4. Newton Method for IK

Game Engineering & XR Technologies

Prof. HyeongYeop Kang
siamiz@khu.ac.kr
IIIXR LAB

Introduction

Newton Methods[1]

- Newton methods are iterative techniques used to solve nonlinear equations:
  - **Problem definition**: The IK problem is formulated as a system of nonlinear equations, where the unknowns are the joint angles or variables, and the goal is to find a solution that brings the end effector as close as possible to the target pose.
  - **Problem linearization**: The Newton method works by linearizing the system of nonlinear equations around the current estimate of the solution.
  - **Iterative update**: In each iteration, the method computes an update step for the joint angles or variables by solving the linearized system of equations and updates joint angles.
  - **Convergence check**: The method iterates until a convergence criterion is met (reaching a specified maximum number of iterations, achieving a sufficiently small change in the joint angles between consecutive iterations, etc.)

Introduction

Newton Methods[1]

- The Newton family of methods is based on a second order Taylor series expansion of the object function f(x):

$$f(x+\sigma) \approx f(x) + [\triangledown f(x)]^{T}\sigma + \frac{1}{2}\sigma^{T}H_f(x)\sigma$$

where $H_f(x)$ is the Hessian matrix.

- However, the calculation of the Hessian matrix is very complex and it results in high computational cost for each iteration.
  - To reduce the computational complexity, several approaches have been proposed that, instead of calculating the Hessian matrix, use an approximation of the Hessian matrix based on a function gradient value.
  - The most well-known methods are Broyden's method, Powell's method, and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

Introduction

Newton Methods vs. Jacobian-based Method

- Advantages of Newton method:
  - **Fast convergence (quadratic convergence)**: When initialized with a good initial guess, this method often converges faster than Jacobian-based method. For example, let's estimate $\sqrt{2}$ using Newton's method.

n = itor num	**0**	**1**	**2**	**3**	**4**	**5**
x_n	1.00000000	1.50000000	$1.\underline{41}666666$	$1.\underline{41421}568$	$\underline{1.414213562}$	1.414213562..

  - **Applicability to nonlinear problems**: The Newton method can directly solve the nonlinear IK problems, which might be more suitable for highly nonlinear systems or when exact solutions are required.

- Disadvantages of Newton method:
  - **Sensitivity to initial conditions**: The convergence of the Newton-Raphson method highly depends on the initial guess. If the initial guess is far from the true solution, the method may fail to converge or converge to an undesired local minimum.
  - **Computation of Jacobian and Hessian**: The Newton-Raphson method requires the computation of the Jacobian matrix (first-order partial derivatives) and sometimes the Hessian matrix (second-order partial derivatives) at each iteration, which can be computationally expensive for complex systems with many joints and degrees of freedom.
  - **Inverse of Jacobian**: In some cases, the Newton-Raphson method may require the inversion of the Jacobian matrix, which can be numerically unstable or ill-conditioned, especially for singular or near-singular configurations.

Basics

Taylor Series[2]

- The functions which are infinitely differentiable can generate a power series called the Taylor series.

- Suppose we have a function $f(x)$ and $f(x)$ has derivatives of all orders on a given interval, then the Taylor series generated by $f(x)$ at $x = a$ is given by:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

$$= \sum_{n=0}^{\infty} \frac{f^{n}(a)}{n!}(x - a)^{n}$$

- If we set $a = 0$, then we have an expansion called the Maclaurin series expansion of $f(x)$.

$$f(x) = f(0) + f'(0)\,x + \frac{f''(0)}{2!}x^2 + \frac{f^{(3)}(0)}{3!}x^3 + \ldots + \frac{f^{(n)}(0)}{n!}x^n + \ldots.$$

Power Series[3]?

A power series is an infinite series of the form

$$\sum_{n=0}^{\infty} a_n (x-c)^n = a_0 + a_1(x-c) + a_2(x-c)^2 + \ldots$$

where a_n represents the coefficient of the nth term and c is a constant value.

