TP 1 - Robot Stäubli TX2-40 / Cs9 / SP2 : Génération de trajectoire

Jean-Louis Boimond
Université d'Angers

Ce TP a pour objectif de permettre à un bras de robot Stäubli TX2-40, initialement tendu à la verticale, muni d'un contrôleur Cs9 et d'une pince électrique Schunk EGP 40-N-N-B :

- de saisir une pièce rectangulaire ($22 \times 22 \times 40\ mm^3$) située sur un emplacement *initial*, noté **E1**,
- de déplacer la pièce sur un emplacement *de travail*, noté **E2**, pour y subir une temporisation (simulant ainsi un certain traitement),
- de déposer la pièce sur un emplacement *final*, noté **E3**,
- de repositionner le bras du robot tendu à la verticale,

sachant que le parcours de la pièce d'un emplacement au suivant est contraint par la présence d'un obstacle dans lequel la pièce doit se déplacer : le premier, situé entre **E1** et **E2**, en forme de couloir, intitulé **Couloir** ; le second, situé entre **E2** et **E3**, en forme de quart de cercle, intitulé **Cercle**.

Pour arriver à cela, une initiation à la programmation d'un robot Stäubli est proposée dans la partie **1** du document. La partie **2** est relative l'utilisation de la pince Schunk. Les points dits *pertinents* de la trajectoire à réaliser sont acquis dans la partie **3**. La programmation de la trajectoire à partir de ces points de passage est faite dans les parties **3** et **4**.

Table des matières

1) Initiation à la programmation d'un robot Stäubli

Il s'agit de prendre en main le robot, ainsi que de vous initier aux bases de sa programmation en VAL 3 avec le Pendant d'Apprentissage (*Teach Pendant*, nommé SP2 chez Stäubli), en utilisant pour cela le document intitulé « Prise en main du robot Stäubli TX2-40/Cs9 – Teach pendant SP2 – Programmation VAL 3 » accessible <ici>. A vous de tester les fonctionnalités qui y sont décrites en réalisant, entre autres, un premier programme intitulé `First_steps`, ceci **en faisant preuve de prudence** lorsque la puissance est mise sur le bras du robot.

Notamment, veuillez réduire la vitesse de déplacement du bras à 25% (voir pour cela la partie 1.3, p. 4, du document sur la prise en main du robot) durant toute la durée du TP.

Par la suite, vous allez avoir à compléter une application, intitulée **`TP_3A`**, située dans le disque dur du contrôleur. Cette application contient le code de base de l'application à réaliser, en incluant notamment certaines variables utilisées (et décrites) par la suite.

2) Utilisation de la pince Schunk

Une pince Schunk EGP 40-N-N-B est fixée à la flasque du bras du robot afin de pouvoir saisir, ou dessaisir, la pièce à déplacer. Pour cela, une variable intitulée `tPinceSchunk`, de type `tool`, a été créée dans l'application **`TP_3A`**. Elle est caractérisée par les valeurs $X = 0, Y = 0, Z = 151.7, Rx = 0, Ry = 0, Rz = 0$, ainsi que par un lien avec 2 sorties logiques intitulées `FastIO/fOut0`, `FastIO/fOut1` qui vont permettre la fermeture ou l'ouverture de la pince. L'utilisation de cette variable, plutôt que la variable `flange` (utilisée par défaut), dans une instruction de mouvement va permettre de situer le *Tool Center Point* (TCP) au niveau de la pièce à manipuler. En effet, le repère R_{Tool} associé à l'outil (la pince) correspond à celui du repère R_{Fl} (associé à la flasque) après une translation de $151.7\ mm$ le long de l'axe z_{Fl}, comme indiqué dans la figure qui suit. Vous noterez que l'origine O_{Tool} du repère R_{Tool} correspond au TCP.

