دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی مکانیک

گزارش پروژه درس رباتیک

عنوان

تحلیل ربات شش درجه آزادیUR10

نگارش
صادق مهدوی ۹۸۲۶۰۳۰
مهدی رحمانی ۹۷۲۶۰۳۱

استاد درس
دکتر حامد غفاری راد

تدریسیار درس
مهندس سیدعلی میرحقگوی

بهار ۱۴۰۲

بسم الله الرحمن الرحيم

تقدیر و تشکر

بدینویسله مراتب قدردانی و تشکر خود را خدمت،

استاد محترم درس، جناب آقای دکتر غفاری راد بابت آموزش عالی مباحث مربوط به درس رباتیک، و همچنین جناب آقای مهندس میرحقگوی بابت کمک‌های بی‌دریغ و راهنمایی‌هایشان برای یادگیری هرچه بهتر نکات درس،

ابراز و از تمامی زحمات ایشان تشکر می نمایم.

عنوان **فهرست مطالب** **صفحه**

عنوان **فهرست اشکال** **صفحه**

فصل اول

مقدمه

۱-۱- معرفی ربات

امروزه در صنایع مختلف، کاربرد بازوهای رباتیک رو به گسترش است. از بازوهای رباتیک، جهت مونتاژ، دمونتاژ، جابجایی قطعات، جوشکاری و ... استفاده می شود. استفاده از ربات ها کاهش خطاهای انسانی، افزایش دقت ساخت، کاهش هزینه و افزایش سرعت را در پی دارد.

شرکت یونیورسال ربات، یک شرکت دانمارکی است که در سال ۲۰۰۵ و با ساخت بازوهای رباتیک همکار صنعتی[۱] کوچک و انعطاف پذیر تاسیس شد. در سال ۲۰۰۸ اولین کوبات های UR5 این شرکت در بازارهای دانمارک و آلمان در دسترس بودند . در سال ۲۰۱۲ دومین کوبات، UR10 ،توسعه داده شد . در سپتامبر ۲۰۱۹ ، این شرکت UR16e را راه‌اندازی کرد که برای کارهای پر بار، مانند جابجایی مواد سنگین، مراقبت از ماشین‌های سنگین، بسته‌بندی و پالت سازی مناسب است. کوباتهای UR هم در مشاغل کوچک تا متوسط و هم در شرکت‌های بزرگ و در صنایعی مانند خودروسازی، الکترونیک، فلز و ماشین کاری، داروسازی و... استفاده می شوند.

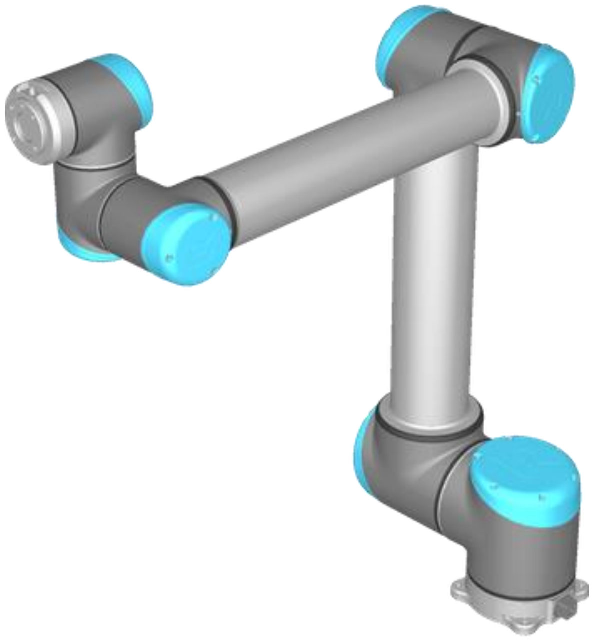
یونیورسال روبات UR10 یک بازوی روباتیک همه کاره است که به طور گسترده در کاربردهای مختلف صنعتی و تحقیقاتی استفاده می شود. بازوی ربات طوری طراحی شده است که برنامه ریزی آسان، انعطاف پذیر و ایمن برای کار در کنار اپراتورهای انسانی باشد. این ربات، یک بازوی رباتیک شش محوره است که قادر است محموله هایی تا وزن ۱۰ کیلوگرم را جابجا کند. دارای برد ۱۳۰۰ میلی متر و دقت تکرارپذیر ۰,۱ میلی متر است. بازوی ربات مجهز به یک سیستم دید سه بعدی و یک حسگر نیرو/گشتاور است که آن را قادر می‌سازد تا وظایف مختلفی مانند جابجایی، مراقبت از ماشین، مونتاژ و کنترل کیفیت را انجام دهد. در این پروپوزال، ما یک نمای کلی از ربات UR10، کاربردهای آن، درجات آزادی، بعد لینک‌ها و فضای کاری ارائه خواهیم داد.

[۱] Cobot

۱-۲- کاربرد ربات

ربات UR10 در صنایع مختلفی از جمله خودروسازی، هوافضا، الکترونیک و داروسازی استفاده می شود. همچنین به طور گسترده در تحقیقات و آموزش استفاده می شود. برخی از کاربردهای خاص ربات UR10 عبارتند از:

- **مراقبت از ماشین:** ربات UR10 می تواند برای بارگیری و تخلیه ماشین ها استفاده شود و نیاز اپراتورهای انسانی به انجام این وظایف تکراری و بالقوه خطرناک را کاهش دهد.
- **مونتاژ:** بازوی ربات می تواند پارت‌ها و قطعات را با دقت بالا مونتاژ کند و خطاها را کاهش دهد و بهره وری را افزایش دهد.
- **کنترل کیفیت:** ربات UR10 می تواند پارت‌ها و قطعات را از نظر نقص بررسی کند و اطمینان حاصل کند که فقط محصولات با کیفیت بالا برای مشتریان ارسال می شود.
- **تحقیق:** ربات UR10 در کاربردهای تحقیقاتی مختلفی از جمله تعامل انسان و ربات، دستکاری رباتیک و بینایی کامپیوتری استفاده می شود.



شکل ۱- نمایی از ربات UR10

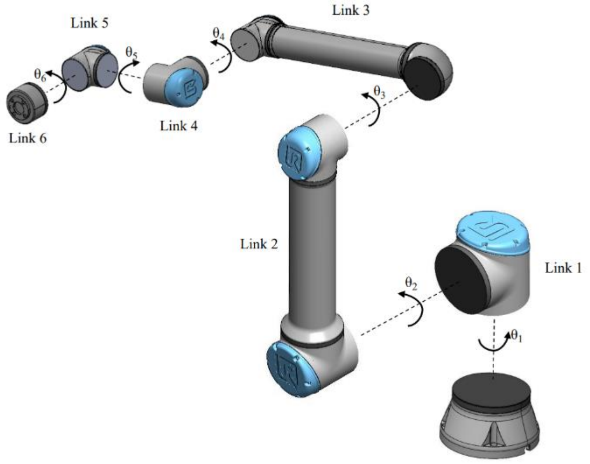
۱-۳- درجات آزادی مورد بررسی

درجات آزادی (DOF) یک ربات درواقع تعداد جهت‌ها و محورهایی است که میتواند به طور مستقل در آن‌ها حرکت داشته باشد. بازوی رباتیک مورد نظر دارای ۶ مفصل روولوت که هر کدام یک درجه آزادی به ربات ما می دهند. در نتیجه UR10 یک ربات ۶ محوره میباشد و ۶ درجه آزادی دارد. محدوده کاری تمامی مفاصل ۳۶۰ درجه می باشد و ظرفیت باربرداری آن ۱۰ کیلوگرم است.

هریک از مفصل‌های این ربات یک درجه آزادی دارد و به این معنی که اجازه میدهد که فقط در یک جهت حرکت انتقالی داشته باشد یا حول یک محور دوران داشته باشد. شش مفصل این ربات به صورت سری کنار هم قرار گرفته‌اند و حرکت در هریک از این جوینت‌ها در مکان end effector تاثیر دارد. همچنین Configuration آن به گونه‌ای است که به ربات اجازه میدهد که end effector خود را در محدوده گسترده‌ای از مکان‌ها و چرخش‌های مختلف حرکت دهد.

شش درجه آزادی این ربات به شرح زیر میباشد:

۱- چرخش حول Base : اجازه میدهد که ربات در یک مسیر دایره‌ای حول base بچرخد. (θ_1)
۲- چرخش حول Shoulder : اجازه میدهد که ربات بازویش را بالا و پایین ببرد. (θ_2)
۳- چرخش حول Elbow : درواقع اجازه میدهد که ربات بازویش را خم کند. (θ_3)
۴- چرخش حول محور pitch از Wrist : این به ربات اجازه می دهد تا مچ دست خود را به سمت بالا و پایین خم کند. (θ_4)
۵- چرخش حول محور roll از Wrist : این به ربات اجازه می دهد تا مچ دست خود را از یک طرف به سمت دیگر بچرخاند. (θ_5)
۶- چرخش حول محور yaw از Wrist: این به ربات اجازه می دهد تا مچ دست خود را حول محور خود بچرخاند. (θ_6)

