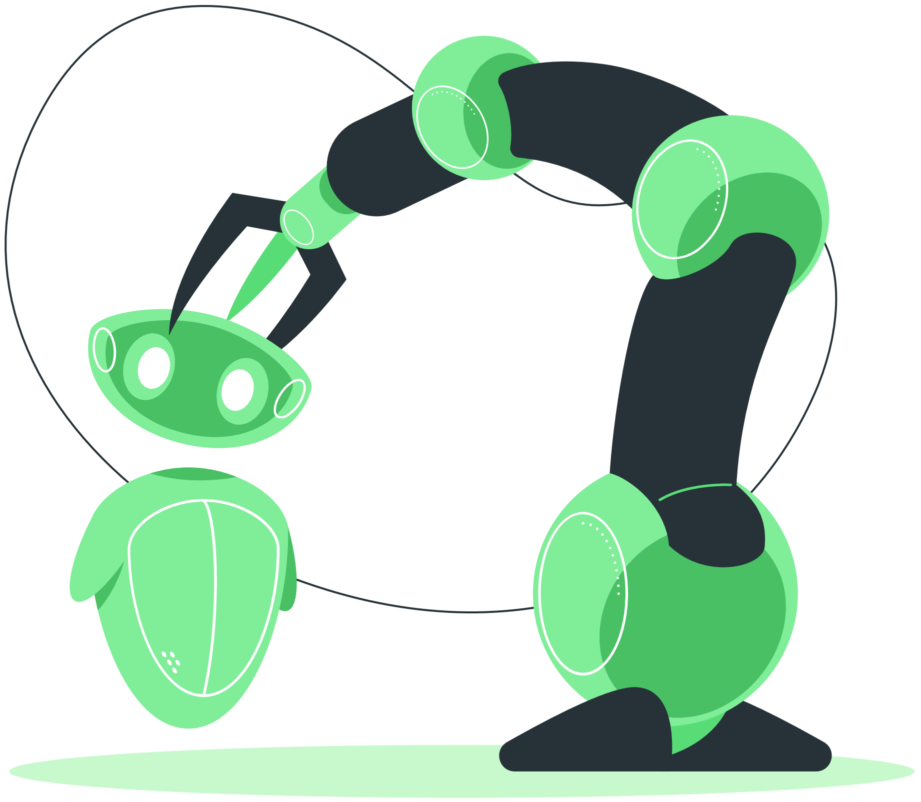
Stanford Manipulator

Trajectory Planning using Genetic Algorithm

22AIE214
Introduction to AI Robotics
Batch B Group 2



The Team

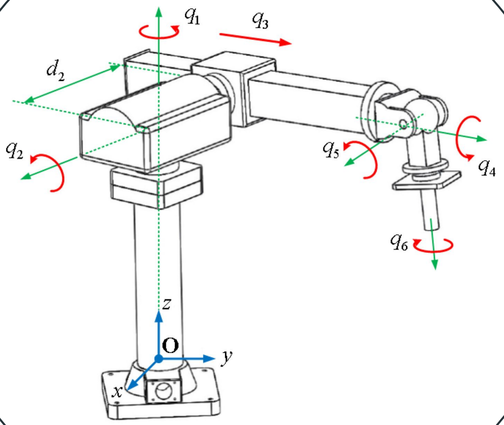
01 **A Manasha**
CB.EN.U4AIE22101

26 **KL Amritha Nandini**
CB.EN.U4AIE22126

51 **Y Sudheer Kumar Chowdary**
CB.EN.U4AIE22151

61 **Burugu Rahul**
CB.EN.U4AIE22161

Manipulator Selected



Stanford Manipulator

The Stanford arm is an industrial robot with **six degrees of freedom**, designed at Stanford University by Victor Scheinman in 1969. It is a serial manipulator whose kinematic chain consists of **5 revolute joints and 1 prismatic joint.**

Because it includes several kinematic pairs, it is often used as an educational example in robot kinematics.

Uses of the Stanford Manipulator:

It has been used in a variety of applications, including:

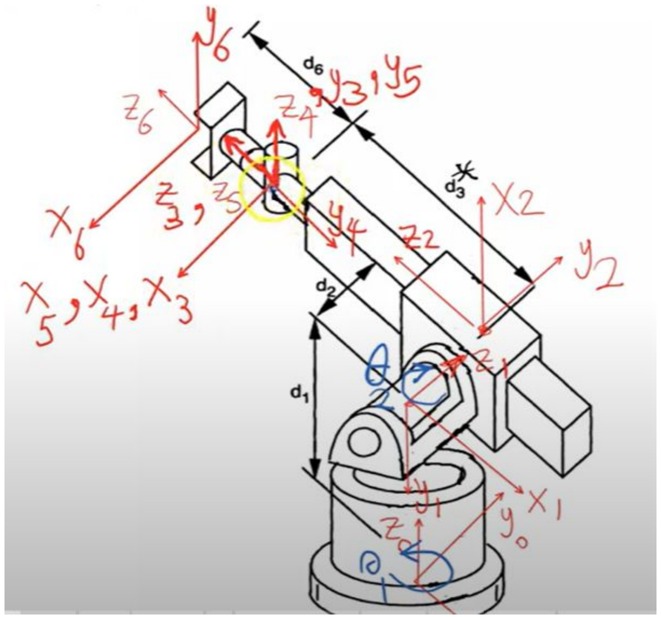
- Industrial Automation: The manipulator can be used for tasks such as welding, painting, and assembly. source
- Research and Development: The manipulator is a valuable tool for researchers studying robotics, control theory, and artificial intelligence. source
- Education: The manipulator is used in robotics courses to teach students about robot kinematics, dynamics, and control. source