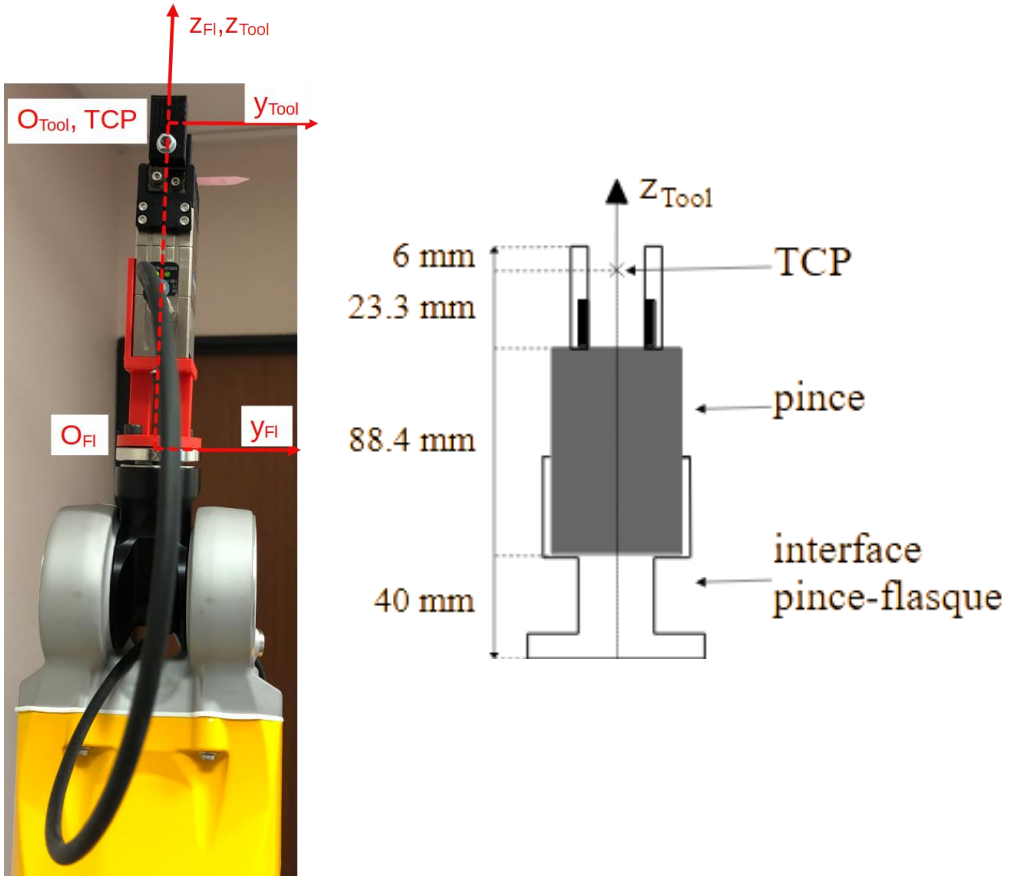

Figure 1 : Schéma de la pince Schunk avec une représentation des repères R_{Fl} associé à la flasque du robot et R_{Tool} associé à la pince.

Question 1 : La pince étant fixée au bras du robot et en admettant (car ce n'est pas le cas) que l'application `First_steps` (décrite dans le document sur la prise en main du robot) contienne la variable `tPinceSchunk` (décrite ci-dessus), quels changements observeriez-vous dans l'application `First_steps` si l'instruction de mouvement :

```
movej(pExamplePoint,flange,mNomSpeed)
```

était remplacée par :

```
movej(pExamplePoint,tPinceSchunk,mNomSpeed) ?
```

Manipulation 2 : Chargez en mémoire vive l'application **`TP_3A`**. Vérifiez que la variable `tPinceSchunk` est bien déclarée dans cette application avec les caractéristiques décrites ci-dessus.

