NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 05
Lecture: 21

Gravity Compensation, and External Forces/Torques

Overview of this lecture

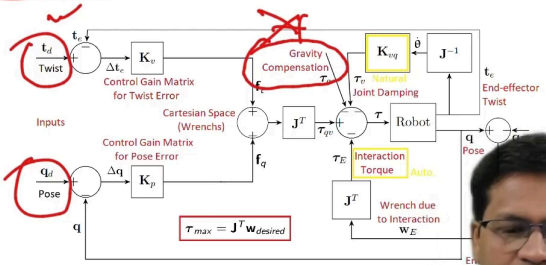
- Gravity Compensation
- Resolving External Forces/Torques to Joint Forces/Torques

Welcome to the second lecture of the module Robot Statics. In this module, I will discuss Gravity Compensation, and I will try to resolve External Forces and Torques into Joint Forces and Torques.

Introduction to Gravity Compensation

Minimum amount of joint torque/force that the joints actuator should provide even when the robot is stationary.

Applications of Gravity Compensation:
Lead-through programming of COBOT.
Controller implementations with floating robot model (uniform behavior in all directions).



Video Link: https://youtu.be/Mnec0zuzqkA?si=tm9IDIxI58vk9XrT

So, let us begin with Gravity Compensation. What is that? Gravity compensation torques are the minimum amount of joint force and torque that the joint actuator should provide even when the robot is stationary.

So, I want to physically make this robot float in the air as if it is in space, as if there is no gravitational pull which is pulling the robot arms downward. So, applications of gravity compensation are led through programming, which you have seen in this video that I have shown in the application video in my first lecture. So, you can see directly here. In this case, what I was doing, I could drag the robot to any different points on the programming frame. I have taken my robot to the first point, second point, third point, fourth point. I can drag the robot physically. Even though the robot is quite heavy, I could not feel it because it was gravity compensated from within. All the joint actuators were providing enough torque to make this robot float in the air. So, I was able to take it to different points. So, once I take it to a different point, I can teach it, and the robot can repeat those points back. This is known as lead-through programming, which is one of the very common things in Cobots.

Controller implementation with a floating robot model behaves uniformly in all directions. So, you see a regular controller scheme that we will be discussing in much detail in the module Robot Control. So this is the gravity compensation. This is an external compensating torque that is continuously acting at the robot joint so that the robot becomes floating in the air and behaves uniformly in all directions. Thereafter, you apply it with input velocity, input position commands, force commands, and everything else. So, this is why it is very important, and these are the places where gravity compensation is utilised.

Introduction to Gravity Compensation

Minimum amount of joint torque/force that the joints actuator should provide even when the robot is stationary.

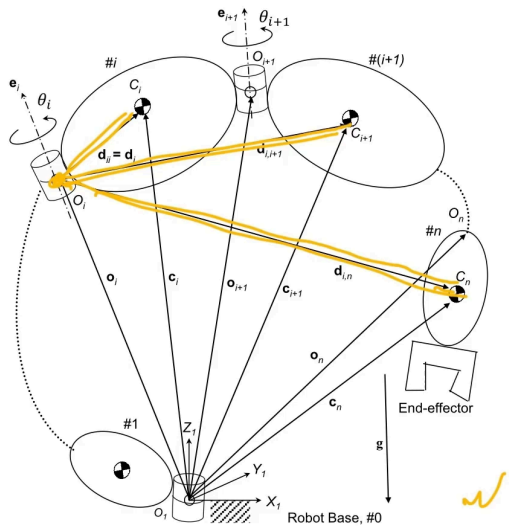
Applications of Gravity Compensation:
Lead-through programming of COBOT.
Controller implementations with floating robot model (uniform behavior in all directions).

Pre-requisite:
Identified model of the robot for its accurate mass and center-of-mass locations with respect to the link.

→ Obtained through precise CAD models and/or through identifications experiments with isolated link or by measuring the joint torques after assembling the link to the robot.

