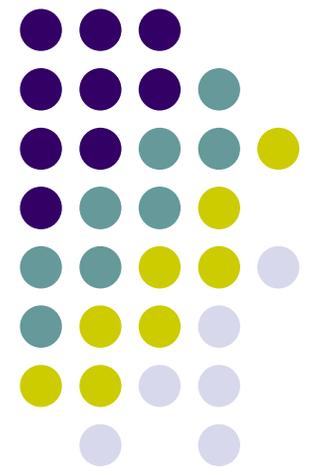# ME451
Kinematics and Dynamics
of Machine Systems

Driving Constraints 3.5
October 19, 2010

Before we get started…

- Last Time
  - Finished kinematic constraints
    - Cam-follower & Cam-flat follower
    - Point-Follower

- Today:
  - Start cover driving constraints

- HW due next Tu:
  - 3.4.7, 3.4.8, 3.4.9 + MATLAB (posted on class website)

- Recall that TA assigned ADAMS homework

- Final Project proposal due on 10/28
  - Can be based on MATLAB simEngine2D
  - Can represent some project you undertake in ADAMS

- Syllabus was updated on the class website

Driving Constraints

- ## The context

  - Up until now, we only expressed time invariant kinematic constraints

  - These constraints describe physical structure of a machine/mechanism

  - Normally the mechanism has a certain number of DOFs

  - Some additional time dependent constraints ("drivers") are added to control these "unoccupied" DOFs
    - You thus control the motion of the mechanism

Kinematic Drivers

Absolute Coordinate Drivers

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

Relative Coordinate Drivers

- You see these more often…

    - Relative distance driver
    - Revolute-rotational driver
    - Translational-distance driver

4

Absolute Driving Constraints

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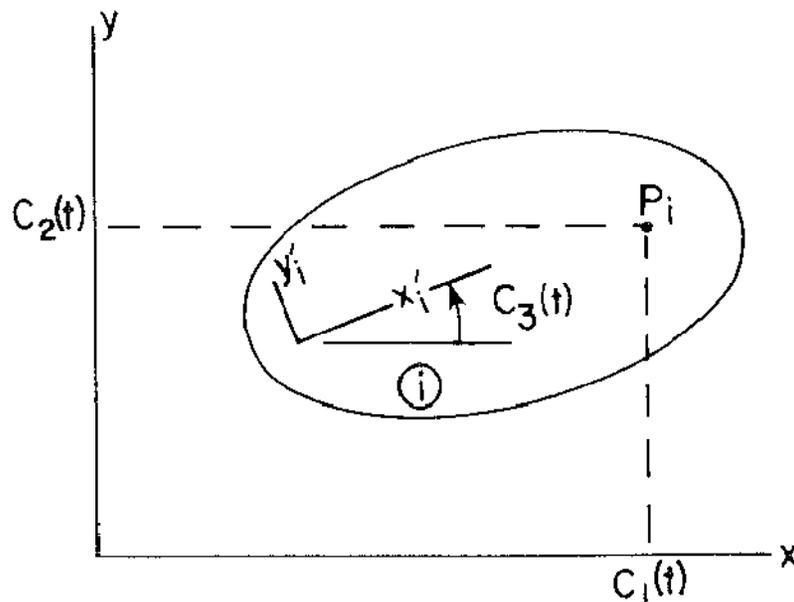

Figure 3.5.1 Absolute coordinate drivers.

- X-position

$$x^{P_i} - C_1(t) = 0$$

- Y-position

$$y^{P_i} - C_2(t) = 0$$

- Orientation angle
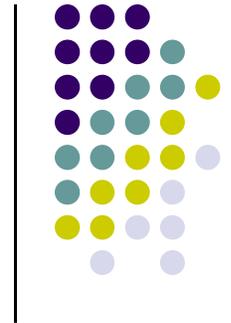
$$\phi_i - C_3(t) = 0$$