Les instructions à même d'ouvrir ou de fermer la pince sont les suivantes :

- pour ouvrir la pince :

```
// ouverture de la pince
  dioSet(diClose,0)
  dioSet(diOpen,1)
```

- pour fermer la pince :

```
// fermeture de la pince
  dioSet(diClose,1)
  dioSet(diOpen,0)
```

où les variables `diOpen` et `diClose` sont reliées aux sorties logiques `FastIO/fOut0`, `FastIO/fOut1`.

Programmation 3 : Comme dans l'application `First_steps`, déclarez dans l'application **`TP_3A`** une variable `jDpt`, de type `jointRx`, telle que $J1 = \cdots = J6 = 0$ (voir 2.2.2, 2.2.3 du document sur la prise en main du robot), ce qui va permettre de tendre le bras à la verticale. Complétez le programme `start()` de l'application afin de : positionner le bras dans cette posture (voir 2.3 du document sur la prise en main du robot), fermer la pince (si elle ne l'est pas déjà !), puis l'ouvrir 2 secondes après afin de tester son fonctionnement.

Demandez à votre encadrant de vérifier votre code avant d'exécuter l'application.

N.B. : Il est possible d'ouvrir ou de fermer la pince en agissant à partir du Pendant d'Apprentissage sur les sorties logiques `FastIO/fOut0` et `FastIO/fOut1` du contrôleur Cs9. Pour cela, appuyer sur ***IO*>*Boards*>*J212 FastIO***, puis sélectionner l'onglet ***Digital Out***. Le changement d'état des sorties se fait en appuyant sur la touche  correspondante. Veiller à ce que les sorties affichées, à savoir, *Fast Output* 1 (pour `FastIO/fOut0`) et *Fast Output* 2 (pour `FastIO/fOut1`), ne soient pas verrouillées à travers un appui sur la touche  sachant que cette touche est opérationnelle lorsque le profil est celui de *maintenance* (pour cela aller dans la fenêtre **Settings>Profiles** en mettant dans l'encadré ***Current Profile*** : *maintenance* (et non *default*) dans le champ *Name* et *spec_cal* (et non une chaîne vide) dans le champ *Password*).
La pince est ouverte lorsque les sorties *Fast Output* 1=Off et *Fast Output* 2=On ; elle est fermée lorsque les sorties *Fast Output* 1=On et *Fast Output* 2=Off.

3) Acquisition des *points* pertinents de la trajectoire de la pince

La trajectoire à réaliser se base sur la donnée de certains *points*, dits *pertinents*, à même de *situer* (*i.e., positionner* et *orienter*) les emplacements (**E1**, **E2**, **E3**) et les obstacles (**Couloir**, **Cercle**) lesquels sont fixés sur une plaque millimétrique. Ces points, représentés par des croix dans la figure qui suit, vont permettre au bras du robot de déplacer le TCP (correspondant au point de « contact » avec la pièce, voir la figure 1) d'un emplacement au suivant en prenant en compte les obstacles. Soient :

- $P0$ le point relatif à l'emplacement *initial* (**E1**),
- $P1$, resp., $P2$, le point relatif à l'entrée, resp., la sortie, de l'obstacle **Couloir**,

- $P3$ le point relatif à l'emplacement *de travail* (**E2**),
- $P4$, resp., $P5$, le point relatif à l'entrée, resp., la sortie, de l'obstacle **Cercle**,
- $P6$ le point relatif à l'emplacement *final* (**E3**).