Components of the Stanford Manipulator:

The manipulator consists of the following components:

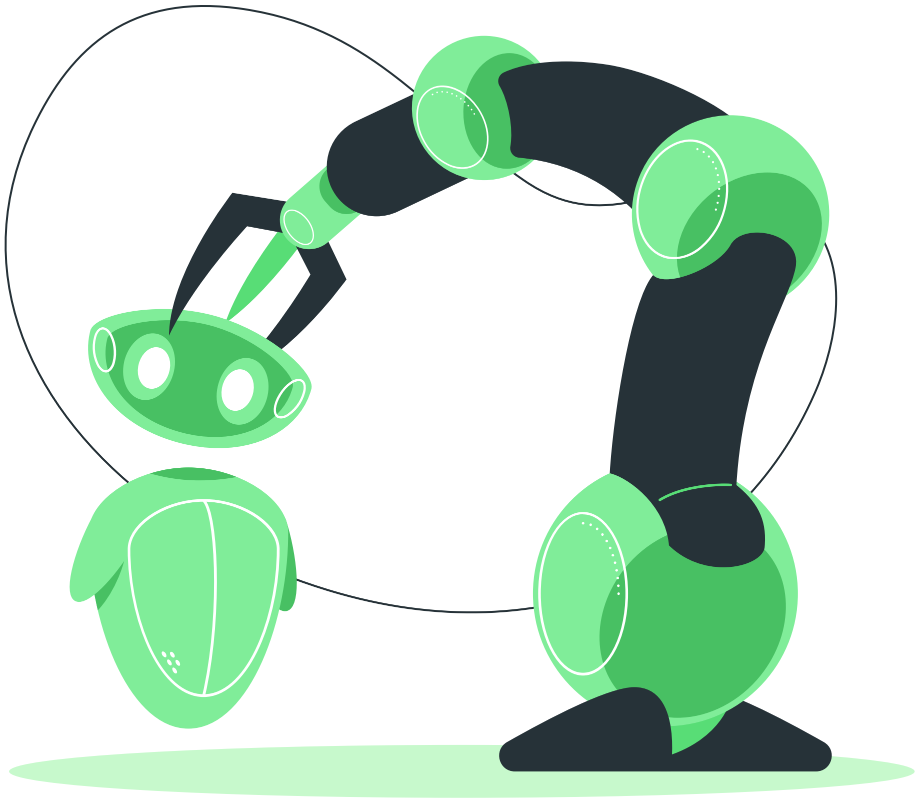
- Rotation Body: Provides the base rotational motion.
- Large Arm: Acts as the main structural component for vertical movement.
- Extension Arm: Facilitates horizontal extension and retraction.
- Wrist: Allows complex orientation adjustments.
- End Effector: Performs the manipulative tasks, such as gripping.

Standard DH Parameter



	ai	αi	di	θi
1	0	-90°	d1	Θ1*
2	0	90°	d2	Θ2*
3	0	0	d3*	0
4	0	-90°	0	Θ4*
5	0	90°	0	Θ5*
6	0	0	0	Θ6*

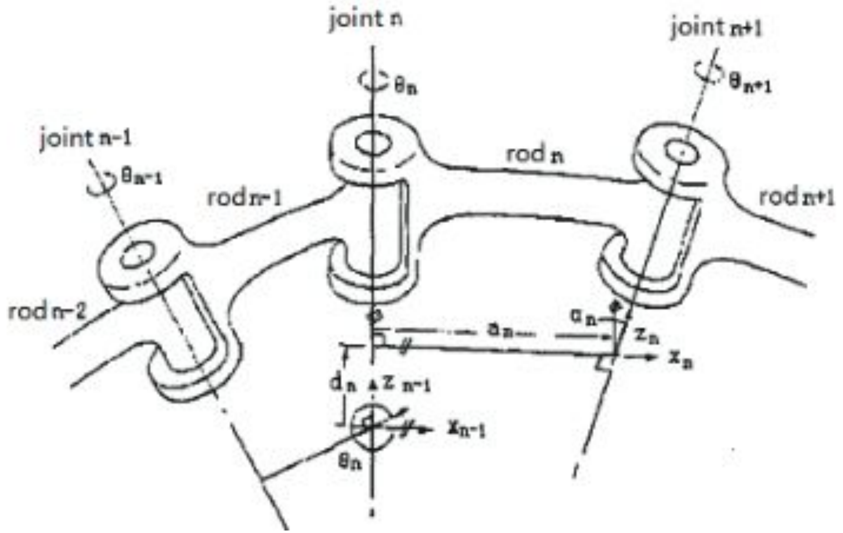
Analytical forward & inverse kinematics



Computing Transformation matrix

Homogeneous transformation matrices are used to represent the position and orientation of a robot manipulator's link in a 3D space.

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- the xn-1 axis is rotated by ***θ***n angle around the zn-1 axis.
- the xn-1 axis moves dn distance along the zn-1 axis.
- the zn-1 axis moves an distance along the rotating xn-1
- axis, that is, xn axis. Finally, the zn-1 axis is twisted by αn angle around the xn axis.

