南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Robot Modeling & Control ME331

Section 11： Dynamics I

Chenglong Fu（付成龙）
Dept. of MEE， SUSTech

Outline

- Review
  - Manipulability Ellipsoid
  - Manipulability Index
- Numerical Solution to Inverse Kinematics
  - Jacobian Inverse Method
  - Jacobian Transpose Method
- The Euler-Lagrange Equations
  - Motivation
  - Holonomic Constraints and Virtual Work
  - D'Alembert's Principle

Review

- **Manipulability:** Manipulability Measure

$$\sum \frac{\omega_i^2}{\sigma_i^2} \leq 1$$

The volume of the ellipsoid is given by

$$\text{volume} = K\sigma_1\sigma_2\cdots\sigma_m$$

in which K is a constant that depends only on the dimension of the ellipsoid.

The **manipulability measure** (可操作性度量) is given by

$$\mu = \sigma_1\sigma_2\cdots\sigma_m$$

Now, consider the special case when the robot is not redundant, that is, $J \in \mathbb{R}^{m\times m}$. Recall that the determinant of a product is equal to the product of the determinants, and that a matrix and its transpose have the same determinant. Thus, we have

$$\det JJ^T = \lambda_1^2\lambda_2^2\cdots\lambda_m^2$$

In which $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ are the eigenvalues of J. This leads to

$$\mu = \sigma_1\sigma_2\cdots\sigma_m = \sqrt{\det JJ^T} = |\lambda_1\lambda_2\cdots\lambda_m| = |\det J|$$

Review

- **Manipulability:** Manipulability Measure

$$\mu = \sqrt{\det JJ^T} = |\lambda_1 \lambda_2 \cdots \lambda_m| = |\det J|$$

The manipulability μ has the following properties.

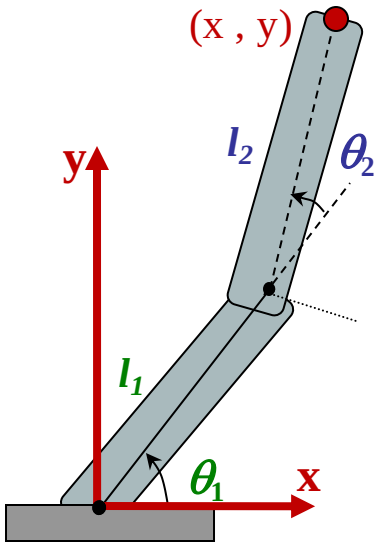
- In general, $\mu = 0$ holds if and only if rank(J) < m, that is, when J is not full rank.
- Suppose that there is some error in the measured velocity $\Delta\xi$. We can bound the corresponding error in the computed joint velocity $\Delta\dot{q}$ by

$$(\sigma_1)^{-1} \leq \frac{\|\Delta\dot{q}\|}{\|\Delta\xi\|} \leq (\sigma_m)^{-1}$$

Example (Two-link Planar Arm). Consider the two-link planar arm and the task of positioning in the plane. The Jacobian is given by

$$J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$

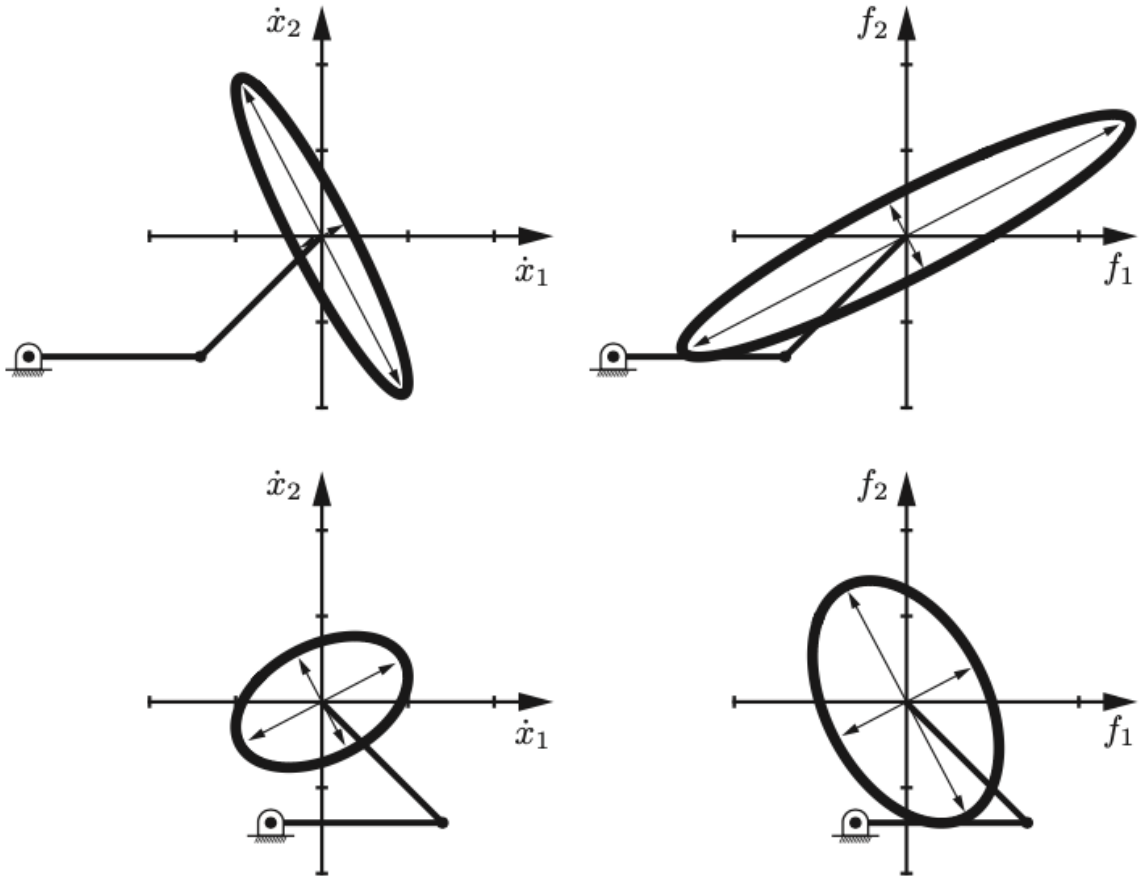
$$\mu = |\det(J)| = l_1 l_2 |\sin\theta_2|$$



Review

- **Manipulability:** Manipulability Measure

  manipulability ellipsoid v.s. force ellipsoid



If it is easy to generate a tip velocity in a given direction then it is difficult to generate a force in that same direction, and vice versa.

