ME451
Kinematics and Dynamics of Machine Systems

Driving Constraints

3.5

October 16, 2014

Dan Negrut
ME451, Fall 2014
University of Wisconsin-Madison

- Quote of the day: “Impossible is a word to be found only in the dictionary of fools.”
-- Napoleon Bonaparte

Before we get started…

- Last time
  - Wrapped up cam-follower
  - Started to talk about motions
- Today
  - Continue discussion on motions (Rheonomic constraints)
  - Cover an example
- HW:
  - Posted online, due in one week
  - Pen-and-paper component: 3.5.1, 3.5.4, 3.5.5, 3.5.6
- MATLAB solutions posted online
  - Posting what the grader recommends as a nice solution

Driving Constraints

- The context
  - Up until now, we only discussed time invariant kinematic constraints
  - Normally the mechanism has a certain number of DOFs
  - Some additional time dependent constraints ("drivers") are added to control these "unoccupied" DOFs
    - Physically, these drivers represent actuators that control the motions of bodies in the mechanism
    - Recall that for Kinematics Analysis, you need NDOF=0
      - You have as many equations as unknowns (that is, generalized coordinates)

Driving Constraints: Types

Absolute Drivers

- Absolute x-coordinate driver
- Absolute y-coordinate driver
- Absolute angle driver

- Absolute distance driver

Relative Drivers

- Relative x-coordinate driver
- Relative y-coordinate driver
- Relative angle driver

- Relative distance driver

- Revolute-rotational driver
- Translational-distance driver

3.5.1

ABSOLUTE DRIVERS

Absolute Coordinate Drivers (1)

- Indicate that the coordinate of a point expressed in the global reference frame assumes a certain value that changes with time

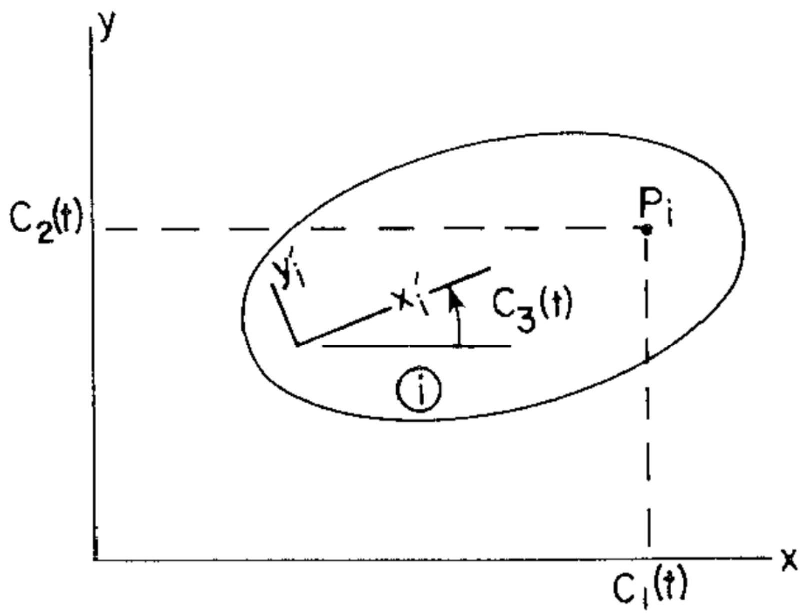


Figure 3.5.1 Absolute coordinate drivers.

$$\Phi^{axd(i)} = x_i^P - c_1(t) = 0$$

$$\Phi^{ayd(i)} = y_i^P - c_2(t) = 0$$

$$\Phi^{a\phi d(i)} = \phi_i - c_3(t) = 0$$

Absolute Coordinate Drivers (2)

Step 2

$$\Phi^{axd(i)} = x_i + {x'}_i^P \cos\phi_i - {y'}_i^P \sin\phi_i - c_1(t) = 0$$

$$\Phi^{ayd(i)} = y_i + {x'}_i^P \sin\phi_i + {y'}_i^P \cos\phi_i - c_2(t) = 0$$

$$\Phi^{a\phi d(i)} = \phi_i - c_3(t) = 0$$

Step 3

$$\Phi^{axd(i)}_{\mathbf{q}_i} = \begin{bmatrix} 1 & 0 & -{x'}_i^P \sin\phi_i - {y'}_i^P \cos\phi_i \end{bmatrix}$$