Computing Transformation matrix

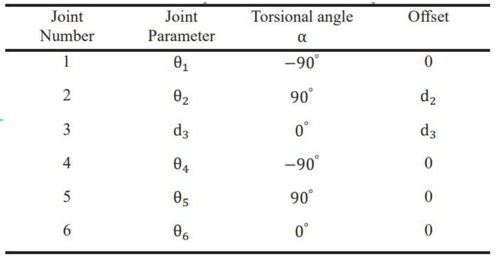
$$A_n = \begin{bmatrix} cos\theta_n & -sin\theta_n & 0 & 0 \\ sin\theta_n & cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_n & -sin\alpha_n & 0 \\ 0 & sin\alpha_n & cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} cos\theta_n & -sin\theta_n & sin\theta_n sin\alpha_n & a_n cos\theta_n \\ sin\theta_n & cos\theta_n & -cos\theta_n sin\alpha_n & a_n sin\theta_n \\ 0 & sin\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Computing Transformation matrix

$$A_n = \begin{bmatrix} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta_n & -\sin\theta_n & \sin\theta_n \sin\alpha_n & a_n \cos\theta_n \\ \sin\theta_n & \cos\theta_n & -\cos\theta_n \sin\alpha_n & a_n \sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



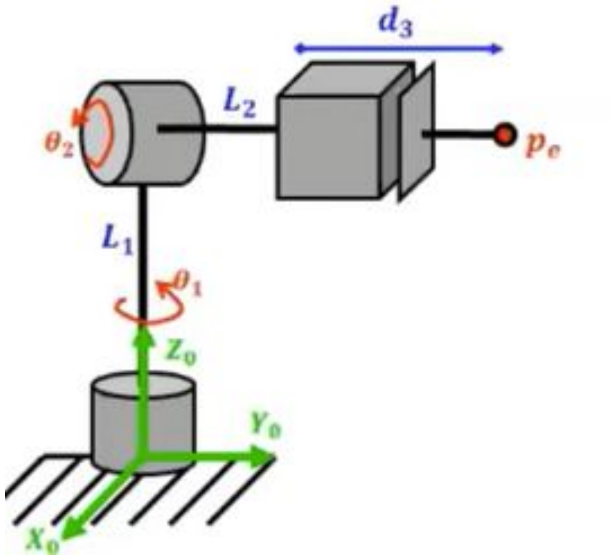
Joint Number	Joint Parameter	Torsional angle α	Offset
1	θ_1	-90°	0
2	θ_2	90°	d_2
3	d_3	0°	d_3
4	θ_4	-90°	0
5	θ_5	90°	0
6	θ_6	0°	0

Forward Kinematics Calculation

Reference coordinate system is $\longrightarrow$ transformed to the **base** coordinate system

$$\mathbf{T6 = A1.\ A2.\ A3.\ A4.\ A5.\ A6} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Kinematics Calculation



End effector position - px, py, pz only depends on *θ*1, *θ*2, d3

$$\theta_1 = tan^{-1}\left(\frac{p_x}{p_y}\right) - tan^{-1}\left(\frac{d_2}{\sqrt{p_x^2 + p_y^2 - d_2^2}}\right)$$

$$\theta_2 = \tan^{-1}\frac{c_1 p_x + s_1 p_y}{p_z}$$

$$d_3 = s_2\left(c_1 p_x + s_1 p_y\right) + c_2 p_z$$

Motion Trajectory Planning

- Motion trajectory planning is the process of determining a path or sequence of movements for a robotic manipulator or any other moving system to follow over time.
- Starting time t=t0 and ending at t=tf

$$\begin{tikzcd}
\Theta 1, \Theta 2, d3 \arrow[r] & px, py, pz \arrow[r] & \text{Trajectory}
\end{tikzcd}$$

Manipulator Joints

Position of end effector

- In trajectory planning, we solve the initial values and final values of joint parameters by inverse kinematics according to the given corresponding initial position px0, py0, pz0 and end position pxf, pyf, pzf.

Motion Trajectory Planning

- Each joint parameter will be performed by **polynomial interpolation** operation to get smooth function θ(t).

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Trajectory

Should be continuous

$$\dot{\theta}(0) = \dot{\theta}(t_f) = 0$$

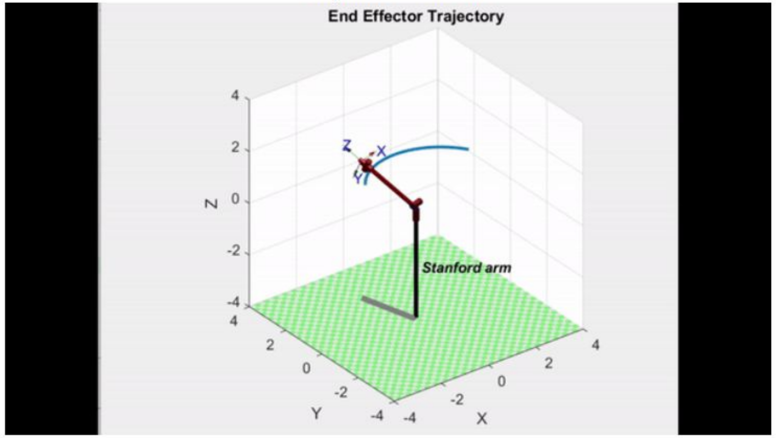
$$\begin{cases} a_0 = \theta_0 \\ a_1 = 0 \\ a_2 = \dfrac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 = -\dfrac{2}{t_f^3}(\theta_f - \theta_0) \end{cases}$$

Unnecessary to make all parameters of joints start changing at the same time.