Review

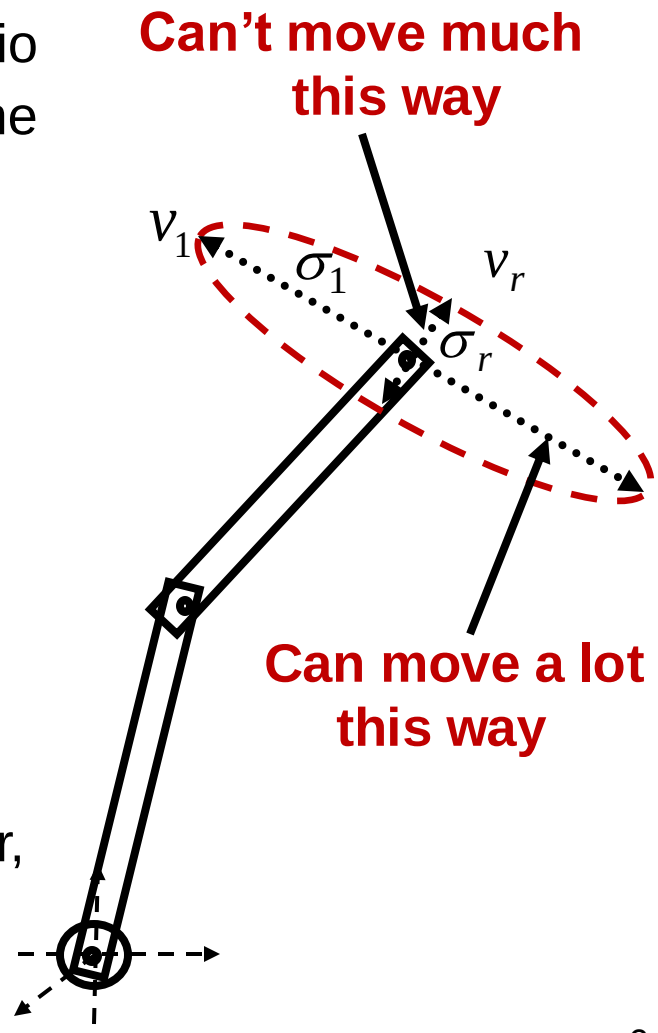
- **Manipulability:** Condition Number & Dexterity Index

The **condition number** is defined as the ratio between the largest singular value and the minimum singular value of the Jacobian matrix J :

$$\kappa \equiv \frac{\sigma_{\max}}{\sigma_{\min}}$$

- $1 \leq \kappa \leq \infty$
- When κ is low (i.e., close to 1) then the manipulability ellipsoid is nearly spherical or isotropic, meaning that it is equally easy to move in any direction. This situation is generally desirable.
- As the robot approaches a singularity, however, κ goes to infinity.

The **dexterity index** is defined as $dex \equiv \frac{\sigma_{\min}}{\sigma_{\max}}$.



Numerical Solution to Inverse Kinematics

- **Problem Definition**

Let $x^d \in \mathbb{R}^m$ be a vector of Cartesian coordinates. For example, x^d could represent the wrist center point ($m = 3$) or the end-effector position and orientation ($m = 6$). The forward kinematics for an n-link manipulator, in this case, is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. If we set

$$G(q) = x^d - f(q)$$

then a solution to the inverse kinematics is a configuration q^d satisfying $G(q^d) = x^d - f(q^d) = 0$.

Below we will give details of the most common algorithms to iteratively solve for q^d given x^d.

1) The first based on the **Jacobian inverse**, which is similar to the Newton–Raphson method for root finding.
2) The second is based on the **Jacobian transpose** and is derived as a gradient search algorithm.

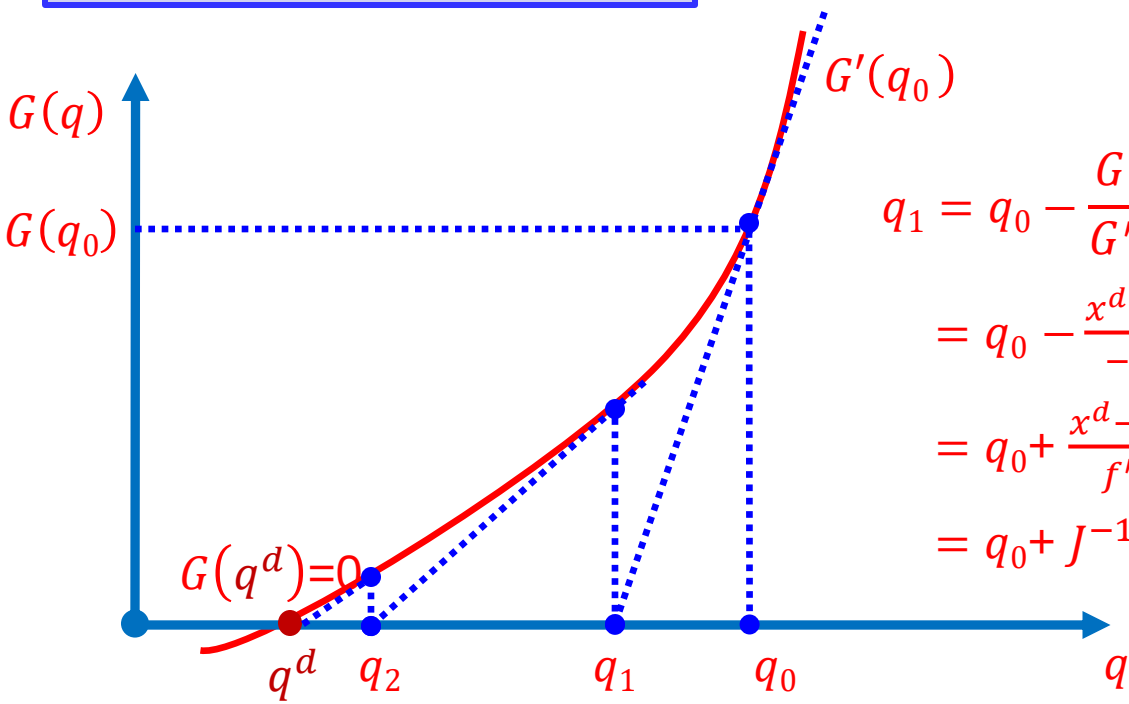
Numerical Solution to Inverse Kinematics

1) Jacobian Inverse Method

The forward kinematics is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

If we set $G(q) = x^d - f(q)$, then a solution to the inverse kinematics is a configuration q^d satisfying $G(q^d) = x^d - f(q^d) = 0$.