So, the prerequisite to begin with gravity compensation is that the whole robot mass and mass centre location should be well known. So, they are identified using identification experiments with respect to the link. So, they are obtained through precise CAD models with the known densities of the material with which these robots are designed. The whole assembly is packed into the CAD environment, and there you can or it can be done after assembly also through identification experiments with the isolated link or by measuring the joint torques after assembling the link to the robot. So, there are various approaches to it. These are a few. So, this is the link of a robot which is shown here. In CAD, it looks like this.

Static Joint Forces/Torques due to Gravity



The forward kinematic transformation matrix for the i^{th} frame with respect to robot's base is:

$$[\mathbf{T}_i]_1 \equiv \begin{bmatrix} [\mathbf{Q}_i]_1 & \mathbf{o}_{i+1} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \text{where } i = 1, 2, \cdots, n$$

The position vector of the center of gravity (CG) $\mathbf{c}_i \equiv [c_{ix}\ c_{iy}\ c_{iz}]^T$ with respect to the base frame O_1

$$\mathbf{c}_i = \mathbf{o}_i + [\mathbf{Q}_i]_1\, \mathbf{d}_i$$

where $\mathbf{d}_i \equiv [d_{ix}\ d_{iy}\ d_{iz}]^T$ is position of the CG of the i^{th} link with respect to the link $i-1$

The intermediate vectors $\mathbf{d}_{ik}$ that joins the origin of the i^{th} link to the CG of the k^{th} link are:

$$\mathbf{d}_{ik} = \mathbf{c}_k - \mathbf{o}_i \text{ where, } k = [i,\ i+1,\ \cdots,\ n-1,\ n]$$

So, Static Joint Forces and Torques due to Gravity. Let us begin calculating that. So, the forward kinematic transformation matrix for the i^{th} frame with respect to the robot's base, you know, can be extracted using forward kinematics directly. You can calculate it. So, T is equal to link transformation matrices. So, Ai, i minus 1($^{i-1}A_i$). So, likewise, you keep on moving i plus 1, i, and so on till n, n minus 1, and it goes from 0 to 1.

$$T_i = {}^{0}A_1\, {}^{1}A_2 \cdots {}^{i-1}A_i\, {}^{i}A_{i+1} \cdots {}^{n-1}A_n$$

So, like that. So, finally, it gives you the link transformation matrix. So, using this, you can calculate the rotation matrix of the link and the position of the link fixed frame. So, that is o_i plus one. So, for each link, if you do forward kinematic transformation till that link, you can get all the o_is. So, all the frame location, you can calculate as well as the orientation of the link. So, this is the first step for that.

Next is the position vector of the centre of gravity ci, cix, ciy, cyz with respect to the base frame is calculated using this. So, how? So, you already know the d_i, d_i is d_{ix}, d_{iy}, d_{iz} is the position of the centre of gravity of the i^{th} link with respect to the link i minus 1. So, you already know the vector this because the link CAD geometry is well known. This can be extracted from the CAD. So, this ci with respect to one of the points which is on the body

itself, can be easily found out. So, di is already known, so di has to be transformed and made with respect to the base frame so that you can do this-

$$\mathbf{c}_i = \mathbf{o}_i + [\mathbf{Q}_i]_1 \, \mathbf{d}_i$$

Because ci is to be calculated in the base frame. So, oi you have already calculated in the base frame using this, from the top, you see this equation, from where you have extracted all the o_is, so o_i is known, Qi is also taken from here, multiply with d_i, it gives you in the base frame. So, all the vectors are in the base frame now. So, using this vector triangle, using this vector triangle, you can calculate ci. So, the center of gravity location is now known with respect to the base frame.

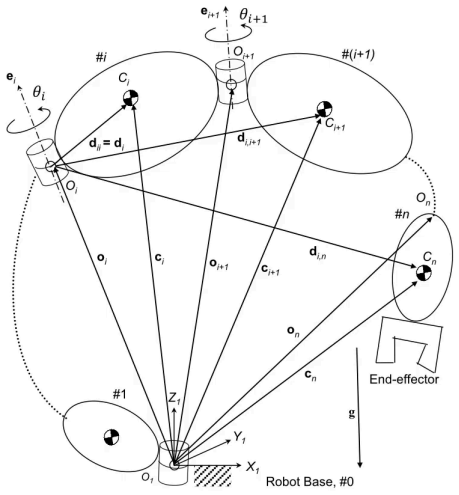
Now, the intermediate vectors that are d_i to k that join the origin of the ith link to the CG of the k^{th} link, origin, let's say this is the ith link. So, these are the vectors. So, all these vectors can be found out using $d_{ik} = c_k - o_i$ (c_k minus oi).