$$\Phi^{ayd(i)}_{\mathbf{q}_i} = \begin{bmatrix} 0 & 1 & {x'}_i^P \cos\phi_i - {y'}_i^P \sin\phi_i \end{bmatrix}$$

$$\Phi^{a\phi d(i)}_{\mathbf{q}_i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Step 4

$$\nu^{axd(i,j)} = 0 + \dot{c}_1(t)$$

$$\nu^{ayd(i,j)} = 0 + \dot{c}_2(t)$$

$$\nu^{a\phi d(i,j)} = 0 + \dot{c}_3(t)$$

Step 5

$$\gamma^{axd(i,j)} = \left({x'}_i^P \cos\phi_i - {y'}_i^P \sin\phi_i\right) \dot{\phi}_i^2 + \ddot{c}_1(t)$$

$$\gamma^{ayd(i,j)} = \left({x'}_i^P \sin\phi_i + {y'}_i^P \cos\phi_i\right) \dot{\phi}_i^2 + \ddot{c}_2(t)$$

$$\gamma^{a\phi d(i,j)} = 0 + \ddot{c}_3(t)$$

Straightforward calculation, starting from the corresponding kinematic constraint:

- Jacobian stays the same
- Add $\dot{c}(t)$ to expression of ν
- Add $\ddot{c}(t)$ to expression of γ

Absolute Distance Driver

- Step 2: Identify $\Phi^{add(i,j)} = 0$ (see Eq. 3.2.1 on page 57)

$$\boldsymbol{\Phi}^{\boldsymbol{add(i)}} = \left(\mathbf{r}_i^P - \mathbf{C}\right)^T \left(\mathbf{r}_i^P - \mathbf{C}\right) - c_4(t) = 0$$

- Step 3: $\Phi_{\mathbf{q}}^{add(i,j)} = ?$ (see Eq. 3.2.2 on page 58)

- Step 4: $\nu^{add(i,j)} = ?$ (see page 58)

- Step 5: $\gamma^{add(i,j)} = ?$ (see page 58)

3.5.2

RELATIVE DRIVERS

Relative Coordinate Drivers

- Indicate that the difference in a certain coordinate of points on the two bodies, expressed in the global reference frame, has a specified time evolution.

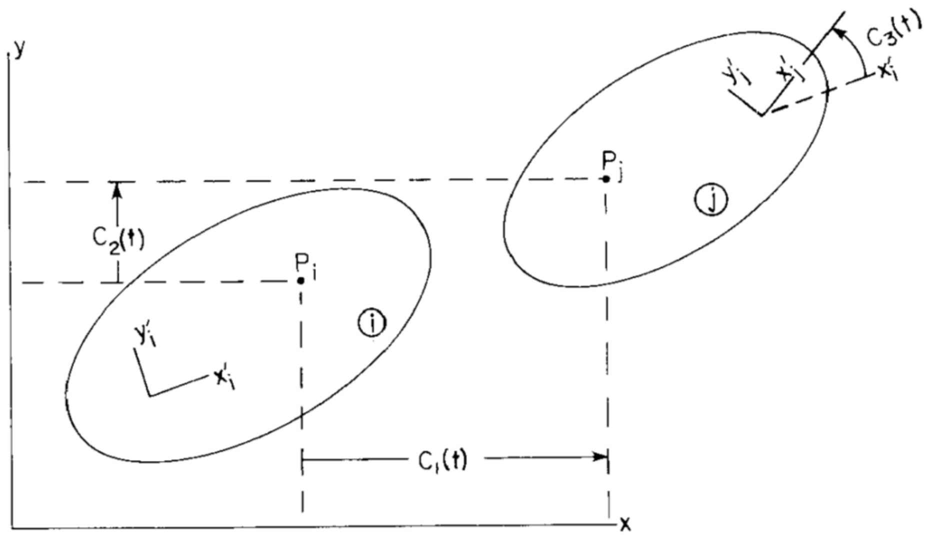


Figure 3.5.4 Relative coordinate drivers.