Newton–Raphson method



$$q_1 = q_0 - \frac{G(q_0)}{G'(q_0)}$$

$$= q_0 - \frac{x^d - f(q_0)}{-f'(q_0)}$$

$$= q_0 + \frac{x^d - f(q_0)}{f'(q_0)}$$

$$= q_0 + J^{-1}(q_0)(x^d - f(q_0))$$

Numerical Solution to Inverse Kinematics

1) Jacobian Inverse Method

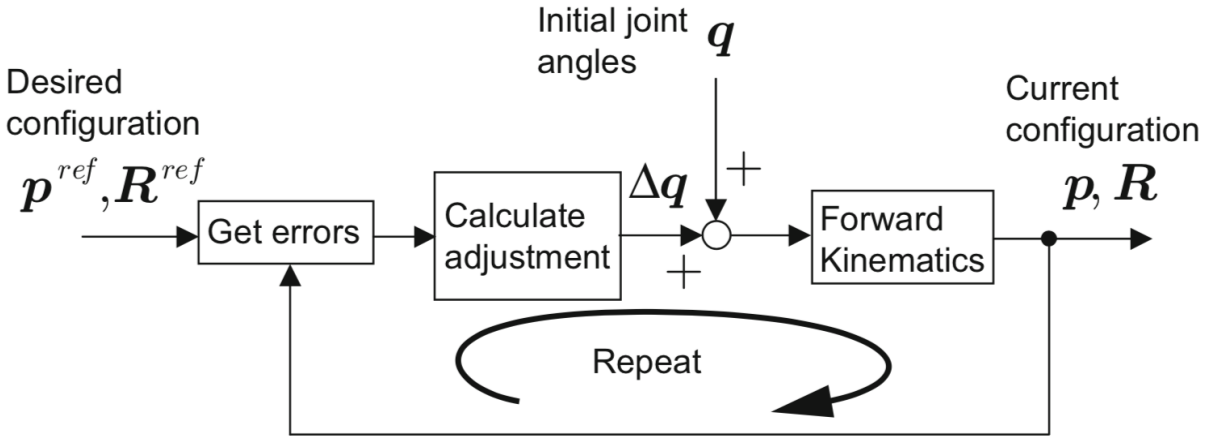
$$q_1 = q_0 + J^{-1}(q_0)(x^d - f(q_0))$$

To find a solution for q^d, we begin with an initial guess, q_0, and form a sequence of successive estimates, q_0, q_1, q_2, . . . , as

$$q_k = q_{k-1} + \alpha_k J^{-1}(q_{k-1})\left(x^d - f(q_{k-1})\right), \qquad k = 1,2,\cdots$$

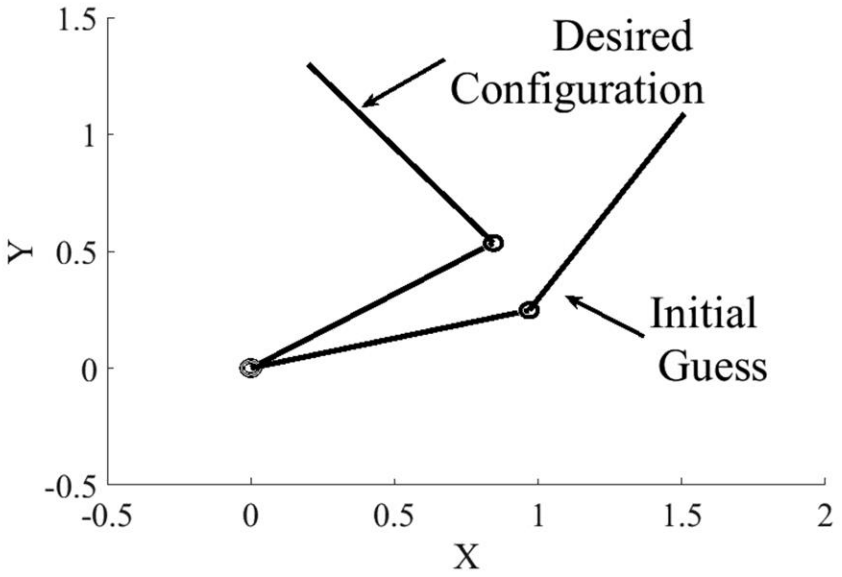
Note that a step size, $\alpha_k > 0$, which can be adjusted to aid convergence.

The step size α_k may be chosen as a constant or as a function of k, a scalar or as a diagonal matrix, the last in order to scale each component of the configuration separately.



Numerical Solution to Inverse Kinematics

1) Jacobian Inverse Method: Example



Iteration	θ_1	θ_2
1	−0.33284	2.6711
2	0.80552	2.1025
3	0.46906	1.9316
4	0.53554	1.7697
5	0.55729	1.7227
6	0.56308	1.7104
7	0.56455	1.7073
8	0.56492	1.7065
9	0.56501	1.7063
10	0.56503	1.7062

Desired end-effector coordinates are $x^d = (0.2, 1.3)$. The joint variables corresponding to x^d are $\theta_1 = 0.5650$, $\theta_2 = 1.7062$. The initial guess is $\theta_1 = 0.25$, $\theta_2 = 0.75$. The step size α was chosen as 0.75.

The algorithm converges to within 10^{-4} of the exact solution after 10 iterations with the given parameters.

Numerical Solution to Inverse Kinematics

1) Jacobian Inverse Method

If the Jacobian is not square or not invertible, then one may use the pseudoinverse J^+ in place of J^{-1}. For $m \leq n$, we defined the right pseudoinverse as $J^+ = J^T(JJ^T)^{-1}$.