$$\mathbf{d}_{ik} = \mathbf{c}_k - \mathbf{o}_i$$

What is it? c_k, you have extracted here all the c_k. o_i, you already know, you have taken it from here. Using that, you can calculate all these vectors. So, that is very, very important. So, you can go k = i, i + 1, i + 2, i + 3, and finally till n - 1 and n. So, let me just vanish for a moment again.

So, yes. So, dik will be very, very helpful in calculating the moment at the ith joint due to the centre of gravity forces, which are gravitational forces, which are here at all the links. So, each of these would cause a moment over the i^{th} joint.

Static Joint Force/Torque due to Gravity



Moment due to gravity at the i^{th} joint $\mathbf{n}_i$ will be the result of moments due to all the link after O_i:

$$\mathbf{n}_i = \sum_{k=i}^{n} m_k \mathbf{d}_{ik} \times \mathbf{g}$$

m_i: mass of the i^{th} link, $\mathbf{g}$: acceleration due to gravity

Net torque due to gravity at any joint i is the moment $\mathbf{n}_i$ projected along the joint axis Z_i, i.e., along $\mathbf{e}_i$.

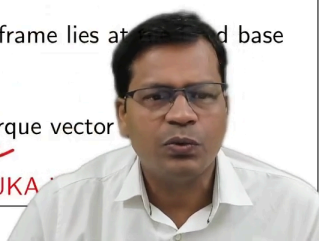
$$\tau_i = \left[\sum_{k=i}^{n} m_k \mathbf{d}_{ik} \times \mathbf{g}\right] \cdot \mathbf{e}_i$$

Note: $\mathbf{e}_1 \equiv [0\ 0\ 1]^T$ of the first frame lies at [illegible] base along the $Z-$Axis of joint 1.

The net gravity compensation torque vector

$$\boldsymbol{\tau}_g = [\tau_1\ \tau_2\ \cdots\ \tau_n]^T$$

Video: Demonstrations using KUKA [illegible]



So, that is what. So, ni will be the final moment, and the total moment at the ith joint would be equal to the sum of all the mk * g, k = i to n, finally. So, all the links that are after the link's joint i will influence the moment which is here.

$$\mathbf{n}_i = \sum_{k=i}^{n} m_k \mathbf{d}_{ik} \times \mathbf{g}$$

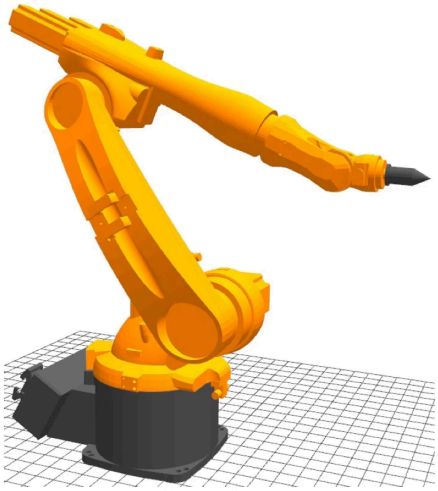
So, the sum of all the moments k = i to n. d i cross mkg, so effectively it gives me this, so total moments. So, this becomes my total moment at the ith joint.

So, now net torque due to gravity at any joint i is the moment, and projected along the Zi axis that is the joint axis. So, the sum of moments you have obtained. So, project that along the axis of rotation, what you get is the torque at that axis. So, torque at that axis will be a projection of this (above equation), that is, dot product with e_i vector. What is ei? e_i is the axis vector that you know it can be extracted from the homogeneous transformation matrix directly, the third column, first three rows. So, if it is a homogeneous transformation matrix, the first, the second, and the third column, you can get the first three elements will be the axis vector. So, if you have already done forward kinematics, A1, A2, A3 and finally, till this, you have to do this point. So, that transformation matrix, the third column, will have ei vector.