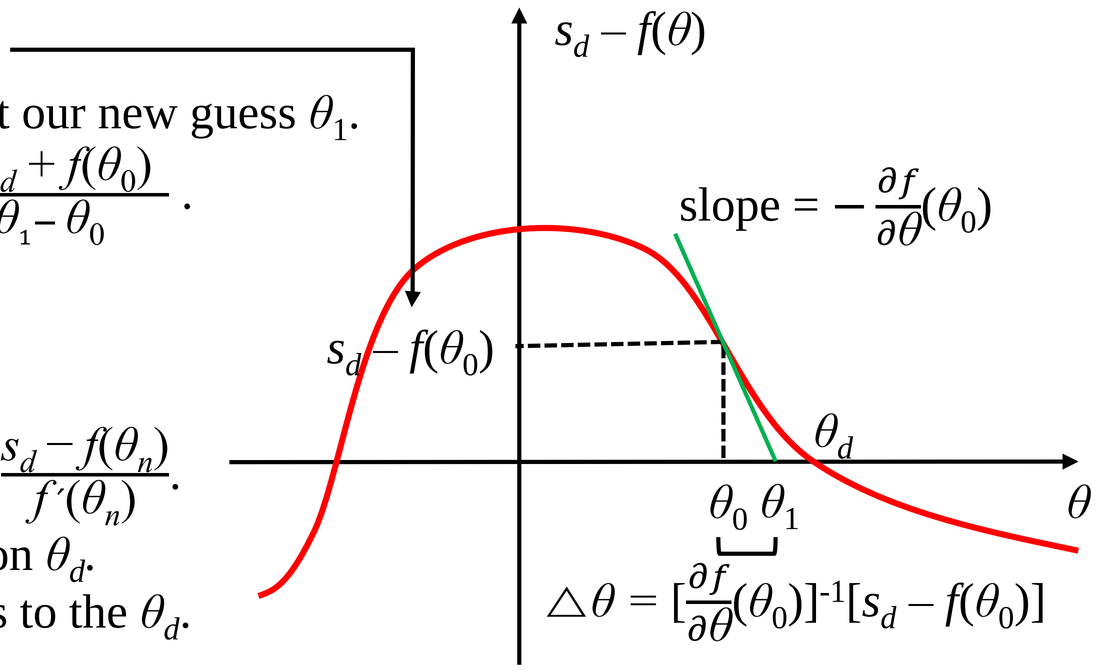
Newton-Raphson Method

Newton-Raphson Method (also known as Newton method) and IK

- In the context of IK, Newton-Raphson method is referred Newton method.
  - The Newton-Raphson method specifically targets finding the roots of a nonlinear function.

- Recall that the objective of IK is to find the joint angle θ such that $F(x) = s_d - f(\theta) = 0$ where s_d is the desired position of the end effector.

- Let's say the solution to this problem is θ_d. Then, we can adopt Newton-Raphson root finding method.
  - The initial guess of the root is typically denoted θ_0.
    - At that guess, we can calculate the value of $s_d - f(\theta_0)$.
  - If we extend the slope to where it crosses the θ-axis, we get our new guess θ_1.
    - The slope can be expressed as: slope $= -\frac{\partial f}{\partial \theta}(\theta_0) = \frac{0 - s_d + f(\theta_0)}{\theta_1 - \theta_0}$.
    - Then, $\theta_1 - \theta_0 = \triangle\theta$ in the guess is given by:
      - $\triangle\theta = \frac{-s_d + f(\theta_0)}{-\frac{\partial f}{\partial \theta}(\theta_0)} = \frac{s_d - f(\theta_0)}{\frac{\partial f}{\partial \theta}(\theta_0)}$
    - Therefore, the update function is given by $\theta_{n+1} = \theta_n + \frac{s_d - f(\theta_n)}{f'(\theta_n)}$.
  - Through the process, our guess becomes closer to a solution θ_d.
    - We can repeat the process until the guess θ_n converges to the θ_d.



