TP Génération de trajectoires rectilignes du *Tool Center Point* dans l'espace des tâches - Application au robot Kuka KR3

Jean-Louis Boimond
Université Angers

Ce TP traite de la **génération de trajectoires** utilisée dans le cadre de l'instruction de mouvement Kuka **LIN** (pour LINear) laquelle permet de réaliser un déplacement **rectiligne** du *Tool Center Point* (TCP), ceci tout en maitrisant l'évolution de son **orientation** au cours du mouvement. Dans notre cas, le bras du robot n'est pas muni d'un outil aussi le TCP correspond au point situé au centre de la flasque du robot.

Rappel : Un ***point*** (tel que le TCP) est assimilable à un ***repère*** défini par une certaine ***situation*** (c'est-à-dire, une ***position*** et une ***orientation***) par rapport au repère de base (R_0) du robot. Par la suite, un *point* sera défini à travers un vecteur de dimension 6 ou une matrice de transformation homogène (de dimension (4×4)).

La trajectoire est définie entre deux points, notés P^{Ini} et P^{Fin}, où :

- P^{Ini} est le *point* dit *initial* où se *situe* (en termes de position et d'orientation) le TCP avant sa mise en mouvement du fait de l'exécution d'une instruction **LIN**,
- P^{Fin} est le *point* dit *final* où se situe le TCP une fois l'instruction **LIN** exécutée.

L'instruction Kuka permettant un tel mouvement, sachant que le point courant avant l'exécution de l'instruction **LIN** correspond à P^{Ini}, est :

LIN $\boldsymbol{P^{Fin}}$

On suppose que l'instruction **LIN** est exécutée à l'instant 0 (le TCP étant situé à cet instant au *point initial* (P^{Ini})) et que sa durée d'exécution est égale à un temps t_f, autrement dit, le point *final* (P^{Fin}) du TCP est atteint à l'instant t_f (à l'issue de l'exécution de l'instruction **LIN**).

Deux évolutions de l'*orientation* du TCP (laquelle a un impact sur l'orientation de l'outil) entre les points *initial* et *final* (P^{Ini}, P^{Fin}) sont possibles selon la valeur de la variable Kuka $\$\mathrm{ORI\_TYPE}$:

- soit l'*orientation* reste constante au cours du mouvement (cas où $\$\mathrm{ORI\_TYPE} = \#\mathrm{CONSTANT}$, c'est alors l'orientation relative au point *initial* qui est prise en compte),
- soit l'*orientation* varie continument/progressivement le long du déplacement entre l'orientation du point *initial* et celle du point *final* (cas où $\$\mathrm{ORI\_TYPE} = \#\mathrm{VAR}$),

ces deux évolutions sont illustrées à travers les figures 1 et 2.

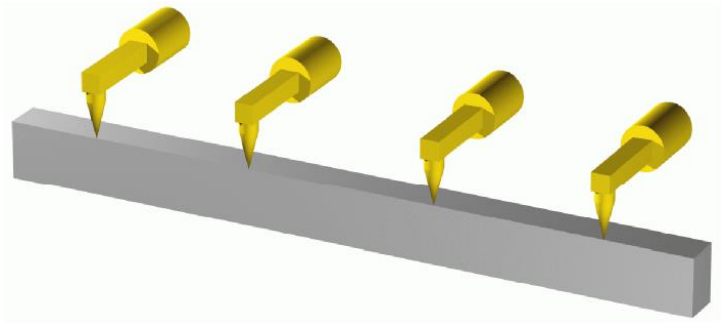


Figure 1 : Pas de changement de l'orientation du TCP au cours du mouvement.