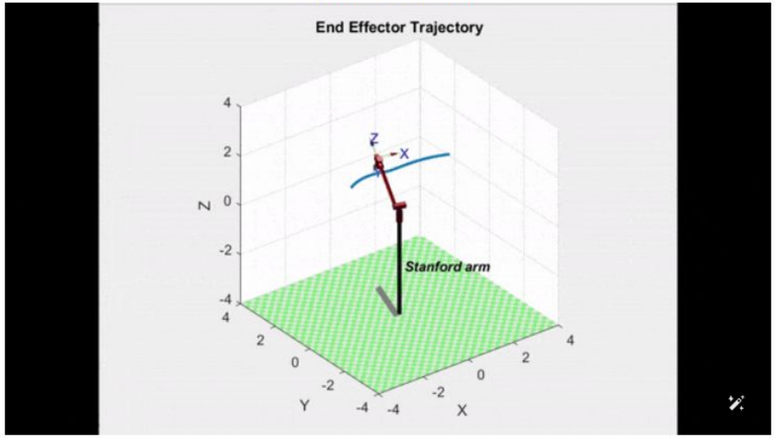
Objective

- There are many kinds of orders sequences between different parameter variations. With a result the trajectory of end effector will be also different.
- Unnecessary to make all parameters of joints start changing at the same time.
- In unoptimized trajectory, all the joints move at the same time t=t0 and end at t=tf.

Unoptimized Path:



Optimized Path:



Objective

When $0 < \mathrm{t} < \mathrm{t}_2$,

$$\begin{cases} a_{10} = \theta_{10} \\ a_{11} = 0 \\ a_{12} = \dfrac{3}{t_f^2}(\theta_{1f} - \theta_{10}) \\ a_{13} = -\dfrac{2}{t_f^3}(\theta_{1f} - \theta_{10}) \end{cases}$$

$$\theta_1(\mathrm{t}) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3$$

$$\begin{cases} a_{20} = \dfrac{\theta_{20}t_f^3 - 3\theta_{20}t_f^2 t_2 + 3\theta_{2f}t_f t_2^2 - \theta_{2f}}{(t_f - t_2)^3} \\ a_{21} = \dfrac{6t_2 t_f(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \\ a_{22} = -\dfrac{3(t_2 + t_f)(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \\ a_{23} = \dfrac{2(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \end{cases}$$

$$\theta_2(\mathrm{t}) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3$$

$$\begin{cases} a_{30} = \dfrac{d_{30}t_f^3 - 3d_{30}t_f^2 t_3 + 3d_{3f}t_f t_3^2 - d_{3f}t_3^3}{(t_f - t_3)^3} \\ a_{31} = \dfrac{6t_3 t_f(d_{30} - d_{3f})}{(t_f - t_3)^3} \\ a_{32} = -\dfrac{3(t_3 + t_f)(d_{30} - d_{3f})}{(t_f - t_3)^3} \\ a_{33} = \dfrac{2(d_{30} - d_{3f})}{(t_f - t_3)^3} \end{cases}$$

$$d_3(\mathrm{t}) = a_{30} + a_{31}t + a_{32}t^2 + a_{33}t^3$$

$$L_1 = \int_{L_1} \vec{p}_1 ds = \int_0^{t_2} \sqrt{\dot{p}_{x1}^2(t) + \dot{p}_{y1}^2(t) + \dot{p}_{z1}^2(t)} dt$$

$$L_2 = \int_{L_2} \vec{p}_2 ds = \int_{t_2}^{t_3} \sqrt{\dot{p}_{x2}^2(t) + \dot{p}_{y2}^2(t) + \dot{p}_{z2}^2(t)} dt$$

$$L_3 = \int_{L_3} \vec{p}_3 ds = \int_{t_3}^{t_f} \sqrt{\dot{p}_{x3}^2(t) + \dot{p}_{y3}^2(t) + \dot{p}_{z3}^2(t)} dt$$

Objective

- The trajectory optimization objective is to **t2 and t3** to **minimize the movement distance of end effector** when the initial position px0, py0, pz0 and end position pxf, pyf, pzf is given.

$$L_1 = \int_{L_1} \vec{p}_1 ds = \int_0^{t_2} \sqrt{\dot{p}_{x1}^2(t) + \dot{p}_{y1}^2(t) + \dot{p}_{z1}^2(t)} dt$$

