SI 211: Comparison of Newton method and Gauss-Newton method of the Inverse Kinematics

Wangshu Zhu * **Chunyan Rong** **

* *ShanghaiTech University, China (e-mail: zhuwsh@shanghaitech.edu.cn).*
** *ShanghaiTech University, China (e-mail: rongchy@shanghaitech.edu.cn).*

Abstract: In this project, we apply the Newton method and Gauss-Newton method (a Newton-type method) on the inverse kinematic (IK) problem of the robot manipulator and compare the iteration time and computing time of the two methods. The robot manipulator with 6 degrees of freedom (DOF) is in 3-dimension space which is a hard problem to compute the IK. And there a constraint we used to illustrate the difference between the two methods.

Keywords: Global optimization, Iterative methods, Newton method, Gauss-Newton method, Jacobian matrices, Space robotics, Robot kinematics, Robot arms, Robotic manipulators

1. INTRODUCTION

For the control of robot manipulators, the inverse kinematic (IK) Kucuk and Bingul (2006) is definitely significant. It transforms the end effector in the Cartesian coordinates to the joint configuration space coordinatesGoldenberg et al. (1985). The common approach to solve this problem is solving a closed-form equation and obtaining the solution of the joint configuration space. But for the robot manipulator high DOF (e.g. 6,7,8...), it will take a long time to solve the equation and obtain numerous solutions which is unnecessary for robot manipulators.
A second approach for solving the IK problem of n DOF robot manipulators is based on the iterative procedures for solving a system of nonlinear equations e.g.Chen et al. (1999),Castellet and Thomas (1998). Starting from the initial position, the method will get the next position with small change aiming to the target position and repeat this procedure until reaching the target position. The most common algorithm proposed to solve the nonlinear kinematic equations uses the Newton method based on simultaneous successive linear interpolation of nonlinear equations.
Another kind of iterative methods is the Newton-type method. While the Newton method computes the exact differential of function f, the Newton-type method tries to use the approximation of the differential to save computing time. Because computing the Jacobian matrix and the Hessian matrix is time-consuming especially for high-dimension problems, some Newton-type methods try to update the approximation of Jacobian without re-computing the Jacobian.
In this project, the Newton method and one of the Newton-type method: Gauss-Newton method will be applied to the IK problem of a robot manipulator with 6 DOF is in 3-dimension space. And a constraint will be conducted on the problem. In Chapter 2 we formulate the problem and the constraint. In Chapter 3 we present the methods that we use for our experiments and introduce the algorithms. In Chapter 4 we present our results and discuss these results, their limitations and their relevance in the face of similar researches. Finally, in Chapter 5 we conclude this work.

2. PROBLEM FORMULATION

We consider the equality optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^3} F(\mathbf{x}) \tag{1}$$

where $F(\mathbf{x}) = \|(\mathbf{x} - \mathbf{x}^*)\|$ representing the distance between two vectors, $\mathbf{x}$ is the current position of the end effector in Cartesian coordinates which is [x,y,z], $\mathbf{x}^*$ is the target postion in Cartesian coordinates. So we need to minimize the distance between two points to make the end effector move from the initial position to the target position.
The constraint in our project is

$$\begin{cases} Ax + By + D = 0 \\ y = c \end{cases} \tag{2}$$

which means the trajectory should be on the plane paralleling to the z axis and y is a constant.

3. SOLUTION METHOD

3.1 Newton (or Newton-type) method

Originally, Newton method is to solve the nonlinear equation $f(x)$, we can start with an initial guess x_0 and solve the linear equation systems

$$f(x_k) + M(x_k)(x_{k+1} - x_k) = 0 \tag{3}$$

where $k \in 0, 1, 2, ...$ If f is differentiable, we might choose M(x_k)=$f^{'}(x)$, which is exact Newton method. If we work with the approximations M$(x_k) \approx f^{'}(x_k)$, then it is the Newton-type method.

Table 1.