In this case, we can define the update rule for q_k as

$$q_k = q_{k-1} + \alpha_k J^+(q_{k-1})\left(x^d - f(q_{k-1})\right), \qquad k = 1,2,\cdots$$

Remark:

1) Only local convergence can be expected.
2) Since there are generally multiple solutions to the inverse kinematics, the particular configuration that results from running the algorithm is dependent on the initial guess.

Numerical Solution to Inverse Kinematics

2) Jacobian Transpose Method

To find a solution q^d satisfying $f(q^d) = x^d$, we define an optimization problem

$$\min_q F(q) = \min_q \frac{1}{2}(f(q) - x^d)^T (f(q) - x^d)$$

The gradient of the above cost function $F(q)$ is given by

$$\nabla F(q) = J^T(q)(f(q) - x^d)$$

A gradient descent algorithm to minimize $F(q)$ is then

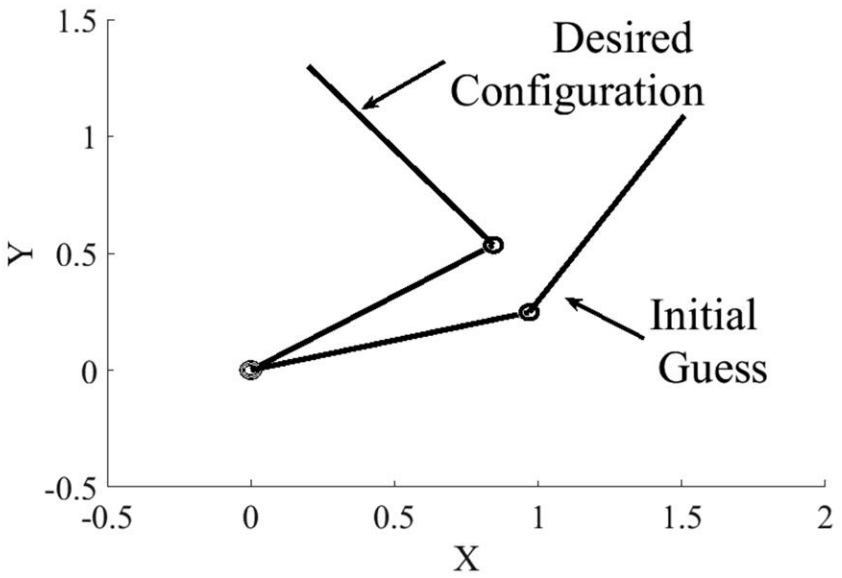
$$q_k = q_{k-1} - \alpha_k \nabla F(q_{k-1}) = q_{k-1} + \alpha_k J^T(q_{k-1})\left(x^d - f(q_{k-1})\right)$$

where, again, $\alpha_k > 0$ is the step size.

Remark: Jacobian transpose is easier to compute than the Jacobian inverse and does not suffer from configuration singularities. However, the convergence, in terms of number of iterations, may be slower with this method.

Numerical Solution to Inverse Kinematics

2) Jacobian Transpose Method: Example

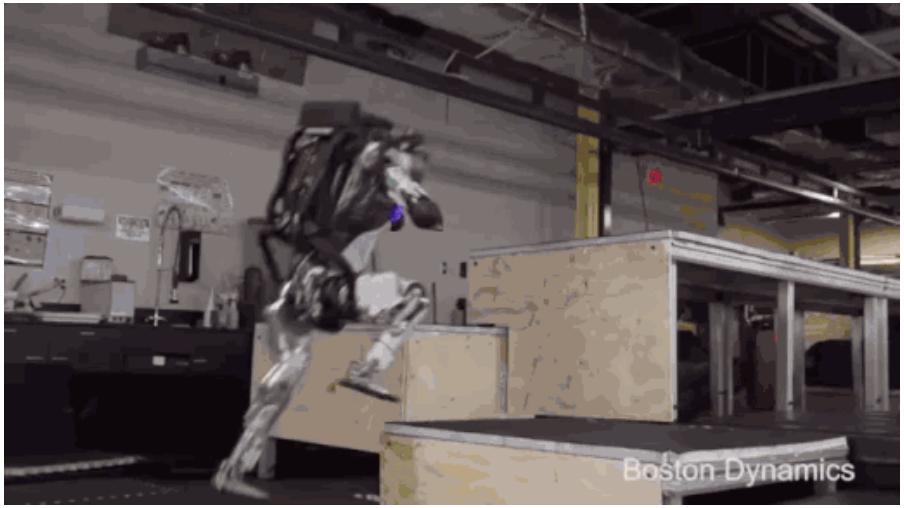


Iteration	θ_1	θ_2
1	1.8362	1.3412
2	0.4667	1.1025
3	1.1215	1.6233
4	0.45264	1.415
5	0.83519	1.7273
26	0.56522	1.7063
27	0.56492	1.7061
28	0.56514	1.7063
29	0.56498	1.7062
30	0.5650	1.7062

Desired end-effector coordinates are $x^d = (0.2, 1.3)$. The joint variables corresponding to x^d are $\theta_1 = 0.5650$, $\theta_2 = 1.7062$. The initial guess is $\theta_1 = 0.25$, $\theta_2 = 0.75$. The step size α was chosen as 0.75.

The algorithm converges to within 10^{-4} of the exact solution after 30 iterations with the given parameters.