5

Absolute Driver Constraints

- Step 2: Constraint Equations $\Phi(\mathbf{q}, t) = ?$

- Step 3: Constraint Jacobian $\Phi_{\mathbf{q}} = ?$

- Step 4: $\nu = ?$

- Step 5: $\gamma = ?$

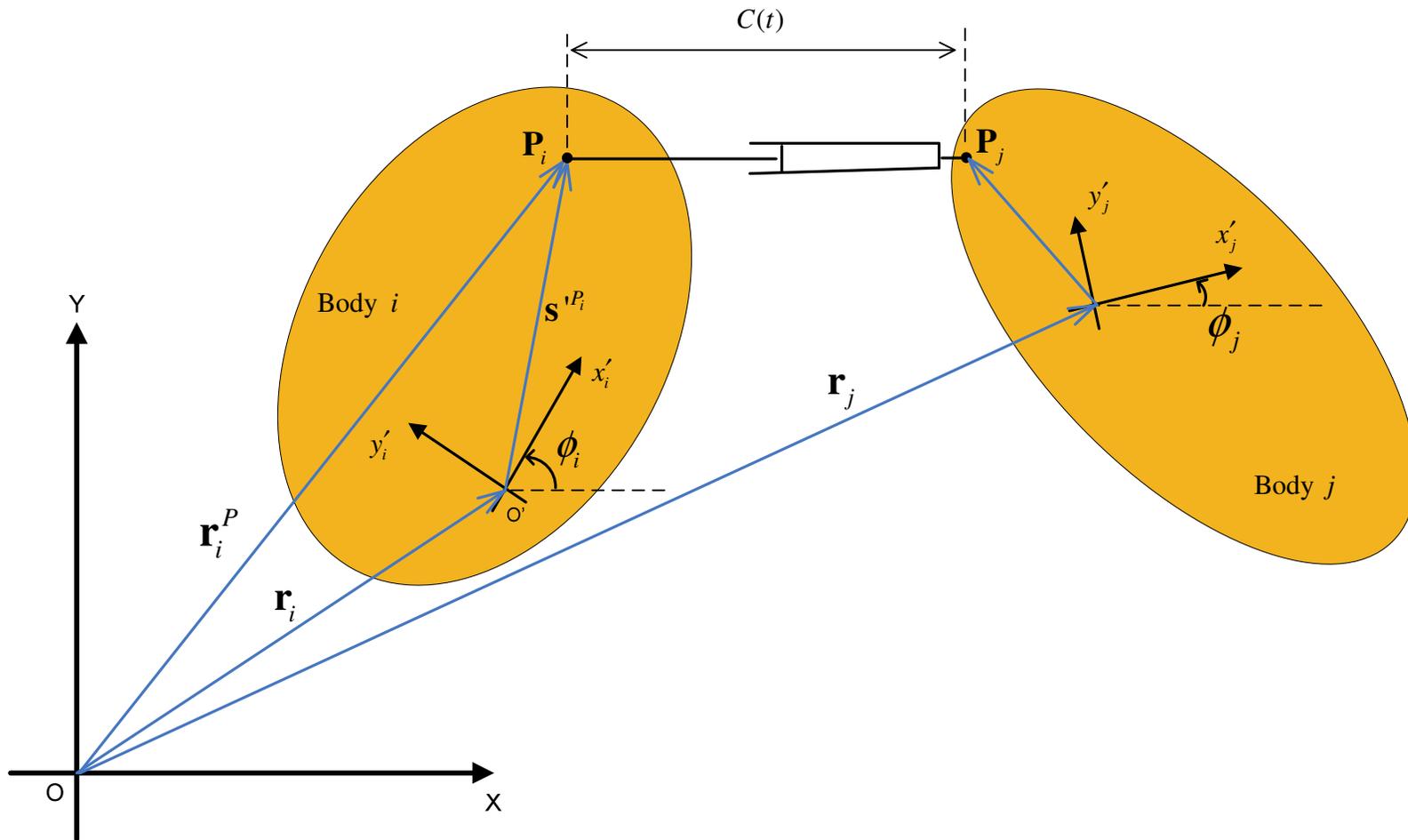**Absolute Coordinate Drivers**
- Very simple to compute this information

6

End Absolute Driving Constraints

Begin Relative Driving Constraints

Relative Distance Driver

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

Relative Distance Driver

- Step 2: Constraint Equations

$$\boldsymbol{\Phi}^{rdc(i,j)}(\mathbf{q}, t) = (\mathbf{r}_i^P - \mathbf{r}_j^P)^T (\mathbf{r}_i^P - \mathbf{r}_j^P) - C^2(t) = 0$$