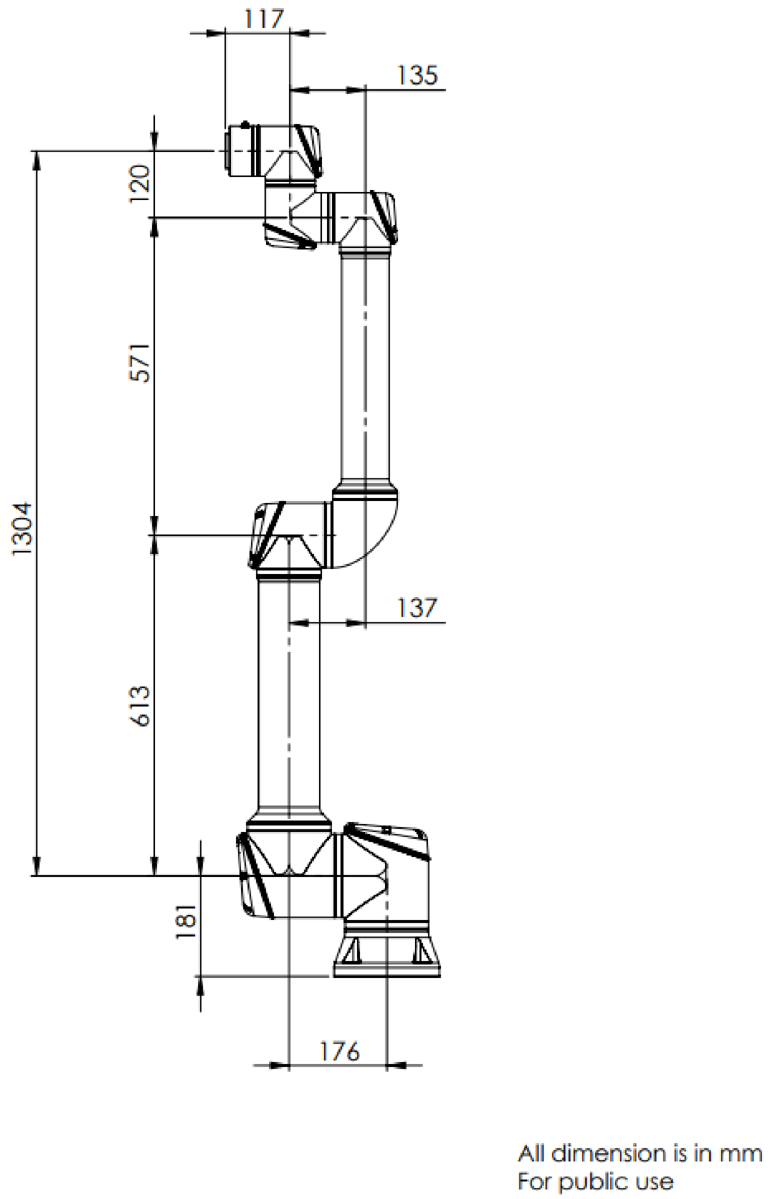


شکل ۲- نمای انفجاری ربات و نمایش متغیر های مفصلی بر روی آن

۱-۴- مشخصات ربات

۱-۴-۱-ابعاد لینک‌ها

ابعاد باتوجه به کاتالوگ‌ها موجود در شکل زیر به وضوح قابل مشاهده میباشد. بر اساس این شکل به راحتی میتوان Offset و سایر موارد لازم در جدول دنویت-هارتنبرگ که در کلاس تا حدی تدریس شده است را پیدا کرد.

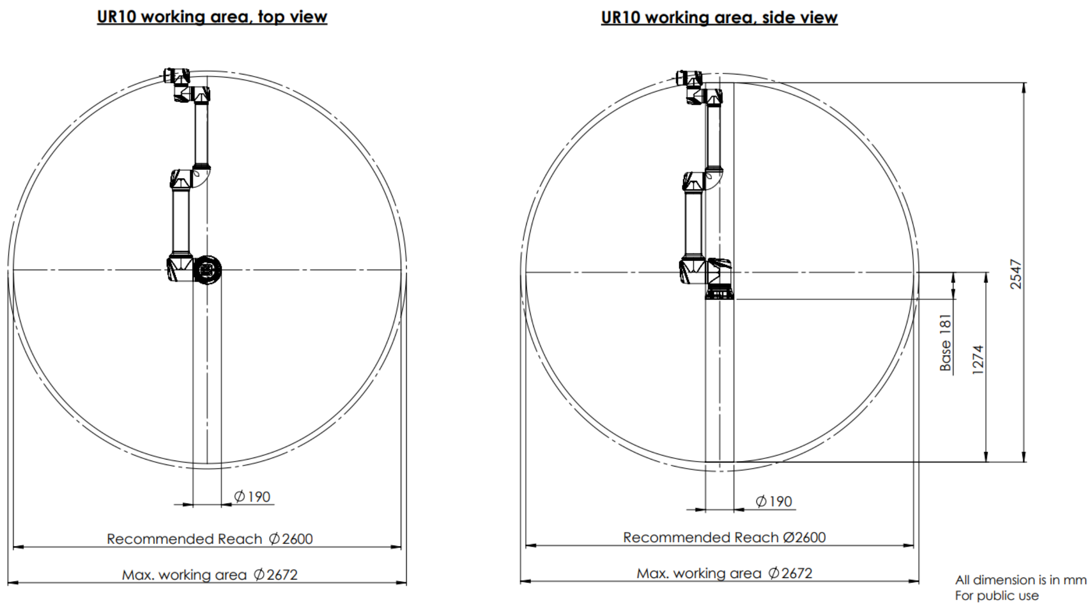


شکل ۳- ابعاد لینک‌ها و آفست‌های موجود بین آن‌ها در ربات UR10

۱-۴-۲- فضای کاری

محدوده کاری تمامی مفاصل این ربات، ۳۶۰ درجه می باشد و ربات UR10 دارای فضای کاری کروی با قطر تقریبی ۲۶۰۰ میلی متر است. این بدان معنی است که بازوی ربات می تواند به هر نقطه ای در این حجم کروی برسد و به آن امکان می دهد طیف گسترده ای از وظایف را انجام دهد.

میتوانید فضای کاری آن را به صورت بهتر و کاملتری در شکل زیر مشاهده کنید.



شکل ۴- فضای کاری ربات UR10

فصل دوم

سینماتیک مستقیم

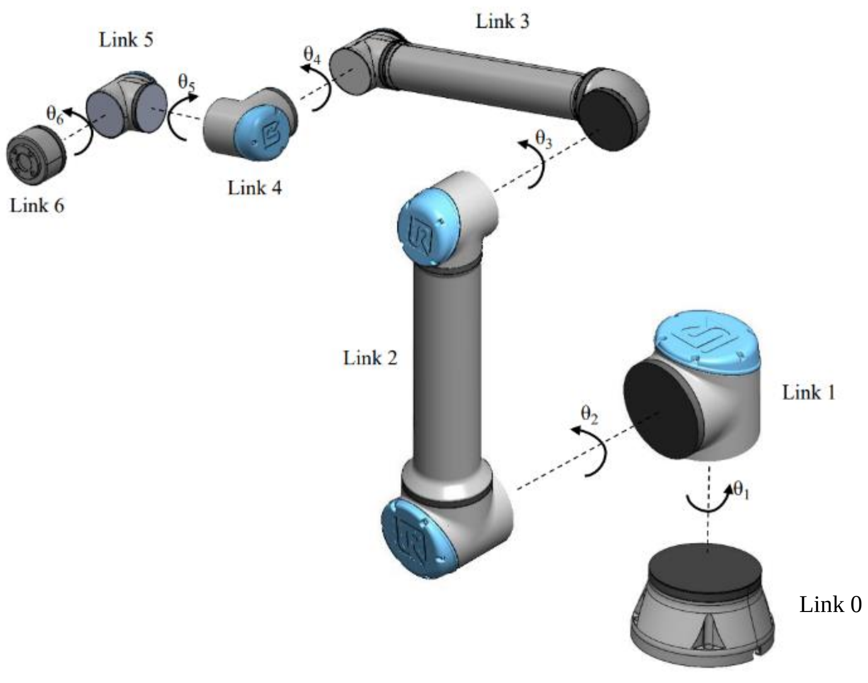
در این قسمت باتوجه به اسلایدها و مرج اصلی درس مراحلی را جهت حل مسئله سینماتیک مستقیم مربوط به ربات UR10 پیش میبریم. به طور مختصر این مراحل عبارتند از:

۱- پیدا کردن درجات آزادی و نام گذاری آن‌ها و شماره گذاری لینک‌ها
۲- انتخاب مناسب فریم‌ها و نمایش آن‌ها بر روی ربات و به دست آوردن ابعاد
۳- به دست آوردن جدول DH و همچنین ماتریس تبدیل ${}^{0}T_6$

در ادامه به ترتیب به این مراحل میپردازیم. پس از این مراحل به سراغ صحه گذاری میرویم و مثال‌هایی را با روش‌های مختلف برای این منظور انجام میدهیم. در واقع در این مرحله لازم است که مقدار درجات آزادی مختلف ربات داده شود و سپس به کمک روابط به دست آمده از سینماتیک مستقیم، موقعیت End effector را تعیین نماییم.

۲-۱- پیدا کردن درجات آزادی و نام گذاری آن‌ها

این مرحله درواقع در همان بخش سوم مقدمه انجام شد اما برای کامل بودن سلسله مراحل کار لازم است تا بار دیگر در این قسمت به آن اشاره کنیم. رباتی که با آن سر و کار داریم، یک ربات ۶ درجه آزادی میباشد که همه مفاصل آن از نوع Revolute میباشند. نام گذاری مفاصل و همچنین لینک‌ها در شکل زیر آمده است. لینک‌ها از ۰ تا ۶ شماره گذاری شده اند که لینک ۰ مربوط به Base میباشد. مفاصل هم از θ_1 تا θ_6 شماره گذاری شده‌اند که میتوانید در شکل زیر مشاهده کنید. توضیحات مربوط به هریک از مفاصل در مقدمه آمده است که از تکرار آن‌ها در این قسمت خودداری میکنیم.



شکل ۵- درجات آزادی ربات

۲-۲- انتخاب فریم‌ها

برای انتخاب فریم‌های مناسب براساس اسلایدهای درس پیش میرویم. برای این منظور ابتدا باید فریم گذاری مربوط به لینک ۱ تا ۵ را براساس توضیحات زیر انجام دهیم. (فریم گذاری همه لینک‌ها به جز اولین و آخرین لینک)

۱- محور $\hat{Z}_i$ از فریم {i} را $\hat{Z}_i$، می‌نامیم که منطبق بر محوری است که مفصل iام حول آن دوران دارد.

۲- مبدأ این فریم در محل تقاطع عمود مشترک بین محور مفصل i و مفصل i+1ام با محور iام میباشد. (جایی که a_i متقاطع با $\hat{Z}_i$ میباشد.) چنانچه دو محور i و i+1ام متقاطع باشند، محل قرارگیری مبدأ در همان نقطه تقاطع میباشد.

۳- محور $\hat{X}_i$ در امتداد عمود مشترک بین محور i و i+1ام میباشد یا به عبارتی در امتداد a_i به سمت محور i+1ام میباشد.

۴- محور $\hat{Y}_i$ به کمک قاعده دست راست به دست می‌آید تا فریم iام را کامل کند.

دراینجا باتوجه به اینکه مفاصل همگی Revolute اند فقط قاعده مربوط به لینک صفرم و آخر مربوط به مفصل Revolute را کافی است که مرور کنیم.