Boston Dynamics



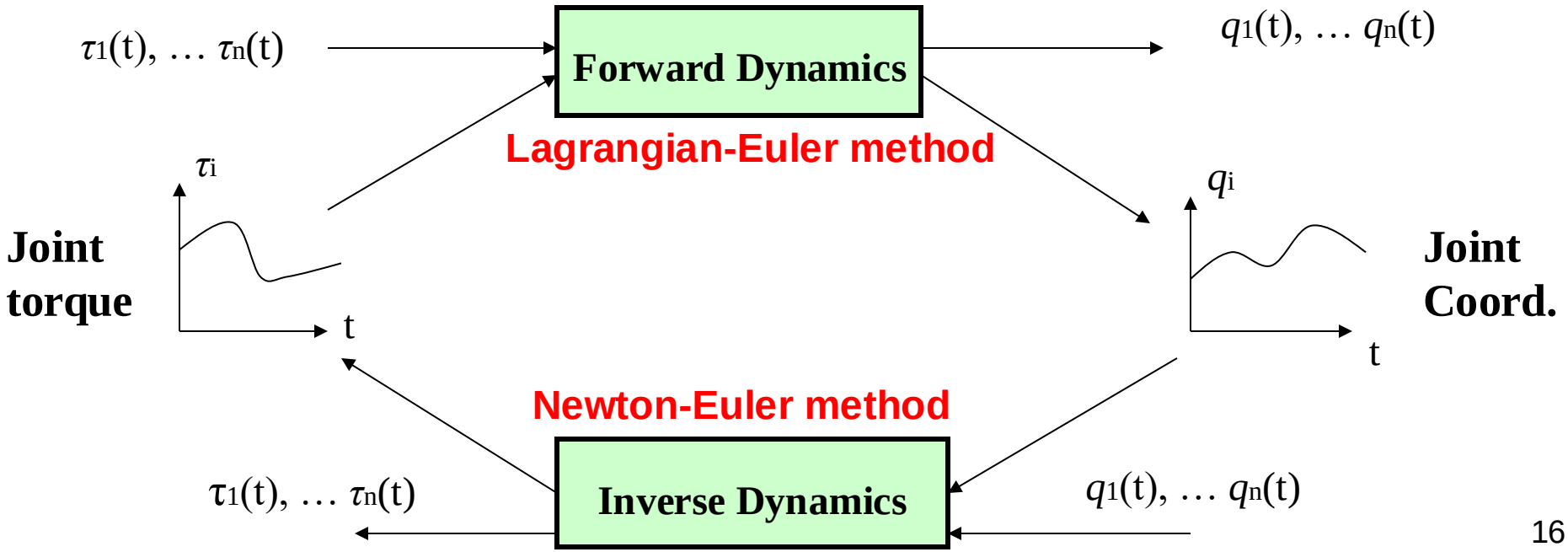
Atlas' controller is what's known as a *model-predictive controller* because it uses **a model of the robot's dynamics** to predict how its motion will evolve into the future. The controller works by solving an optimization that computes the best thing to do *right now* to produce the best possible motion over time.

Dynamics

Dynamics

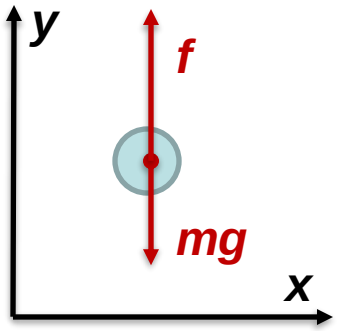
- **Mathematical equations describing the dynamic behavior of the robot**
  - For physical simulation
  - Design of suitable controller
  - Evaluation of robot structure

Force ⟺ **Motion**



The Euler–Lagrange Equations

- From **newton's second law** to **Lagrange**



$$m\ddot{y} = f - mg$$

$$m\ddot{y} = \frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}\left(\frac{1}{2}m\dot{y}^2\right) = \frac{d}{dt}\frac{\partial K}{\partial \dot{y}}$$

$$mg = \frac{\partial}{\partial y}(mgy) = \frac{\partial P}{\partial y}$$

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy$$

$$\frac{\partial L}{\partial \dot{y}} = \frac{\partial K}{\partial \dot{y}} \qquad \frac{\partial L}{\partial y} = -\frac{\partial P}{\partial y}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f$$

The Euler–Lagrange Equations

- **Lagrange-Euler Formulation**

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i$$

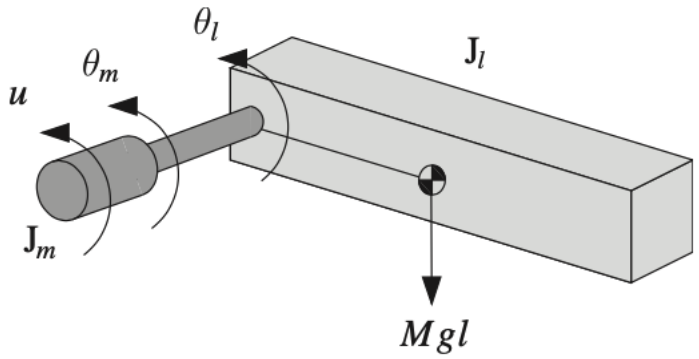
  - Lagrange function is defined

$$L = K - P$$

    - K : Total kinetic energy of robot
    - P : Total potential energy of robot
    - q_i: Joint variable of *i*-th joint
    - $\dot{q}_i$: first time derivative of q_i
    - τ_i: Generalized force associated with q_i

The Euler–Lagrange Equations

Example 1: Single-Link Manipulator



Consider the single-link robot arm, consisting of a rigid link coupled through a gear train to a DC motor. Let θ_l and θ_m denote the angles of the link and motor shaft, respectively.

Then, $\theta_m = r\theta_l$ where $r:1$ is the gear ratio.

If we choose as generalized coordinate $q = \theta_l$, the kinetic energy of the system is given by

$$K = \frac{1}{2}J_m\dot{\theta}_m^2 + \frac{1}{2}J_l\dot{\theta}_l^2 = \frac{1}{2}(r^2J_m + J_l)\dot{q}^2$$

The potential energy is given as

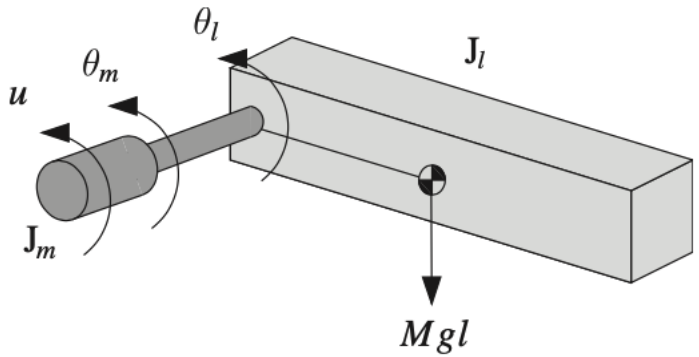
$$P = Mgl(1 - \cos q)$$

Defining $I = r^2J_m + J_l$, the Lagrangian $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2}I\dot{q}^2 - Mgl(1 - \cos q)$$

The Euler–Lagrange Equations

Example 1: Single-Link Manipulator



$$\mathcal{L} = \frac{1}{2} I\dot{q}^2 - Mgl(1 - \cos q)$$

Substituting this expression into the equation $\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k$ yields the equation of motion

$$I\ddot{q} + Mgl \sin q = \tau_l$$

The generalized force τ_l represents those external forces and torques that are not derivable from a potential function. For this example, τ_l consists of

1) the input motor torque $u = r\tau_m$;

2) reflected to the link, and damping torques $B_m\dot{\theta}_m$ and $B_l\dot{\theta}_l$.

