16-642 Fall 2017: Reference Notes for Velocity Kinematics and Statics

George Kantor

25 and 27 September 2017

1 Velocity Kinematics Introduction

In this section we learn how to go back and forth between joint space velocities and task space velocities, $\dot{\Theta} \leftrightarrow \dot{\mathbb{X}}$.

Let n be the number of joints. $\Theta \in \mathbb{R}^n$.

Assume the task space can be represented with an m dimensional vector, $X \in \mathbb{R}^m$. We call this "the simple case" (as opposed to the more difficult case where the task space is $SE(3)$).

So in the simple case, the forward kinematic map can be written as a vector function

$$X = f(\Theta) = \begin{bmatrix} f_1(\theta_1, \theta_2, \ldots, \theta_n) \\ f_2(\theta_1, \theta_2, \ldots, \theta_n) \\ \vdots \\ f_m(\theta_1, \theta_2, \ldots, \theta_n) \end{bmatrix}$$

Then

$$\dot{X} = J(\Theta)\dot{\Theta}$$

where

$$J(\Theta) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_n} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1} & \frac{\partial f_m}{\partial \theta_2} & \cdots & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix}$$

Important Fact (if you only learn one thing about velocity kinematics, learn this: The ith column of the Jacobian is

$$J_i = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_i} \\ \frac{\partial f_2}{\partial \theta_i} \\ \vdots \\ \frac{\partial f_m}{\partial \theta_i} \end{bmatrix}$$

this is the end-effector velocity that is achieved when you move joint i at a speed of 1.

1.1 When the Jacobian is Square

In the case where $m = n$, J is square and will invertible if and only if all columns are linearly independent, which intuitively means that the the end effector can move in m different directions!

And when J^{-1} exists:

$$\dot{\Theta} = J(\Theta)^{-1}\dot{X}$$

1.2 When the Jacobian is Tall and Skinny

What if J is not square? In that case it is obviously not invertible, but there still may a way to solve the inverse problem.

J has m rows and n columns

If $m > n$ J cannot be invertible. This makes sense, in this case the number of task space degrees of freedom is greater than the number of joints, so in general there will not be a solution to the inverse problem. But we may be able to do something:

$$\dot{X} = J\dot{\Theta}$$

$$J^T\dot{X} = J^TJ\dot{\Theta}$$

Note that J^TJ is $n \times n$, so it is square. It might be invertible. If it is then we can write

$$\boxed{\dot{\Theta} = \left(J^TJ\right)^{-1}J^T\dot{X}.}$$

The matrix

$$J_{\text{left}}^{-1} = \left(J^TJ\right)^{-1}J^T$$

is called the **left psuedoinverse** since

$$J_{\text{left}}^{-1}J = \left(J^TJ\right)^{-1}J^TJ = I.$$

So we can sometimes use the boxed equation above to solve the velocity kinematics. That equation will always give an answer, but it might not be the correct answer! Here are the caveats:

1. if a solution exists, then the boxed equation will give the correct answer.
2. if a solution does not exist, the boxed equation will give the wrong answer. You can check to see if the solution exists by substituting the psuedoinverse answer back into the original $\dot{X} = J\dot{\Theta}$ equation to see if that equation holds. In other words, the solution is correct if the following equation is true:

   $$\dot{X} = JJ_{\text{left}}^{-1}\dot{X} = J\left(J^TJ\right)^{-1}J^T\dot{X}.$$

   If a solution does not exist, then the answer you get from the pseudoinverse will be the best answer in the sense the error in task space velocity will be smaller for that choice of $\dot{\Theta}$ than it would be for any other choice.

1.3 When the Jacobian is Short and Fat

If $m < n$ we can "invert" J provided that all of rows are linearly independent (which is the same as saying there are m linearly independent columns). In this case there are more joints than task space degrees of freedom, which (assuming the linear independence condition holds) means there are an infinite number of solutions. The **right pseudoinverse** will give one of the solutions as follows:

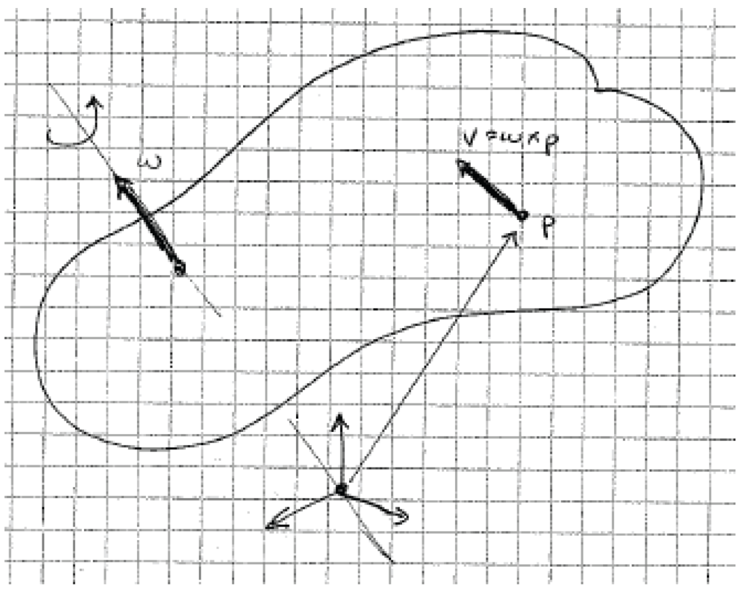
$$\boxed{\dot{\Theta} = \underbrace{J^T \left(JJ^T\right)^{-1}}_{J^{-1}_{\text{right}}} \dot{X}.}$$

Not only will this give a correct solution, it will give the best correct solution in the sense that the solution it gives minimizes the sum of the squares of the joint velocities.

What's Next? For most of the systems we deal with, we don't specify the task space with a vector. Instead, we specify it with a homogeneous matrix. So we can't get the Jacobian doing plain old differentiation.

2 Angular Velocity

Consider a rigid body moving arbitrarily in space. At any instant in time, its change in orientation can be thought of as a rotation about an axis. So we introduce the concept of *angular velocity* vector ω. The direction of ω defines the axis of rotation, and the magnitude of ω defines the rate of rotation.



Some comments:

1. the thing about the axis is true *instantaneously*. It is possible to have complicated motions where the axis (and ω) changes with time. But at any intant, the rotation will always be about a fixed axis.

2. Rigid bodies and frames can have angular velocity. Points can only have linear velocity, not angular velocity.
3. The angular velocity is the same for all frames attached to a rigid body.
4. the location of the axis of rotation is not important to the rotatation rate, only its direction. So ω is a vector (more explicitly, it is a *free* vector).
5. ω can be represented in different frames just like any other vector. (i.e. ω^0 is the angular velocity represented in the zero frame.)
6. the location of the axis of rotation *is* important to the linear velocities of points a rigid body.
7. if the axis of rotation contains the origin, then the linear velocity of a point p on a rigid body spinning with angular velocity ω is

$$v = \omega \times p.$$

3 Review Representing General Velocities

Now we can generally write the velocity of a frame (and a rigid body attached to it) with a vector:

$$\xi = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

We're looking for the Jacobian, i.e., the $6 \times n$ ($n =$ number of joints) $J(\Theta)$ such that

$$\xi_n^0 = J(\Theta)\dot{\Theta}$$

Define $3 \times n$ matrices J_v and J_ω so that

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

Then we can worry about angular and linear velocities independently:

$$v_n^0 = J_v \dot{\Theta}$$
$$\omega_n^0 = J_\omega \dot{\Theta}$$

We can now build J_v and J_ω using the fact that the ith column of the Jacobian is the velocity of the end effector when the velocity of the ith joint is 1 and all of the other joint velocities are zero.

3.1 Angular Velocity

Let's start by figuring out how to build J_ω. We will build J_ω bydeterming the value of each column $J_{\omega i}$, for $i = 1, 2, \ldots, n$.

