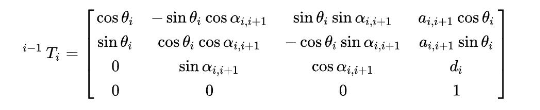
1. PRELIMINARIES

The following section deals with the important concepts behind the approach that is used in this paper. The approach is based on the the Denavit Hartenberg (DH) convention, Newton–Raphson method for Inverse Kinematic(IK) and our joint angle computing algorithm.

1. Denavit Hartenberg Notation

The kinematic information of robot components of each link i is described by rigidly attaching a coordinate frame to each joint i+1 (R. N. Jazar, Theory of applied robotics. Springer, 2010. ). The standard method of Denavit-Hartenberg (DH) parameterization is used for the same. By applying the DH model, four parameters characterized each coordinate frame. The link length a which is the distance between zi−1 and zi ,the link twist α which is the amount of rotation about xi to make zi−1 parallel to zi , the joint offset d which is the length between xi−1 and xi along zi−1 and the joint angle θi , the required angle about zi−1 to make xi−1 parallel to xi. Then using the DH convention we can get the transformation matrix of one joint to the next one.



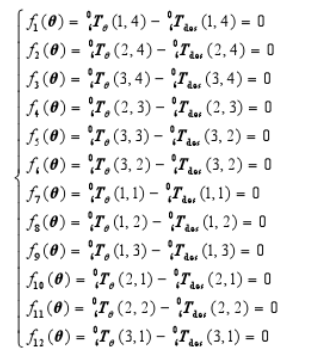
We can multiply the all joints’ transformation matrices by the base coordinate homogeneous form T\_base=[0,0,0,1]^T to get the final pose. In our six-joint robot model we can get:

T\_aim=T6\*T5\*T4\*T3\*T2\*T1\*T\_base

1. Newton-Raphson method

We need to solve the Inverse Kinematic(IK) problem of our manipulator, which is given the x, y and z position, we should find the corresponding joint angles of every joints of the manipulator. The Newton-Raphson method is for finding the minima in optimization. In our problem we need to find the right angles to minimize the difference between the now state position of end effector of the manipulator and the aim position.

First, we need to build nonlinear equations: F(**θ**)=(f1,f2,...,f12)^T=0, where **θ**=(θ1，θ2，θ3，θ4，θ5，θ6)^T. The each element f\_i of F(**θ**) is derivated by the following way. Given the aim homogeneous transformation matrix T\_aim of the hand coordinate system to the base coordinate system. And for each iteration, we can get a now state homogeneous transformation matrix T\_θ（θ1，θ2，θ3，θ4，θ5，θ6）. The two transformation matrices are both 4\*4. But we just use the upper 3\*4 elements( T11,T12,T13,T14,T21,T22,T23,T24,T31,T32,T33,T34) because the remaining 4 elements never change. We can subscribe the all 12 corresponding entries from the T\_θ and T\_aim one on one to get f\_i, i=1,2,3...11,12.



Then we can obtain the partial derivatives on (θ1，θ2，θ3，θ4，θ5，θ6)

from F（θ）as the Jacobian matrix J which is 12\*6. And we can get the left inverse of J by (JT\*J)-1\*JT as the pseudo inverse J+. At last use the Newton Iteration we can get the new **θ**.

new**θ**=**θ**-J+F(**θ**)

1. Joint angle computing algorithm.

...

1. Selfmodelling algorithm

In this paper, we try to discuss the problem of robot manipulator with some broken joints

on the industrial assembly line. The motors of the joints are broken so their angles are fixed in some values. By our C part above, we can obtain the values of some broken joints. Then in each iteration if we know which joints are broken, we can update the new angles values at last of each iteration by the values we computed from the joint angle computing algorithm above:

new θi=θi\_computed,assuming the ith joint is broken

Or we can set the ith column of Jacobian matrix J to 0, which means the ith joint does not have the effect on minimizing the difference between the now state position of end effector of the manipulator and the aim position. Depends on the adaptive characteristics of Newton iteration method, we can still go to the aim position.