Reflecting the motor damping to the link yields

$$\tau_l = u - B\dot{q}$$

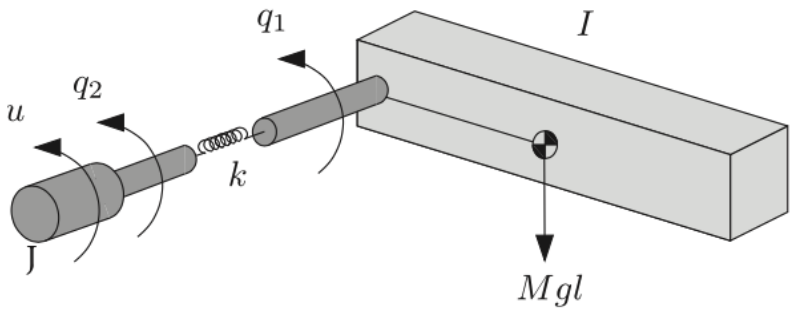
where $B = rB_m + B_l$.

Therefore, the complete expression for the dynamics of this system is

$$I\ddot{q} + B\dot{q} + Mgl \sin q = u$$

The Euler–Lagrange Equations

Example 2: Single-Link Manipulator with Elastic Joint



Consider a single-link manipulator including the transmission flexibility.

Let θ_l and θ_m denote the angles of the link and motor shaft, respectively.

In this case the motor angle $q_1 = \theta_l$ and the link angle $q_2 = \theta_m$ are independent variables and so the system possesses two degrees of freedom.

The kinetic energy of this system is

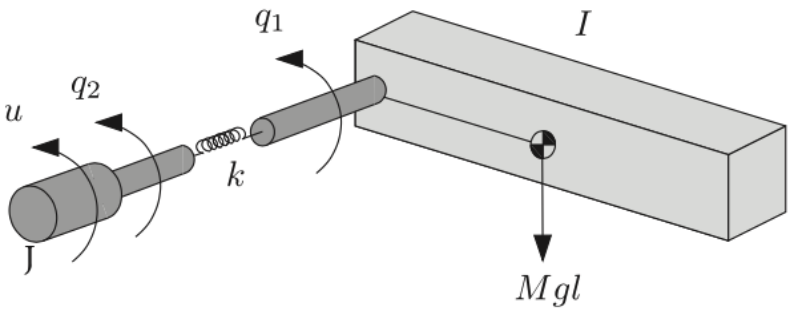
$$K = \frac{1}{2} J_l \dot{q}_1^2 + \frac{1}{2} J_m \dot{q}_2^2$$

The potential energy includes the spring potential energy in addition to the gravitational potential energy,

$$P = Mgl(1 - \cos q_1) + \frac{1}{2} k (q_1 - q_2)^2$$

The Euler–Lagrange Equations

Example 2: Single-Link Manipulator with Elastic Joint



The kinetic energy of this system is

$$K = \frac{1}{2} J_l \dot{q}_1^2 + \frac{1}{2} J_m \dot{q}_2^2$$

The potential energy of this system is

$$P = Mgl(1 - \cos q_1) + \frac{1}{2} k (q_1 - q_2)^2$$

Forming the Lagrangian $\mathcal{L} = K - P$ and ignoring damping for simplicity, the equations of motion are found from the Euler-Lagrange equations as

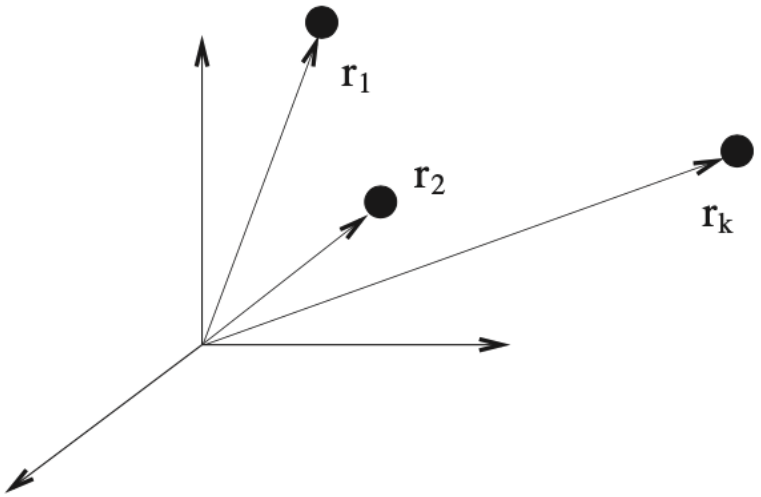
$$J_l \ddot{q}_1 + Mgl \sin(q_1) + k(q_1 - q_2) = 0$$

$$J_m \ddot{q}_2 + k(q_2 - q_1) = u$$

The details are left as a **homework**.

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work



Now, consider a system of k particles with corresponding position vectors r_1,..., r_k.

If the motion of the particles is constrained in some fashion, then one must take into account not only the externally applied forces, but also the so-called **constraint forces**, that is, the forces needed to make the constraints hold.

Suppose the system consists of two particles joined by a massless rigid wire of length l. Then the two coordinates r_1 and r_2 must satisfy the constraint $\|r_1 - r_2\| = l$ or $(r_1 - r_2)^T(r_1 - r_2) = l^2$.

If one applies some external forces to each particle, then the particles experience not only these external forces but also the force exerted by the wire, which is along the direction $r_2 - r_1$ and of appropriate magnitude.

In order to analyze the motion of the two particles, we can search for a method of analysis that does not require us to know the constraint force.

