“Without music to decorate it, time is just a bunch of boring production deadlines or dates by which bills must be paid.”

--Frank Zappa
Kinematics Analysis: Comments on the Three Stages

- The three stages of Kinematics Analysis: position analysis, velocity analysis, and acceleration analysis they each follow *very* similar recipes for finding for each body of the mechanism its position, velocity, and acceleration, respectively.

- ALL STAGES RELY ON THE CONCEPT OF JACOBIAN MATRIX:
  - $\Phi_q$ – the partial derivative of the constraints wrt the generalized coordinates

- ALL STAGES REQUIRE THE SOLUTION OF A SYSTEM OF EQUATIONS

$$\Phi_q \ x = b$$

Before we get started…

● Last Time:
  ● Wrap up, describing the orientation of a body using Euler parameters
  ● Connection between ang. velocity and time derivatives of Euler parameters
  ● Start discussion of Kinematic Analysis of mechanisms
    ● Kinematic constraints & driving constraints
    ● Degrees of freedom
    ● Position, Velocity, and Acceleration analysis of a mechanism

● Today:
  ● How to model joints typically encountered in mechanisms
    ● Start with four building blocks that are subsequently used in various combinations to capture the net effect of several mechanical joints
Step 1: Identify the **geometry of the motion** (the geometric constraint) whenever a physical joint is limiting the absolute or relative motion of a body

Step 2: Identify the **attributes** needed to fully describe the geometry of the motion

Step 3: Formulate the algebraic constraint equations $\Phi(q,t)=0$, that capture the effect of the geometric constraint

Step 4: Compute the Jacobian (or the sensitivity matrix) $\Phi_q$

Step 5: Compute $\nu$, the right side of the velocity equation

Step 6: Compute $\gamma$, the right side of the acceleration equation (tedious…)
Nomenclature & Notation

Conventions

- Geometric Constraint (GCon): a real world geometric feature of the motion of the mechanical system
  - Examples:
    - Particle moves around point (1,2,3) on a sphere of radius 2.0
    - A unit vector $\mathbf{u}_6$ on body 6 is perpendicular on a certain unit vector $\mathbf{u}_9$ on body 9
    - The y coordinate of point Q on body 8 is 14.5

- Algebraic Constraint Equations (ACEs): in the virtual world, a collection of one or more algebraic constraints, involving the generalized coordinates of the mechanism and possibly time $t$, that capture the geometry of the motion as induced by a certain Geometric Constraint
  - Examples:
    - $(x - 1)^2 + (y - 2)^2 + (z - 3)^2 - 4 = 0$
    - $\mathbf{u}_6^T \cdot \mathbf{u}_9 = 0$
    - $[0 \ 1 \ 0] \cdot \mathbf{r}_8^Q - 14.5 = 0$

- Modeling: the process that starts with the idealization of the real world to yield a GCon and continues with the GCon abstracting into a set of ACEs
The GCon Zoo: Basic GCons

- We have four basic GCons:
  - DP1: the dot product of two vectors on two bodies is specified
  - DP2: the dot product of a vector of on a body and a vector between two bodies is specified
  - D: the distance between two points on two different bodies is specified
  - CD: the difference between the coordinates of two bodies is specified

- Note:
  - DP1 stands for Dot Product 1
  - DP2 stands for Dot Product 2
  - D stands for distance
  - CD stands for coordinate difference
The GCon Zoo:
Intermediate + High Level GCons

- We have two Intermediate GCons:
  - B 1: a vector is B on a plane belonging to a different body
  - B 2: a vector between two bodies is B on a plane belonging to the different body

- We have a large number of High Level GCons (joints):
  - Spherical Joint (SJ)
  - Universal Joint (UJ)
  - Cylindrical Joint (CJ)
  - Revolute Joint (RJ)
  - Translational Joint (TJ)
  - Other composite joints (spherical-spherical, translational-revolute, etc.)
## The GCon Zoo: Overview

<table>
<thead>
<tr>
<th></th>
<th>( DP_1 )</th>
<th>( DP_2 )</th>
<th>( D )</th>
<th>( CD )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>××</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>×</td>
<td></td>
<td>×××</td>
</tr>
<tr>
<td>( S.J )</td>
<td></td>
<td></td>
<td></td>
<td>([×××]_{S.I})</td>
</tr>
<tr>
<td>( U.J )</td>
<td>×</td>
<td></td>
<td></td>
<td>([×××]_{S.I})</td>
</tr>
<tr>
<td>( C.J )</td>
<td>([××]_1)</td>
<td>([××]_2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T.J )</td>
<td>× ([××]_1)</td>
<td>([××]_2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note that there are other GCons that are used, but they see less mileage.
Basic GCon: DP1
Basic GCon: DP1

[Cntd.]

• Step 1. GCon $\Phi^{DP1}$ reflects the fact that motion is such that the dot product between a vector on body $i$ and a second vector on body $j$ assumes a specified value.