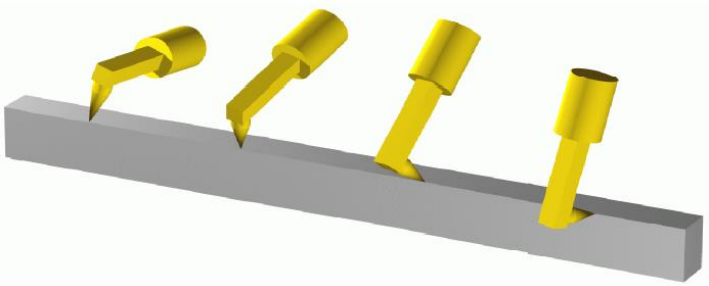
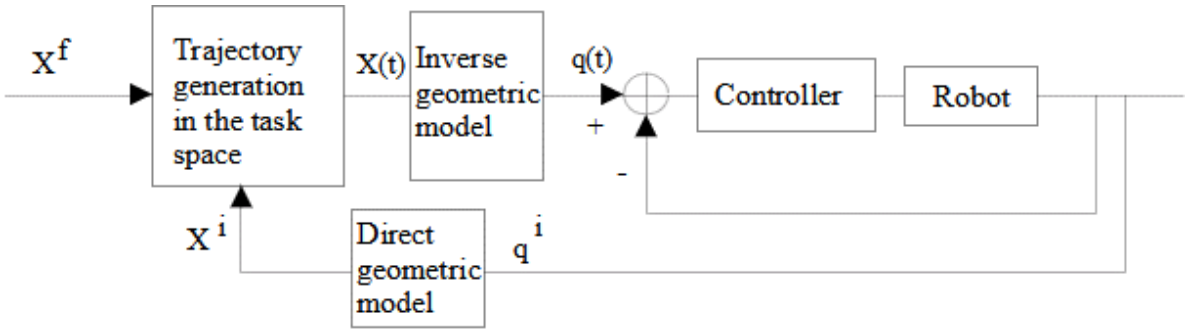


Figure 2 : Changement progressif de l'orientation du TCP au cours du mouvement.

On sait que le calcul du mouvement que l'on souhaite réaliser ne se fait pas directement dans *l'espace articulaire*, en effet, le fait de se situer dans cet espace ne permet pas de prévoir aisément le comportement du TCP (sa *position* et son *orientation*) entre les points initial et final (voir, à titre d'exemple, le robot 2R considéré au §5.1 du cours de robotique). Ce calcul est réalisé dans l'*espace des tâches* (appelé également *espace opérationnel*), espace dans lequel il est alors possible de maîtriser à tout instant le comportement du TCP. Le principe de génération de mouvement du TCP *dans l'espace opérationnel* est illustré à travers le schéma suivant.



où X^i, respectivement X^f, désigne la matrice de transformation homogène du point *initial* (P^{Ini}) du TCP, respectivement du point *final* (P^{Fin}), relativement au repère de base (R_0) du robot.

Dans ce TP, on s'intéresse à la génération de la trajectoire du TCP dans l'espace opérationnel réalisée dans le bloc '*Trajectory generation in the task space*' (représenté dans le schéma précédent) ; les modèles direct et inverse du modèle du robot étant supposés connus. Notons que le fait d'utiliser le modèle géométrique inverse du robot -pour disposer des valeurs articulaires q à appliquer au niveau des axes du robot- peut poser des problèmes de singularité.

La suite du TP est constituée de deux parties comme suit :

- dans la **partie A**, une description de deux méthodes de génération de trajectoires,
- dans la **partie B**, la programmation d'une méthode parmi les deux décrites dans la partie A, ainsi qu'une comparaison des résultats obtenus avec ceux obtenus *via* l'instruction **LIN**.

A) Description de deux méthodes de génération de trajectoires

Soient $X^i = \begin{bmatrix} A^i & P^i \\ 0\ 0\ 0 & 1 \end{bmatrix}$ et $X^f = \begin{bmatrix} A^f & P^f \\ 0\ 0\ 0 & 1 \end{bmatrix}$ où :

- A^i, respectivement A^f, est la matrice de rotation de dimension (3×3) représentant l'*orientation* par rapport à R_0 du point *initial* (P^{Ini}), respectivement du point *final* (P^{Fin}),
- P^i, respectivement P^f, est le vecteur de dimension (3×1) représentant la *position* par rapport à R_0 du point *initial*, respectivement du point *final*.

Dans ce qui suit, on s'intéresse au cas d'une évolution progressive de l'*orientation* du TCP (de A^i vers A^f), le cas d'une *orientation* constante du TCP étant un cas particulier (plus simple) de celle dont l'évolution est progressive.

Deux démarches sont à mener pour réaliser une trajectoire, il s'agit d'effectuer :

- d'une part, une **translation rectiligne** de la *position* du TCP entre les *positions* P^i et P^f, décrite dans la partie A.1,
- d'autre part, une **évolution progressive** de l'*orientation* du TCP entre les *orientations* A^i et A^f, décrite dans la partie A.2.