Classic DH parameters

i	alpha(i-1)	a(i-1)	di	theta
1	$\pi/2$	0	0.2755	q1
2	π	0.4100	0	q2
3	$\pi/2$	0	0.0098	q3
4	$\pi/3$	0	- 0.2501	q4
5	$\pi/3$	0	-0.0856	q5
6	π	0	-0.2028	q6

3.2 Newton method for IK

Denavit-Hartenberg parameters In mechanical engineering, the Denavit—Hartenberg parameters Denavit and Hartenberg (1955)(also called DH parameters) are the four parameters associated with a particular convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator. We conduct our simulation on the Kinova 6 DOF robot arm whose DH parameters are shown in Table 1.

Transformation Matrix The transformation matrix from coordinate system A to coordinate system B has the following form:

$$ {}^{B}_{A}T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{4}$$

Where t is the translation vector and R is the rotation matrix. The form of t is shown as follows:

$$ t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5}$$

There are three forms of rotation matrix R according to the rotary axis:

$$ R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{6}$$

$$ R_Y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{7}$$

$$ R_Y(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

We use the Newton method to solve the inverse kinematic problem of our manipulator. We should find the corresponding joint angles for every single joint, and the Newton method enables us to find the root in an iterative process, given a specific x, y, z position. In our problem we need to find the right angles to minimize the difference between the current state position of the end effector of our manipulator and the target position.

Firstly we need to build nonlinear equations:

$$ \begin{cases} F(\theta) = (f_1, f_2, \ldots, f_{12})^T = \mathbf{0} \\ \theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)^T \end{cases} \tag{9}$$

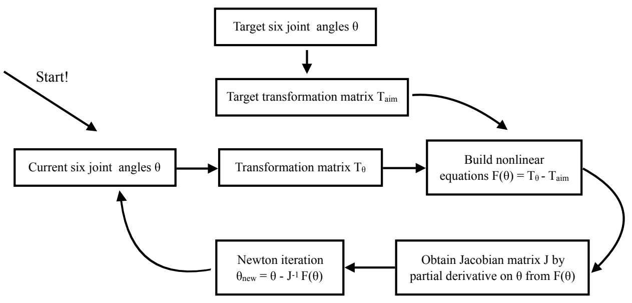
Target six joint angles θ
Start!
Target transformation matrix T_aim
Current six joint angles θ
Transformation matrix T_θ
Build nonlinear equations F(θ) = T_θ - T_aim
Newton iteration θ_new = θ - J^-1 F(θ)
Obtain Jacobian matrix J by partial derivative on θ from F(θ)

Fig. 1. The pipeline of the Newton method for IK. It starts from the current position, getting the current joint angles.

Each element f_i of $F(\theta)$ is derived by the following way: given the aim homogeneous transformation matrix $\mathbf{T_{aim}}$ of the hand coordinate system to the base coordinate system, $\mathbf{T_{aim}} = \mathbf{T_6 T_5 T_4 T_3 T_2 T_1}$, and for each iteration, we can find a current state homogeneous transformation matrix $\mathbf{T}_\theta$, $\mathbf{T}_\theta = \mathbf{T}_{\theta_6}\mathbf{T}_{\theta_5}\mathbf{T}_{\theta_4}\mathbf{T}_{\theta_3}\mathbf{T}_{\theta_2}\mathbf{T}_{\theta_1}$. The two transformation matrices are both 4×4, but we just use the upper 3×4 elements ($\mathbf{T_{11}}$,$\mathbf{T_{12}}$,$\mathbf{T_{13}}$,$\mathbf{T_{14}}$,$\mathbf{T_{21}}$,$\mathbf{T_{22}}$,$\mathbf{T_{23}}$,$\mathbf{T_{24}}$,$\mathbf{T_{31}}$,$\mathbf{T_{32}}$,$\mathbf{T_{33}}$,$\mathbf{T_{34}}$), as the remaining four elements never change. We can subscribe all 12 corresponding entries from the $\mathbf{T}_\theta$ and $\mathbf{T_{aim}}$, one by one, to find $f_i, i = 1, 2, 3 \ldots 11, 12$.

Then we can obtain the partial derivatives on $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ from $F(\theta)$ as the Jacobian matrix J, which is 12×6, and we can find the left inverse of J as $(J^TJ)^{-1}J^T$. Lastly, we use the iteration in the Newton algorithm to obtain θ^{i+1}. This workflow is demonstrated in Fig. 1. If the distance between the current position and the target position is below a constant we set at first, the loop will end. In the simulation the default constant is 0.001.