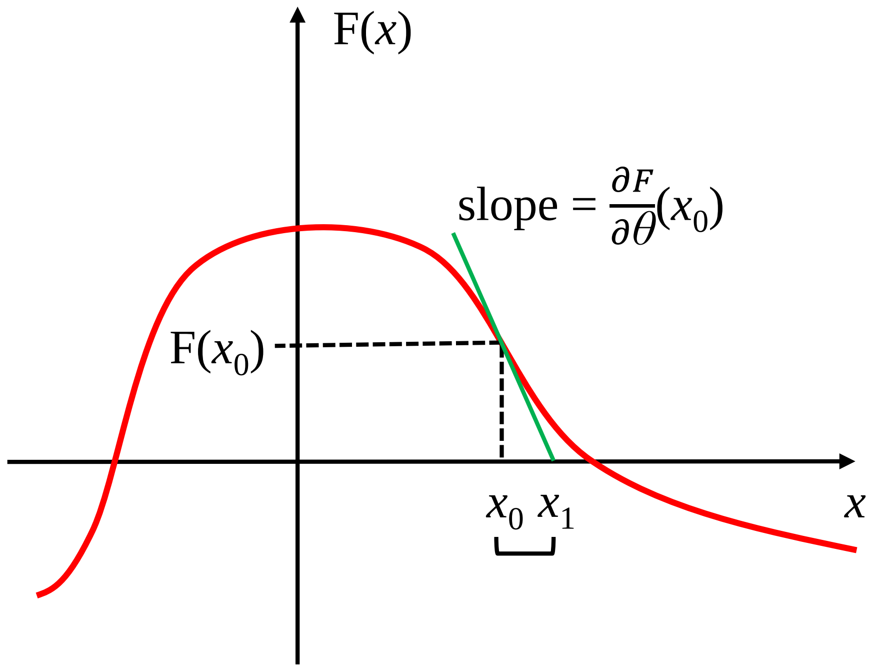
$s_d - f(\theta)$
slope $= -\frac{\partial f}{\partial \theta}(\theta_0)$
$s_d - f(\theta_0)$
θ_d
θ_0 θ_1
θ
$\triangle\theta = [\frac{\partial f}{\partial \theta}(\theta_0)]^{-1}[s_d - f(\theta_0)]$

Practice

Q. Use the Newton-Raphson method to find an one-step improvement on the initial estimation.

- Q1. $F(x) = e^x - 4$ from initial estimate $x_0 = 1.5$.

- Q2. $G(x) = x^3 - 2x - 4$ from initial estimate $x_0 = 2.5$.

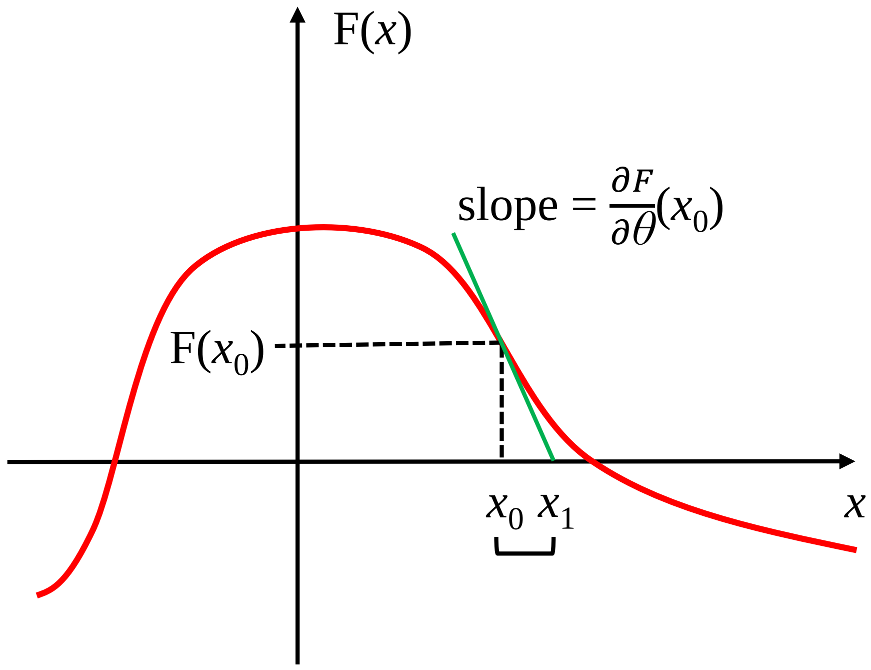


Solution

Q. Use the Newton-Raphson method to find an one-step improvement on the initial estimation.

- Q1. $F(x) = e^x - 4$ from initial estimate $x_0 = 1.5$.
  - In this example, $x_{n+1} = x_n - \frac{F(xn)}{F\prime(xn)}$. Then, $x_1 = 1.5 - \frac{0.4817}{4.4817} = 1.3925$.