A.1) Translation rectiligne de la position P^i vers la position P^f

On note $P(t)$ la *position*, de coordonnées $x(t), y(t), z(t)$ dans l'espace (R^3), du TCP à l'instant t avec $0 \leq t \leq t_f$. La trajectoire entre les positions P^i et P^f est déterminée par l'équation vectorielle :

$$P(t) = P^i + r(t)(P^f - P^i),$$

où $r(t)$ est une fonction d'interpolation telle que $r(0) = 0$ et $r(t_f) = 1$.

Deux fonctions d'interpolation sont considérées : l'interpolation linéaire (*cf*. A.1.1.) et l'interpolation polynomiale cubique (*cf*. A.1.2.).

A.1.1) Interpolation linéaire

Soit $r(t) = \frac{t}{t_f}$ la fonction d'interpolation linéaire (en temps), il en résulte l'équation :

$$P(t) = P^i + \frac{t}{t_f}\,(P^f - P^i).$$

La trajectoire est continue vis-à-vis de la position mais pas vis-à-vis de la vitesse $(\dot{P}(t))$, ce qui est source d'à-coups au niveau des actionneurs. En effet, on a : $\dot{P}(t) = \frac{P^f - P^i}{t_f}$ pour $0 \leq t \leq t_f$ alors que $\dot{P}(t) = 0$ aux autres instants (le bras du robot étant à l'arrêt aux points P^{Ini} et P^{Fin}).

A.1.2) Interpolation polynomiale cubique

Le polynôme d'interpolation est d'ordre 3, soit $r(t) = a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3$, afin d'assurer une continuité vis-à-vis de la position, mais également de la vitesse (en assurant des vitesses nulles aux instants 0 et t_f), ceci tout en assurant d'avoir (comme dans le cas de l'interpolation linéaire) $P(0) = P^i$ et $P(t_f) = P^f$.

Question 1 : Montrer que la fonction d'interpolation $r(t)$ résultante est égale à :

$$3\left(\frac{t}{t_f}\right)^2 - 2\left(\frac{t}{t_f}\right)^3.$$

N.B. : Le calcul de $r(t)$ peut être raffiné en utilisant une interpolation polynomiale d'ordre 5 pour permettre également une continuité vis-à-vis de l'accélération $(\ddot{P}(t))$ (en assurant des accélérations nulles aux instants 0 et t_f), voir §5.1 du cours de robotique pour plus de détails.

A.2) Evolution progressive de l'orientation A^i vers l'orientation A^f

On note $A(t)$ la matrice de *rotation* de dimension (3×3) représentant l'*orientation* du TCP par rapport à R_0 à l'instant t avec $0 \leq t \leq t_f$.

Deux méthodes sont habituellement utilisées :

- l'une, décrite dans la partie A.2.1, est basée sur l'interpolation des angles d'Euler $\phi = (\psi, \theta, \varphi)$ (sachant qu'il est possible d'exprimer les angles d'Euler correspondant à une matrice d'orientation, et réciproquement) ;
- l'autre, décrite dans la partie A.2.2., est basée sur l'interpolation d'une matrice de rotation d'un angle γ autour d'un vecteur unitaire *U* sachant que cette matrice permet, pour γ et *U* convenablement choisis, l'obtention de la matrice de rotation *finale* (A^f) à partir de celle *initiale* (A^i).

Remarque : La démarche consistant à faire une interpolation directement sur les matrices de transformation homogènes X^i, X^f n'est pas utilisée car elle ne garantit pas l'orthonormalité de la matrice de rotation A à tout instant.

A.2.1) Angles d'Euler ϕ

On effectue une interpolation linéaire de l'équation vectorielle suivante :

$$\phi(t) = \phi^i + \frac{t}{t_f}\,(\phi^f - \phi^i),$$

où ϕ^i, ϕ^f et $\phi(t)$ représentent les angles d'Euler respectivement initiaux (obtenus à l'instant 0), finaux (obtenus à l'instant t_f) et courant (obtenus à l'instant *t*).