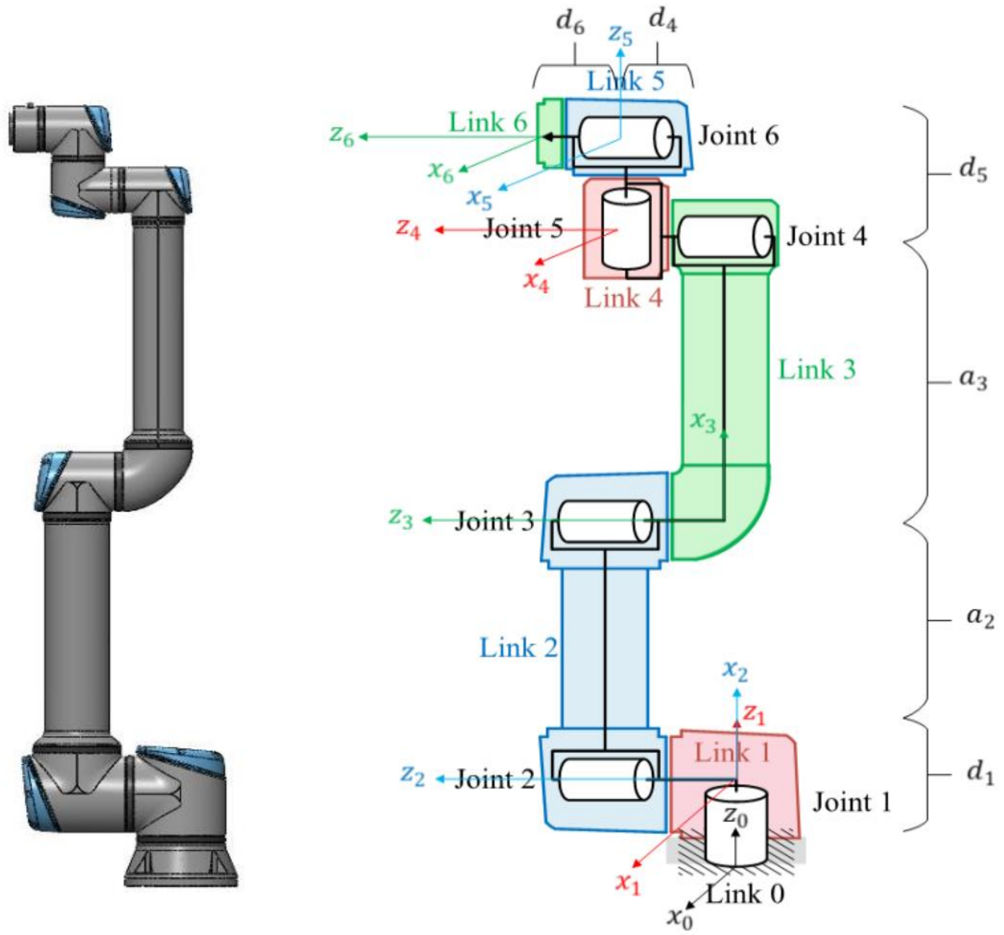
- برای فریم {0} باید به صورت زیر عمل کنیم:

این فریم دلخواه است ولی برای آنکه در محاسبات راحت باشیم، $\hat{Z}_0$ را در جهت محور مفصل ۱ انتخاب میکنیم (مانند $\hat{Z}_1$) و باید فریم‌های {0} و {1} به گونه‌ای قرار گیرند که اگر $\theta_1 = 0$ شود، آنگاه این دو فریم روی هم قرار گیرند.

- برای فریم {n} بهتر است به صورت زیر عمل میکنیم:

باید $\hat{X}_n$ به گونه‌ای قرار گیرد که چنانچه $\theta_n = 0$ شد، سپس $\hat{X}_n$ در امتداد $\hat{X}_{n-1}$ قرار گیرد. همچنین باید به نحوی باشد که بین $\hat{X}_n$ و $\hat{X}_{n+1}$ در راستای $\hat{Z}_{n+1}$ هیچ گونه فاصله‌ای نباشد و در حقیقت $d_n = 0$ باشد.

آنچه در صفحه قبل گفته شد فریم گذاری اساندارد است اما برای مثال به خصوص در فریم $\{0\}$ و $\{6\}$ میتوان کمی متفاوت تر عمل کرد. در نهایت با توجه به توضیحات داده شده در صفحه قبل میتوان فریم گذاری را به صورت زیر انجام داد. دقت شود که x_2 و x_3 در راستای z_3 فاصله‌شان ۰ است.



شکل ۶- فریم گذاری ربات

۲-۳- استخراج جدول پارامترهای DH

برای این منظور ابتدا لازم است تا ابتدا ابعاد مربوطه را به دست آوریم. برای این کار هم میتوان از کاتالوگ مربوط به این ربات استفاده کرد و هم از فایل CAD اندازه‌ها را به دست آورد. به هرجهت باید اندازه‌ها به نحوی باشد تا بعدا برای صحه سنجی دچار مشکل نشویم. اندازه‌ها در جدول زیر قرار گرفته است.

جدول ۱- ابعاد مربوط به ربات

	اندازه (برحسب mm)
d_1	128
a_2	612.9
a_3	517.6
d_4	163.9
d_5	115.7
d_6	92.2

برای به دست آوردن جدول DH مطابق اسلاید، تعاریف زیر را برای هر پارامتر داریم:

- a_{i-1}: فاصله میان محورهای $\hat{Z}_{i-1}$ و $\hat{Z}_i$ در راستای محور $\hat{X}_{i-1}$
- α_{i-1}: زاویه میان محورهای $\hat{Z}_{i-1}$ و $\hat{Z}_i$ حول محور $\hat{X}_{i-1}$
- d_i: فاصله میان محورهای $\hat{X}_{i-1}$ و $\hat{X}_i$ در راستای محور $\hat{Z}_{i-1}$
- θ_i: زاویه میان محورهای $\hat{X}_{i-1}$ و $\hat{X}_i$ در راستای محور $\hat{Z}_{i-1}$

براین اساس برای جدول DH داریم:

جدول ۲- جدول پارامترهای DH

i	$\alpha_{i-1}\ (mm)$	$a_{i-1}\ (rad)$	$d_i\ (mm)$	$\theta_i\ (rad)$
1	0	0	128	θ_1
2	$\frac{\pi}{2}$	0	0	θ_2
3	0	612.9	0	θ_3
4	0	571.6	163.9	θ_4
5	$-\frac{\pi}{2}$	0	115.7	θ_5
6	$\frac{\pi}{2}$	0	92.2	θ_6

۴-۲- ماتریس‌های تبدیل همگن

در گام بعدی لازم است تا براساس جدول DH ماتریس‌های تبدیل همگن بین هردو لینک متوالی را بیابیم.
برای این منظور باتوجه به اطلاعات هر سطر باید از فرمول کلی زیر استفاده نماییم:

$$^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i\ c\alpha_{i-1} & c\theta_i\ c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}\, d_i \\ s\theta_i\ s\alpha_{i-1} & c\theta_i\ s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

حال براین اساس برای iهای ۱ تا ۶ باید این ماتریس‌ها را محاسبه نماییم. کافی است مقادیر α_{i-1} و a_{i-1} و d_i و θ_i را از جدول DH در ماتریس فوق جاگذاری نماییم.

$$^{0}T_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \quad {}^{1}T_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \quad {}^{3}T_4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}T_5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \quad {}^{5}T_6 = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & -1 & -d_6 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

۲-۵- سینماتیک مستقیم موقعیت و جهت گیری

برای این منظور باید ابتدا ${}^{0}T_6$ بیابیم. باتوجه به ماتریس‌های تبدیل همگن به دست آمده در مرحله قبل و ضرب آن‌ها این امر امکان پذیر میباشد:

$$ {}^{0}T_6 = {}^{0}T_1 \; {}^{1}T_2 \; {}^{2}T_3 \; {}^{3}T_4 \; {}^{4}T_5 \; {}^{5}T_6 $$

$$ {}^{0}T_6 = $$

$$ \begin{bmatrix} c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1 & -s_6(-s_1s_5 + c_1c_5c_{234}) - s_{234}c_1c_6 & s_1c_5 + s_5c_1c_{234} & c_1(a_2c_2 + a_3c_{23}) + d_4s_1 - d_5s_{234}c_1 + d_6(s_1c_5 + s_5c_1c_{234}) \\ c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1 & -s_6(c_1s_5 + s_1c_5c_{234}) - s_{234}s_1c_6 & -c_1c_5 + s_5s_1c_{234} & s_1(a_2c_2 + a_3c_{23}) - d_4c_1 - d_5s_{234}s_1 - d_6(c_1c_5 - s_5s_1c_{234}) \\ s_6c_{234} + s_{234}c_5c_6 & c_6c_{234} - s_{234}c_5s_6 & s_5s_{234} & a_2s_2 + a_3s_{23} + d_1 + d_5c_{234} + d_6s_5s_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

حال از برابر قرار دادن طرفین میتوان معادلات مربوط به سینماتیک مستقیم موقعیت و جهت گیری را به دست آورد:

$$ r_{11} = c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1 $$

$$ r_{21} = c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1 $$

$$ r_{31} = s_6c_{234} + s_{234}c_5c_6 $$

$$ r_{12} = -s_6(-s_1s_5 + c_1c_5c_{234}) - s_{234}c_1c_6 $$

$$ r_{22} = -s_6(c_1s_5 + s_1c_5c_{234}) - s_{234}s_1c_6 $$

$$ r_{32} = c_6c_{234} - s_{234}c_5s_6 $$

$$ r_{13} = s_1c_5 + s_5c_1c_{234} $$

$$ r_{23} = -c_1c_5 + s_5s_1c_{234} $$

$$ r_{33} = s_5s_{234} $$

$$ p_x = c_1(a_2c_2 + a_3c_{23}) + d_4s_1 - d_5s_{234}c_1 + d_6(s_1c_5 + s_5c_1c_{234}) $$

$$ p_y = s_1(a_2c_2 + a_3c_{23}) - d_4c_1 - d_5s_{234}s_1 - d_6(c_1c_5 - s_5s_1c_{234}) $$

$$ p_z = a_2s_2 + a_3s_{23} + d_1 + d_5c_{234} + d_6s_5s_{234} $$

$$ \rightarrow \; {}^{0}R_6 = \begin{bmatrix} c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1 & -s_6(-s_1s_5 + c_1c_5c_{234}) - s_{234}c_1c_6 & s_1c_5 + s_5c_1c_{234} \\ c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1 & -s_6(c_1s_5 + s_1c_5c_{234}) - s_{234}s_1c_6 & -c_1c_5 + s_5s_1c_{234} \\ s_6c_{234} + s_{234}c_5c_6 & c_6c_{234} - s_{234}c_5s_6 & s_5s_{234} \end{bmatrix} $$

$$ \rightarrow \; {}^{0}P_6 = \begin{bmatrix} c_1(a_2c_2 + a_3c_{23}) + d_4s_1 - d_5s_{234}c_1 + d_6(s_1c_5 + s_5c_1c_{234}) \\ s_1(a_2c_2 + a_3c_{23}) - d_4c_1 - d_5s_{234}s_1 - d_6(c_1c_5 - s_5s_1c_{234}) \\ a_2s_2 + a_3s_{23} + d_1 + d_5c_{234} + d_6s_5s_{234} \end{bmatrix} $$

پس از تعیین ماتریس تبدیل عضو آخر ربات نسبت به دستگاه پایه می توان ماتریس دوران آن که بیانگر وضعیت دورانی عضو انتهایی ربات است را از ماتریس تبدیل استخراج نمود. برای این موضوع طبق اسلایدها ۴ روش داشتیم که در ادامه به کمک هریک از روش‌ها میتوان جهت گیری نسبت به دستگاه پایه را بیان کرد.