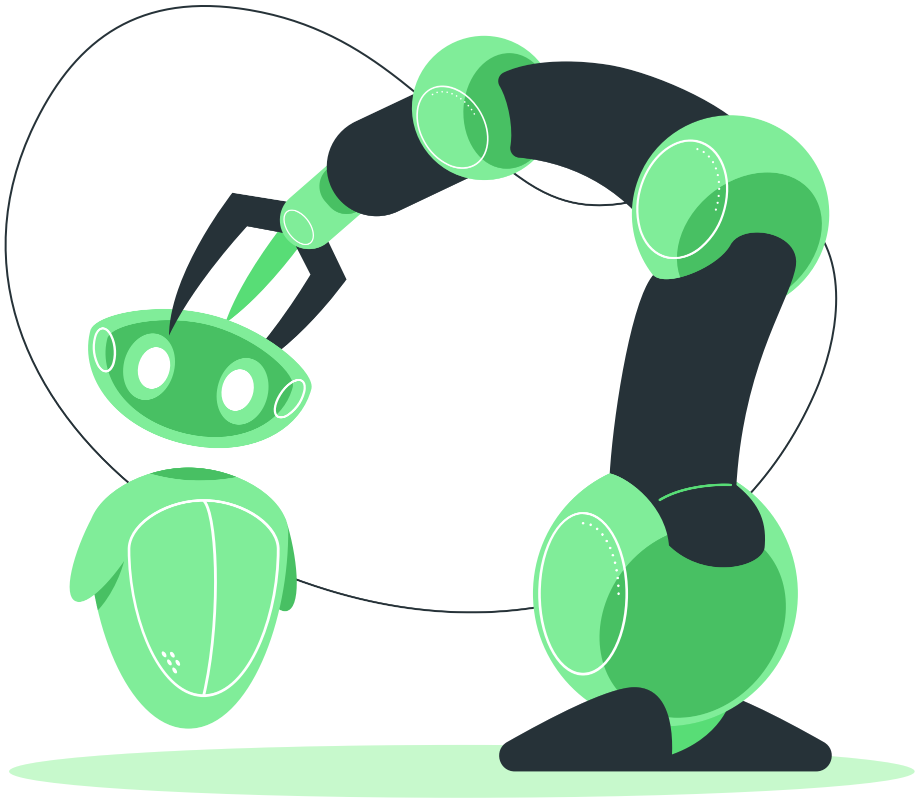
$$L_2 = \int_{L_2} \vec{p}_2 ds = \int_{t_2}^{t_3} \sqrt{\dot{p}_{x2}^2(t) + \dot{p}_{y2}^2(t) + \dot{p}_{z2}^2(t)} dt$$

$$L_3 = \int_{L_3} \vec{p}_3 ds = \int_{t_3}^{t_f} \sqrt{\dot{p}_{x3}^2(t) + \dot{p}_{y3}^2(t) + \dot{p}_{z3}^2(t)} dt$$

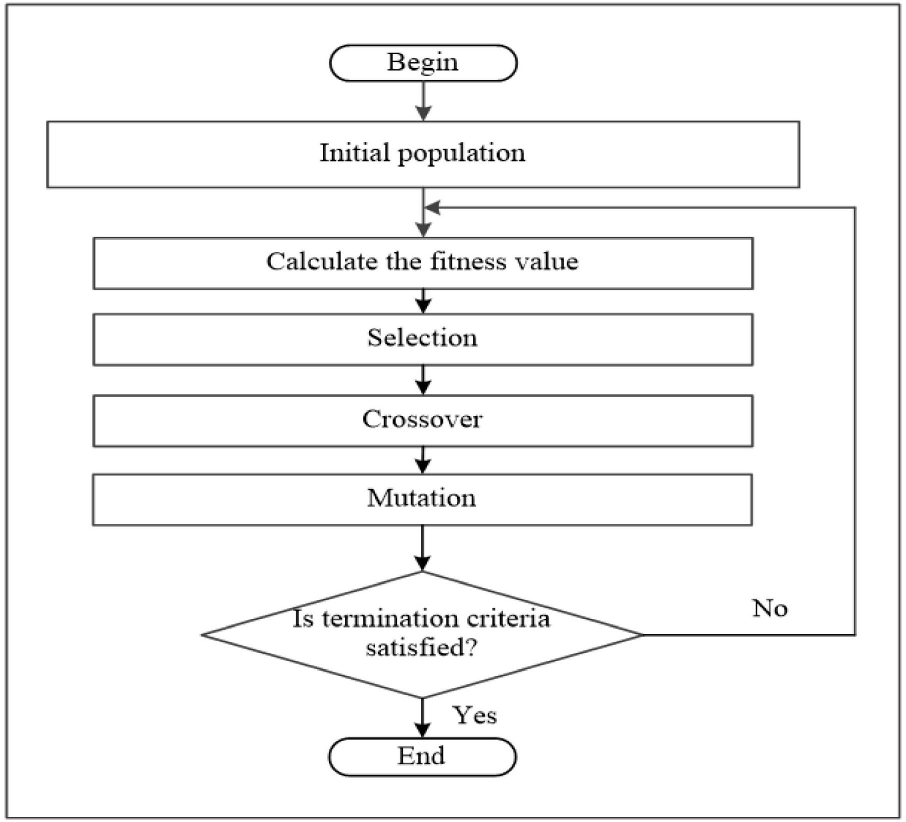
$$\mathrm{L} = \mathrm{L}_1 + \mathrm{L}_2 + \mathrm{L}_3$$

Finding the optimal t2 and t3 is given by **Genetic Algorithm**

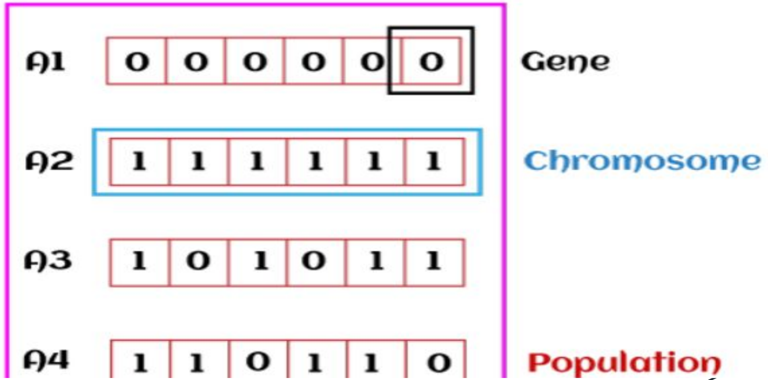
Genetic Algorithm



Flow Chart


Begin
Initial population
Calculate the fitness value
Selection
Crossover
Mutation
Is termination criteria satisfied?
No
Yes
End

- **Population:** Population is the subset of all possible or probable solutions, which can solve the given problem.
- **Chromosomes:** A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome. (In this problem each chromosome is t2,t3 starting time of joint parameters theta2 and d3)
- **Gene:** A chromosome is divided into a different gene, or it is an element of the chromosome.