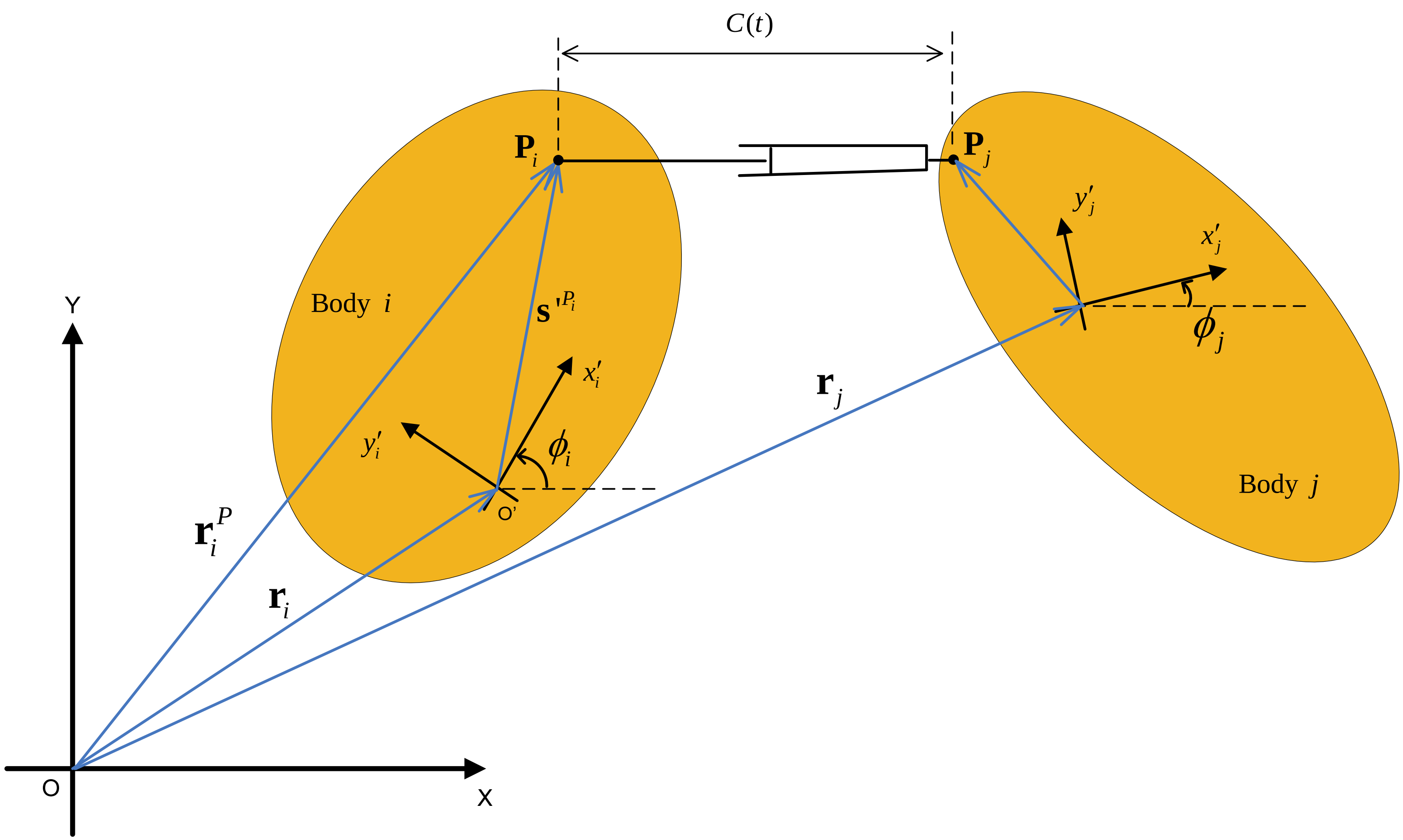
$$\Phi^{rxd(i,j)} = x_j^P - x_i^P - c_1(t) = 0$$

$$\Phi^{ryd(i,j)} = y_j^P - y_i^P - c_2(t) = 0$$

$$\Phi^{r\phi d(i,j)} = \phi_j - \phi_i - c_3(t) = 0$$

Relative Distance Driver (1)

- The distance between P_i and P_j is a prescribed function of time: $\|P_i P_j\| = c_4(t)$



Relative Distance Driver (2)

- Identify $\Phi^{rp(i,j)} = 0$ (see Eq. 3.3.7 on page 63)

$$\mathbf{\Phi}^{rdd(i,j)}(\mathbf{q}, t) = \left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T \left(\mathbf{r}_i^P - \mathbf{r}_j^P\right) - C_4^2(t) = 0$$