روش اول) Z-Y-X Euler angles

$$R_{ZYX}(\alpha,\beta,\gamma)=\begin{bmatrix} c\alpha\ \ c\beta & c\alpha\, s\beta\, s\gamma - s\alpha\, c\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma \\ s\alpha\, c\beta & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & s\alpha\, s\beta\, c\gamma - c\alpha\, s\gamma \\ -s\beta & c\beta\, s\gamma & c\beta\, c\gamma \end{bmatrix}$$

$${}^{0}R_{6}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

حال از برابر قرار دادن این دو ماتریس خواهیم داشت:

$$\rightarrow \begin{cases} \beta = atan2\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) \\ \alpha = atan2\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right) \\ \gamma = atan2\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right) \end{cases}$$

$$\rightarrow \begin{cases} \beta = atan2\left(-(s_6c_{234} + s_{234}c_5c_6), \sqrt{(c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1\ )^2 + (c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1)^2}\right) \\ \alpha = atan2\left(\frac{c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1}{\cos\beta}, \frac{c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1}{\cos\beta}\right) \\ \gamma = atan2\left(\frac{c_6c_{234} - s_{234}c_5s_6}{\cos\beta}, \frac{s_5s_{234}}{\cos\beta}\right) \end{cases}$$

روش دوم) X-Y-Z Fixed angles

$$R_{XYZ}(\gamma,\beta,\alpha)=\begin{bmatrix} c\alpha\ \ c\beta & c\alpha\, s\beta\, s\gamma - s\alpha\, c\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma \\ s\alpha\, c\beta & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & s\alpha\, s\beta\, c\gamma - c\alpha\, s\gamma \\ -s\beta & c\beta\, s\gamma & c\beta\, c\gamma \end{bmatrix}$$

$${}^{0}R_{6}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

حال از برابر قرار دادن این دو ماتریس نتایج مثل مرحله قبل خواهد شد و خواهیم داشت:

$$\rightarrow \begin{cases} \beta = atan2\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) \\ \alpha = atan2\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right) \\ \gamma = atan2\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right) \end{cases}$$

$$\rightarrow \begin{cases} \beta = atan2\left(-(s_6c_{234} + s_{234}c_5c_6), \sqrt{(c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1)^2 + (c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1)^2}\right) \\ \alpha = atan2\left(\frac{c_6(s_5c_1 + s_1c_5c_{234}) - s_6s_{234}s_1}{\cos\beta}, \frac{c_6(-s_1s_5 + c_1c_5c_{234}) - s_6s_{234}c_1}{\cos\beta}\right) \\ \gamma = atan2\left(\frac{c_6c_{234} - s_{234}c_5s_6}{\cos\beta}, \frac{s_5s_{234}}{\cos\beta}\right) \end{cases}$$

روش سوم) **Equivalent Angle-Axis**

$$\begin{bmatrix} k_xk_xv\theta + c\theta & k_xk_yv\theta - k_z\,s\theta & k_xk_zv\theta + k_y\,s\theta \\ k_xk_yv\theta + k_z\,s\theta & k_yk_yv\theta + c\theta & k_yk_zv\theta - k_x\,s\theta \\ k_xk_zv\theta - k_y\,s\theta & k_yk_zv\theta + k_x\,s\theta & k_yk_yv\theta + c\theta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\rightarrow \begin{cases} \theta = Acos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \\ \hat{K} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{cases}$$

روش چهارم) **Euler Parameters**

$$\rightarrow \begin{cases} \epsilon_4 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \\ \epsilon = \frac{1}{4\epsilon_4}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{cases}$$

۲-۶- بررسی سینماتیک مستقیم برای چندین وضعیت متفاوت

برای بررسی جواب‌ها ما کد را به زبان پایتون در فایل Jupyter notebook نوشته ایم. در ادامه نیز برای چندین حالت کارکرد قسمت سینماتیک مستقیم را بررسی می‌کنیم.

الف) بررسی وضعیت صفر ربات

در این حالت باید تمامی زوایای ورودی را برابر با صفر درجه بگذاریم. در این صورت خواهیم داشت

A) Zero state

- in this state all joint variables have zero value

```
In [10]: theta1 = math.radians(0)
         theta2 = math.radians(0)
         theta3 = math.radians(0)
         theta4 = math.radians(0)
         theta5 = math.radians(0)
         theta6 = math.radians(0)

         desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
         T_0_6_A = forward(desired_theta)

         print('T_0_6 : ')
         for row in T_0_6_A:
             print(row)

         T_0_6 :
         [1.0, 0.0, 0.0, 1184.5]
         [0.0, 6.123233995736766e-17, -1.0, -256.1]
         [0.0, 1.0, 6.123233995736766e-17, 243.7]
         [0.0, 0.0, 0.0, 1.0]
```

ب) بررسی وضعیت کاملا کشیده

باتوجه به فریم گذاری که داشتیم، زمانی ربات به حالت کشیده در می‌آید که زوایای تتا به صورت زیر باشند:

تتا۱: دلخواه | تتا۲: ۹۰ درجه | تتا۳: ۰ درجه | تتا۴: ۹۰-درجه | تتا۵: ۰ درجه | تتا۶: دلخواه

که تتا۱ را مقدار ۴۵ درجه و تتا۶ را هم مقدار ۶۰ درجه به عنوان مقادیر دلخواه می‌دهیم.

B) fully-stretched state

- We have below values for joint variables in fully-stretched state:

theta1 = arbitrary | theta2 = 0 | theta3 = 90 | theta4 = 0 | theta5 = -90 | theta6 = arbitrary

We consider theta1 = -20 and theta6 = 60 degrees.

```
In [11]: theta1 = math.radians(45)
         theta2 = math.radians(90)
         theta3 = math.radians(0)
         theta4 = math.radians(-90)
         theta5 = math.radians(0)
         theta6 = math.radians(60)

         desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
         T_0_6_B = forward(desired_theta)

         print('T_0_6 : ')
         for row in T_0_6_B:
             print(row)

         T_0_6 :
         [0.3535533905932738, -0.6123724356957946, 0.7071067811865476, 181.09004666187485]
         [0.3535533905932739, -0.6123724356957946, -0.7071067811865476, -181.09004666187474]
         [0.8660254037844386, 0.5000000000000001, 6.123233995736766e-17, 1428.2]
         [0.0, 0.0, 0.0, 1.0]
```

ج) بررسی یک وضعیت دلخواه

برای بررسی وضعیت دلخواه ما زوایای مفاصل را به صورت دلخواه انتخاب کردیم و برابر مقادیر زیر است:

تتا۱: ۳۰ درجه | تتا۲: ۴۵ درجه | تتا۳: ۲۶ درجه | تتا۴: ۵۰ درجه | تتا۵: ۶۰ درجه | تتا۶: ۸۰ درجه

در این صورت خواهیم داشت:

C) Arbitrary state

- We have below values for joint variables in arbitrary state:

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

In [12]:
```
theta1 = math.radians(30)
theta2 = math.radians(45)
theta3 = math.radians(26)
theta4 = math.radians(50)
theta5 = math.radians(60)
theta6 = math.radians(80)

desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
T_0_6_C = forward(desired_theta)

print('T_0_6 : ')
for row in T_0_6_C:
    print(row)
```

```
T_0_6 :
[-0.8449695581951862, 0.5171601307605163, -0.1362785561825407, 519.983544410926]
[-0.3141952242140994, -0.6862252123034819, -0.6560307302863893, 57.72552628018109]
[-0.432790719406602, -0.5115079248172085, 0.7423286577013642, 1110.696960974777]
[0.0, 0.0, 0.0, 1.0]
```

فصل سوم

سینماتیک معکوس

۳-۱- حل تحلیلی و بررسی تعداد جواب

در مسئله سینماتیک معکوس در حقیقت به دنبال آن هستیم که با داشتن مکان و چرخش End-effector نسبت به فریم پایه، بتوانیم مقادیر زوایای مفاصل را بیابیم و درواقع بیابیم که با چه تتاهایی میتوان به موقعیت دلخواه رسید. از نظر تعداد جواب‌ها ممکن است اصلا برای آن موقعیت خواسته شده، جوابی پیدا نشود یا حتی چندین جواب پیدا شود.

ربات UR10 از آنجایی که محور ۳ جوینت آخر مربوط به Wrist باهم در یک نقطه intersection ندارند، میتوان گفت که مسئله decouple نمیباشد که مانند اسلایدها از روش Piper حل کنیم. در ادامه از روش هندسی و جبری مسئله سینماتیک معکوس را حل کردیم.

ابتدا مسئله سینماتیک معکوس را به روش جبری و با تکنیک separating out variables حل می‌کنیم.

یافتن زاویه θ_1 :

$$ {}^0T_6 = {}^0T_1 \ {}^1T_2 \ {}^2T_3 \ {}^3T_4 \ {}^4T_5 \ {}^5T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

سمت راست رابطه بالا همان ماتریس تبدیل مطلوب و ورودی مسئله می‌باشد. بنابراین مقادیر عددی درایه‌های آن برای ما معلوم است.