Fitness

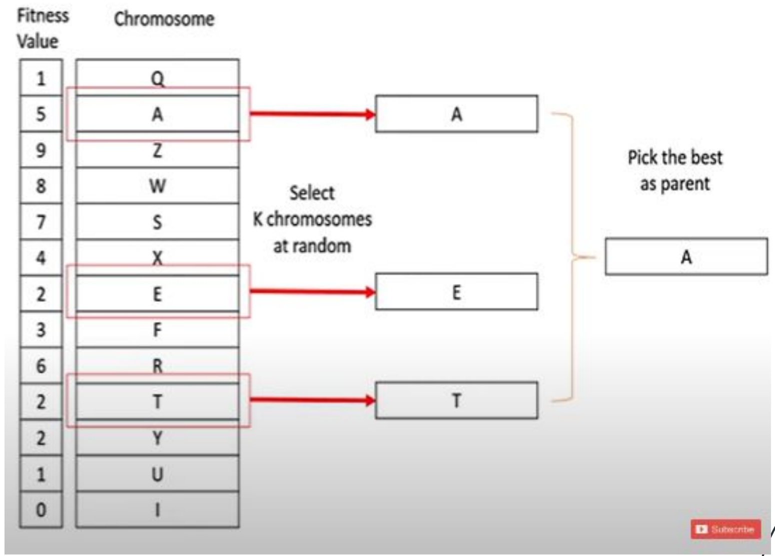
- Fitness function is used to determine how fit an individual is? It means the ability of an individual to compete with other individuals.
- In every iteration, individuals are evaluated based on their fitness function.
- The fitness function provides a fitness score to each individual. This score further determines the probability of being selected for reproduction.
- The distance will serves as the fitness score for the individual.

Selection

- The selection phase involves the selection of individuals for the reproduction of offspring. All the selected individuals are then arranged in a pair of two to increase reproduction.
- Then these individuals transfer their genes to the next generation.
- We use k-Tournament selection for selecting the individuals.

K-Tournament

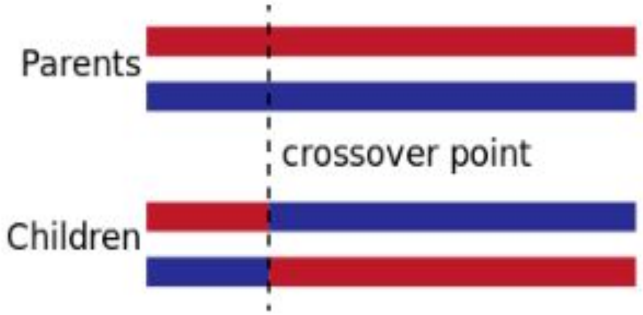
- Select k individuals from the population and perform a tournament amongst them
- Select the best individual from the k individuals
- Repeat process 1 and 2 until you have the desired amount of population



Crossover

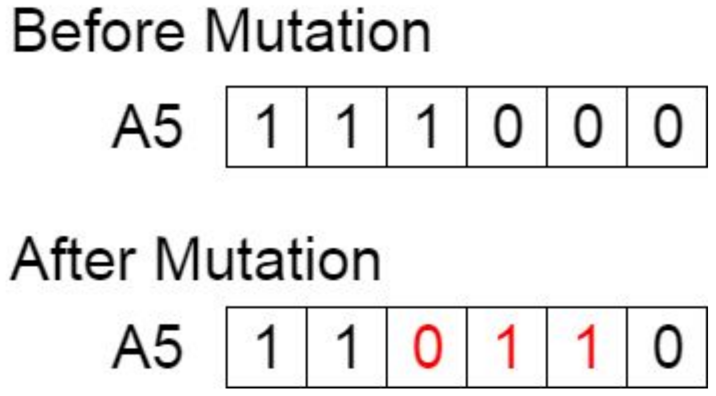
In this process, a crossover point is selected at random within the genes.

Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing the offspring.

