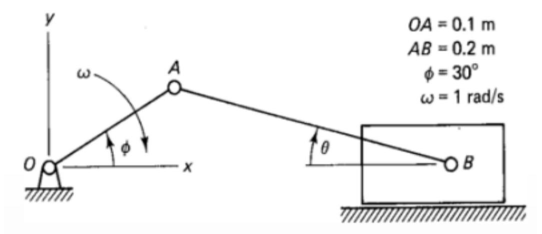
My repository link: https://github.com/Narges-Derakhshandeh/CMIM11

- Solve the problem from previous slide using $\phi = \pi\ 6 + \omega t$
- Implement translational joints and Kinematic analysis on acceleration level



Source: P. E. Nikravesh, *Computer-aided analysis....*

1. Making the system:

Globally I have a complete multibody system including bodies, joints, analysis settings. To store the information of the system I need to use data structure which named struct in MATLAB. So I have a global structure which include all information for my system and it is kind of tree which contain the body data (body struct).The drawback of keeping it is, it decrease the efficiency because it adds extra coordinates which they must be solved. But because I are not concern about the efficiency in this problem, I can keep ground as the body it makes the process more easy for us because it is fully constrained.

2. Defining my bodies

What do I need to describe my body? to describe the system I need: name of body, its location and orientation. In this system there are 4 bodies named as: ground, cranck, link, slider which are defined by their location and orientation in the MATLAB code.

I also need a fuction to generate the body which is named "add_body". output of the function is sys because I want to generate and add the body to previous system so the output is sys in this function. In this function I want to define that as a default the orienttation is zero but if I want to change it I can do it here. it means that in this function I provide orientation=0 for the arguments which are loIr 4 (arguments=sys, name, location, orientation) and it means I cannot provide the rotation.

Then I can add the body in the slider by define the body to the system by its name, location and orientation. Add the bodies in order(1.ground 2.cranck 3.link 4.slider). For the cranck, link I can add the location in meter and the orientation in rad

How to account the change? it is necessary to define a function as(make_system) to add the changes like adding gravity.

3. Joints-kinematic (revolute, simple and driving)

3.1 Revolute joints

then I want to define the data struct related to each joints(I have 3 revolute joints and a simple joint) as function of 2 bodies which the joint is connecting them together (named ad body_i and body_j). For body_i and body_j I need a coordinate .for the first revolute joint the body i is ground and the body j is cranck. I continue by adding the other revolute joint betIen cranck and link. remember for the new revolute joint I must follow exactly the previous steps. Then the last revolute joint betIen link and slider was added the same as previous ones.

3.2 Simple Joints

Now I need to add the simple joint which is related to the slider and the ground. I need to provide two more information here: the value(scaler) which make the constraints and secondly in which coordinate the simple joint works. what I need to constraint is y axis and angle to be zero.

Then I creat the add_joint_simple function for them. then in this, function I can add the check_body_exists for asserting the name of the bodies. then I need to change the coordinate name from string to the value, by defining if_end this puporse is provided and the coordinate name can cahnge to the value.

3.3 Driving joint

Now I need to add the simple joint which is related to the driving.it is applied on the cranck and I define the angle for as the initial value for this driving joint. The last argument for driving constraint (time function) is 1.2*t -deg2rad(30). Then I create the add_joint_simple_driving function for driving cranck. Then in this function I can add the check_body_exists for asserting the name of the bodies

3.4 Translational joint

In this stage, the translational joint betIen two bodies has been defined like revolute joint. For body_i and body_j I need a coordinate . the first joint body i is slider and the body j is ground. So I add translational joint function to my system and main file.

4. Constraints

Then I need to get constraints vector and because I have 4 types of joints so I have 3 types of constraints (revolute, simple and driving joints) and one more constraints vector for the translational joint.

so the first function is all constrains for revolute joints. I need qi for body i and qj for the body j. I also need to define the rotational matrix for computing S_i and S_j. I need to slove first constraint equation (qi(1:2)+Ai*s_i - qj(1:2)-Aj*j.s_j) to find the zeros. then I add the constraint function(constraints_revolute(sys,q)) related to the revolute joints to the main file of constraints.

The next constraint equation is for the simple joint (constraints_simple(sys,q)) which needs the body coordinate and values.

The next constraint equation is for the simple driving joint which is function of time and body coordinate function(constraints_simple_driving(sys,q,0)).

The last constraint equation is for the translational joint which is function of time and body coordinate. This joint is betIen two bodies (slider and ground). Its function is defined liked the revolute joint function.

I add these functions to the main file and the initial value for the time is zero. before solving the problem, I need to define the function for all constraints

Then I can check the length of C and q that they must be equal and I understand that I need to add 3 more constraints for the ground(x, y,angle). and now the length of both C and q are equal. if I want to solve the problem just in terms of position level I can use fsolve function. So, I can solve the problem (constraints(sys,q,0)) by the fsolve and initial coordinates are q0 and time=0

5. Derivative of constraints

To solve the problem in acceleration level, I need to have the derivative(respect to the coordinate q) of constraints matrix. Therefore for each of my constraint function (revolute, simple, driving and translational) , I have written their own derivative function with dq suffix. (note that : because the constraints have different equation so their derivative is also different and for each of them the derivative function must be written separately).