3.3 Gauss-Newton method for IK

In order to solve the nonlinear least-squares optimization problem $f(x)$, we start with an initial guess x_0 and solve the standard least-squares problem

$$ \min_{\Delta x_k} \|f(x_k) + f^{'}(x_k)\Delta x_k)\|^2 \tag{10}$$

and ... is a line search parameter.
If the Jacobian matrix $f^{'}(x_k)$ has full-rank, the step direction can alternatively be written in the form

$$ \Delta\, x_k = -(f^{'}(x_k)^T f^{'}(x_k))^{-1} f^{'}(x_k)^T f^{'}(x_k) \tag{11}$$

Gauss-Newton methods are special class of Newton type methods which employ the Hessian approximation

$$ \overset{(2)}{F}(x_k) \approx M(x_k) = f^{'}(x_k)^T f^{'}(x_k) \tag{12}$$

3.4 Newton (type) for IK with constraints

For a general nonlinear equality constrained optimization problem

$$ \min_x F(x) s.t. G(x) = 0 \tag{13}$$

The first order necessary KKT condition is: if x* is a minimizer at which he matrix A* =G'(x*) has full rank, then there exists a multiplier $\lambda \in R^m$ such that

$$0 = F'(x^*)^T + G'(x^*)^T \lambda^* \tag{14}$$

$$0 = G(x^*) \tag{15}$$

The process is:
1.Choose initial guesses x_0 and λ_0, tolerance error<0.
2.Repeat:
2.1 Choose a positive definit Hessian approximation
2.2 Solve the KKT system

$$\begin{pmatrix} M(x_k) & A(x_k)^T \\ A(x_k) & 0 \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \tilde{\lambda}_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla_x L(x_k, \lambda_k) \\ -G(x_k) \end{pmatrix} \tag{16}$$

2.3 Terminate if $|F'(x_k)\Delta x_k| + \sum_{i=0}^{m} |\lambda_i| \, |G_i(x_k)| < \epsilon$
2.4 Choose a line-search parameter $\alpha \in [0,1]$ and set $x_{k+1} = x_k + \alpha\delta x_k$ as well as $\lambda_{k+1} = \lambda_k + \alpha(\lambda_{k+1} - \lambda_k)$ and k=k+1

4. NUMERICAL RESULTS

4.1 Results

We test our code in a simple case in 2-D which means the z value does not change in the experiments.The initial angles of six joints are q1 = 0.0, q2 = 2.9, q3 = 1.3, q4 = 4.2, q5 = 1.4, q6 = 0.0 in radian. The coresponding Cartesian coordinates is x=-0.269895400271165, y=-0.232693755610215, z=0.619562132325957. The target position is x = -0.2, y = 0.6, z = 0.4.
We use the Newton method and the Gauss Newton method to find the optimal angles of all six joints. The solution for the Newton method is [-0.0571889823272353], [1.15685918021512], [2.57969997208474], [3.968672982397 64], [-1.64732821473317], [1.57079632679490] which transforms the base from [0, 0, 0] to the [-0.200734813393614, -0.232642445544987, 0.602854017242225]. The iteration time is 688.8872308731079 seconds while the iteration is 46. The trajectory can be seen in Fig. 2.
The solution for the Gauss Newton method is [-0.008178 91134471990], [1.14767194683545], [2.85464287169995], [4. 22052115421740], [-1.70144983583778], [1.57079632679490] which transforms the base from [0, 0, 0] to the [-0.200512665156810, -0.232707102568481, 0.6486953855117 30]. The iteration time is 397.81528997421265 seconds while the iteration is 23 which is better than exact Newton method. The trajectory can be seen in Fig. 3. The Gauss-Newton method achieves a better trajectory which is closer to the path line. In general Gauss Newton method works better in this IK problem.

We also test the Gauss Newton method in 3-D space which can be seen in Fig.5 and the initial position and target position are illustrated in Fig.6 and Fig.7. We do not apply the exact Newton method in 3D because the Hessian matrix in 3D is too hard for us to compute.