- Q2. $F(x) = x^3 - 2x - 4$ from initial estimate $x_0 = 2.5$.
  - In this example, $x_{n+1} = x_n - \frac{F(xn)}{F\prime(xn)}$. Then, $x_1 = 2.5 - \frac{6.625}{16.75} = 2.1045$.



Newton Method in Optimization Problem

Finding Critical Points

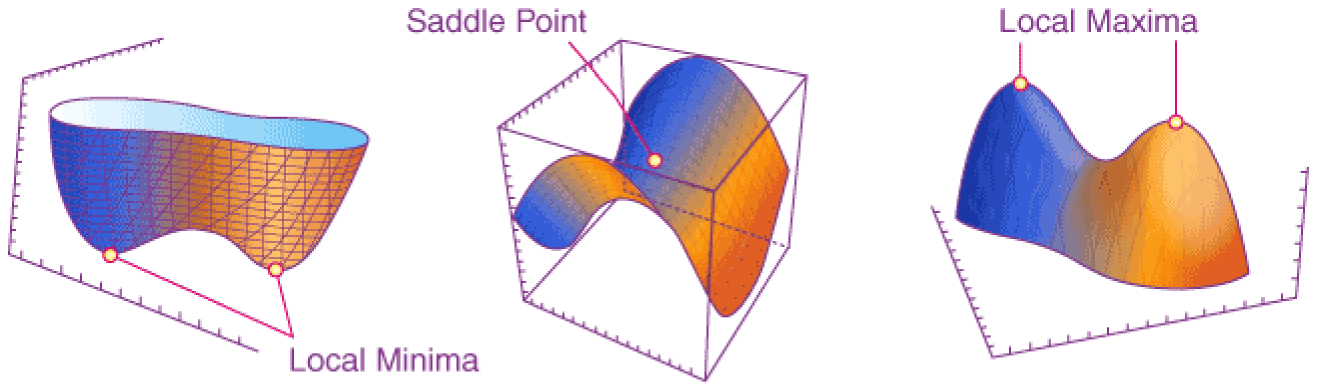
- Instead of finding the joint angle θ_d making $s_d - f(\theta_d) \approx 0$, Newton method can be used for minimizing $s_d - f(\theta_d)$. We call this optimization problem.

- Let's abbreviate $s_d - f(\theta)$ as $F(\theta)$.

- Recall that the Taylor Series approximates $F(\theta)$ by polynomials of increasing powers:

$$F(\theta_0 + \triangle\theta) = F(\theta_0) + \frac{F'(\theta_0)}{1!}(\triangle\theta) + \frac{F''(\theta_0)}{2!}(\triangle\theta)^2 + \frac{F'''(\theta_0)}{3!}(\triangle\theta)^3 + \cdots$$

- We want to find $\triangle\theta$ such that $(\theta_0 + \triangle\theta)$ is the solution to minimizing the equation.
  - This means that $(\theta_0 + \triangle\theta)$ is the stationary point where the first derivative of the function is equal to zero.
  - Stationary point can be of three types: local minimum, local maximum, or a saddle point.



[spoint]