If the ith joint is prismatic, then the answer is easy because a P joint does not rotate, which means it does not cause any angular velocity at the end effector. In other words,

$$J_{\omega i} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

If the ith joint is revolute things are just a bit trickier. Assuming the frames were assigned using the DH convention, then the axis of the ith joint is z_{i-1}. The R joint rotates about its axis, so when the ith joint is moving at a rate of 1 and all of the other joints are fixed, then the angular velocity of the end effector (and every other part of the manipulator past the ith joint) is a vector of magnitude 1 that points along the z_{i-1} axis. Written in the $(i-1)$th frame, this vector is just

$$\hat{\omega}^{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

But the ith column of the Jacobian is the velocity *represented in the* $\{0\}$ *frame*. Since the forward kinematics have already been solved using DH, we know the transforms between every frame and we can easily convert the free vector $\hat{\omega}^{i-1}$:

$$J_{\omega i} = R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

3.2 Linear Velocity

direct differentiation Note that since the end effector position is a vector, then we can compute J_v with direct differentiation, i.e., do forward kinematics to get

$$T^0_n = \begin{bmatrix} R^0_n & o^0_n \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

then compute J_v as

$$J_v = \frac{\partial o^0_n}{\partial \Theta}$$

constructing the Jacobian: If you don't like big ugly differentiation, we can also construct J_v just like we did angular velocity.

For prismatic joints, the ith joint will impart linear motion in the direction of the $i-1$th z axis, i.e.,

$$J_{vi} = R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

For revolute joints, the ith joint will impart linear motion due the the angular velocity applied at axis $i-1$ using the $v = \omega \times p$ rule. In the $i-1$ frame, this is represented as

$$v^{i-1}_n = \omega^{i-1}_{i-1} \times o^{i-1}_n$$

Representing it in the 0 frame and using the DH convention gives the ith column of J_v:

$$J_{vi} = R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(o^0_n - o^0_{i-1}\right).$$

4 Manipulator Statics

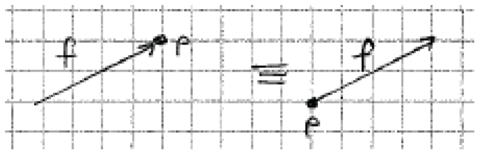
We're just about to finish kinematics, which was described as "the study of motion without regard to the forces that cause it". Next we want to study the motion of a robotic manipulator *with* regard to the forces that cause it. This area is called **dynamics**. So the first thing we need to talk about is...

Forces "push" and "pull".

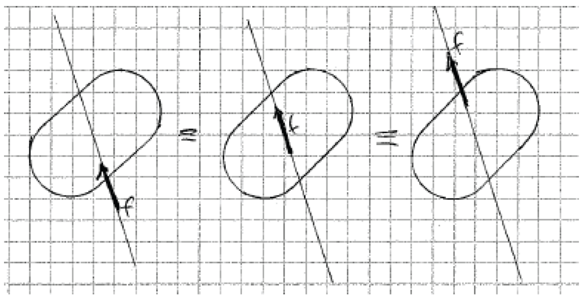
We represent forces with vectors. The direction of the vector describes the direction of the push, and magnitude of the vector describes how hard the push is.

Forces are usually denoted with an f (or F). We use the same notation conventions we use with any other vector, i.e. f^0 is the force expressed in the 0 frame.

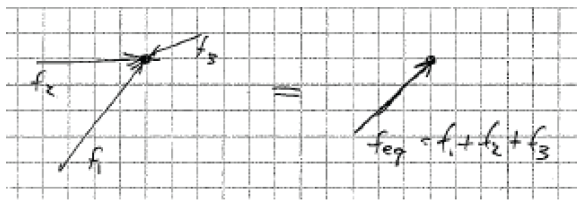
A force is applied at a point. When we draw the vectors, it doesn't matter which end of the vector we put at the point.



When a force is applied to a rigid body, it is equivalent to any other force with the same direction and magnitude applied at any point along the "line of force". By "equivalent" we mean that motion produced will be identical.



Multiple forces applied at the same point can be combined and equivalently expressed as a single force. Assuming that the force vectors are expressed in the same coordinate frame, this combination can be achieved by simply adding the vectors.

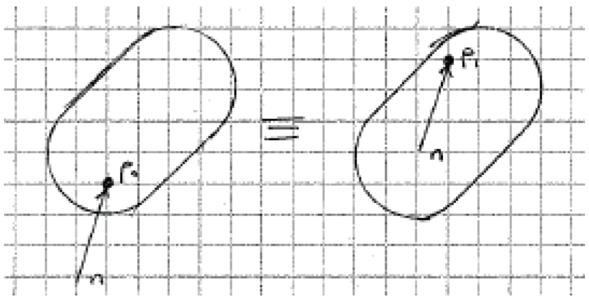


Moments are like "twisty" versions of forces; they cause rigid bodies to rotate.