- Step 3: $\Phi_{\mathbf{q}}^{rp(i,j)} = ?$ (see Eq. 3.3.8 on page 63)

$$\mathbf{\Phi}_{\mathbf{q}_i}^{rdd(i,j)} = \left[2\left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T, 2\left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T \mathbf{B}_i \mathbf{s}'^P_i\right]$$

$$\mathbf{\Phi}_{\mathbf{q}_j}^{rdd(i,j)} = \left[-2\left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T, -2\left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T \mathbf{B}_i \mathbf{s}'^P_i\right]$$

- Step 4: $\nu^{rp(i,j)} = ?$ (see page 63)

$$\nu^{rdd(i,j)} = 0 + 2C_4(t)\dot{C}_4(t)$$

- Step 5: $\gamma^{rp(i,j)} = ?$ (see page 63)

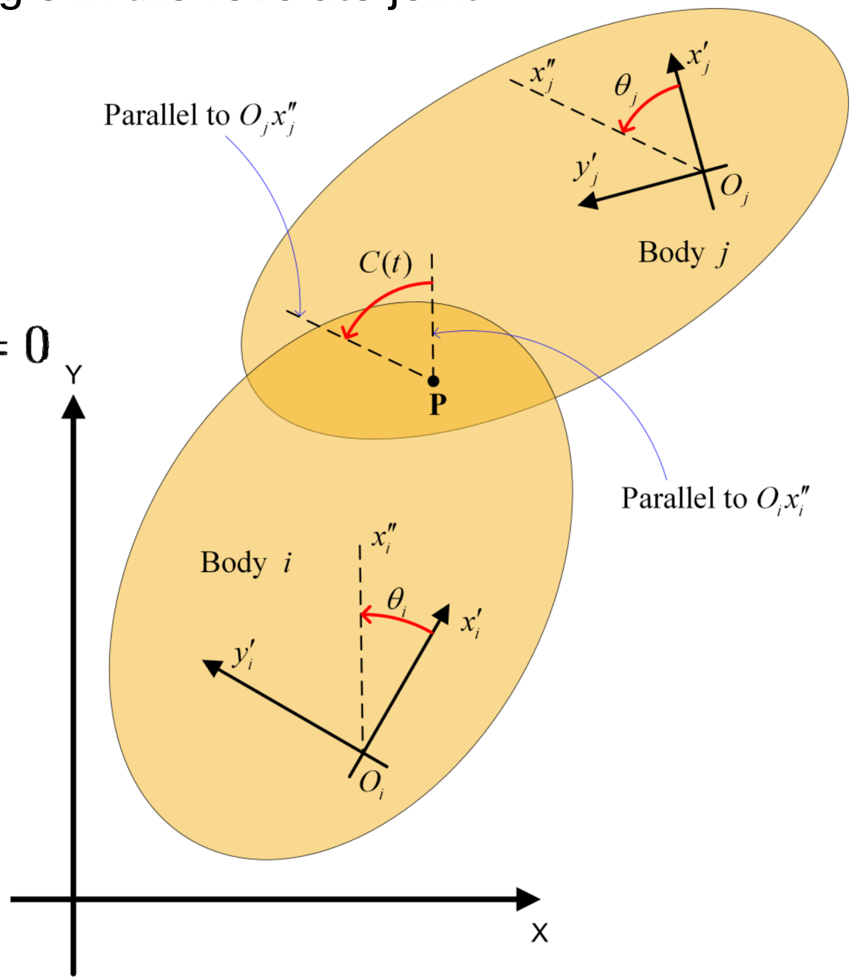
$$\gamma^{rdd(i,j)} = -2\left(\dot{r}_i^P - \dot{r}_j^P\right)^T \left(\dot{r}_i^P - \dot{r}_j^P\right) - 2\left(\mathbf{r}_i^P - \mathbf{r}_j^P\right)^T \left(\dot{\phi}_i^2 \mathbf{A}_i \mathbf{s}'^P_i - \dot{\phi}_j^2 \mathbf{A}_j \mathbf{s}'^P_j\right) + 2C_4(t)\ddot{C}_4(t) + 2\dot{C}_4^2(t)$$

Revolute-Rotational Driver

- Setup
  - Two bodies connected by a revolute joint at point P
  - We prescribe the time evolution of the angle in the revolute joint

- Constraint equation:

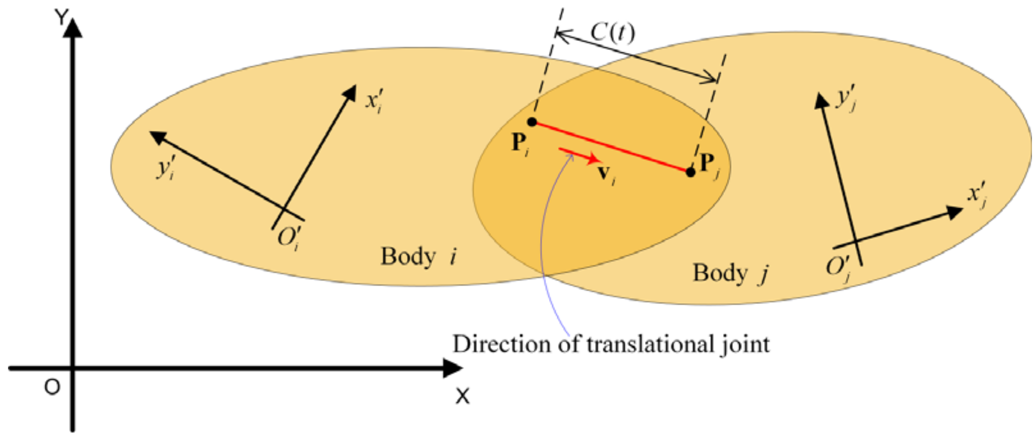
$$\Phi^{rrd(i,j)} = (\phi_j + \theta_j) - (\phi_i + \theta_i) - C(t)$$
$$\equiv (\phi_j - \phi_i) + (\theta_j - \theta_i) - C(t) = 0$$



- Notes
  - θ_i and θ_j are attributes of the constraint
  - With an appropriate choice of the LRFs and/or modifying $C(t)$, θ_i and θ_j can be made equal to zero

Translational-Distance Driver (1)

- Setup
  - Two bodies connected by a translational joint
  - We prescribe the displacement in the translational joint



- Model
  - Direction of translational joint on body i is defined by the vector $\mathbf{v}_i$
  - This driver says that the distance between point P_i on body i and point P_j on body j, measured along the direction of $\mathbf{v}_i$, changes in time according to a user prescribed function $C(t)$:

$$\frac{\mathbf{v}_i^T \mathbf{d}_{ij}}{v_i} - C(t) = 0$$

Translational Distance Driver (2)

- The book complicates the formulation for no good reason
  - There is nothing to prevent us from specifying the direction $\mathbf{v}_i$ using a unit vector (that is making $v_i = 1$)

- The mathematical representation of this driver is then simply:

$$\mathbf{v}_i^T \mathbf{d}_{ij} - C(t) = 0$$

resulting in:

$$\mathbf{\Phi}^{tdd(i,j)} = \mathbf{v}_i'^T \mathbf{A}_i^T (\mathbf{r}_j - \mathbf{r}_i) + \mathbf{v}_i'^T \mathbf{A}_{ij} \mathbf{s}_j'^P - \mathbf{v}_i'^T \mathbf{s}_i'^P - C(t) = 0$$

- Important: the direction of translation is indicated now through a unit vector (you are going to get the wrong motion if you work with a $\mathbf{v}_i$ that is not unit length)

- Keep this in mind when working on HW problem 3.5.6

Driver Constraints: Conclusions (1)

- What is after all a driving constraint?
  - We start with a kinematic constraint, which indicates that a certain *kinematic quantity* should stay equal to zero
  - Rather than equating this *kinematic quantity* to zero, we allow it to change with time:

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0} \qquad \text{versus} \qquad \mathbf{\Phi}(\mathbf{q}) = C(t)$$

or equivalently:

$$\underbrace{\mathbf{\Phi}(\mathbf{q}) = \mathbf{0}}_{\text{Kinematic Constraint}} \qquad \text{versus} \qquad \mathbf{\Phi}^D(\mathbf{q}, t) = \underbrace{\mathbf{\Phi}(\mathbf{q}) - C(t) = \mathbf{0}}_{\text{Driver Constraint}}$$

Driver Constraints: Conclusions (2)

- Notation used:
  - Kinematic Constraints: $\mathbf{\Phi}^K(\mathbf{q})$
  - Driver Constraints: $\mathbf{\Phi}^D(\mathbf{q}, t)$
  - Note the arguments: for Kinematic Constraints, there is no explicit time dependency