Newton Method in Optimization Problem

Finding Critical Points

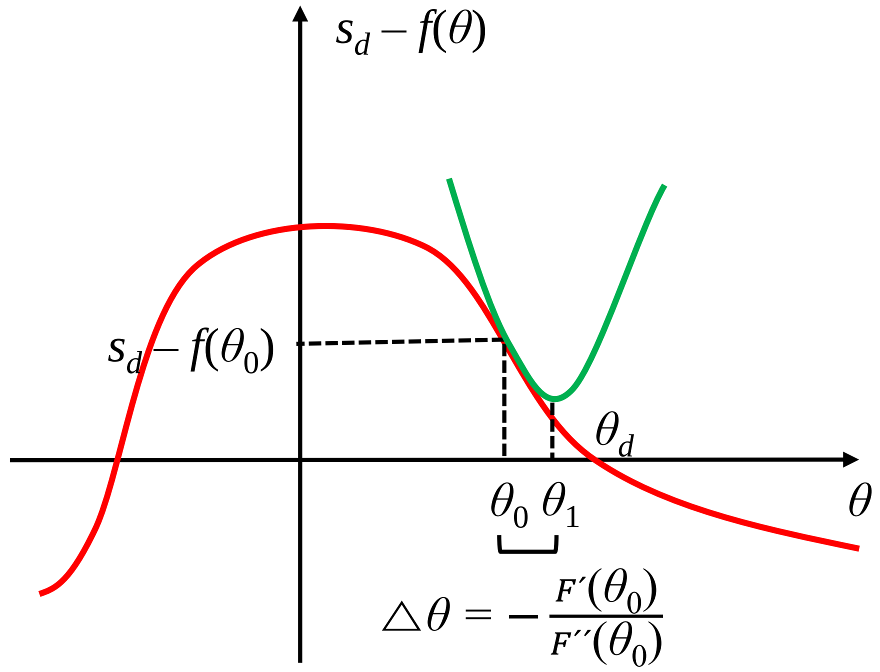
- To get an estimate of x when $F'(\theta) = 0$ (stationary point), we can truncate the second-order Taylor polynomial, and solve by setting the derivative to 0.

$$F(\theta_0+\triangle\theta) = F(\theta_0) + \frac{F'(\theta_0)}{1!}(\triangle\theta) + \frac{F''(\theta_0)}{2!}(\triangle\theta)^2 + \frac{F'''(\theta_0)}{3!}(\triangle\theta)^3 + \cdots$$

$$F'(\theta) = \frac{df}{d\triangle\theta}[(F(\theta_0) + \frac{F'(\theta_0)}{1!}(\triangle\theta)] = 0 \qquad \text{(truncated)}$$

$$\rightarrow F'(\theta_0) + F''(\theta_0)\triangle\theta = 0$$

$$\rightarrow \triangle\theta = -\frac{F'(\theta_0)}{F''(\theta_0)}$$

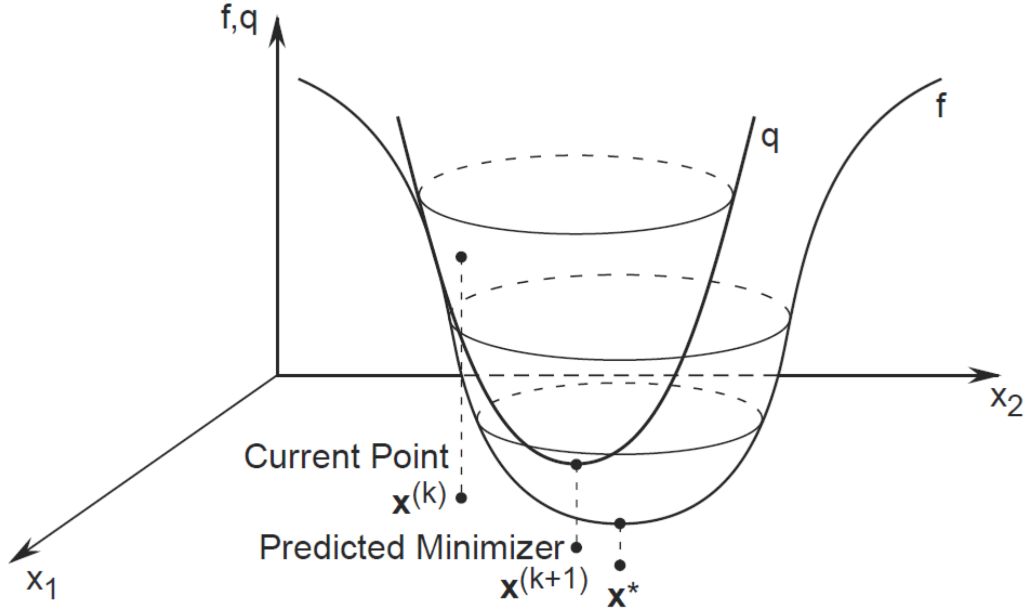


Multivariate Newton Method

Multivariate Newton Method

- The IK problem is to solve the multivariate problem. Therefore, the single variable case (univariate case) should be converted to the multivariate case.