4.2 Simulation environment

We choose the Robot Operating System (ROS) as our simulation environment. ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety

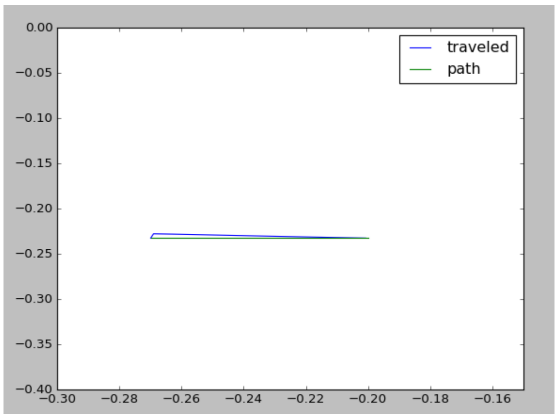

Fig. 2. The Newton method in 2D space. The green line is connecting the initial position and the target position. The blue line is the accurate travelled trajectory.

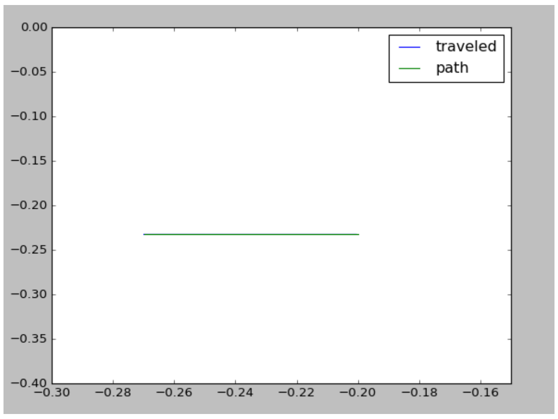

Fig. 3. The Gauss-Newton method in 2D space. The green line is connecting the initial position and the target position. The blue line is the accurate travelled trajectory.

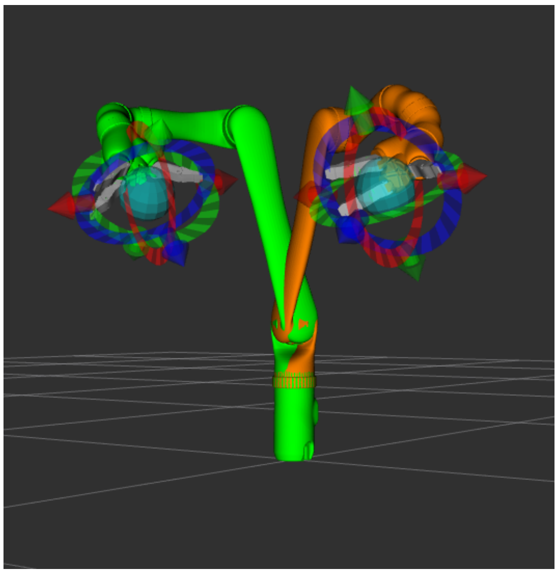

Fig. 4. The simulation of the change of the arm in Rviz.

of robotic platforms.
Creating truly robust, general-purpose robot software is hard. From the robot's perspective, problems that seem

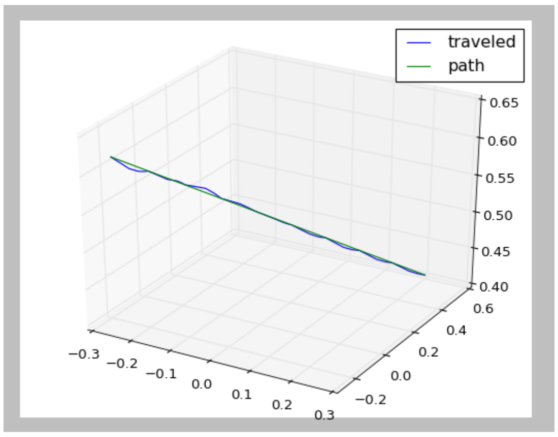

Fig. 5. The Gauss-Newton method in 3D space. The green line is connecting the initial position and the target position. The blue line is the accurate travelled trajectory.