رابطه بالا را از سمت چپ در وارون ماتریس ${}^0T_1$ ضرب می‌کنیم تا زاویه θ_1 تنها مجهول در سمت راست تساوی باشد.

$$ {}^1T_6\big|_{para} = {}^1T_2 \ {}^2T_3 \ {}^3T_4 \ {}^4T_5 \ {}^5T_6 = \left({}^0T_1\right)^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

در اینصورت ماتریس تبدیل ${}^1T_6$ پارامتریک به شکل زیر حاصل می‌شود:

$$ \begin{bmatrix} c_{234}c_5c_6 - s_{234}s_6 & -s_{234}c_6 - c_{234}c_5s_6 & c_{234}s_5 & a_2c_2 + a_3c_{23} - d_5s_{234} + d_6c_{234}s_5 \\ c_6s_5 & -s_6s_5 & -c_5 & -d_4 - d_6c_5 \\ s_{234}c_5c_6 + c_{234}s_6 & c_{234}c_6 - s_{234}c_5s_6 & s_{234}s_5 & a_2s_2 + a_3s_{23} + d_5c_{234} + d_6s_{234}s_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

ماتریس تبدیل ${}^{1}T_6$ عددی نیز به فرم زیر می‌باشد:

$$ {}^{1}T_6\big|_{Num} = \left({}^{0}T_1\right)^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$ {}^{1}T_6\big|_{Num} = \begin{bmatrix} r_{11}c_1 + r_{21}s_1 & r_{12}c_1 + r_{22}s_1 & r_{13}c_1 + r_{23}s_1 & p_xc_1 + p_ys_1 \\ r_{21}c_1 - r_{11}s_1 & r_{22}c_1 - r_{12}s_1 & r_{23}c_1 - r_{13}s_1 & p_yc_1 - p_xs_1 \\ r_{31} & r_{32} & r_{33} & p_z - d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

از تساوی درایه‌های T_{23} و T_{24} دو ماتریس عددی و پارامتریک خواهیم داشت:

$$ -c_5 = r_{23}c_1 - r_{13}s_1 \tag{5} $$

$$ -d_4 - d_6c_5 = p_yc_1 - p_xs_1 \tag{6} $$

با جایگذاری مقدار عبارت c_5 از رابطه (۵) در رابطه (۶) خواهیم داشت:

$$ -d_4 + d_6(r_{23}c_1 - r_{13}s_1) = p_yc_1 - p_xs_1 \tag{7} $$

با کمی ساده سازی، این معادله به فرم معادله نوع اول مثلثاتی جزوه در می‌آید.

$$ (p_y - d_6r_{23})\, c_1 + (d_6r_{13} - p_x)\, s_1 = -d_4 \tag{8} $$

پاسخ این معادله از رابطه (۹) قابل دسترسی است

$$ \theta_1 = 2Atan\left(\frac{b \pm \sqrt{b^2 + a^2 - c^2)}}{a + c}\right) \tag{9} $$

که در آن پارامترهای c, b, a تعریفی مطابق زیر دارند:

$a =$ ضریب کسینوس زاویه

$b =$ ضریب سینوس زاویه

$c =$ مقدار ثابت سمت راست تساوی

$$ \begin{cases} a = p_y - d_6r_{23} \\ b = d_6r_{13} - p_x \\ c = -d_4 \end{cases} $$

با توجه به علامت $\pm$ موجود در آرگومان تانژانت وارون، دو پاسخ برای زاویه θ_1 خواهیم داشت.

یافتن زاویه θ_5 :

چون در قسمت قبل مقدار زاویه θ_1 را پیدا کردیم، اکنون به راحتی می‌توان از رابطه (۵)، زاویه θ_5 را پیدا نمود:

$$-c_5 = r_{23}c_1 - r_{13}s_1 \;\Rightarrow c_5 = r_{13}s_1 - r_{23}c_1$$

می‌دانیم تابع کسینوس تابعی زوج است، یعنی $c_5 = c_{-5}$

پس به ازای هر مقدار از زاویه θ_1، دو مقدار برای θ_5 بدست می‌آید:

$$\theta_5 = \pm Acos(r_{13}s_1 - r_{23}c_1) \tag{9}$$

با توجه به اینکه θ_1 خود دو مقدار دارد، چهار پاسخ برای θ_5 حاصل می‌شود.

یافتن زاویه $\theta_2+\theta_3+\theta_4$:

از این قسمت به بعد، زوایای مفصلی بطور مستقیم از ماتریس تبدیل فریم نهایی به زمین، قابل محاسبه است.

از آنجا که محورهای دوران مفاصل ۲ و ۳ و ۴ موازی اند، ابتدا مقدار مجموع این زوایا بدست می‌آید.

از ماتریس تبدیل فریم نهایی به زمین (${}^0T_6$) داریم:

$$r_{13} = s_1c_5 + s_5c_1c_{234} \Rightarrow c_{234} = \frac{r_{13} - s_1c_5}{s_5c_1}$$

$$r_{33} = s_5s_{234} \Rightarrow s_{234} = \frac{r_{33}}{s_5}$$

$$\theta_{234} = Atan2\left(\frac{r_{33}}{s_5}, \frac{r_{13} - s_1c_5}{s_5c_1}\right) \tag{10}$$

زاویه θ_1، دارای دو تعدد پاسخ و زاویه θ_5، دارای چهار تعدد پاسخ می‌باشد.

بنابراین θ_{234} دارای $2 \times 4 = 8$ تعدد پاسخ خواهد بود.

یافتن زاویه θ_6 :

$$r_{32} = c_{234}c_6 - s_{234}c_5s_6$$

$$r_{31} = c_{234}s_6 + s_{234}c_5c_6$$

با کمی دقت می‌توان دریافت که این دستگاه معادلات، به فرم معادلات جبری نوع سوم اند که در جزوه درمورد روش حل آنها بحث شده است.

$$k_1 = c_{234}, k_2 = s_{234}c_5$$

$$r_{32} = k_1c_6 - k_2s_6$$

$$r_{31} = k_1s_6 + k_2c_6$$

$$\theta_6 = Atan2(r_{31}, r_{32}) - Atan2(s_{234}c_5, c_{234}) \tag{11}$$

یافتن زاویه θ_3 :

$$p_y = s_1(a_2c_2 + a_3c_{23}) - d_4c_1 - d_5s_{234}s_1 - d_6(c_1c_5 - s_5s_1c_{234})$$

$$p_z = a_2s_2 + a_3s_{23} + d_1 + d_5c_{234} + d_6s_5s_{234}$$

با کمی مرتب سازی و گرفتن تغییر متغیر داریم:

$$\frac{p_y + d_4c_1 + d_5s_{234}s_1 + d_6(c_1c_5 - s_5s_1c_{234})}{s_1} = F_2 = a_2c_2 + a_3c_{23}$$

$$p_z - d_1 - d_5c_{234} - d_6s_5s_{234} = F_3 = a_2s_2 + a_3s_{23}$$

$$F_2^2 + F_3^2 = a_2^2 + a_3^2 + 2a_2a_3c_3 \Rightarrow c_3 = \frac{F_2^2 + F_3^2 - a_2^2 - a_3^2}{2a_2a_3}$$

$$\theta_3 = \pm Acos\left(\frac{F_2^2 + F_3^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \tag{12}$$

یافتن زاویه θ_2 :

با تعریف دو پارامتر Q_1, Q_2 می‌توان به فرم سوم معادلات جبری دست یافت:

$$\begin{cases} Q_1 = a_2 + a_3 c_3 \\ Q_2 = a_3 s_3 \end{cases} \Rightarrow \begin{cases} F_2 = Q_1 c_2 - Q_2 s_2 \\ F_3 = Q_1 s_2 + Q_2 c_2 \end{cases}$$

$$\theta_2 = Atan2(F_3, F_2) - Atan2(Q_2, Q_1) \tag{13}$$

یافتن زاویه θ_4 :

$$\theta_4 = (\theta_2 + \theta_3 + \theta_4) - \theta_2 - \theta_3$$

۳-۲- بررسی سینماتیک معکوس برای چندین وضعیت متفاوت

در این قسمت برای همان حالاتی که در سینماتیک مستقیم مورد بررسی قرار دادیم، سینماتیک معکوس را بررسی میکنیم. برای این منظور ماتریس‌های تبدیل خروجی در سینماتیک مستقیم را به عنوان ورودی بخش سینماتیک معکوس قرار می‌دهیم.

الف) بررسی وضعیت صفر ربات

در این حالت در قسمت سینماتیک مستقیم تمامی زوایای ورودی را برابر با صفر درجه گذاردیم. حال اگر ماتریس تبدیل به دست آمده از آن بررسی را در این جا نیز قرار دهیم، انتظار داریم تمامی زاویای ۰ را به عنوان خروجی بگرداند.

A) Zero state

- We gave zero values to joint varibles in forward kinematics and it took us Transformation matrix (T_0_6_first) as a result. now we give this matrix as an input to inverse function.

In [13]:
```
T_0_6_A
```

Out[13]:
```
[[1.0, 0.0, 0.0, 1184.5],
 [0.0, 6.123233995736766e-17, -1.0, -256.1],
 [0.0, 1.0, 6.123233995736766e-17, 243.7],
 [0.0, 0.0, 0.0, 1.0]]
```

In [24]:
```
theta_A = inverse(T_0_6_A)
i = 0
for th in theta_A:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  0.0  theta2:  0.0  theta3:  0.0  theta4:  0.0  theta5:  0.0  theta6:  0.0
ans 2  theta1:  164.24  theta2:  180.0  theta3:  0.0  theta4:  -180.0  theta5:  164.24  theta6:  0.0
```

ب) بررسی وضعیت کاملا کشیده

باتوجه به فریم گذاری گفتیم، زمانی ربات به حالت کشیده در می‌آید که زویای تتا به صورت زیر باشند:

تتا۱: دلخواه | تتا۲: ۹۰ درجه | تتا۳: ۰ درجه | تتا۴: ۹۰-درجه | تتا۵: ۰ درجه | تتا۶: دلخواه

که تتا۱ را مقدار ۴۵ درجه و تتا۶ را هم مقدار ۶۰ درجه به عنوان مقادیر دلخواه دادیم. حال اگر ماتریس تبدیل نهایی را به تابع سینماتیک معکوس دهیم انتظار داریم که همین زوایای گفته شده را بگیریم.