- On the RHS issue…
  - Computing the right hand sides (RHS) of the velocity and acceleration equations; i.e., $\boldsymbol{\nu}$ and $\boldsymbol{\gamma}$, for driver constraints is straightforward
    - Once we know how to compute these quantities for $\mathbf{\Phi}^K(\mathbf{q})$, converting to $\mathbf{\Phi}^D(\mathbf{q}, t)$ is just a matter of correcting…
      - … $\boldsymbol{\nu}$ (RHS of velocity equation) with the first derivative of $C(t)$
      - … $\boldsymbol{\gamma}$ (RHS of acceleration equation) with second derivative of $C(t)$
      - Section 3.5.3 discusses these issues

MATLAB Implications:
How to Handle Arbitrary Motions

Problem: Given a string that represents the expression of some function of time t, how do you evaluate the function, as well as its first two derivatives?

```
% Assume the string ‘str’ contains the expression of the function
% (e.g. as read from the ‘fun’ property of a driver constraint in the ADM file)
fun_str = ‘(1.5 * sin(t) + 3 * t^2)^2’;

% Declare a symbolic variable for time
syms t;

% Evaluate the given string as a symbolic expression.
% NOTE: we must play a trick here to deal with the case where ‘fun_str’
%       represents a constant function; i.e. there is no explicit dependency
%       on t. In this case, ‘eval’ by itself would return a double, not a sym!
fun_sym = sym(eval(fun_str));

% Symbolically differentiate the function.
funD_sym = diff(fun_sym);
funDD_sym = diff(fnuD_sym);

% Create Matlab function handles from the three symbolic functions above
fun_handle    = matlabFunction(fun_sym, ‘vars’, t)
funD_handle   = matlabFunction(funD_sym, ‘vars’, t);
fundDD_handle = matlabFunction(funDD_sym, ‘vars’, t);
```

[handout]

Example: Relative Distance Drivers

- Generalized coordinates: $\mathbf{q} = [\phi_1 \,,\, x_2 \,,\, y_2 \,,\, \phi_2]^T$

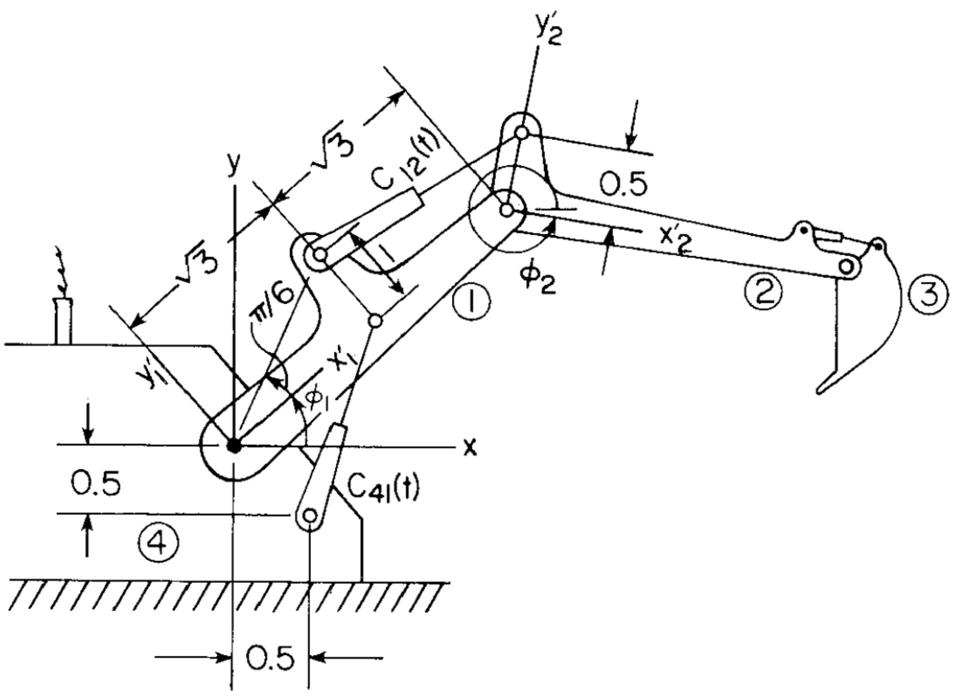


Figure 3.5.6 Excavator boom assembly with two distance drivers.

- Prescribed motions:

$$C_{41}(t) = \frac{1}{5}t + 1.8$$

$$C_{12}(t) = \frac{1}{10}t + 1.9$$

- Derive the constraints acting on the system
- Derive the linear system whose solution provides the generalized velocities