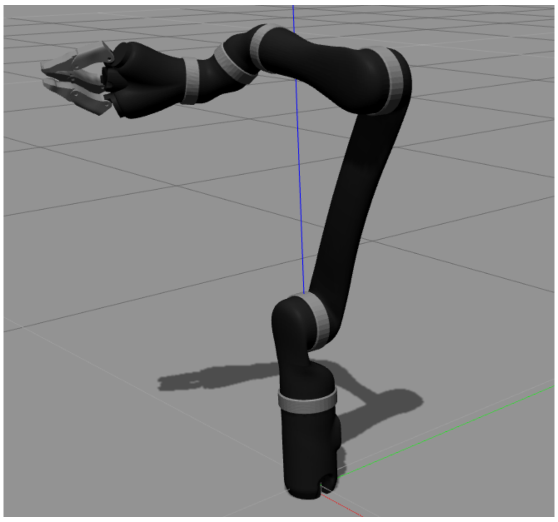

Fig. 6. The initial position of the arm in Gazebo.

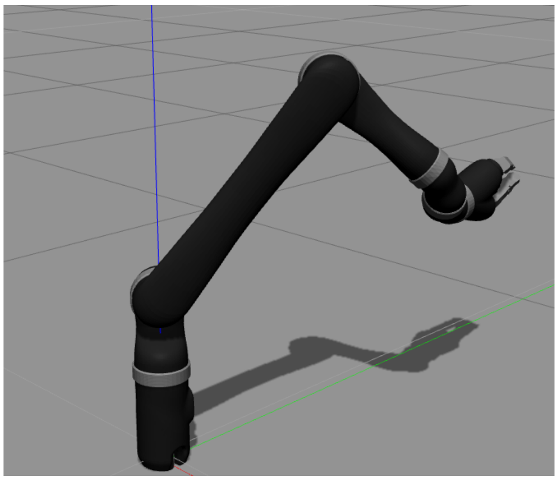

Fig. 7. The changed position the arm in Gazebo

trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own.

As a result, ROS was built from the ground up to encourage collaborative robotics software development. For example, one laboratory might have experts in mapping indoor environments, and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed specifically for groups like these to collaborate and build upon each other's work, as is described throughout this site.
If we connect to the computer to the real manipulator, our code will work and move the joints the same as simulation which is the greatest thing of ROS.

5. CONCLUSION

In this project, we apply the Newton method and Gauss-Newton method (one Newton-type method) on the inverse kinematic (IK) problem of the robot manipulator and compare the iteration time and computing time of the two methods. The constraint is that the trajectory should be on the plane paralleling to the z axis and y is a constant. We evaluate the efficiency of both methods in 2D space and find that the Gauss Newton method has a better performance. And Gauss Newton method is able to find the optimal angles of joints to minimize the distance between the current position and the target position in 3D.
In the future, we may apply some other constraint on the problem to change the trajectory e.g. on cylindrical surface crossing the initial position and the target position which is excellent to avoid obstacles. And computing IK in Newton method is relatively time-consuming which means we can compute the trajectory in advance and apply it to the manipulators which is more practical in industry.

ACKNOWLEDGEMENTS

In the project and the whole course we really appreciate Professor Boris Houska and all TAs. Your efforts and knowledge help us a lot in understanding the basic processes of numerical analysis.

REFERENCES

Castellet, A. and Thomas, F. (1998). An algorithm for the solution of inverse kinematics problems based on an interval method. In *Advances in Robot Kinematics: Analysis and Control*, 393–402. Springer.

Chen, I.M., Yang, G., and Kang, I.G. (1999). Numerical inverse kinematics for modular reconfigurable robots. *Journal of Robotic Systems*, 16(4), 213–225.

Denavit, J. and Hartenberg, R.S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME E, Journal of Applied Mechanics*, 22, 215–221.

Goldenberg, A., Benhabib, B., and Fenton, R. (1985). A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation*, 1(1), 14–20.

Kucuk, S. and Bingul, Z. (2006). Robot kinematics: Forward and inverse kinematics. In *Industrial Robotics: Theory, Modelling and Control*. IntechOpen.