B) fully-stretched state

- We should use T_0_6_B as input and get the below angles in results:

theta1 = 45 | theta2 = 90 | theta3 = 0 | theta4 = -90 | theta5 = 0 | theta6 = 60

In [17]:
```
T_0_6_B
```

Out[17]:
```
[[0.3535533905932738,
  -0.6123724356957946,
  0.7071067811865476,
  181.09004666187485],
 [0.3535533905932739,
  -0.6123724356957946,
  -0.7071067811865476,
  -181.09004666187474],
 [0.8660254037844386, 0.5000000000000001, 6.123233995736766e-17, 1428.2],
 [0.0, 0.0, 0.0, 1.0]]
```

In [26]:
```
theta_B = inverse(T_0_6_B)
i = 0
for th in theta_B:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  45.0  theta2:  90.0  theta3:  0.0  theta4:  -90.0  theta5:  0.0  theta6:  60.0
ans 2  theta1:  45.0  theta2:  -90.0  theta3:  0.0  theta4:  90.0  theta5:  0.0  theta6:  60.0
ans 3  theta1:  135.0  theta2:  90.0  theta3:  0.0  theta4:  -90.0  theta5:  0.0  theta6:  60.0
ans 4  theta1:  135.0  theta2:  -90.0  theta3:  0.0  theta4:  90.0  theta5:  0.0  theta6:  120.0
```

ج) بررسی یک وضعیت دلخواه

برای بررسی وضعیت دلخواه ما زوایای مفاصل را به صورت دلخواه انتخاب کردیم و برابر مقادیر زیر بود:

تتا۱: ۳۰ درجه | تتا۲: ۴۵ درجه | تتا۳: ۲۶ درجه | تتا۴: ۵۰ درجه | تتا۵: ۶۰ درجه | تتا۶: ۸۰ درجه

حال اگر ماتریس تبدیل نهایی را به تابع سینماتیک معکوس دهیم انتظار داریم که همین زوایای گفته شده را به عنوان یکی از خروجی‌های قابل قبول بگیریم.

C) Arbitrary state

- We should use T_0_6_B as input and get the below angles in results:

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

In [27]:
```
T_0_6_C
```

Out[27]:
```
[[-0.8449695581951862,
  0.5171601307605163,
  -0.1362785561825407,
  519.983544410926],
 [-0.3141952242140994,
  -0.6862252123034819,
  -0.6560307302863893,
  57.72552628018109],
 [-0.432790719406602,
  -0.5115079248172085,
  0.7423286577013642,
  1110.696960974777],
 [0.0, 0.0, 0.0, 1.0]]
```

In [28]:
```
theta_C = inverse(T_0_6_C)
i = 0
for th in theta_C:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  30.0  theta2:  45.0  theta3:  26.0  theta4:  50.0  theta5:  60.0  theta6:  -280.0
ans 2  theta1:  30.0  theta2:  70.08  theta3:  -26.0  theta4:  76.92  theta5:  60.0  theta6:  -280.0
ans 3  theta1:  30.0  theta2:  28.77  theta3:  72.95  theta4:  -160.72  theta5:  -60.0  theta6:  -100.0
ans 4  theta1:  30.0  theta2:  98.77  theta3:  -72.95  theta4:  -84.82  theta5:  -60.0  theta6:  -100.0
ans 5  theta1:  175.03  theta2:  82.09  theta3:  66.85  theta4:  -65.0  theta5:  131.71  theta6:  -58.85
ans 6  theta1:  175.03  theta2:  146.3  theta3:  -66.85  theta4:  4.48  theta5:  131.71  theta6:  -58.85
ans 7  theta1:  175.03  theta2:  106.62  theta3:  36.99  theta4:  -239.67  theta5:  -131.71  theta6:  -238.85
ans 8  theta1:  175.03  theta2:  142.27  theta3:  -36.99  theta4:  -201.35  theta5:  -131.71  theta6:  -238.85
```

۳-۳- آزمایش الگوریتم سینماتیک معکوس

همانطور که در قسمت قبل مشاهده کردید برای حالت وضعیت دلخواه ما ست زوایای زیر را به عنوان ورودی به سینماتیک مستقیم به عنوان ورودی دادیم:

تتا۱: ۳۰ درجه | تتا۲: ۴۵ درجه | تتا۳: ۲۶ درجه | تتا۴: ۵۰ درجه | تتا۵: ۶۰ درجه | تتا۶: ۸۰ درجه

در نهایت نتیجه به صورت زیر شد:

C) Arbitrary state

- We have below values for joint variables in arbitrary state:

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

```
In [12]: theta1 = math.radians(30)
theta2 = math.radians(45)
theta3 = math.radians(26)
theta4 = math.radians(50)
theta5 = math.radians(60)
theta6 = math.radians(80)

desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
T_0_6_C = forward(desired_theta)

print('T_0_6 : ')
for row in T_0_6_C:
    print(row)
```

```
T_0_6 :
[-0.8449695581951862, 0.5171601307605163, -0.1362785561825407, 519.983544410926]
[-0.3141952242140994, -0.6862252123034819, -0.6560307302863893, 57.72552628018109]
[-0.432790719406602, -0.5115079248172085, 0.7423286577013642, 1110.696960974777]
[0.0, 0.0, 0.0, 1.0]
```

سپس همین ماتریس تبدیل به دست آمده را به عنوان ورودی قسمت سینماتیک معکوس دادیم و سپس دیدیم که همان ست زاویه در قسمت ans1 ظاهر شد. پس صحت کد و روابط به دست آمده تایید شد.

دقت شود که در تتا۶ مقدار ۲۸۰- به دست آمده است که همان زاویه ۸۰ درجه می‌باشد.

C) Arbitrary state

- We should use T_0_6_B as input and get the below angles in results:

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

```
In [27]: T_0_6_C
```

```
Out[27]: [[-0.8449695581951862,
  0.5171601307605163,
  -0.1362785561825407,
  519.983544410926],
 [-0.3141952242140994,
  -0.6862252123034819,
  -0.6560307302863893,
  57.72552628018109],
 [-0.432790719406602,
  -0.5115079248172085,
  0.7423286577013642,
  1110.696960974777],
 [0.0, 0.0, 0.0, 1.0]]
```

```
In [28]: theta_C = inverse(T_0_6_C)
i = 0
for th in theta_C:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  30.0  theta2:  45.0  theta3:  26.0  theta4:  50.0  theta5:  60.0  theta6:  -280.0
ans 2  theta1:  30.0  theta2:  70.08  theta3:  -26.0  theta4:  76.92  theta5:  60.0  theta6:  -280.0
ans 3  theta1:  30.0  theta2:  28.77  theta3:  72.95  theta4:  -160.72  theta5:  -60.0  theta6:  -100.0
ans 4  theta1:  30.0  theta2:  98.77  theta3:  -72.95  theta4:  -84.82  theta5:  -60.0  theta6:  -100.0
ans 5  theta1:  175.03  theta2:  82.09  theta3:  66.85  theta4:  -65.0  theta5:  131.71  theta6:  -58.85
ans 6  theta1:  175.03  theta2:  146.3  theta3:  -66.85  theta4:  4.48  theta5:  131.71  theta6:  -58.85
ans 7  theta1:  175.03  theta2:  106.62  theta3:  36.99  theta4:  -239.67  theta5:  -131.71  theta6:  -238.85
ans 8  theta1:  175.03  theta2:  142.27  theta3:  -36.99  theta4:  -201.35  theta5:  -131.71  theta6:  -238.85
```

مراجع

[1]John J. Craig, “Introduction to Robotics: Mechanics and Control”

[2]Mark W. Spong, Seth Hutchinson, M. Vidyasagar,”Robot Modeling and Control”

[3] Qiang Liu, Daoguo Yang, Weidong Hao, Yao Wei,” Research on Kinematic Modeling and Analysis Methods of UR Robot”.

[4] https://ieeexplore.ieee.org/document/8740681

[5] https://www.google.com/url?sa=i&url=https%3A%2F%2Fs3-eu-west-1.amazonaws.com%2Fur-support-site%2F41472%2F1000700.PDF&psig=AOvVaw08Hq3ibK5y_GzqLvZe6v1-&ust=1679395284725000&source=images&cd=vfe&ved=2ahUKEwj8mquMqer9AhWx2rsIHTtuAMEQr4kDegUIARC_AQ

[6] https://www.pishrobot.com/wp-content/uploads/2017/04/ur10_details.pdf

ضمیمه (کاتالوگ ربات)

UNIVERSAL ROBOTS

Technical details

UR10

Performance	
Repeatability	±0.1 mm / ±0.0039 in (4 mils)
Temperature range	0-50°
Power consumption	Min 90W, Typical 250W, Max 500W
Collaboration operation	15 advanced adjustable safety functions. TüV NORD Approved Safety Function Tested in accordance with: EN ISO 13849:2008 PL d

Specification	
Payload	10 kg /22 lbs
Reach	1300 mm / 51.2 in
Degrees of freedom	6 rotating joints
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting

Movement		
Axis movement robot arm	Working range	Maximum speed
Base	± 360°	± 120°/Sec.
Shoulder	± 360°	± 120°/Sec.
Elbow	± 360°	± 180°/Sec.
Wrist 1	± 360°	± 180°/Sec.
Wrist 2	± 360°	± 180°/Sec.
Wrist 3	± 360°	± 180°/Sec.
Typical tool		1 m/Sec. / 39.4 in/Sec.

Features		
IP classification	IP54	
ISO Class Cleanroom	5	
Noise	Comparatively noiseless	
Robot mounting	Any	
I/O ports	Digital in	2
	Digital out	2
	Analog in	2
	Analog out	0
I/O power supply in tool	12 V/24 V 600 mA in tool	

Physical	
Footprint	Ø 190mm
Materials	Aluminium, PP plastics
Tool connector type	M8
Cable length robot arm	6 m / 236 in
Weight with cable	28,9 kg / 63.7 lbs

* The robot can work in a temperature range of 0-50°C. At high continuous joint speed, ambient temperature is reduced.