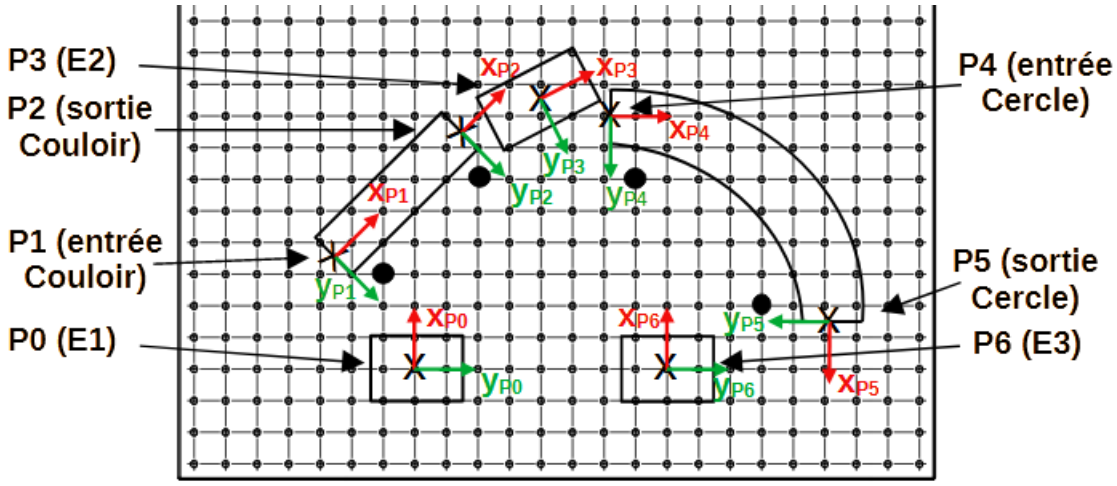

Figure 2 : *Situations* des *points* $P_0, \ldots, P_6$ sur la plaque millimétrique.

Question 4 : Rappelez pourquoi l'axe z de ces 7 points est opposé à l'axe z_0 du repère de base du robot.

La **manipulation** qui suit va permettre l'acquisition dans l'application **`TP_3A`** de ces 7 points (afin que le TCP soit à même de se situer en ces points).

La déclaration de ces points se fera, non pas dans l'espace articulaire (à travers une variable de type `jointRx`), mais dans **l'espace cartésien** à travers la donnée de 3 coordonnées pour *positionner* le point et 3 autres pour l'*orienter* par rapport à un repère de référence (par défaut le repère de base du robot) dans une variable de type `pointRx`.

Afin de disposer de mesures précises, l'acquisition de ces points va se faire alors que la pince tiendra la pièce (à déplacer). De plus, la posture adoptée pour atteindre l'ensemble des points sera telle que l'épaule sera à gauche avec le coude en haut.

Les valeurs en z de ces points devront être telles que la position du TCP corresponde à la description donnée dans la figure qui suit lorsque la pince se ferme, ou s'ouvre, pour saisir, ou relâcher, la pièce (de hauteur égale à $40\ mm$).

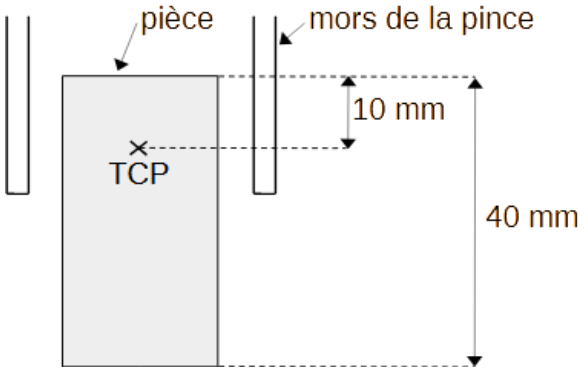

Figure 3 : Position du TCP lors de la saisie, ou de la relâche, de la pièce par la pince.

Manipulation 5 : Il s'agit de mémoriser les points $P0, \ldots, P6$ dans l'application **`TP_3A`** réalisée dans la **Programmation 3**. Pour cela, déplacer le bras du robot au point $P0$ afin de mémoriser ce