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

A constraint on the k coordinates r_1 ,..., r_k is called **holonomic** (完整的) if it is an equality constraint of the form

$$g_i(r_1, \ldots, r_k) = 0, \quad i = 1, \cdots, l$$

The constraint $\|r_1 - r_2\| = l$ imposed by connecting two particles by a massless rigid wire is an example of a holonomic constraint. By differentiating the above equation we have an expression of the form

$$\sum_{j=1}^{k} \frac{\partial g_i}{\partial r_j} dr_j = 0$$

A constraint of the form $\sum_{j=1}^{k} \omega_j dr_j = 0$ is called **nonholonomic** (非完整) if it cannot be integrated to an equality constraint of the form $g_i(r_1, \ldots, r_k) = 0$.

If a system is subjected to l holonomic constraints, then one can think in terms of the constrained system having l fewer degrees of freedom than the unconstrained system.

In this case, it may be possible to express the coordinates of the k particles in terms of n **generalized coordinates** $q_1, \ldots, q_n$.

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

In other words, we assume that the coordinates of the various particles, subjected to the set of constraints, can be expressed in the form

$$r_i = r_i(q_1, \dots, q_n), \quad i = 1, \cdots, k$$

where $q_1, \dots, q_n$ are all independent.

- The idea of generalized coordinates can be used even when there are infinitely many particles.
- A physical rigid object contains an infinity of particles; but since the distance between each pair of particles is fixed, six coordinates are sufficient to specify completely the coordinates of any particle in the bar.
- One could use three position coordinates to specify the location of the CoM of the body, and three Euler angles to specify the orientation of the body. Typically, generalized coordinates are positions, angles, etc.
- We chose to denote the joint variables by the symbols $q_1, \dots, q_n$ precisely because these joint variables form a set of generalized coordinates for an n-link robot manipulator.

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

One can now speak of **virtual displacements**, which are any set of infinitesimal displacements, $\delta r_1, \ldots, \delta r_k$, that are consistent with the constraints.

For example, consider once again the constraint $\|r_1 - r_2\| = l$ and suppose r_1, r_2 are perturbed to $r_1 + \delta r_1$, $r_2 + \delta r_2$, respectively.

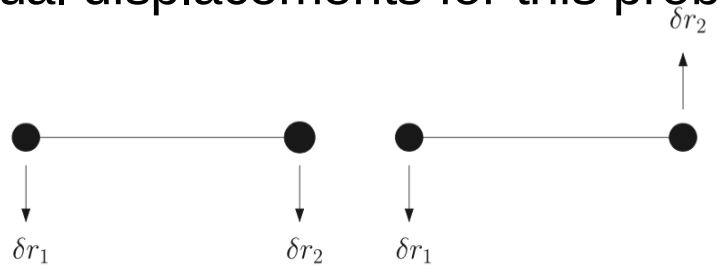
Then, in order that the perturbed coordinates continue to satisfy the constraint, the length of the bar must not change and so we must have

$$(r_1 + \delta r_1 - r_2 - \delta r_2)^T (r_1 + \delta r_1 - r_2 - \delta r_2) = l^2.$$

If we neglect quadratic terms in δr_1, δr_2 we obtain after some algebra

$$(r_1 - r_2)^T (\delta r_1 - \delta r_2) = 0.$$

Any pair of infinitesimal vectors δr_1,δr_2 that satisfy the above equation constitutes a set of virtual displacements for this problem.



The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

The reason for using generalized coordinates is to avoid dealing with complicated constraint relationships.

The set of all virtual displacements is precisely

$$\delta r_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \delta q_j \,, \qquad i = 1, \cdots, k$$

where the virtual displacements $\delta q_1, \cdots, \delta q_n$ of the generalized coordinates are unconstrained (that is what makes them generalized coordinates).

Next, we begin a discussion of constrained systems in equilibrium.

Suppose each particle is in equilibrium. Then the net force on each particle is zero, which in turn implies that the work done by each set of virtual displacements is zero. Hence, the sum of the work done by any set of virtual displacements is also zero; that is,

$$\sum_{i=1}^{k} F_i^T \delta r_i = 0$$

where F_i is the total force on particle i.

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

As mentioned earlier, the force F_i is the sum of two quantities, namely

(i) the externally applied force f_i ;

(ii) the constraint force f_i^a .

Now, suppose that the total work done by the constraint forces corresponding to any set of virtual displacements is zero, that is,

$$\sum_{i=1}^{k} f_i^{a^T} \delta r_i = 0$$

This will be true whenever the constraint force between a pair of particles is directed along the radial vector connecting the two particles. Therefore,

$$\sum_{i=1}^{k} f_i^T \delta r_i = 0$$

The beauty of this equation is that it does not involve the unknown constraint forces, but only the known external forces. This equation expresses the **principle of virtual work**, which can be stated in words as: *The work done by external forces corresponding to any set of virtual displacements is zero.*

The Euler–Lagrange Equations

Holonomic Constraints and Virtual Work

Principle of virtual work: *The work done by external forces corresponding to any set of virtual displacements is zero.*

The principle is not universally applicable; it requires the constraint forces do no work. Thus, if the principle of virtual work applies, one can analyze the dynamics of a system *without* having to evaluate the constraint forces.

It is easy to verify that the principle of virtual work applies whenever the constraint force between a pair of particles acts along the vector connecting the position coordinates of the two particles.

Consider a single constraint of the form. The force exerted on the first particle by the wire must be of the form

$$f_1^a = c(r_1 - r_2)$$

The force exerted on the second particle by the wire must be just the negative of the above, that is,

$$f_2^a = -c(r_1 - r_2)$$

Now, the work done by the constraint forces corresponding to a set of virtual displacements is

$$f_1^{aT}\delta r_1 + f_2^{aT}\delta r_2 = c(r_1 - r_2)^T(\delta r_1 - \delta r_2) = 0.$$

The Euler–Lagrange Equations

D'Alembert's Principle

Principle of virtual work:

$$\sum_{i=1}^{k} f_i^T \delta r_i = 0 \quad (\text{for static system})$$

where the virtual displacements δr_i are not independent. We cannot conclude from this equation that each coefficient f_i individually equals zero.

In order to apply such reasoning, we must transform to generalized coordinates. Before doing this, we consider systems that are not necessarily in equilibrium.

For such systems, **D'Alembert's principle** states that, if one introduces a fictitious additional force $-\dot{p}_i$ on each particle, where p_i is the momentum of particle i, then each particle will be in equilibrium.