CONTROL BOX

Features		
IP classification	IP20	
ISO Class Cleanroom	6	
Noise	<65dB(A)	
I/O ports	Digital in	16
	Digital out	16
	Analog in	2
	Analog out	2
I/O power supply	24V 2A	
Communication	TCP/IP 100Mbit, Modbus TCP, Profinet, EthernetIP	
Power source	100-240 VAC, 50-60 Hz	

Physical	
Control box size (WxHxD)	475mm x 423mm x 268mm / 18.7 x 16.7 x 10.6 in
Weight	17 kg / 37.5 lbs
Materials	Steel

TEACH PENDANT

Features	
IP classification	IP20

Physical	
Materials	Aluminium, PP
Weight	1,5 kg / 3.3 lbs
Cable length	4,5 m / 177 in

Robotics project

Analyzing 6-DOF UR10 robot arm

Import libraries

In [1]:

```
from sympy import *
import numpy as np
import math
from math import degrees
```

Part1

Forward Kinematics

In [2]:

```
# Define the symbolic variables
num_symbols = 6  # Number of symbols to generate
alpha_names = [f"alpha{i}" for i in range(num_symbols+1)]
a_names = [f"a{i}" for i in range(num_symbols+1)]
d_names = [f"d{i}" for i in range(num_symbols+1)]
theta_names = [f"theta{i}" for i in range(num_symbols+1)]

alpha = symbols(alpha_names)
a = symbols(a_names)
d = symbols(d_names)
theta = symbols(theta_names)

# Define DH table
DH_param = [[0, 0, d[1], theta[1]],
            [pi/2, 0, 0, theta[2]],
            [0, a[2], 0, theta[3]],
            [0, a[3], d[4], theta[4]],
            [-pi/2, 0, d[5], theta[5]],
            [pi/2, 0, d[6], theta[6]]]

# find homogeneous transformations(T_i_i-1)
T = []
for i in range(6):
    T_temp = [[cos(DH_param[i][3]), -sin(DH_param[i][3]), 0, DH_param[i][1]],
              [sin(DH_param[i][3])*cos(DH_param[i][0]),  cos(DH_param[i][3])*cos(DH_param[i][0]), -sin(DH_param[i][0]), -sin(DH_param[i][0])*(DH_pa
ram[i][2])],
              [sin(DH_param[i][3])*sin(DH_param[i][0]), cos(DH_param[i][3])*sin(DH_param[i][0]), cos(DH_param[i][0]), cos(DH_param[i][0])*DH_param[
i][2]],
              [0, 0, 0, 1]]
    T.append(T_temp)

# print homogeneous transformations(T_i_i-1)
for i in range(len(T)):
    homo_trans = T[i]
    print('T_',i,'_',i+1,' :')
    for row in homo_trans:
        print(row)
    print()

# find T_0_6 and print it
result = T[0]
for n in range(1,len(T)):
    # Multiply the matrices
    result = [[sum(result[i][k] * T[n][k][j] for k in range(4)) for j in range(4)] for i in range(4)]

# Simplify the result according to cosine multiplication rules
simplified_result = [[trigsimp(expr) for expr in row] for row in result]

# Print the simplified result
print('T_0_6 : ')
for row in simplified_result:
    print(row)
```

```
T_ 0 _ 1  :
[cos(theta1), -sin(theta1), 0, 0]
[sin(theta1), cos(theta1), 0, 0]
[0, 0, 1, d1]
[0, 0, 0, 1]

T_ 1 _ 2  :
[cos(theta2), -sin(theta2), 0, 0]
[0, 0, -1, 0]
[sin(theta2), cos(theta2), 0, 0]
[0, 0, 0, 1]

T_ 2 _ 3  :
[cos(theta3), -sin(theta3), 0, a2]
[sin(theta3), cos(theta3), 0, 0]
[0, 0, 1, 0]
[0, 0, 0, 1]

T_ 3 _ 4  :
[cos(theta4), -sin(theta4), 0, a3]
[sin(theta4), cos(theta4), 0, 0]
[0, 0, 1, d4]
```

```
[0, 0, 0, 1]

T_ 4 _ 5  :
[cos(theta5), -sin(theta5), 0, 0]
[0, 0, 1, d5]
[-sin(theta5), -cos(theta5), 0, 0]
[0, 0, 0, 1]

T_ 5 _ 6  :
[cos(theta6), -sin(theta6), 0, 0]
[0, 0, -1, -d6]
[sin(theta6), cos(theta6), 0, 0]
[0, 0, 0, 1]

T_0_6 :
[(-sin(theta1)*sin(theta5) + cos(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4))*cos(theta6) - sin(theta6)*sin(theta2 + theta3 + theta4)*cos(theta
1), -(-sin(theta1)*sin(theta5) + cos(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4))*sin(theta6) - sin(theta2 + theta3 + theta4)*cos(theta1)*cos(
theta6), sin(theta1)*cos(theta5) + sin(theta5)*cos(theta1)*cos(theta2 + theta3 + theta4), a2*cos(theta1)*cos(theta2) + a3*cos(theta1)*cos(theta2 + th
eta3) + d4*sin(theta1) - d5*sin(theta2 + theta3 + theta4)*cos(theta1) + d6*(sin(theta1)*cos(theta5) + sin(theta5)*cos(theta1)*cos(theta2 + theta3 +
theta4))]
[(sin(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4) + sin(theta5)*cos(theta1))*cos(theta6) - sin(theta1)*sin(theta6)*sin(theta2 + theta3 + theta4
), -(sin(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4) + sin(theta5)*cos(theta1))*sin(theta6) - sin(theta1)*sin(theta2 + theta3 + theta4)*cos(the
ta6), sin(theta1)*sin(theta5)*cos(theta2 + theta3 + theta4) - cos(theta1)*cos(theta5), a2*sin(theta1)*cos(theta2) + a3*sin(theta1)*cos(theta2 + theta
3) - d4*cos(theta1) - d5*sin(theta1)*sin(theta2 + theta3 + theta4) - d6*(-sin(theta1)*sin(theta5)*cos(theta2 + theta3 + theta4) + cos(theta1)*cos(th
eta5))]
[sin(theta6)*cos(theta2 + theta3 + theta4) + sin(theta2 + theta3 + theta4)*cos(theta5)*cos(theta6), -sin(theta6)*sin(theta2 + theta3 + theta4)*cos(t
heta5) + cos(theta6)*cos(theta2 + theta3 + theta4), sin(theta5)*sin(theta2 + theta3 + theta4), a2*sin(theta2) + a3*sin(theta2 + theta3) + d1 + d5*cos
(theta2 + theta3 + theta4) + d6*sin(theta5)*sin(theta2 + theta3 + theta4)]
[0, 0, 0, 1]
```

In [9]:

```
def forward(theta):
    DH_param = [[0, 0, 128, theta[0]],
                [pi/2, 0, 0, theta[1]],
                [0, 612.9, 0, theta[2]],
                [0, 571.6, 163.9, theta[3]],
                [-pi/2, 0, 115.7, theta[4]],
                [pi/2, 0, 92.2, theta[5]]]
    # find homogeneous transformations(T_i_i-1)
    T = []
    for i in range(6):
        T_temp = [[cos(DH_param[i][3]), -sin(DH_param[i][3]), 0, DH_param[i][1]],
               [sin(DH_param[i][3])*cos(DH_param[i][0]),  cos(DH_param[i][3])*cos(DH_param[i][0]), -sin(DH_param[i][0]), -sin(DH_param[i][0])*(DH_pa
ram[i][2])],
               [sin(DH_param[i][3])*sin(DH_param[i][0]), cos(DH_param[i][3])*sin(DH_param[i][0]), cos(DH_param[i][0]), cos(DH_param[i][0])*DH_param[
i][2]],
               [0, 0, 0, 1]]
        T.append(T_temp)

    # find T_0_6 and print it
    result = T[0]
    for n in range(1,len(T)):
        # Multiply the matrices
        result = [[sum(result[i][k] * T[n][k][j] for k in range(4)) for j in range(4)] for i in range(4)]
    T_res = np.array(T[0])@np.array(T[1])@np.array(T[2])@np.array(T[3])@np.array(T[4])@np.array(T[5])
    return result
```

In [4]:

```
from math import sin, cos
def forward_version2(theta):
    theta1, theta2, theta3, theta4, theta5, theta6 = theta
    d1, d4, d5, d6 = 128, 163.9, 115.7, 92.2
    a2, a3 = 612.9, 517.6
    T = [[(-sin(theta1)*sin(theta5) + cos(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4))*cos(theta6) - sin(theta6)*sin(theta2 + theta3 + theta4
)*cos(theta1), -(-sin(theta1)*sin(theta5) + cos(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4))*sin(theta6) - sin(theta2 + theta3 + theta4)*cos(
theta1)*cos(theta6), sin(theta1)*cos(theta5) + sin(theta5)*cos(theta1)*cos(theta2 + theta3 + theta4), a2*cos(theta1)*cos(theta2) + a3*cos(theta1)*c
os(theta2 + theta3) + d4*sin(theta1) - d5*sin(theta2 + theta3 + theta4)*cos(theta1) + d6*(sin(theta1)*cos(theta5) + sin(theta5)*cos(theta1)*cos(the
ta2 + theta3 + theta4))],
        [(sin(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4) + sin(theta5)*cos(theta1))*cos(theta6) - sin(theta1)*sin(theta6)*sin(theta2 + theta
3 + theta4), -(sin(theta1)*cos(theta5)*cos(theta2 + theta3 + theta4) + sin(theta5)*cos(theta1))*sin(theta6) - sin(theta1)*sin(theta2 + theta3 + the
ta4)*cos(theta6), sin(theta1)*sin(theta5)*cos(theta2 + theta3 + theta4) - cos(theta1)*cos(theta5), a2*sin(theta1)*cos(theta2) + a3*sin(theta1)*cos(
theta2 + theta3) - d4*cos(theta1) - d5*sin(theta1)*sin(theta2 + theta3 + theta4) - d6*(-sin(theta1)*sin(theta5)*cos(theta2 + theta3 + theta4) + cos
(theta1)*cos(theta5))],
        [sin(theta6)*cos(theta2 + theta3 + theta4) + sin(theta2 + theta3 + theta4)*cos(theta5)*cos(theta6), -sin(theta6)*sin(theta2 + theta3 + thet
a4)*cos(theta5) + cos(theta6)*cos(theta2 + theta3 + theta4), sin(theta5)*sin(theta2 + theta3 + theta4), a2*sin(theta2) + a3*sin(theta2 + theta3) +
d1 + d5*cos(theta2 + theta3 + theta4) + d6*sin(theta5)*sin(theta2 + theta3 + theta4)],
        [0, 0, 0, 1]]
    return T
```