point (correspondant au point courant dans lequel se trouve le TCP à l'issue du déplacement) dans la variable cartésienne `pP0` ; procédez de même concernant les 6 autres points, à savoir $P1, \ldots, P6$, afin de les mémoriser dans les variables `pP1,…,pP6`. Pour réaliser cela, voir la N.B. située à la fin de l'annexe A.3 du document sur la prise en main du robot.
Plus précisément :
- concernant le point `pP0`, positionnez préalablement la pièce sur l'emplacement *initial* (**E1**). Déplacez alors le bras lentement (à 1% de sa vitesse) à proximité de **E1** jusqu'à ce que la pince soit prête à saisir la pièce, fermez alors la pince (voir N.B. précédente). Il restera alors à mémoriser ce point dans la variable `pP0` ;
- **pour des raisons de sécurité**, les points relatifs aux deux obstacles, à savoir, `pP1,pP2,pP4,pP5,` doivent être tels que la face inférieure de la pièce (à déplacer) soit à une distance de sécurité de $5\ mm$ par rapport aux supports des obstacles (relativement au point `pP3,` la pièce ne sera pas relâchée une fois située sur l'emplacement **E2**) ;
- enfin le point `pP6` doit être tel que la pince puisse s'ouvrir en ce point afin de déposer (délicatement) la pièce sur l'emplacement *final* (**E3**).

Demandez à votre encadrant de vérifier vos résultats avant de poursuivre.

4) Programmation du robot permettant de situer le TCP au point $P0$

Le mouvement d'approche vers le point $P0$ n'étant pas (encore) géré, la pièce (à déplacer) n'est pas censée être positionnée sur l'emplacement *initial* (**E1**) (au risque que la pince ne la saisisse pas correctement !!). L'objectif est de compléter l'application **`TP_3A`** pour permettre :

- de situer le TCP au *point* `jDpt` pour tendre verticalement le bras du robot, d'ouvrir la pince et d'attendre 1 seconde (au point `jDpt`) ;
- de situer le TCP au *point* $P0$, de fermer la pince (pour saisir la pièce), puis de l'ouvrir au bout de 2 secondes (afin de libérer la pièce) ;
- de repositionner le bras du robot à la verticale.

Programmation 6 : Complétez/modifiez le programme `start()` de l'application **TP_3A** afin que le bras du robot réalise l'objectif décrit ci-dessus.

Demandez à votre encadrant de vérifier votre code avant d'exécuter l'application.

5) Programmation de la trajectoire du robot

On suppose que la pièce (à déplacer) est positionnée sur le socle de l'emplacement *initial* (**E1**) et que l'application **`TP_3A`** est chargée en mémoire vive.

La trajectoire à réaliser est telle que le bras du robot doit :
a) Se positionner dans sa configuration initiale, à savoir, situer le TCP au point `jDpt` ; puis ouvrir la pince et attendre 1 seconde ;
b) Saisir la pièce située sur l'emplacement *initial* **E1**. Pour cela :
- situer le TCP à une hauteur de $10\ cm$ à la verticale du point $P0$,
- descendre le TCP en ligne droite au point $P0$,
- fermer la pince, puis attendre 1 seconde (la pièce est saisie) ;

c) Déplacer la pièce vers l'entrée de l'obstacle **Couloir**. Pour cela :
- monter le TCP en ligne droite à une hauteur de $10\ cm$ (à la verticale du point $P0$),
- situer le TCP à une hauteur de $10\ cm$ à la verticale du point $P1$,
- descendre le TCP en ligne droite au point $P1$;

d) Déplacer la pièce vers la sortie de l'obstacle **Couloir**. Pour cela :
- déplacer le TCP en ligne droite vers le point $P2$;

e) Déplacer la pièce sur l'emplacement *de travail* **E2**. Pour cela :
- monter le TCP en ligne droite à une hauteur de $10\ cm$ (à la verticale du point $P2$),
- situer le TCP à une hauteur de $10\ cm$ à la verticale du point $P3$,
- descendre le TCP en ligne droite au point $P3$,
- attendre 2 secondes (temporisation simulant un certain traitement de la pièce) ;