- Step 3: Constraint Jacobian $\boldsymbol{\Phi_q}$ = ? (see Eq. 3.3.8)

$$\boldsymbol{\Phi}_{\mathbf{q}_i}^{rdc(i,j)} = 2(\mathbf{r}_i^P - \mathbf{r}_j^P)^T \left[\begin{array}{cc} \mathbf{I} & \mathbf{B}_i \mathbf{s}_i'^P \end{array} \right] \qquad \boldsymbol{\Phi}_{\mathbf{q}_j}^{rdc(i,j)} = 2(\mathbf{r}_j^P - \mathbf{r}_i^P)^T \left[\begin{array}{cc} \mathbf{I} & \mathbf{B}_j \mathbf{s}_j'^P \end{array} \right]$$
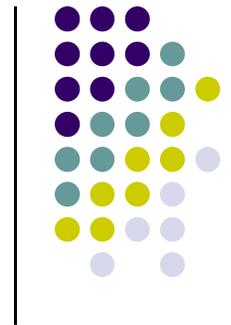
- Step 4: ν = ?

$$\nu = 2C(t)\dot{C}(t)$$

- Step 5: γ = ?

$$\gamma^{rdc(i,j)} = -2(\dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P)^T (\dot{\mathbf{r}}_i^P - \dot{\mathbf{r}}_j^P) + 2(\mathbf{r}_i^P - \mathbf{r}_j^P)^T (\dot{\phi}_i^2 \mathbf{A}_i \mathbf{s}_i'^P - \dot{\phi}_j^2 \mathbf{A}_j \mathbf{s}_j'^P) + 2C(t)\ddot{C}(t) + 2\dot{C}^2(t)$$
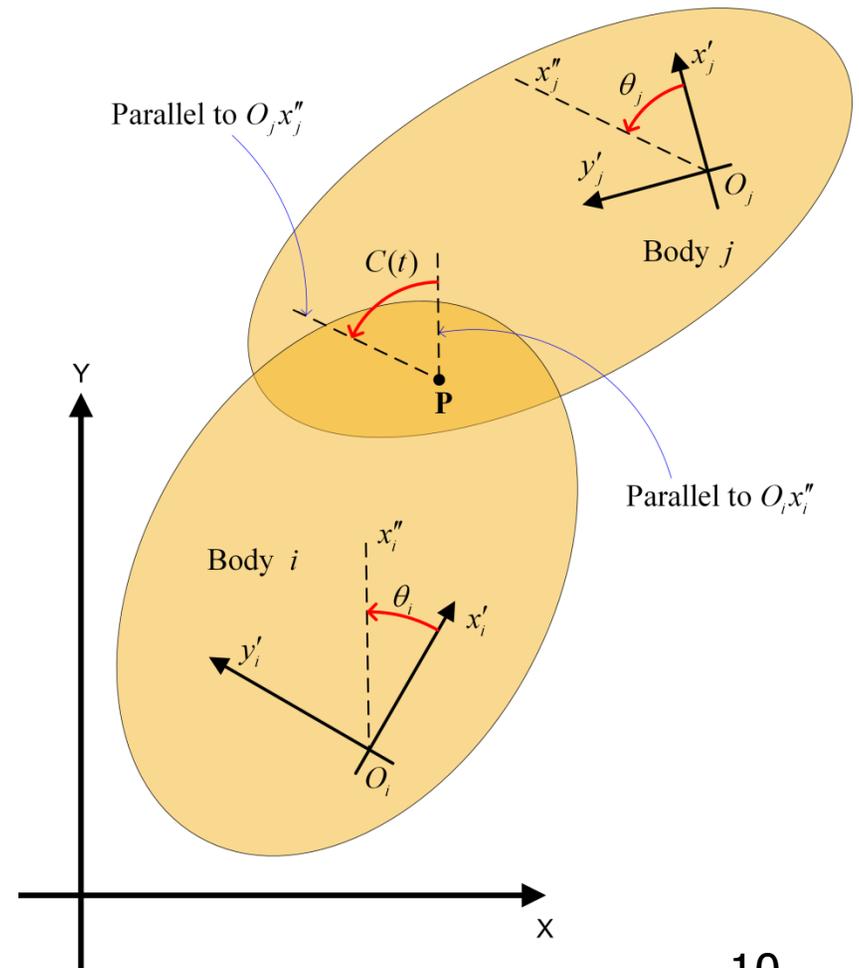
Revolute Rotational Driver