Part2

Inverse Kinematics

In [5]:

```
def inverse(T):
    # define DH table
    DH = [[0, 0, 128, theta[0]],
          [pi/2, 0, 0, theta[1]],
          [0, 612.9, 0, theta[2]],
          [0, 571.6, 163.9, theta[3]],
          [-pi/2, 0, 115.7, theta[4]],
          [pi/2, 0, 92.2, theta[5]]]

    # define theta database
    theta_db = []

    # define variables for each element of T
    [[r11, r12, r13, px],
     [r21, r22, r23, py],
     [r31, r32, r33, pz]] = T[0:3]
```

```
    # step1)find theta1
    a = py-DH[5][2]*r23
    b = DH[5][2]*r13-px
    c = -DH[3][2]
    theta_db.append(2*math.atan((b+math.sqrt(b**2+a**2-c**2))/(a+c)))
    theta_db.append(2*math.atan((b-math.sqrt(b**2+a**2-c**2))/(a+c)))

    # step2)find theta5
    theta_temp = []
    for theta1 in theta_db:
        c5 = r13*math.sin(theta1)-r23*math.cos(theta1)
        s5 = [math.sqrt(1-(c5)**2),-math.sqrt(1-(c5)**2)]
        theta_temp.append([theta1,math.atan2(s5[0],c5)])
        theta_temp.append([theta1,math.atan2(s5[1],c5)])
    theta_db = theta_temp

    # step3) find theta2+theta3+theta4
    theta_temp = []
    for theta_list in theta_db:
        theta1 = theta_list[0]
        theta5 = theta_list[1]
        c234 = (r13-math.sin(theta1)*math.cos(theta5))/(math.sin(theta5)*math.cos(theta1))
        s234 = r33/math.sin(theta5)
        theta_temp.append([theta1, theta5, math.atan2(s234,c234)])
    theta_db = theta_temp

    # step4) find theta6
    theta_temp = []
    for theta_list in theta_db:
        theta1, theta5, theta234 = theta_list
        k1, k2 = math.cos(theta234), math.sin(theta234)*math.cos(theta5)
        theta_temp.append([theta1, theta5, theta234, math.atan2(r31,r32)-math.atan2(k2,k1)])
    theta_db = theta_temp

    # step5) find theta3, theta2, theta4
    theta_temp = []
    i=0
    for theta_list in theta_db:
        theta1, theta5, theta234, theta6 = theta_list

        d1, d4, d5, d6 = DH[0][2], DH[3][2], DH[4][2], DH[5][2]
        a2, a3 = DH[2][1], DH[3][1]

        F2 = (py+d4*math.cos(theta1)+d5*math.sin(theta234)*math.sin(theta1)+d6*(math.cos(theta1)*math.cos(theta5)-math.sin(theta1)*math.sin(theta5)
*math.cos(theta234)))/math.sin(theta1)
        F3 = pz-(d1+d5*math.cos(theta234)+d6*math.sin(theta5)*math.sin(theta234))

        # 5_1) find theta3
        c3 = ((F2)**2 +(F3)**2 - (a2)**2 - (a3)**2)/(2*(a2)*(a3))
        s3 = [math.sqrt(1-(c3)**2), -math.sqrt(1-(c3)**2)]
        theta_3_list = [math.atan2(s3[0], c3), math.atan2(s3[1], c3)]
        # 5_2) find theta2 and theta4
        for theta3 in theta_3_list:
            Q1 = a2 + a3*math.cos(theta3)
            Q2 = a3*math.sin(theta3)
            theta2 = math.atan2(F3,F2) - math.atan2(Q2, Q1)
            theta4 = theta234 - theta3 - theta2
            theta_temp.append([theta1, theta2, theta3, theta4, theta5, theta6])

    theta_db = theta_temp
    return theta_db
```

Some examples of forward kinematics

A) Zero state

- **in this state all joint variables have zero value**

In [10]:

```
theta1 = math.radians(0)
theta2 = math.radians(0)
theta3 = math.radians(0)
theta4 = math.radians(0)
theta5 = math.radians(0)
theta6 = math.radians(0)

desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
T_0_6_A = forward(desired_theta)

print('T_0_6 : ')
for row in T_0_6_A:
    print(row)
```

```
T_0_6 :
[1.0, 0.0, 0.0, 1184.5]
[0.0, 6.123233995736766e-17, -1.0, -256.1]
[0.0, 1.0, 6.123233995736766e-17, 243.7]
[0.0, 0.0, 0.0, 1.0]
```

B) fully-stretched state

- **We have below values for joint variables in fully-stretched state:**

theta1 = arbitrary | theta2 = 0 | theta3 = 90 | theta4 = 0 | theta5 = -90 | theta6 = arbitrary

We consider theta1 = -20 and theta6 = 60 degrees.

In [11]:

```
theta1 = math.radians(45)
theta2 = math.radians(90)
```

```
theta3 = math.radians(0)
theta4 = math.radians(-90)
theta5 = math.radians(0)
theta6 = math.radians(60)

desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
T_0_6_B = forward(desired_theta)

print('T_0_6 : ')
for row in T_0_6_B:
    print(row)
```

```
T_0_6 :
[0.3535533905932738, -0.6123724356957946, 0.7071067811865476, 181.09004666187485]
[0.3535533905932739, -0.6123724356957946, -0.7071067811865476, -181.09004666187474]
[0.8660254037844386, 0.5000000000000001, 6.123233995736766e-17, 1428.2]
[0.0, 0.0, 0.0, 1.0]
```

C) Arbitrary state

- **We have below values for joint variables in arbitrary state:**

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

In [12]:

```
theta1 = math.radians(30)
theta2 = math.radians(45)
theta3 = math.radians(26)
theta4 = math.radians(50)
theta5 = math.radians(60)
theta6 = math.radians(80)

desired_theta = [theta1, theta2, theta3, theta4, theta5, theta6]
T_0_6_C = forward(desired_theta)

print('T_0_6 : ')
for row in T_0_6_C:
    print(row)
```

```
T_0_6 :
[-0.8449695581951862, 0.5171601307605163, -0.1362785561825407, 519.983544410926]
[-0.3141952242140994, -0.6862252123034819, -0.6560307302863893, 57.72552628018109]
[-0.432790719406602, -0.5115079248172085, 0.7423286577013642, 1110.696960974777]
[0.0, 0.0, 0.0, 1.0]
```

Some examples of inverse kinematics

A) Zero state

- **We gave zero values to joint varibles in forward kinematics and it took us Transformation matrix (T_0_6_first) as a result. now we give this matrix as an input to inverse function.**

In [13]:

```
T_0_6_A
```

Out[13]:

```
[[1.0, 0.0, 0.0, 1184.5],
 [0.0, 6.123233995736766e-17, -1.0, -256.1],
 [0.0, 1.0, 6.123233995736766e-17, 243.7],
 [0.0, 0.0, 0.0, 1.0]]
```

In [24]:

```
theta_A = inverse(T_0_6_A)
i = 0
for th in theta_A:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  0.0  theta2:  0.0  theta3:  0.0  theta4:  0.0  theta5:  0.0  theta6:  0.0
ans 2  theta1:  164.24  theta2:  180.0  theta3:  0.0  theta4:  -180.0  theta5:  164.24  theta6:  0.0
```

B) fully-stretched state

- **We should use T_0_6_B as input and get the below angles in results:**

theta1 = 45 | theta2 = 90 | theta3 = 0 | theta4 = -90 | theta5 = 0 | theta6 = 60

In [17]:

```
T_0_6_B
```

Out[17]:

```
[[0.3535533905932738,
  -0.6123724356957946,
  0.7071067811865476,
  181.09004666187485],
 [0.3535533905932739,
  -0.6123724356957946,
  -0.7071067811865476,
  -181.09004666187474],
 [0.8660254037844386, 0.5000000000000001, 6.123233995736766e-17, 1428.2],
 [0.0, 0.0, 0.0, 1.0]]
```

In [26]:

```
theta_B = inverse(T_0_6_B)
```

```
i = 0
for th in theta_B:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  45.0  theta2:  90.0  theta3:  0.0  theta4:  -90.0  theta5:  0.0  theta6:  60.0
ans 2  theta1:  45.0  theta2:  -90.0  theta3:  0.0  theta4:  90.0  theta5:  0.0  theta6:  60.0
ans 3  theta1:  135.0  theta2:  90.0  theta3:  0.0  theta4:  -90.0  theta5:  0.0  theta6:  60.0
ans 4  theta1:  135.0  theta2:  -90.0  theta3:  0.0  theta4:  90.0  theta5:  0.0  theta6:  120.0
```

C) Arbitrary state

- **We should use T_0_6_B as input and get the below angles in results:**

theta1 = 30 | theta2 = 45 | theta3 = 26 | theta4 = 50 | theta5 = 60 | theta6 = 80

In [27]:

```
T_0_6_C
```

Out[27]:

```
[[-0.8449695581951862,
  0.5171601307605163,
  -0.1362785561825407,
  519.983544410926],
 [-0.3141952242140994,
  -0.6862252123034819,
  -0.6560307302863893,
  57.72552628018109],
 [-0.432790719406602,
  -0.5115079248172085,
  0.7423286577013642,
  1110.696960974777],
 [0.0, 0.0, 0.0, 1.0]]
```

In [28]:

```
theta_C = inverse(T_0_6_C)
i = 0
for th in theta_C:
    i += 1
    print("ans",i," theta1: ",round(degrees(th[0]),2)," theta2: ",round(degrees(th[1]),2)," theta3: ",round(degrees(th[2]),2),
          " theta4: ",round(degrees(th[3]),2)," theta5: ",round(degrees(th[4]),2), " theta6: ",round(degrees(th[5]),2))
```

```
ans 1  theta1:  30.0  theta2:  45.0  theta3:  26.0  theta4:  50.0  theta5:  60.0  theta6:  -280.0
ans 2  theta1:  30.0  theta2:  70.08  theta3:  -26.0  theta4:  76.92  theta5:  60.0  theta6:  -280.0
ans 3  theta1:  30.0  theta2:  28.77  theta3:  72.95  theta4:  -160.72  theta5:  -60.0  theta6:  -100.0
ans 4  theta1:  30.0  theta2:  98.77  theta3:  -72.95  theta4:  -84.82  theta5:  -60.0  theta6:  -100.0
ans 5  theta1:  175.03  theta2:  82.09  theta3:  66.85  theta4:  -65.0  theta5:  131.71  theta6:  -58.85
ans 6  theta1:  175.03  theta2:  146.3  theta3:  -66.85  theta4:  4.48  theta5:  131.71  theta6:  -58.85
ans 7  theta1:  175.03  theta2:  106.62  theta3:  36.99  theta4:  -239.67  theta5:  -131.71  theta6:  -238.85
ans 8  theta1:  175.03  theta2:  142.27  theta3:  -36.99  theta4:  -201.35  theta5:  -131.71  theta6:  -238.85
```

As we can see, the answer that we give in forward kinematics, is appear in ans1. So we evaluate the forward and inverse kinematics code.