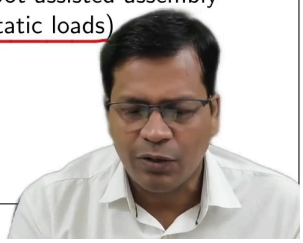
So, e_i for the first will be the first joint will be 001, that lies at the fixed base along the ZI of the joint 1. The net gravity compensation torque would be an array which contains T1, T2, T3, T4, T5, T6 likewise. So, you compile them all together in a column, and that is the gravity compensation torque. This is what is to be applied to the robot continuously, even to maintain this robot in its position. It has to continuously apply torque so as to be there.

Let me show you a small video. So, you see what I am doing. I am able to manipulate this robot and take it to different places directly without feeling the due to gravity. Each of the links is not very light, as I am feeling it. So, you see, I am able to drag this robot anywhere. I can orient it, I can position it, I can put it to any point anywhere. So, this is the beauty of having such an algorithm. Now, if the robot is made to float like this in space, you can understand it becomes very easy to apply the control algorithms now, and this may this also allow me to drag this robot and do teach-repeat kind of operations. So, this is the application of gravity compensation. Now, coming to the next.

Resolving External Forces/Torques to Joint Forces/Torques

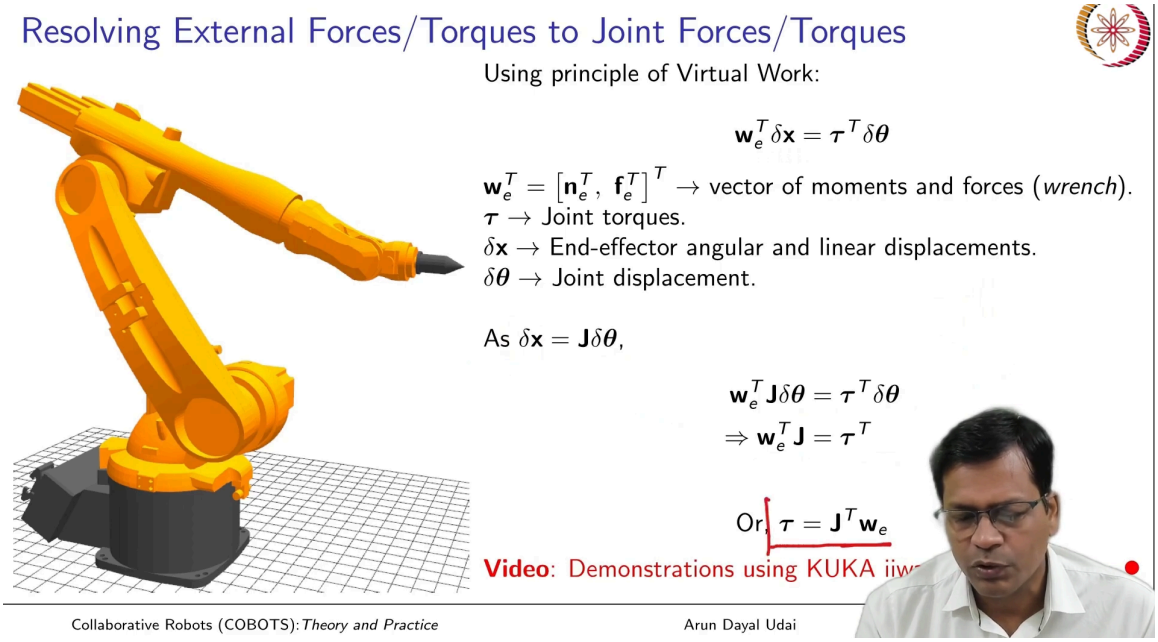


- What are we going to do?
- **Applications**:
  Collaborative manipulation (Robot-Robot), Human Robot Interaction.
  Handling tasks in contact (Continuous Control): Surface finishing tasks, Robot assisted assembly
  Pick-and-Place (Discrete Static loads)
  Controller design.