By replacing F_i by $F_i - \dot{p}_i$ then the resulting equation is valid for dynamic systems. One can then remove the constraint forces as before using the principle of virtual work. This results in the equation

$$\sum_{i=1}^{k} f_i^T \delta r_i - \sum_{i=1}^{k} \dot{p}_i^T \delta r_i = 0 \quad (\text{for dynamic system})$$

The Euler–Lagrange Equations

D'Alembert's Principle

$$\sum_{i=1}^{k} f_i^T \delta r_i - \sum_{i=1}^{k} \dot{p}_i^T \delta r_i = 0 \quad \text{(for dynamic system)}$$

The remainder of this derivation is aimed at expressing the above equation in terms of the generalized coordinates q_i , which are independent.

Recall $\delta r_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \delta q_j$. The virtual work done by the forces f_i is given by

$$\sum_{i=1}^{k} f_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} f_i^T \frac{\partial r_i}{\partial q_j} \delta q_j = \sum_{j=1}^{n} \psi_j \delta q_j$$

where

$$\psi_j = \sum_{i=1}^{k} f_i^T \frac{\partial r_i}{\partial q_j}$$

is called the j^{th} **generalized force**.

Note that ψ_j need not have dimensions of force, just as q_j need not have dimensions of length; however, $\psi_j \delta q_j$ must always have dimensions of work.

The Euler–Lagrange Equations

D'Alembert's Principle

$$\sum_{i=1}^{k} f_i^T \delta r_i - \sum_{i=1}^{k} \dot{p}_i^T \delta r_i = 0 \quad \text{(for dynamic system)}$$

Now, let us study the **second summation**. Since $p_i = m_i \dot{r}_i$, it follows that

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{i=1}^{k} m_i \ddot{r}_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \delta q_j$$

Next, using the product rule of differentiation, we have

$$\frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] = m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} + m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right]$$

Rearranging the above and summing over all $i = 1, \cdots, n$ yields

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] - m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] \right\}$$

Differentiating equation $r_i = r_i(q_1, \ldots, q_n)$ using the chain rule gives

$$v_i = \dot{r}_i = \sum_{j=1}^{n} \frac{\partial r_i}{\partial q_j} \dot{q}_j \quad \Rightarrow \quad \frac{\partial v_i}{\partial \dot{q}_j} = \frac{\partial r_i}{\partial q_j}$$

The Euler–Lagrange Equations

D'Alembert's Principle

Now, let us study the **second summation**. Since $p_i = m_i \dot{r}_i$, it follows that

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \delta q_j$$

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] - m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] \right\}$$

$$\frac{\partial v_i}{\partial \dot{q}_j} = \frac{\partial r_i}{\partial q_j}$$

$$\frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] = \sum_{l=1}^{n} \frac{\partial^2 r_i}{\partial q_j \partial q_l} \dot{q}_l = \frac{\partial}{\partial q_j} \sum_{l=1}^{n} \frac{\partial r_i}{\partial q_l} \dot{q}_l = \frac{\partial v_i}{\partial q_j}$$

Recall $$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i \dot{r}_i^T \frac{\partial r_i}{\partial q_j} \right] - m_i \dot{r}_i^T \frac{d}{dt}\left[\frac{\partial r_i}{\partial q_j} \right] \right\}$$

$$= \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i v_i^T \frac{\partial v_i}{\partial \dot{q}_j} \right] - m_i v_i^T \frac{\partial v_i}{\partial q_j} \right\}$$

The Euler–Lagrange Equations

D'Alembert's Principle

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \sum_{i=1}^{k} \left\{ \frac{d}{dt}\left[m_i v_i^T \frac{\partial v_i}{\partial \dot{q}_j} \right] - m_i v_i^T \frac{\partial v_i}{\partial q_j} \right\}$$

If we define the **kinetic energy** K to be the quantity

$$K = \sum_{i=1}^{k} \frac{1}{2} m_i v_i^T v_i$$

Then the above equation can be compactly expressed as

$$\sum_{i=1}^{k} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} = \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j}$$

Recall the **second summation**:

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{i=1}^{k} \sum_{j=1}^{n} m_i \ddot{r}_i^T \frac{\partial r_i}{\partial q_j} \delta q_j = \sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \right\} \delta q_j$$

The Euler–Lagrange Equations

D'Alembert's Principle

$$\sum_{i=1}^{k} f_i^T \delta r_i - \sum_{i=1}^{k} \dot{p}_i^T \delta r_i = 0 \quad \text{(for dynamic system)}$$

$$\sum_{i=1}^{k} f_i^T \delta r_i = \sum_{j=1}^{n} \psi_j \delta q_j$$

$$\sum_{i=1}^{k} \dot{p}_i^T \delta r_i = \sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} \right\} \delta q_j$$

$$\sum_{j=1}^{n} \left\{ \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} - \psi_j \right\} \delta q_j = 0$$

Now, since the virtual displacements δq_j are independent, we can conclude that each coefficient in the above Equation is zero, that is,

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j, \qquad j = 1, \cdots n$$

The Euler–Lagrange Equations

D'Alembert's Principle

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j, \qquad j = 1, \cdots n$$

If the generalized force ψ_j is the sum of an externally applied generalized force and another one due to a potential field, then a further modification is possible.

Suppose there exist functions τ_j and a potential energy function $P(q)$ such that

$$\psi_j = -\frac{\partial P}{\partial q_j} + \tau_j$$

Then $\frac{d}{dt}\frac{\partial K}{\partial \dot{q}_j} - \frac{\partial K}{\partial q_j} = \psi_j$ can be written in the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_j} - \frac{\partial \mathcal{L}}{\partial q_j} = \tau_j$$

where $\mathcal{L} = K - P$ is the Lagrangian.

We have recovered the **Euler–Lagrange Equation**.

Summary

- **Numerical Solution to Inverse Kinematics**
  - **Jacobian Inverse Method**
  - **Jacobian Transpose Method**
- **The Euler-Lagrange Equations**
  - **Motivation**
  - **Holonomic Constraints and Virtual Work**
  - **D'Alembert's Principle**

Homework 11

Homework 11 is posted at http://bb.sustech.edu.cn

Due date: April 9, 2025

作业要求（Requirements）：

1. 文件格式为以自己姓名学号作业序号命名的pdf文件；

（File name: YourSID_YourName_11.pdf）

2. 作业里也写上自己的姓名和学号。

（Write your name and SID in the homework）