- As we learned earlier, Newton method can refer to two different things, depending on the context:
  - Finding the roots (zeros) of a system of equations → in this context, we will use Jacobian.
  - Finding the minima or maxima of a function → in this context, we will use Hessian.



Multivariate Newton Method

Multivariate Newton Method for Root Finding

- Let’s consider a multivariate root finding problem.
  - The goal is to find the joint angles that satisfy the given end-effector pose.
  - In this case, the problem is formulated as a vector-valued function $F(\theta)$:
    - $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$.
    - $F(\theta) = s_d - f(\theta_d)$ which represents the difference between the desired end-effector pose and the current end-effector pose calculated using forward kinematics.

- Therefore, we will solve the system of m equations with n variables:
  - As this is root finding problem, we will find $F(\theta) = 0$.
    - $F(\theta)$ is a vector-valued function $F{:}R^m \rightarrow R^m$.
    - $F(\theta) = \begin{bmatrix} F_1(\theta_1, \theta_2, \ldots, \theta_n) \\ F_2(\theta_1, \theta_2, \ldots, \theta_n) \\ \vdots \\ F_m(\theta_1, \theta_2, \ldots, \theta_n) \end{bmatrix}$

Multivariate Newton Method

Multivariate Newton Method for Root Finding

- The update function for the multivariate Newton-Raphson method can be derived using Taylor series expansion:
  - $F(\theta_{n+1}) = F(\theta_n) + J(\theta_n)(\triangle\theta) + O(\triangle\theta^2)$ (multivariate) $\leftrightarrow F(\theta_{n+1}) = F(\theta_n) + F'(\theta_n)(\triangle\theta) + O(\triangle\theta^2)$ (univariate)
  - θ_n: current guess
  - θ_{n+1}: updated guess
  - $J(\theta_n)$: Jacobian matrix of F evaluated at $\theta_n = \begin{bmatrix} \frac{\partial F_1}{\partial \theta_1} & \frac{\partial F_1}{\partial \theta_2} & \cdots & \frac{\partial F_1}{\partial \theta_n} \\ \frac{\partial F_2}{\partial \theta_1} & \frac{\partial F_2}{\partial \theta_2} & \cdots & \frac{\partial F_2}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial \theta_1} & \frac{\partial F_m}{\partial \theta_2} & \cdots & \frac{\partial F_m}{\partial \theta_n} \end{bmatrix}$
  - $O(\triangle\theta^2)$: higher-order terms (which will be ignored in this approximation)

- Root finding problem finds θ_{n+1} such that $F(\theta_{n+1}) = 0$.
  - Therefore, we can rewrite the equation as $0 = F(\theta_n) + J(\theta_n)(\triangle\theta)$.
  - Then, $\theta_{n+1} - \theta_n = -\frac{F(\theta_n)}{J(\theta_n)} \rightarrow \theta_{n+1} = \theta_n - \frac{F(\theta_n)}{J(\theta_n)}$

Multivariate Newton Method

Multivariate Newton Method for Root Finding Example (root finding)

- Assume that we have a robotic system with a single end effector located at (x, y, z) defined by a set of joint angle $\theta = [\theta_1, \theta_2, \ldots, \theta_8]^T$.

- In $F(\theta) = s_d - f(\theta_d)$, $f(\theta)$ would be a 3-dimensional vector: $f(\theta) = \begin{bmatrix} f_1(\theta_1, \theta_2, \ldots, \theta_8) \\ f_2(\theta_1, \theta_2, \ldots, \theta_8) \\ f_3(\theta_1, \theta_2, \ldots, \theta_8) \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

Then, $F(\theta) = \begin{bmatrix} s_d(x) - f_1(\theta_1, \theta_2, \ldots, \theta_8) \\ s_d(y) - f_2(\theta_1, \theta_2, \ldots, \theta_8) \\ s_d(z) - f_3(\theta_1, \theta_2, \ldots, \theta_8) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

- Then, the Jacobian matrix $J(\theta)$ would be a 3×8 matrix containing the first-order partial derivatives of $F(\theta)$ with respect to each of the 8 joint angles.