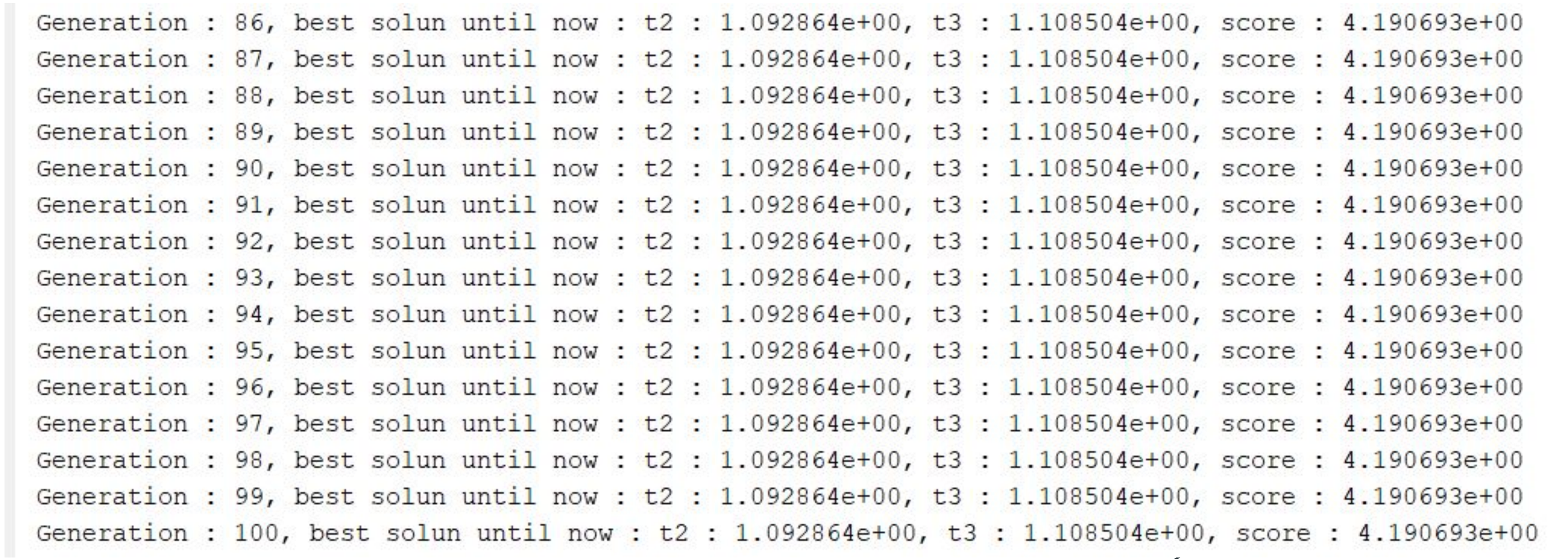


Mutation

- The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population.
- It can be done by flipping some bits in the chromosomes.

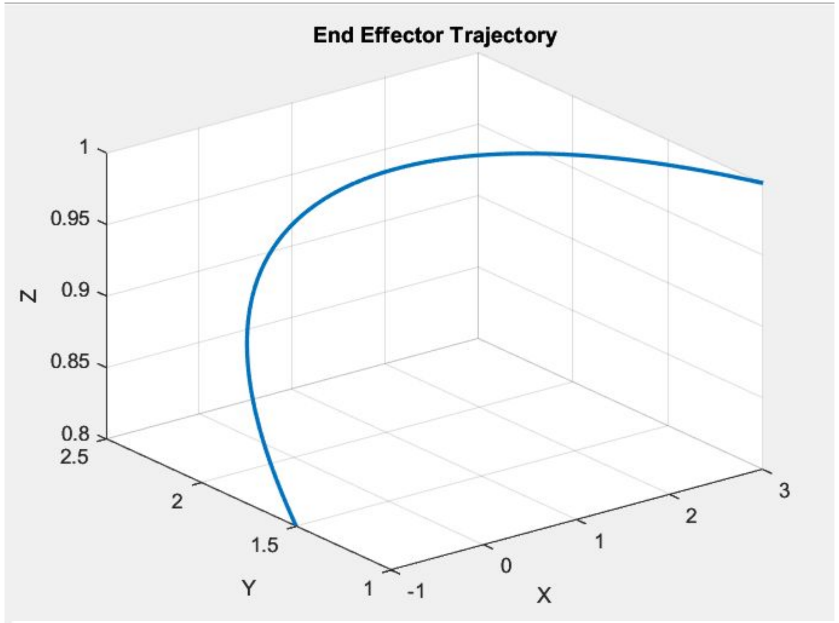


Results

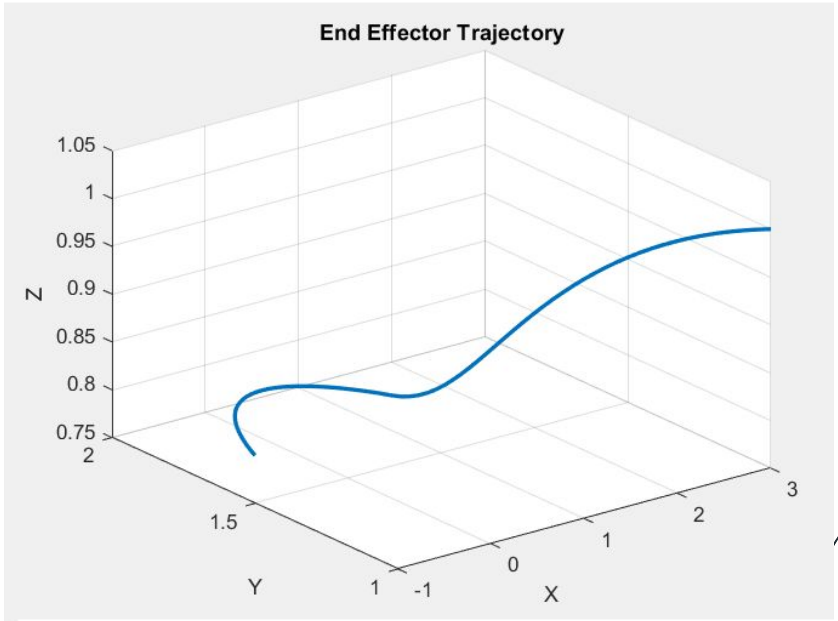


```
Generation : 86, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 87, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 88, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 89, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 90, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 91, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 92, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 93, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 94, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 95, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 96, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 97, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 98, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 99, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
Generation : 100, best solun until now : t2 : 1.092864e+00, t3 : 1.108504e+00, score : 4.190693e+00
```