f) Déplacer la pièce vers l'entrée de l'obstacle **Cercle**. Pour cela :
- monter le TCP en ligne droite à une hauteur de $10\ cm$ (à la verticale du point $P3$),
- situer le TCP à une hauteur de $10\ cm$ à la verticale du point $P4$,
- descendre le TCP en ligne droite au point $P4$;

g) Déplacer la pièce vers la sortie de l'obstacle **Cercle**. Pour cela :
- déplacer le TCP en effectuant un quart de cercle vers le point $P5$;

h) Relâcher la pièce sur l'emplacement *final* **E3**. Pour cela :
- monter le TCP en ligne droite à une hauteur de $10\ cm$ (à la verticale du point $P5$),
- situer le TCP à une hauteur de $10\ cm$ à la verticale du point $P6$,
- descendre le TCP en ligne droite au point $P6$,
- ouvrir la pince, puis attendre 1 seconde (la pièce est déposée sur le socle de l'emplacement) ;

i) Monter le TCP en ligne droite à une hauteur de $10\ cm$ (à la verticale du point $P6$) ; repositionner le bras du robot dans sa configuration initiale.

En plus des instructions présentées dans le document concernant la prise en main du robot, comme `movej()`, `waitEndMove()`, `delay()`, décrivons trois instructions utiles pour réaliser la trajectoire souhaitée.

a) Approche et de dégagement par rapport à un point

L'instruction `appro` permet de définir un *point* cartésien à une certaine distance par rapport à un autre *point*. Par exemple, l'instruction `appro` combinée avec l'instruction `movej` comme suit :

```
movej(appro(pP,{0,0,h,0,0,0}),tPinceSchunk,mNomSpeed)
```

situe le TCP à une distance h (en mm) par rapport à un *point* `pP` le long de son axe z_{pP}. Une illustration est donnée dans la figure suivante avec $h = -100$ où le *point* atteint après l'exécution de la précédente instruction de mouvement est noté `pP1`.

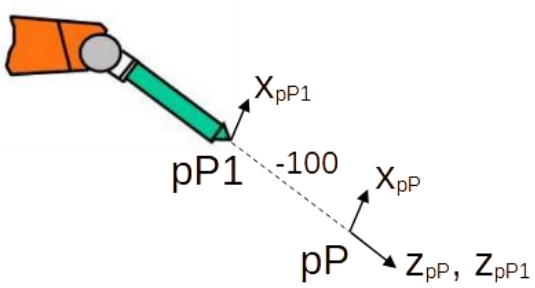

Figure 4 : Exemple d'utilisation de l'instruction `appro()`.

Soit un *point* `pP2` tel que son axe z_{pP2} soit opposé à l'axe z_0 du repère de base du robot, ce qui est le cas des *points* $P0, \ldots, P6$. Alors, les trois instructions suivantes :

```
movej(appro(pP2,{0,0,-50,0,0,0}),tPinceSchunk,mNomSpeed)
movej(pP2,tPinceSchunk,mNomSpeed)
movej(appro(pP2,{0,0,-50,0,0,0}),tPinceSchunk,mNomSpeed)
```

permettent :

- de situer le TCP à la verticale du *point* `pP2` à une hauteur égale à $50\ mm$ (1ère instruction),
- d'opérer un mouvement d'*approche* en situant le TCP au *point* `pP2` (2ème instruction),
- d'opérer un mouvement de *dégagement* en resituant le TCP à la verticale du *point* `pP2` à une hauteur égale à $50\ mm$ (3ème instruction).

