraient en collision dans une petite zone de l'espace de travail souhaité. Pour résoudre ce problème, une forme conique a été implémentée, comme indiqué dans la section 5.3.6.3. Cela a beaucoup amélioré

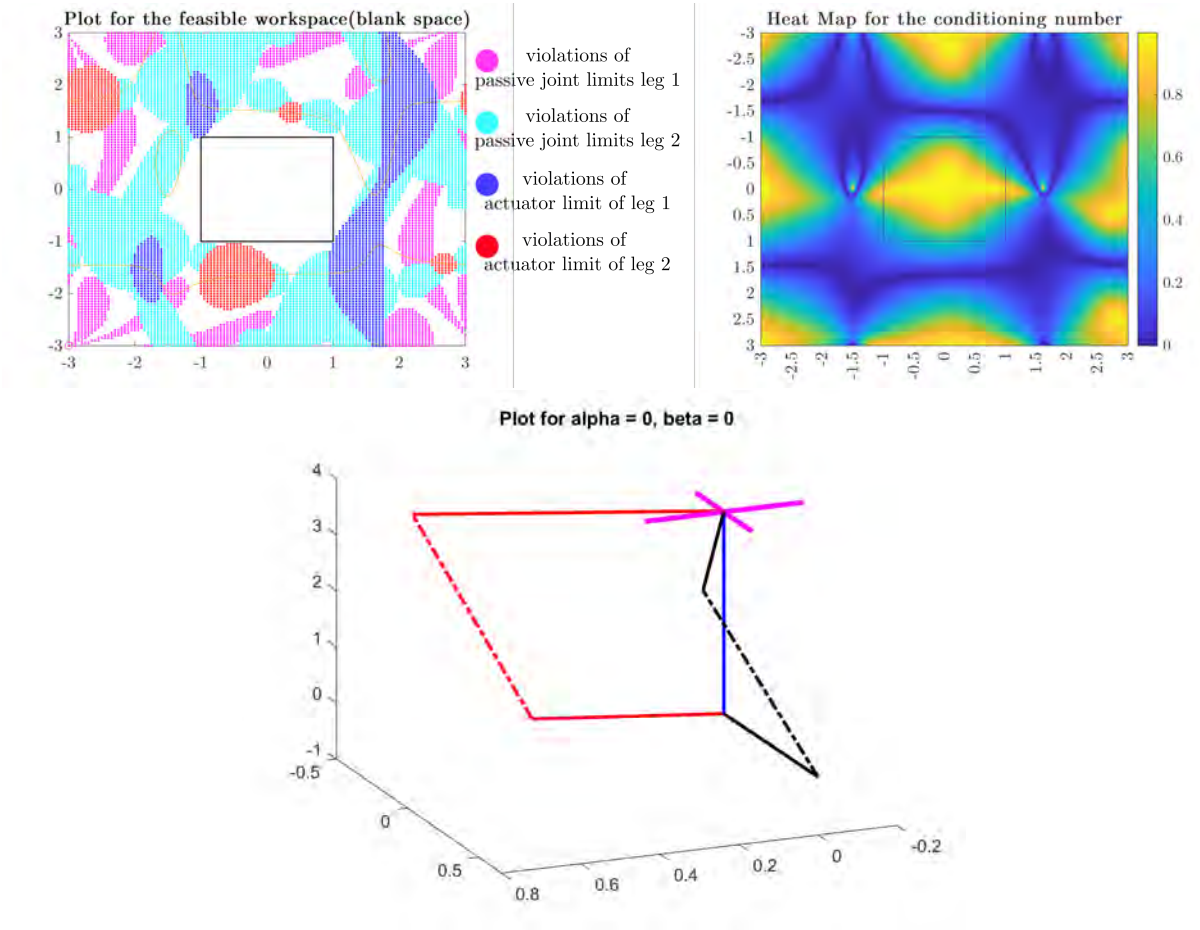


FIGURE 5.41 – Résultats pour l'un des meilleurs optima locaux acquis tout en maximisant le conditionnement global. Paramètres = $[0.458, -0.443, -0.084, 0.742, -0.409, 0.018, 0.631, 1.588, -0.030, 0.724, 1.157, 0.012, 3.6]$, gamme d'actionneurs = $[2.986, 4.355]$