A.2.2) Matrice de rotation d'un angle γ autour d'un vecteur unitaire *U*

Avant de déduire les valeurs de γ et *U* permettant d'obtenir la matrice de rotation *finale* (A^f) à partir de celle *initiale* (A^i), on s'intéresse à établir l'expression de la matrice de rotation, notée $Rot(U,\ \gamma)$, d'un angle γ autour d'un axe quelconque porté par un vecteur unitaire *U*, à savoir :

$$Rot(U,\gamma) = \begin{pmatrix} U_x^2(1-\cos(\gamma)) + \cos(\gamma) & U_xU_y(1-\cos(\gamma)) - U_z\ \sin(\gamma) & U_xU_z(1-\cos(\gamma)) + U_y\ \sin(\gamma) \\ U_xU_y(1-\cos(\gamma)) + U_z\ \sin(\gamma) & U_y^2(1-\cos(\gamma)) + \cos(\gamma) & U_yU_z(1-\cos(\gamma)) - U_x\ \sin(\gamma) \\ U_xU_z(1-\cos(\gamma)) - U_y\ \sin(\gamma) & U_yU_z(1-\cos(\gamma)) + U_x\ \sin(\gamma) & U_z^2(1-\cos(\gamma)) + \cos(\gamma) \end{pmatrix} \quad \text{(éq.1)}$$

où U_x, U_y, U_z représentent les coordonnées du vecteur unitaire U défini dans le repère $R = (O, \vec{x}, \vec{y}, \vec{z})$.

On notera que cette approche (dont l'objectif est de calculer la matrice $A(t)$ à tout instant) n'est pas minimale au sens où elle nécessite 4 paramètres -à savoir le vecteur *U* et l'angle γ- alors que 3 paramètres suffiraient.

- **Expression de la matrice $Rot(U,\ \gamma)$**

Une solution consiste à considérer, comme indiqué dans la figure suivante, que le vecteur *U* est défini à l'aide des angles α, β où :

- α est l'angle entre x et x' autour de *z*,
- β est l'angle entre *z* et *U* autour de x',

on note que x' est orthogonal à z (puisque x' est dans le plan (O, x, y)) et à U.

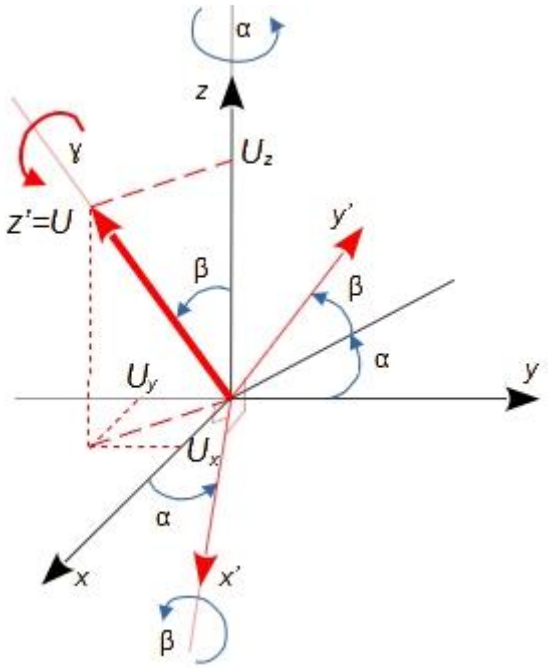


Figure 3 : Définition du vecteur U.

Au vu de cette figure, une solution décomposable en trois étapes permet de réaliser une rotation d'un angle γ autour de U $(= z')$, notée $Rot(U,\ \gamma)$, à savoir :

- étape 1 : on fait en sorte que le repère $(x', y', z'\ (= U))$ soit confondu avec le repère (x, y, z). Pour cela, on effectue une rotation d'un angle $-\alpha$ autour de z (x' est alors confondu avec x), puis une rotation d'un angle $-\beta$ autour de x ($z'\ (= U)$ est alors confondu avec z et y' est confondu avec y) ;
- étape 2 : U étant confondu avec z, on réalise la rotation d'un angle γ autour de $z\ (= U)$;
- étape 3 : on modifie le repère (x', y', z') de sorte que $z'\ (= U)$ revienne à sa position initiale (celle représentée dans la figure 3). Pour cela, on effectue une rotation d'un angle β autour de x', puis une rotation d'un angle α autour de z.