One of my constraints which is named as driving joints is time dependent, so I need to define a time derivative too for it (with dt suffix).

6. Mass matrix

The mass matrix is required for the kinematic analysis. So the mass matrix is written to include the bodies masses and inertia as the diagonal matrix.

7. Force matrix

For the kinematic analysis, it is necessary to determine the external and internal forces. Here the only external force applied on the bodies is the gravity force. So, the matrix of include the gravity force for each body. Remember that I use gravity (9.81 m/s2) and bodies mass to obtain the forces.

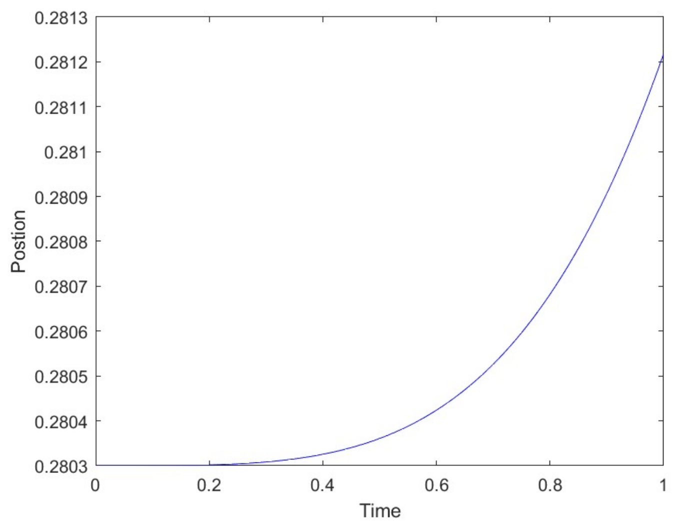
8. Solver part

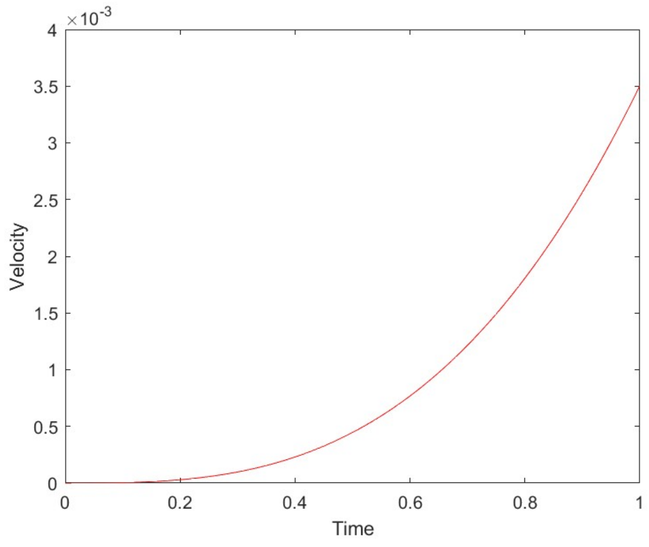
Then I want to solve the system for the acceleration level, I can use the odeEulerCromer function which its 3 outputs are the total time, the global coordinate matrix of bodies (Q) in every time steps and the last output is derivative of this matrix (Qd). For solving this function, I need initial values for the coordinates (q0) and the initial derivate coordinates (qd0).

We also can solve the problem by comparing the left side and right side of the equation in a for loop to find the Qdd matrix in acceleration level.

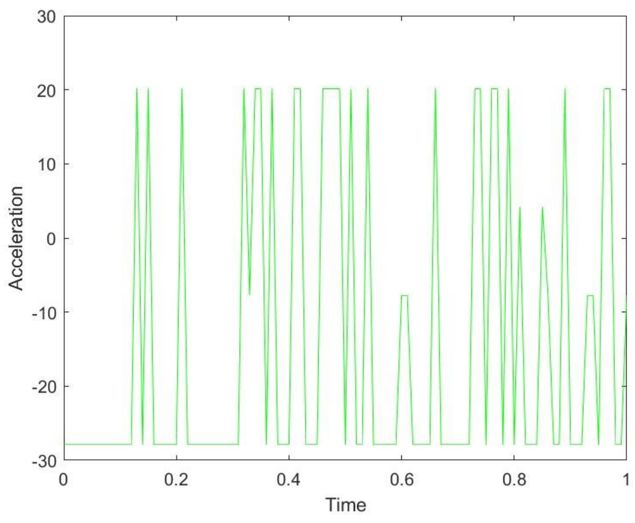
9. Plots

The results of position, velocity, and acceleration with the respect to the time is plotted.





×10⁻³
Velocity
Time
0
0.2
0.4
0.6
0.8
1
0.5
1.5
2
2.5
3
3.5
4



Acceleration
Time
30
20
10
0
-10
-20
-30
0
0.2
0.4
0.6
0.8
1