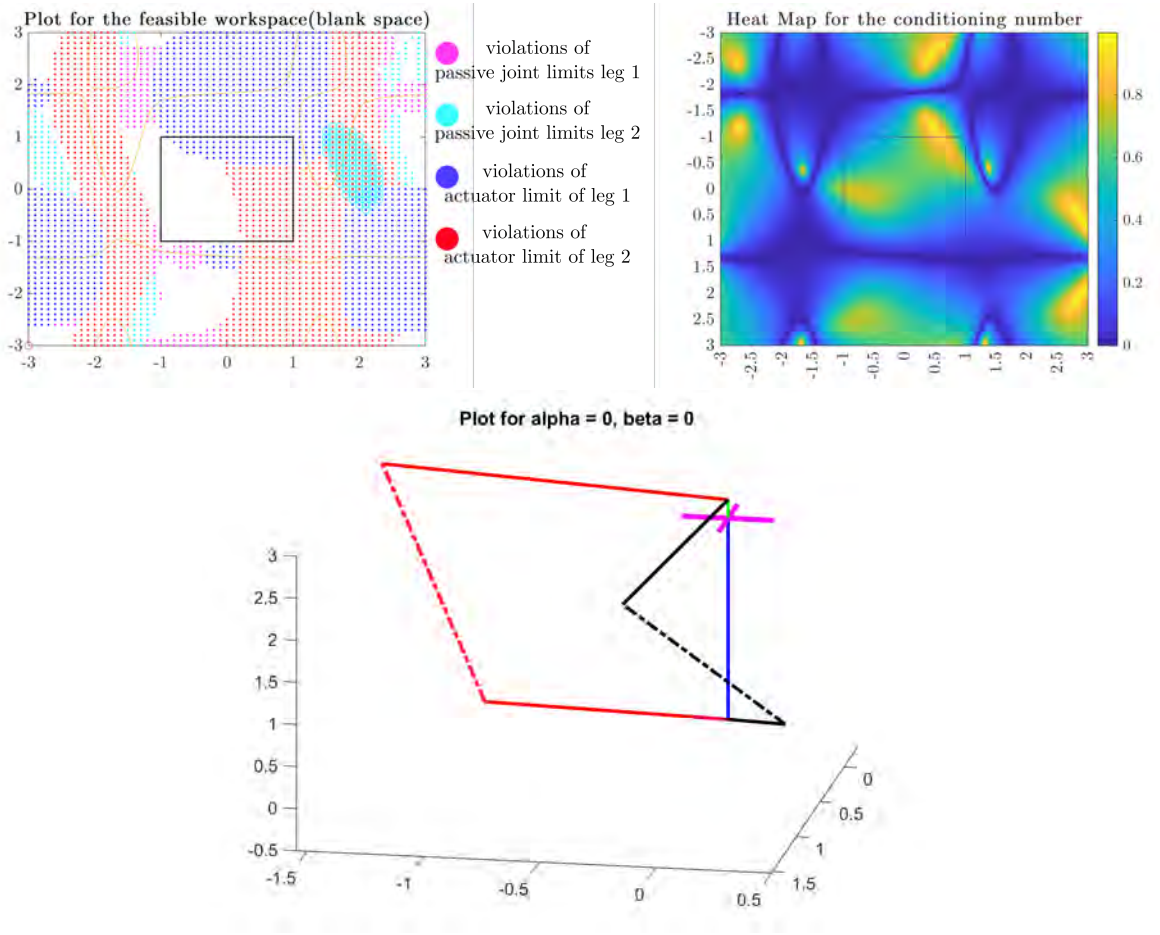


FIGURE 5.42 – Résultats pour la plus grande gamme d'actionneurs implémentée. Paramètres = $[1.056, -1.611, 0.036, 1.568, -1.745, 0.219, 0.25, 1.502, -0.011, 1.531, -0.092, 0.220, 2.39]$, gamme d'actionneurs = $[2.585, 3.877]$