- The framework: at point **P** we have a revolute joint
- It boils down to this: you prescribe the time evolution of the angle in the revolute joint

- Driver constraint formulated as

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

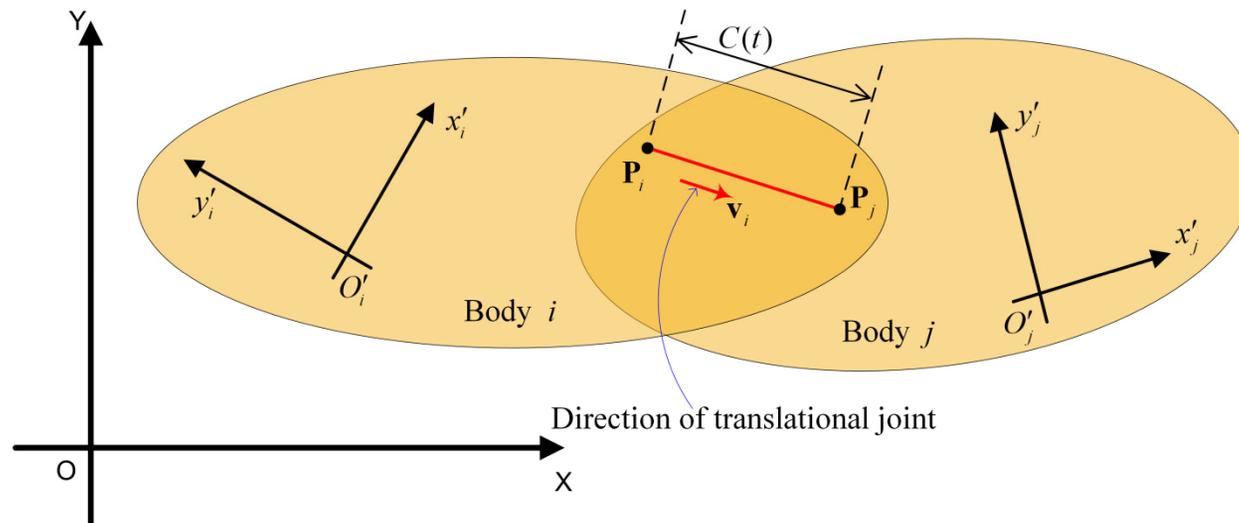- Note that θ_i and θ_j are attributes of the constraint

Parallel to $O_j x_j''$

x_j'' θ_j x_j'

y_j' O_j

Body j

$C(t)$

Y

P

Parallel to $O_i x_i''$

Body i

x_i''

θ_i x_i'

y_i'

O_i

X

Translational Distance Driver

- The framework: we have a translational joint between two bodies
  - Direction of translational join on "Body i" is defined by the vector $\mathbf{v}_i$

Direction of translational joint

- This driver says that the distance between point P_i on "Body i" and point P_j on "Body j" measured along in the direction of $\mathbf{v}_i$ changes in time according to a user prescribed function $C(t)$:

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

11

Translational Distance Driver (Cntd.)

- The book complicates the formulation with no good reason

  - There is nothing to prevent me to specify the direction **v**$_i$ by selecting this quantity to have magnitude 1

- Equivalently, the constraint then becomes

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

- Keep in mind that the direction of translation is indicated now through a unit vector (you are going to get the wrong motion if you work with a **v**$_i$ that is not unit length)

- This is the reason why I wanted to discuss this prior to assigning the problem 3.5.6

Driver Constraints, Departing Thoughts

- What is after all a driving constraint?