- Start with an initial guess of the joint angles $\theta_{0,}$ θ_n is iteratively updated.

- The iteration is stopped when the error is below a certain threshold or the maximum number of iterations is reached.

Multivariate Newton Method

Multivariate Newton Method in Optimization Problem

- Let's consider a multivariate optimization problem.
  - The goal is to find the joint angles that minimize a scalar-valued objective function.
  - The object function could be the sum of squared differences between the desired end-effector pose and the current end-effector pose.
  - In this case, the problem is formulated as a scalar-valued function $F(\theta)$:
    - $\theta = [\theta_1, \theta_2, ..., \theta_n]^T$.
    - $F(\theta) = \sum (s_d - f(\theta_d))^2$ which represents the sum of squared difference between the desired end-effector pose and the current end-effector pose calculated using forward kinematics.

- Therefore, we will solve the system with n variables:
  - As this is optimization problem, we will find $\triangledown F(\theta) = 0$.
    - $F(\theta)$ is a scalar-valued function $F{:}R^n \rightarrow R$.
    - $F(\theta) = [F(\theta_1, \theta_2, ..., \theta_n)]$
    - $\triangledown F(\theta) = \begin{bmatrix} \frac{\partial F}{\partial \theta_1} \\ \frac{\partial F}{\partial \theta_2} \\ \vdots \\ \frac{\partial F}{\partial \theta_n} \end{bmatrix}$

Newton-Raphson Method

Multivariate Newton Method in Optimization Problem

- The update function for the multivariate Newton-Raphson method can be derived using Taylor series expansion:
  - $F(\theta_{n+1}) = F(\theta_n) + \triangledown F(\theta_n)(\triangle\theta) + O(\triangle\theta^2)$ (multivariate) $\leftrightarrow F(\theta_{n+1}) = F(\theta_n) + F'(\theta_n)(\triangle\theta) + O(\triangle\theta^2)$ (univariate)
  - (after differentiate) $\triangledown F(\theta_{n+1}) = \triangledown F(\theta_n) + H_F(\theta_n)(\triangle\theta)$
  - $H_F(\theta_n)$: Hessian matrix of F evaluated at $\theta_n = \begin{bmatrix} \frac{\partial^2 F}{\partial^2\theta_1} & \frac{\partial^2 F}{\partial\theta_1\partial\theta_2} & \cdots & \frac{\partial^2 F}{\partial\theta_1\partial\theta_n} \\ \frac{\partial^2 F}{\partial\theta_2\partial\theta_1} & \frac{\partial^2 F}{\partial^2\theta_2} & \cdots & \frac{\partial^2 F}{\partial\theta_2\partial\theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial\theta_n\partial\theta_1} & \frac{\partial^2 F}{\partial\theta_n\partial\theta_2} & \cdots & \frac{\partial^2 F}{\partial^2\theta_n} \end{bmatrix}$

- Optimization problem finds θ_{n+1} such that $\triangledown F(\theta_{n+1}) = 0$.
  - Therefore, we can rewrite the equation as $0 = \triangledown F(\theta_n) + H_F(\theta_n)(\triangle\theta)$
  - Then, $\theta_{n+1} - \theta_n = -\frac{\triangledown F(\theta_n)}{H_F(\theta_n)} \rightarrow \theta_{n+1} = \theta_n - \frac{\triangledown F(\theta_n)}{H_F(\theta_n)}$

Newton-Raphson Method

Multivariate Newton Method in Optimization Problem Example (optimization)

- Assume that we have a robotic system with a single end effector located at (x, y, z) defined by a set of joint angle $\theta = [\theta_1, \theta_2, ..., \theta_8]^T$.

- Then we will define a scalar-valued objective function $F(\theta)$ that measures the error between the desired end-effector poses and the current end-effector poses.
  - Then, $F(\theta)$ would be the sum of squared differences.