Notez qu'il est possible de réaliser des mouvements dans l'espace cartésien (et non dans l'espace articulaire à travers l'instruction `movej`) afin de maîtriser **à tout instant** la trajectoire effectuée par le TCP, voir pour cela les sections **b** et **c** qui suivent.

b) Mouvement en ligne droite

L'instruction permettant un mouvement du TCP **en ligne droite** à partir du *point courant* (*i.e.*, celui atteint par le TCP (juste) avant l'exécution de l'instruction de mouvement) vers un *point* `pPoint` (de type `pointRx`) est la suivante :

```
movel(pPoint,tPinceSchunk,mNomSpeed)
```

Dans le cas de l'obstacle **Couloir**, une telle instruction s'avère particulièrement intéressante vis-à-vis de l'orientation de la pièce durant son déplacement au sein de l'obstacle, au sens où l'orientation au cours du mouvement du repère associé au TCP ne change pas du fait que l'orientation du *point courant* (correspondant au *point* $P1$ situé à l'entrée de l'obstacle **Couloir**) est la même que celle du *point* $P2$ (situé à la sortie de l'obstacle), voir la figure 2.

c) Mouvement circulaire

L'instruction permettant un mouvement **circulaire** du TCP à partir du *point courant* vers un *point de destination* `pPointDestination` (de type `pointRx`), passant par un *point intermédiaire* `pPointIntermediaire` (de type `pointRx`), est la suivante :

```
movec(pPointIntermediaire,pPointDestination,tPinceSchunk,mNomSpeed)
```

L'orientation du repère associé au TCP au cours du mouvement est issue d'une interpolation entre les orientations du *point courant*, du *point intermédiaire* et du *point de destination*. Cette caractéristique s'avère particulièrement intéressante pour effectuer le déplacement de la pièce au sein de l'obstacle **Cercle**, compte tenu de l'orientation des *points* $P4, P5$ (voir la figure 2) et du *point intermédiaire* $P4\_5$ mesuré ci-dessous.

Manipulation 7 : A l'image de ce que vous avez fait lors de la **Manipulation 5**, il s'agit de déplacer le bras du robot afin d'acquérir dans l'application **`TP_3A`** le *point* cartésien $P4\_5$ (situé sur l'arc de cercle à parcourir), décrit dans la figure qui suit. Soient :

- X, Y, Z les coordonnées de position du *point* $P4\_5$ (situé sur l'arc de cercle à parcourir) ;
- RX, RY, RZ les coordonnées d'orientation du *point* $P4\_5$ sachant que son axe x forme un angle de 45° avec l'axe x du *point* $P4$, voir la figure qui suit.

Soit `pP4_5` la variable de type `pointRx` contenant ces coordonnées. Comme pour les points `pP1, …, pP5`, le point `pP4_5` doit être tel que la face inférieure de la pièce (à déplacer) soit à une distance de sécurité de 5 mm par rapport à l'obstacle **Cercle**.

Demandez à votre encadrant de vérifier vos résultats avant de poursuivre.

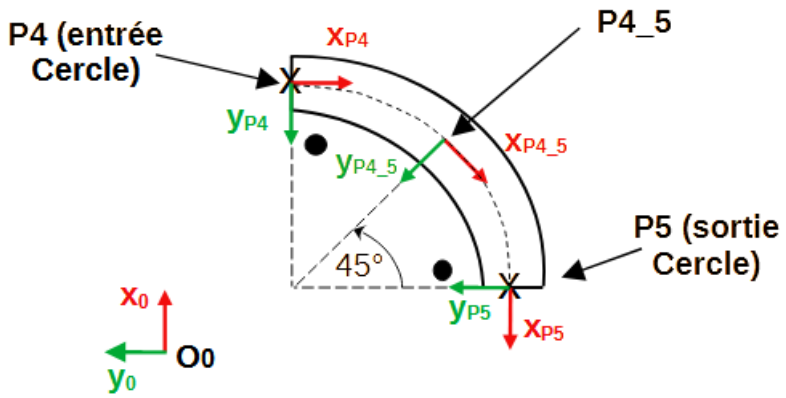

Figure 5 : *Situation* du point `P4_5` dans l'obstacle **Cercle**.

Programmation 8 : Complétez/modifiez le programme `start()` de l'application **TP_3A** afin que le bras du robot réalise la trajectoire décrite à travers les étapes *a*, *b*, …, *i* au début de la section **5**.