Comparison b/w unOptimized and optimized

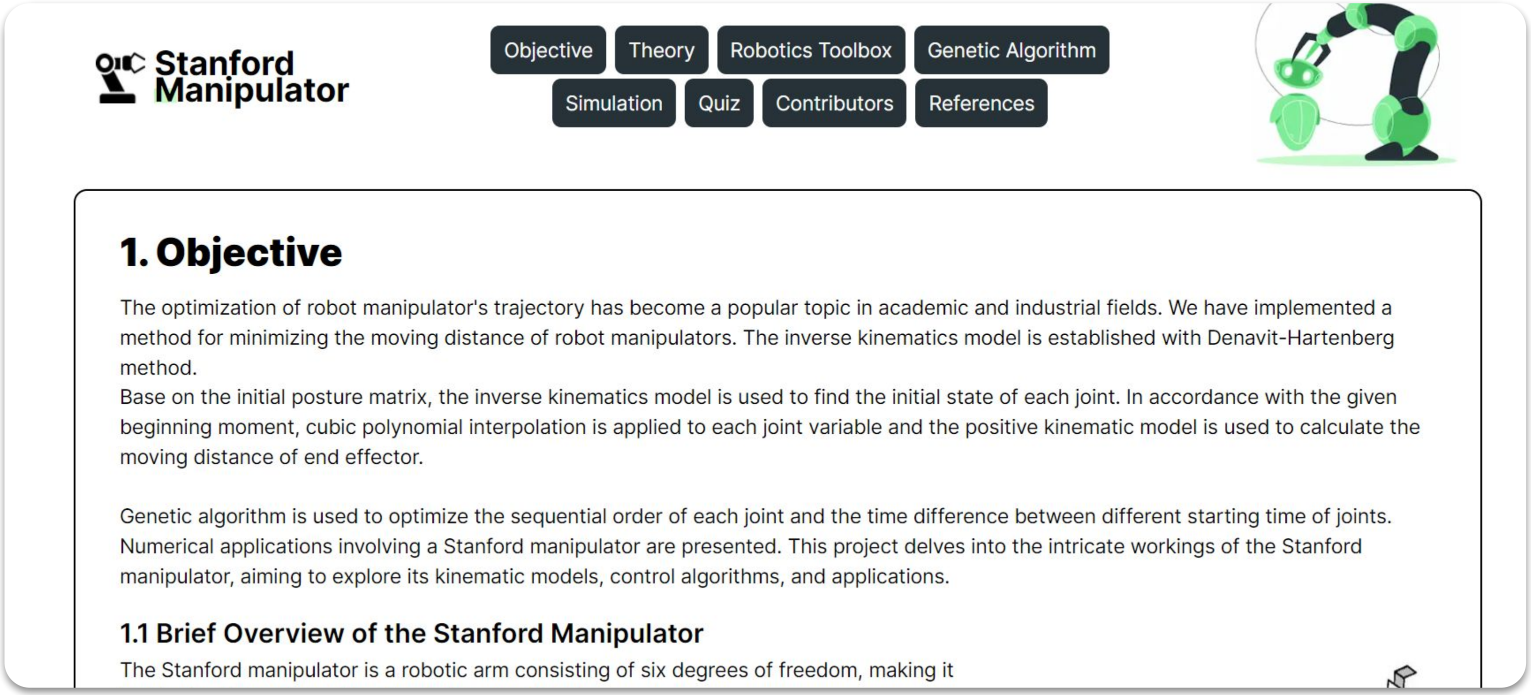


```
L =

    4.7520

Distance Travelled is 4.752017e+00 >>
```



```
The optimized distance is 4.190693e+00>>
```

Website



1. Objective

The optimization of robot manipulator's trajectory has become a popular topic in academic and industrial fields. We have implemented a method for minimizing the moving distance of robot manipulators. The inverse kinematics model is established with Denavit-Hartenberg method.
Base on the initial posture matrix, the inverse kinematics model is used to find the initial state of each joint. In accordance with the given beginning moment, cubic polynomial interpolation is applied to each joint variable and the positive kinematic model is used to calculate the moving distance of end effector.

Genetic algorithm is used to optimize the sequential order of each joint and the time difference between different starting time of joints. Numerical applications involving a Stanford manipulator are presented. This project delves into the intricate workings of the Stanford manipulator, aiming to explore its kinematic models, control algorithms, and applications.

1.1 Brief Overview of the Stanford Manipulator

The Stanford manipulator is a robotic arm consisting of six degrees of freedom, making it

Website Link: https://stanford-robotic-arm-b2.vercel.app/
Github Link: https://github.com/BURUGURAHUL/Stanford-Robotic-Arm

References

Optimization on Trajectory of Stanford Manipulator based on Genetic Algorithm, Xi Han, MATEC Web Conf., 128 (2017) 04001
DOI: https://doi.org/10.1051/matecconf/201712804001

Solution of Inverse Kinematics Problem using Genetic Algorithms, Applied Mathematics & Information Sciences, https://doi:10.18576/amis/100122

https://www.maplesoft.com/content/EngineeringFundamentals/14/MapleDocument_14/Forward%20Kinematics.pdf

https://medium.com/@fxb6476/genetic-algorithms-robot-control-6ff1af0450a7

Thank you!