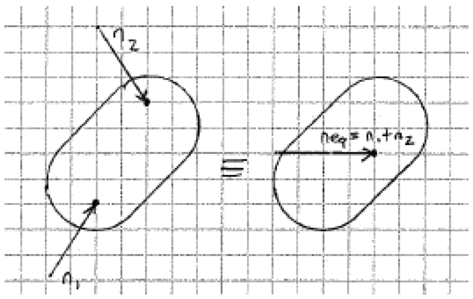
We represent moments with vectors that are analogous to the vectors we used earlier for angular velocity. The direction defines an axis about which the moment is applied, the magnitude is, well, the magnitude.

Moments are usually denoted by n.

Moments are applied to a rigid body. A moment applied to a point p_0 on a rigid body is equivalent to a moment with the same direction and magnitude applied to any other point p_1 on the body.



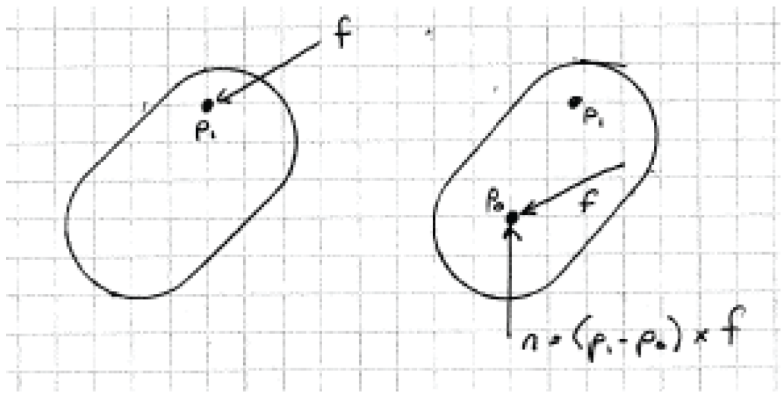
Moments applied to the same body can be combined into a single equivalent moment. Like forces, this combination is achieved by adding the vectors.



Of course, it doesn't matter where on the body the equivalent moment is applied because moments can be moved around on the body.

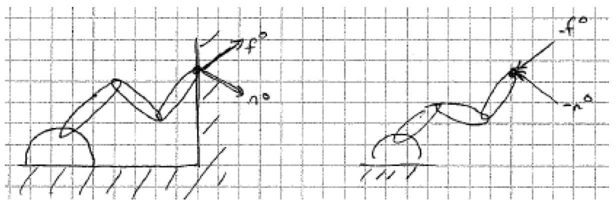
Moments are also sometimes called torques, though the word torque more accurately describes the magnitude of a moment vector.

A force f applied point p_1 is equivalent to a force of the same magnitude and direction applied at the point p_0 together with a moment $n = (p_1 - p_0) \times f$.



4.1 Static Forces applied by a Manipulator

Consider the case where a robot is applying force but not moving. For example, it could be pushing against a wall or holding a mass aloft in a gravity field. We call this the static case since nothing is moving.



Two equivalent ways of thinking about this:

one: the end effector is held still by some unmovable object, what are the joint torques/forces necessary in order for the end effector to exert a force/moment pair of (f^0, n^0)?

two: a force/moment pair of $(-f^0, -n^0)$ is applied to the end effector. What are the joint torques/forces necessary to keep the end effector from moving?

I like to think in terms of two.

Now we can state the relationship between joint torques and the end effector forces (f^0, n^0). Let τ be the vector of joint torques, i.e.,

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix},$$

where the τ_i is the joint at torque i, and let

$$F = \begin{bmatrix} f^0 \\ n^0 \end{bmatrix}$$

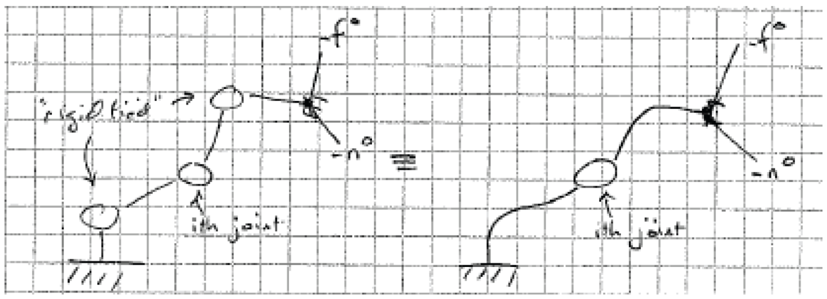
Then the relationship between the τ and F is:

$$\tau = J(\Theta)^T F.$$

4.2 Derivation

For this class, that is all you really need to know. But information about the derivation of this fact is included here for reference:

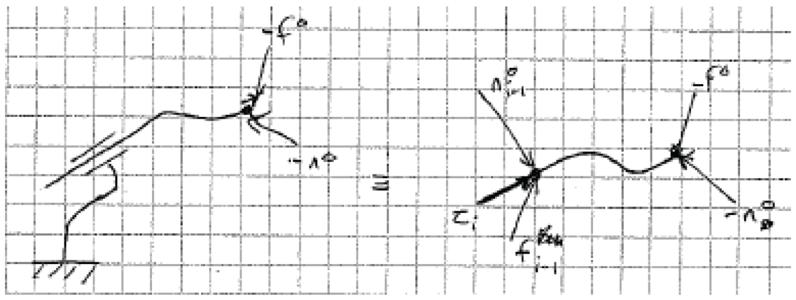
Lets boil it down even further and think about one joint at a time. Consider the ith joint (remember, ith joint is at the frame $i - 1$). Assume all of the other joints are “rigidified”, what force/torque must the ith joint apply in order to hold the manipulator static?



The "bottom" will stay static since it is bolted to the ground. In order for the top to be static, we must satisfy

1. the sum of forces applied to the body must be 0
2. the sum of moments (including equivalent moments generated by applied forces) about any one fixed point must be 0

case 1: joint i is prismatic



Moment balance is achieved automatically from joint constraints.

Force balance gives

$$f^0_{i-1} - f^0 = 0$$

Rearranging and applying coordinate transformations

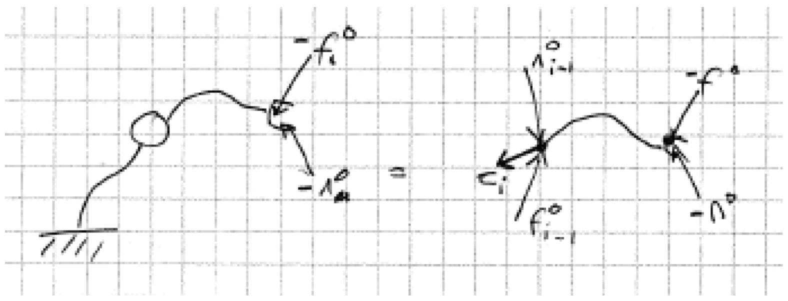
$$f^{i-1}_{i-1} = (R^0_{i-1})^T f^0$$

and since we only need to worry about the z direction (other directions are taken care of by joint constraints) we get

$$\tau_i = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} (R^0_{i-1})^T f^0 = \left(R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)^T f^0$$

Note that we use the letter τ even though it is force.

case 2: joint i is revolute



Force balance is achieved automatically by the joint constraint forces.

Moment balance says

$$n_{i-1}^0 - n^0 - (o_n^0 - o_{i-1}^0) \times f^0 = 0$$

Rearranging and applying coordinate transformations

$$n_{i-1}^{i-1} = (R_{i-1}^0)^T \left(n^0 + (o_n^0 - o_{i-1}^0) \times f^0\right)$$

Moment balance about x_{i-1}^{i-1} and y_{i-1}^{i-1} are also taken care of by joint constraints. The z component must be taken care of with the joint torque:

$$\tau_i = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left(R_{i-1}^0\right)^T \left(n^0 + (o_n^0 - o_{i-1}^0) \times f^0\right) = \left(R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right)^T \left(n^0 + (o_n^0 - o_{i-1}^0) \times f^0\right)$$

One more step seemingly unmotivated step (the reason for taking it will become clearer next time...). Using the scalar triple product ($c^T(a \times b) = a^T(b \times c) = b^T(c \times a)$) and the fact that $a^T b = b^T a$) we get

$$\tau_i = \left(R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right)^T n^0 + \left(R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (o_n^0 - o_{i-1}^0)\right)^T f^0$$

So we have derived the relationships between joint torques/forces and a the force/moment pair applied by the end effector. Those relationships are linear, which means we can write the set of equations as a matrix multiplication:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} = A \begin{bmatrix} f^0 \\ n^0 \end{bmatrix}$$

Where A is some $n \times 6$ matrix.