- Then, the gradient vector $\triangledown F(\theta)$ would be an 8 dimensional vector containing the first-order partial derivatives of $F(\theta)$ with respect to each of the 8 joint angles:
  - $\triangledown F(\theta) = \begin{bmatrix} \frac{\partial F}{\partial \theta_1} \\ \frac{\partial F}{\partial \theta_2} \\ \vdots \\ \frac{\partial F}{\partial \theta_8} \end{bmatrix}$

- The Hessian matrix $H_F(\theta)$ would be an 8×8 matrix containing the second-order partial derivatives of $F(\theta)$ with respect to each pair of the 8 joint angles.

- Start with an initial guess of the joint angles $\theta_{0,}$ θ_n is iteratively updated.

- The iteration is stopped when the error is below a certain threshold or the maximum number of iterations is reached.

Newton-Raphson Method

Multivariate Newton Method (Briefly says..)

- The IK problem is to solve the multivariate problem. Therefore, the single variable case (univariate case) should be converted to the multivariate case.
  - Therefore, the derivative $F'(\theta_n)$ should be replaced with the gradient $\nabla F(\theta_n)$.
  - The reciprocal of the second derivative $\frac{1}{F''(\theta_n)}$ should be replaced with the inverse of the Hessian matrix, $\frac{1}{H_F(\theta_0)}$.
    - The reciprocal of the second derivative (inverse of the second derivative) provide information about how the curvature of the function behaves.
      - When the second derivative is large, the reciprocal will be close to zero, indicating that the function has a high curvature at that point.
      - When the second derivative is close to zero, the reciprocal will be large, indicating that the function has a low curvature at that point.
    - In some optimization or numerical methods (including Newton method), the reciprocal is used as a scaling factor or to approximate the step size.

Newton-Raphson Method

Multivariate Newton Method (Briefly says..)

- As a result, the update function in a single variable case

$$\theta_{n+1} = \theta_n - \frac{F(\theta_n)}{F'(\theta_n)} \quad and \quad \theta_{n+1} = \theta_n - \frac{F'(\theta_n)}{F''(\theta_n)}$$

can be rewritten as:

$$\theta_{n+1} = \theta_n - \frac{F(\theta_n)}{J(\theta_n)} \quad and \quad \theta_{n+1} = \theta_n - \frac{\nabla F(\theta_n)}{\mathrm{H}_F(\theta_n)}$$

- Note that the second-order methods often converge much more quickly but it can be very expensive to calculate and store the inverse of the Hessian matrix. In general, most people prefer quasi-newton methods to approximate the Hessian.

Newton-Raphson Method

Root Finding VS. Optimization

- Advantages of root finding
  - It can be computationally less expensive since only the first derivative is required.
  - It can be applied to a wider range of functions, as some functions may not have a second derivative.
  - It is suitable for gradient-based optimization techniques like gradient descent, where the first derivative is used to determine the direction of the search.

- Disadvantages of root finding
  - It may not provide enough information to distinguish between different types of critical points (local minima, local maxima, or saddle points) since only the first derivative is used.
  - Convergence might be slower compared to methods that use second-order derivatives.
  - The method can be sensitive to the choice of the initial guess and might converge to different solutions depending on the starting point.

Newton-Raphson Method

Root Finding VS. Optimization

- Advantages of optimization
  - The second derivative provides additional information about the curvature of the function, allowing for better identification of the type of critical point (local minimum, local maximum, or saddle point).
  - Convergence can be faster as it leverages both first and second derivatives.
  - It may provide more accurate results since it takes into account the function's curvature.

- Disadvantages of optimization
  - It requires the computation of both the first and second derivatives, which can be computationally expensive and may not always be possible, especially for complex or non-differentiable functions.
  - It is limited to functions that have a second derivative, which may not be the case for all optimization problems.
  - Similar to methods using only the first derivative, it can be sensitive to the choice of the initial guess and might converge to different solutions depending on the starting point.

Reference

[1] http://www.cs.cmu.edu/~15464-s13/lectures/lecture6/iksurvey.pdf
[2] https://machinelearningmastery.com/a-gentle-introduction-to-taylor-series/
[3] https://en.wikipedia.org/wiki/Power_series
[4] https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/partial-derivative-and-gradient-articles/a/the-gradient
[5] https://en.wikipedia.org/wiki/Gradient
[6] https://en.wikipedia.org/wiki/Hessian_matrix
[spoint] https://byjus.com/maths/saddle-points/