  - You take your kinematic constraint, which indicates that a certain *kinematic quantity* should stay equal to zero

  - Rather than equating this *kinematic quantity* to zero, you have it change with time…

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

- Or equivalently…

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

# Driver Constraints, Departing Thoughts (cntd.)

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

- Correcting the RHS…
  - Computing for <u>Driver</u> Constraints the right hand side of the velocity equation and acceleration equation is straightforward

  - Once you know who to compute these quantities for $\Phi^K(\mathbf{q})$, when dealing with $\Phi^D(\mathbf{q},t)$ is just a matter of correcting…
    - … ν (RHS of velocity equation) with the first derivative of C(t)
    - … γ (RHS of acceleration equation) with second derivative of C(t)
    - Section 3.5.3 discusses these issues

14

MATLAB:
How to Handle Arbitrary Motions
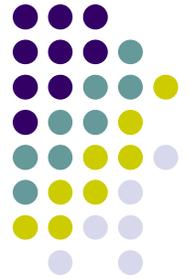
- The function C(t) should be read from an input file and you need to be able to evaluate it as well as its first and second time derivatives

```
% I assume that you already read from an input file the string that is
% supposed to define the motion prescribed:
yourStringThatDefinesC = '(1.5*sin(t) + 3*t^2)^2'

cFunction = eval(yourStringThatDefinesC)
functionHandleValue = matlabFunction(cFunction)

cFunctionPrime = diff(yourStringThatDefinesC)
functionHandleFirstDeriv = matlabFunction(cFunctionPrime)

cFunctionPrimePrime = diff(diff(yourStringThatDefinesC))
functionHandleSecondDeriv = matlabFunction(cFunctionPrimePrime)

[v,f,s] = somePhiConstraint(functionHandleValue, functionHandleFirstDeriv, functionHandleSecondDeriv, 2.3)
```

```
function [value,firstD,secondD] = somePhiConstraint(f,fPrime,fPrimePrime,t)
% this is where you need to use C(t) and its derivatives...
value = f(t);
firstD = fPrime(t);
secondD = fPrimePrime(t);
```
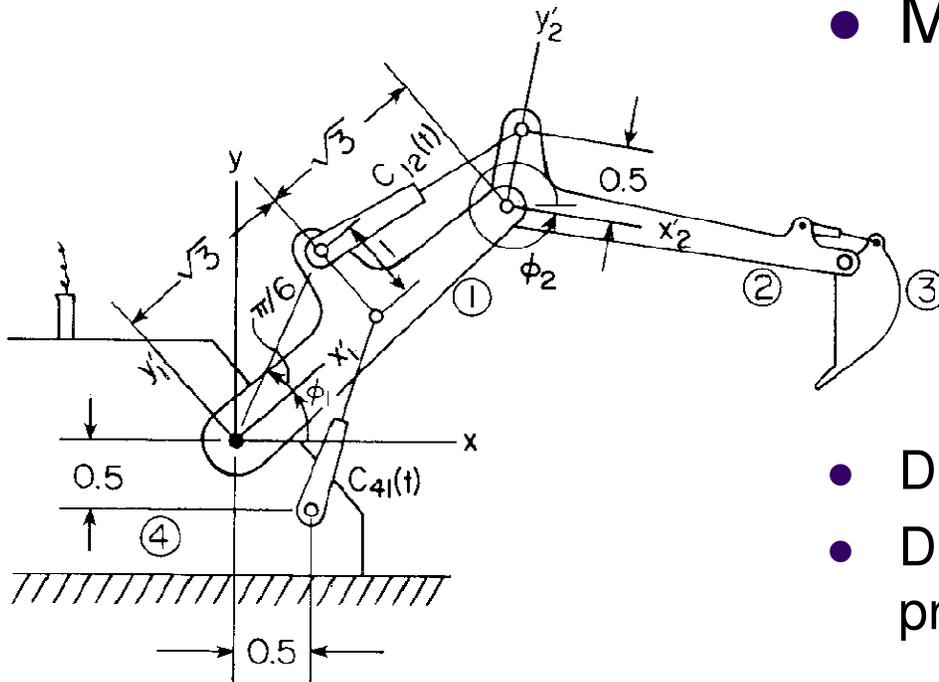
15

Example:
Specifying 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.

- Motions prescribed:

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

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

- Derive the constraints acting on system
- Derive linear system whose solution provides velocities $\dot{\mathbf{q}} = [\dot{\phi}_1, \dot{x}_2, \dot{y}_2, \dot{\phi}_2]^T$

16