So, Resolving External Forces and Torques to the Joint Forces and Torques. So, what we are going to do here is, any external force, if it is applied from here, let us say it is doing a grinding operation, polishing operation. So, any assembly task, so any quasi-static task, will also create interaction forces at the end effector, so how does that influence each of the robot joints. So, I can counter that external force using the joint torques. That is what I want to do here. So, how much is the additional torque, apart from the gravity compensation torque, I have to apply at the joint so as to maintain that contact, or maybe you want to maintain a constant reaction force of some value, against this wall or maybe any object. So, this is what my intention here is.

So, applications are collaborative manipulation, that is, robot-robot interaction or human-robot interaction. Handling tasks in contact that are in continuous control, like surface finishing tasks, grinding, polishing, buffing, and deburring. All these tasks, this is very, very important. So, in the case of grinding, if you vary the amount of force, grinding surfaces will be different. So, you have to maintain a constant reaction force against that surface. So, that is what I want. Robot-assisted assembly. Pick and place Discrete Static Loads you have to do. I am picking up and placing a 5 kg load, let's say. Tomorrow, I want a 10 kg load to be handled. So, that load will be different. So, that can also be taken care of. Controller design is very, very important. We'll use that. We'll discuss this very much in detail when we'll be discussing controls also.



So, how we'll do it using the principle of Virtual Work? If you remember, the external range is w_e. It is nx, ny, nz; these are the moments, and fx, fy, fz; this is the force. So, clubbed together, it creates the wrench vector, a vector of moment and force. This is known as a wrench. So, the external wrench here.

Tau is the just now we have calculated in gravity compensation; similar to that, this is a torque vector. So, this is T1, T2, T3, T4 up to the T joint. Let us say if it is a 7-DoF robot, it is still 7. So, that will be the joint torque.

Delta x, let us assume that is the small end effector angular and linear displacements. That is delta x. So, end effector displacement and delta theta is the joint displacement. So, using the principle of virtual work, what you find is the input work done should be equal to the output work done. So, we have assumed the infinitesimally small incremental displacement along the joint space and the task space. So, delta x into the range, that is the linear and angular displacement into the range moments and forces. So, the product of that is the work done at the end effector, and the work done at the joint would be torque into joint displacement. This is the axis torque into joint angular displacement delta theta. So, that is, input work and output work are both equated here.

$$w_e^T \delta x = \tau^T \delta\Theta$$

So, this is the principle of virtual work we have applied here.

So, as you already know, delta x is equal to Jacobian times delta theta. Delta x is equal to j delta theta. So, that relates, Jacobian relates the input and output rates here. Substituting that delta x here, what we get is wrench transpose into Jacobian delta theta is equal to Tau transpose delta theta.

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

If we cancel out delta theta here, what we get is wrench Jacobian is equal to torque.

$$w_e^T J = \tau^T$$

Wrench times Jacobian is equal to torque. Mind it, this can be a square matrix or it may not be a square matrix also. So, in that case, you have to use J dagger, which is the pseudo-inverse, if you want to calculate the wrench for the corresponding joint torque. So, the Jacobian may or may not be a square matrix. So, tau is equal to what? It is equal to J transpose of we.

$$\tau = J^T w_e$$

So, this is one of the most fundamental equations which we are going to use in control later on.

So, let me again show you a small demonstration here with a small video of external force. So, this is what you see: the robot is continuously moving downwards, and as soon as it comes in contact, it will try to maintain a constant force against the surface. So, this is the operation where I am trying to move the robot with a constant normal reaction force. This may be helpful for doing any grinding work. So, this is what I am doing.

So, this kind of calculation is very important in making my robot perform any such task. Even with the changing environment, I have moved the table, and it is still maintaining the constant reaction force. So, in this case, a moving environment is already something I have done.

I hope you understand the significance of why we are trying to do this and the gravity convention. Both of these are very helpful in designing my control algorithm. Both of these are going to become the major components of the control algorithm, so we'll create the model based on this.

So that is all for this lecture. In the next lecture, I will discuss Kinetostatic Measures for Robot Design, which are Velocity and Force Ellipsoids.

That is all. Thanks a lot.