Question 2 : Déduire de ces étapes la matrice de rotation $Rot(U,\ \gamma)$ en fonction des matrices de rotation d'angles α, β, γ autour de x ou z.

Question 3 : Calculer l'expression des éléments $(1,1), (2,1)$ et $(3,1)$ de la matrice $Rot(U,\ \gamma)$ en fonction des cosinus, ou des sinus, des angles α, β, γ.

Question 4 : Exprimer les termes $\cos(\alpha), \sin(\alpha), \cos(\beta), \sin(\beta)$ en fonction des coordonnées U_x, U_y, U_z.

Question 5 : En déduire une expression des éléments $(1,1), (2,1)$ et $(3,1)$ de la matrice $Rot(U,\ \gamma)$ uniquement en fonction de l'angle γ et des coordonnées U_x, U_y, U_z sachant que vous devriez retrouver les résultats donnés dans l'équation (1).

➢ **Calcul de γ et *U* permettant d'obtenir la matrice de rotation finale (A^f) à partir de celle initiale (A^i)**

Inspiré des notations du Cours introduites lors de la présentation de la méthode de Paul, on note :

- $A^i = (s^i \quad n^i \quad a^i)$ où les vecteurs, de dimension (3×1), $s^i = \begin{pmatrix} s_x^i & s_y^i & s_z^i \end{pmatrix}^t$, $n^i = \begin{pmatrix} n_x^i & n_y^i & n_z^i \end{pmatrix}^t$, $a^i = \begin{pmatrix} a_x^i & a_y^i & a_z^i \end{pmatrix}^t$ expriment les projections dans le repère de base du robot des vecteurs du repère associé au point *initial* (P^{Ini}),

- $A^f = (s^f \quad n^f \quad a^f)$ où les vecteurs $s^f = \begin{pmatrix} s_x^f & s_y^f & s_z^f \end{pmatrix}^t$, $n^f = \begin{pmatrix} n_x^f & n_y^f & n_z^f \end{pmatrix}^t$, $a^f = \begin{pmatrix} a_x^f & a_y^f & a_z^f \end{pmatrix}^t$ expriment les projections dans le repère de base du robot des vecteurs du repère associé au point *final* (P^{Fin}).

Il s'agit de calculer l'angle γ et le vecteur *U* de la matrice de rotation $Rot(U, \gamma)$ qui permettent d'obtenir la matrice de rotation *finale* (A^f) à partir de celle *initiale* (A^i), ce qui revient à résoudre l'équation suivante :

$$A^i \, Rot(U, \gamma) = A^f. \qquad \text{(éq. 2)}$$

Question 6 : Il résulte de l'équation 2 l'expression suivante de $Rot(U, \gamma)$:

$$Rot(U, \gamma) = \begin{pmatrix} s^i \cdot s^f & s^i \cdot n^f & s^i \cdot a^f \\ n^i \cdot s^f & n^i \cdot n^f & n^i \cdot a^f \\ a^i \cdot s^f & a^i \cdot n^f & a^i \cdot a^f \end{pmatrix}. \qquad \text{(éq. 3)}$$

Retrouvez cette expression.

Question 7 : Il résulte des équations (1) et (3) les valeurs suivantes pour l'angle γ et le vecteur *U* :

$$\begin{cases} \cos(\gamma) = \frac{1}{2}\,(s^i \cdot s^f + n^i \cdot n^f + a^i \cdot a^f - 1) \\ \sin(\gamma) = \frac{1}{2}\,\sqrt{(a^i \cdot n^f - n^i \cdot a^f)^2 + (s^i \cdot a^f - a^i \cdot s^f)^2 + (n^i \cdot s^f - s^i \cdot n^f)^2} \\ \gamma = atan2(\sin(\gamma),\, \cos(\gamma)) \\ U = \frac{1}{2\,\sin(\gamma)} \begin{pmatrix} a^i \cdot n^f - n^i \cdot a^f \\ s^i \cdot a^f - a^i \cdot s^f \\ n^i \cdot s^f - s^i \cdot n^f \end{pmatrix} \end{cases}. \qquad \text{(éq. 4)}$$

Retrouvez l'expression de ce résultat.