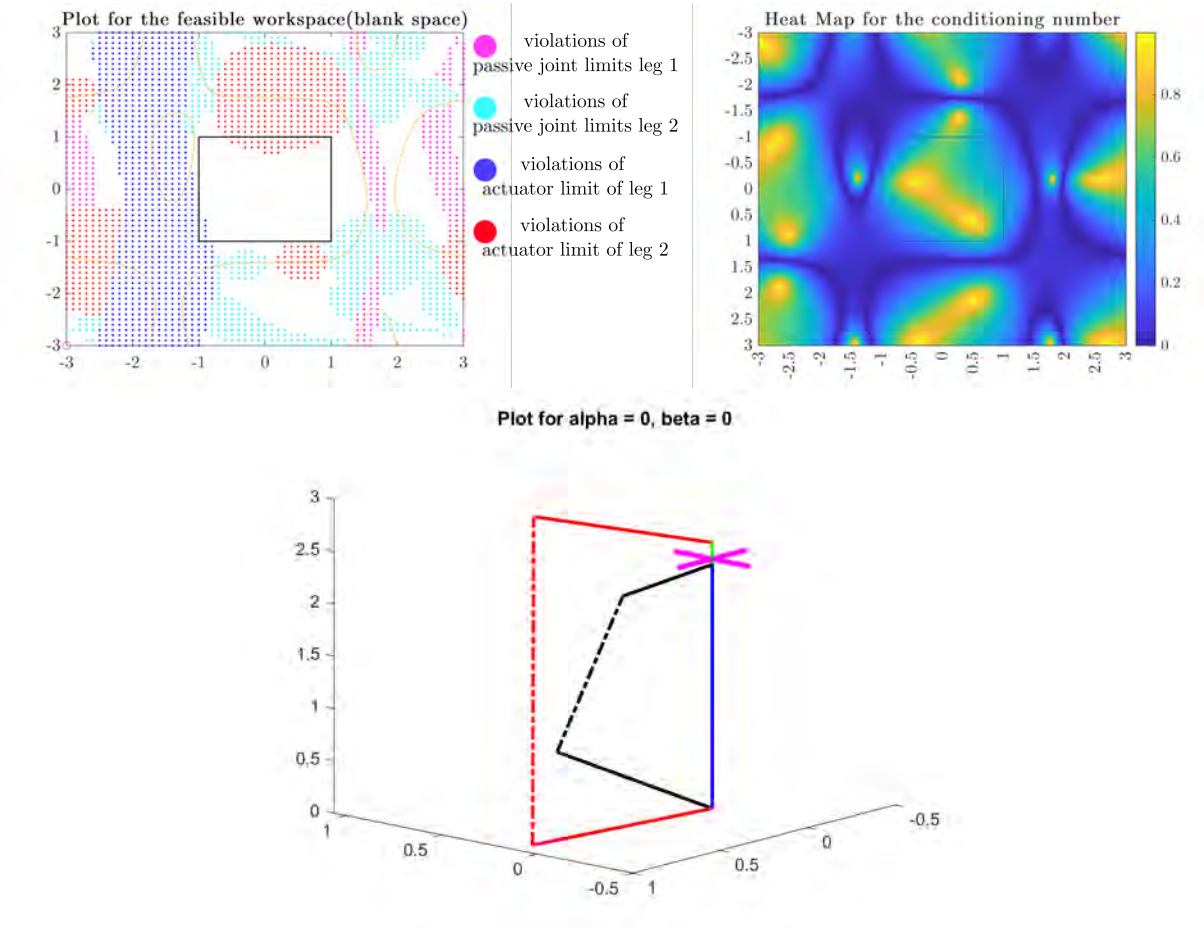


FIGURE 5.43 – Résultats pour la meilleure gamme d'actionneurs implémentée.
Paramètres =
$[0.994, 1.541, 0.072, 0.829, 0.151, 0.156, 1.096, -0.251, 0.031, 0.624, 1.725, -0.053, 2.384]$,
gamme d'actionneurs = $[2.209, 3.313]$