Recall that if the ith joint is prismatic, then

$$\tau_i = \left(R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right)^T f^0 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} n^0$$

so the ith row of A is

$$\begin{bmatrix} \left(R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right)^T & 0 & 0 & 0 \end{bmatrix}$$

which, if you remember, just the transpose of the ith column of the Jacobian.

Likewise, if the joint is revolute, then

$$\tau_i = \left(R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)^T n^0 + \left(R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (o^0_n - o^0_{i-1}) \right)^T f^0$$

so the ith row of A is

$$\left[\left(R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (o^0_n - o^0_{i-1}) \right)^T \quad \left(R^0_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)^T \right]$$

which again is just transpose of the ith column of the Jacobian. So $A = J(\Theta)^T$, which give us a very important relationship between joint forces and end effector forces in the static case:

$$\boxed{\tau = J(\Theta)^T F = J(\Theta)^T \begin{bmatrix} f^0 \\ n^0 \end{bmatrix}}$$

4.3 Virtual Work

There is another nice way to derive this relationship that uses the principle of virtual work. Normally work is force times distance moved. Virtual work is force times an infinitesimally small displacement δX. The virtual work done by the force at the end effector is

$$W_e = F^T \delta X$$

and the work done by the actuators is

$$W_j = \tau^T \delta\Theta$$

J give the relationship between infinitessimal motions in joint space and task space, it is the same as for velocities:

$$\delta X = J \delta\Theta$$

Noting that the work done by the joints must be equal to the work done at the end effector and substituting for δX yields

$$F^T J \delta\Theta = \tau^T \delta\Theta$$

This equation must be true for every $\delta\Theta$, which means that we must have

$$F^T J = \tau^T$$

Transposing both sides gives the result.

5 Dynamics: Euler Lagrange Equations

Now we begin studying "dynamics", i.e., we study how a manipulator moves in response to applied joint torques and forces. We use the Lagrangian formulation, which is an energy-based approach.

First some notation: we'll switch to the book's notation and let q be the vector of joint variables (i.e., $q = \Theta$).

τ is the vector of joint torques/forces, just like in the static case.

The Euler-Lagrange equations give the equations of motion for a manipulator as follows:

$$\boxed{\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k}$$

for $k = 1, 2, \ldots, n$. Here $\mathcal{L}$ is called the Lagrangian. It is a function of both q and $\dot{q}$. It is defined to be

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{P}(q)$$

where $\mathcal{K}(q, \dot{q})$ is the kinetic energy of the robot and $\mathcal{P}(q)$ is the potential energy.

I know this all seems confusing and abstract at this point. We will make it more concrete over the course of the next few days. Here is what you should get now:

- the stuff in the box defines n equations
- the left side of each equations is a function of all of the joint variables q and their first and second derivatives ($\dot{q}$ and $\ddot{q}$).
- the right side of the kth equation is the torque applied to the kth joint
- we call this "a system of n coupled ordinary differential equations (or ODEs)"
- given the following:
  - the initial joint positions $q(0)$
  - the initial joint velocities $\dot{q}(0)$
  - the joint torques as function of time $\tau(t)$, defined for some interval of time $t \in [0, t_f]$.

  we can solve the Euler-Lagrange equations to get $q(t)$, $\dot{q}(t)$, and $\ddot{q}(t)$ for the time interval $t \in [0, t_f]$. In other words *we can solve for the motion that results from applying the torques in $\tau(t)$*.
- it is hard to solve these equations!
- it is easy to find approximate solutions using numerical methods. (we'll learn how to do this).

For this class, we will just take these equations as gospel truth. They can be derived in a number of ways, all of which are beyond the scope of this class. There is a very nice discussion in the book (section 7.1.2) that derives the Euler-Lagrange equations using the principle of virtual work. You won't be responsible for this material on an exam, but reading it and understanding will make you a better person.