➢ **Calcul par interpolation de la matrice de rotation $A(t)$**

On a :

$$A(t) = A^i \, Rot(U, r(t)\,\gamma) \quad \text{avec} \quad 0 \le t \le t_f,$$

où $r(t)$ est une fonction d'interpolation linéaire ou cubique (telle que $r(0) = 0$ et $r(t_f) = 1$), à savoir, $r(t) = \frac{t}{t_f}$ pour une interpolation linéaire ou $r(t) = 3\left(\frac{t}{t_f}\right)^2 - 2\left(\frac{t}{t_f}\right)^3$ pour une interpolation polynomiale cubique (*cf.* partie A.1.).

B) Programmation et test d'une méthode de génération de trajectoire parmi celles décrites précédemment

Il s'agit de réaliser un script MatLab permettant de générer la trajectoire entre deux points donnés (le point *initial* P^{Ini} et le point *final* P^{Fin}). Pour cela, on calcule à un instant courant t (t étant pris à intervalles de temps réguliers) entre les instants 0 et 1 $(= t_f)$:

- le point $P(t)$, de dimension (3×1), pour cela voir la partie A.1.,

- la matrice de rotation $A(t)$, de dimension (3×3), pour cela voir la partie A.2..

On utilisera la fonction d'interpolation $r(t)$ polynomiale cubique pour calculer à un instant t le point $P(t)$ et la matrice de rotation $A(t)$ sachant que $A(t)$ sera calculée en utilisant la méthode basée sur la matrice de rotation d'un angle γ autour de *U*.

Les points initial P^{Ini} et final P^{Fin} du TCP considérés ont été relevés sur le *smartPAD* du robot Kuka, à savoir :

$$P^{Ini} = [X = 242 \quad Y = -397 \quad Z = 527 \quad A = -66 \quad B = 0 \quad C = -10]$$
$$P^{Fin} = [X = 292 \quad Y = -318 \quad Z = 420 \quad A = -66 \quad B = 0 \quad C = 170]'$$

où les 3 premières coordonnées de ces vecteurs sont des distances (en *mm*) alors que les 3 dernières sont des angles (en degré) définis selon la convention (Z, Y, X).

Question 8 : Un script MatLab, intitulé 'Gen_Trajectoire.m', est fourni avec ses fonctions associées afin de vous aider. Après avoir compris son fonctionnement (sans vous focaliser sur la partie du script consacrée au calcul de U lorsque $\sin(\gamma)$ est proche de 0), il va s'agir pour vous de :

- modifier le script afin que la fonction d'interpolation soit polynomiale cubique plutôt que linéaire,

- compléter le script afin de permettre le calcul du vecteur P et de la matrice A au cours du temps effectué dans la boucle '$\mathbf{for}\ \boldsymbol{n} = \mathbf{1}: \boldsymbol{N} + \mathbf{1}\ \cdots \mathbf{end}$' où $\boldsymbol{N} + \mathbf{1}$ est le nombre, initialement égal à 11, de points calculés et $\boldsymbol{n}$ est l'indice courant de la boucle permettant le calcul de P et A à l'instant $\boldsymbol{t(n)} = \frac{\boldsymbol{n-1}}{\boldsymbol{N}}\ \boldsymbol{t_f}$, ces valeurs étant mémorisées dans $\boldsymbol{P}(:, \boldsymbol{n})$ et $\boldsymbol{A}(:, :, \boldsymbol{n})$.

La vidéo accessible à travers le lien 'Video-SimPro.MOV', produite *via* le progiciel de simulation KUKA SimPro, permet de visualiser la trajectoire réalisée par le robot Kuka à l'issue des instructions suivantes :

PTP P3
LIN P1

où le point $P3$, respectivement $P1$, joue le rôle du point P^{Ini}, respectivement P^{Fin}. L'instruction : **PTP P3** (PTP pour Point-To-Point) permet au bras d'aller au point P3 à travers un « simple » calcul dans l'espace articulaire du robot.
Dans la vidéo, le repère de base (R_0) du robot est tel que le plan de visualisation de la simulation (écran de l'ordinateur) correspond au plan (x_0, z_0).

Question 9 : Que dire des résultats obtenus par calcul par rapport à ceux issus de la simulation, par Kuka SimPro, de la trajectoire réalisée par le robot.