• Step 2. We have the following attributes (quantities required to properly define the GCon above):
  
  – Body $i$ and the associated L-RF$_i$. On that body, we need to know the algebraic vector $\bar{a}_i$
  – Body $j$ and the associated L-RF$_j$. On that body, we need to know the algebraic vector $\bar{a}_j$
  – The prescribed value that the dot product should assume. This prescribed value does not depend on GCs, but might depend on time. The prescribed value is specified through the function $f(t)$.

  * Most often, $f(t) = 0$, which indicates that the two vectors are orthogonal.
  * If $f(t)$ actually depends on time, this leads to $\Phi^{DP1}$ being a driving (rheonomic) constraint.

• Step 3. The ACE asserts that:

$$\Phi^{DP1}(i, \bar{a}_i, j, \bar{a}_j, f(l)) = \bar{a}_i^T A_i^T A_j \bar{a}_j - f(l) = 0$$
Basic GCon: DP1

[Cntd.]

- Step 4. The Jacobian for DP1: discussed next lecture

- Step 5. The $\nu$ term that enters the righthand side of the velocity equation assumes the form:

$$\nu^{DP1} = -\frac{\partial \Phi^{DP1}}{\partial t} = \frac{\partial f}{\partial t}$$

- Step 6. The $\gamma$ term that enters the righthand side of the acceleration equation assumes the form:

$$\gamma^{DP1} = -\tilde{a}_j^T (A_j^T A_i \ddot{\omega}_i \ddot{\omega}_i + \ddot{\omega}_j \ddot{\omega}_j A_j^T A_i) \ddot{a}_i + 2\ddot{\omega}_j^T \tilde{a}_j A_j^T A_i \ddot{a}_i \ddot{\omega}_i + \frac{\partial^2 f}{\partial t^2}$$

- **Note:** The $+$ term only depends on position and velocity information - Important since it is used to compute the acceleration and therefore it should not depend on acceleration (to prevent a circular argument)

- GCon-DP1 imposes one ACE and removes one DOF
Basic GCon: DP2
Basic GCon: DP2

[Cntd.]

- Step 1. GCon $\Phi^{DP2}$ reflects the fact that motion is such that the dot product between a vector $\vec{a}_i$ on body $i$ and a second vector $\vec{P}_iQ_j$ from body $i$ to body $j$ assumes a specified value.

- Step 2. We have the following attributes (quantities required to properly define the GCon above):
  - Body $i$ and the associated L RF$_i$. On that body we need to know (1) the algebraic vector $\vec{a}_i$, and (2) the location $\vec{s}_i^P$ of the point $P_i$.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\vec{s}_j^Q$ of the point $Q_j$
  - The prescribed value that the dot product should assume. This prescribed value does not depend on GCs, but might depend on time. The prescribed value is specified through the function $f(t)$.
    * Most often, $f(t) = 0$, which indicates that $\vec{a}_i$ and $\vec{P}_iQ_j$ are orthogonal.
    * If $f(t)$ actually depends on time, this leads to $\Phi^{DP2}$ being a driving (rheonomic) constraint.

- Step 3. The DP2-ACE asserts that:

\[
\Phi^{DP2}(i, \vec{a}_i, \vec{s}_i^P, j, \vec{s}_j^Q, f(t)) = \vec{a}_i^T A_i^T d_{ij} - f(t) = \vec{a}_i^T A_i^T (r_j + A_j s_j^Q - r_i - A_i s_i^P) - f(t) = 0
\]
Basic GCon: DP2

[Cntd.]

- Step 4. The Jacobian for DP2: discussed next lecture
- Step 5. The $\nu$ term that enters the righ-hand side of the velocity equation assumes the form:

$$\nu^{DP2} = -\frac{\partial \Phi^{DP2}}{\partial t} = \frac{\partial f}{\partial t}$$

- Step 6. The $\gamma$ term that enters the righ-hand side of the acceleration equation assumes the form:

$$\gamma^{DP2} = 2\tilde{\omega}_i^T \tilde{a}_i A_i^T (\dot{r}_i - \dot{r}_j) + 2[\bar{s}_j^Q]^T \tilde{\omega}_j A_j^T A_i \tilde{\omega}_i \tilde{a}_i - [\bar{s}_i^P]^T \tilde{\omega}_i \tilde{\omega}_i \tilde{a}_i$$

$$- [\bar{s}_j^Q]^T \tilde{\omega}_j \tilde{\omega}_j A_j^T A_i \tilde{a}_i - d_{ij}^T A_i \tilde{\omega}_i \tilde{\omega}_i \tilde{a}_i + \frac{\partial^2 f}{\partial t^2}$$

- GCon-DP2 imposes one ACE and as such removes one DOF
Basic GCon: D [Distance]
Basic GCon: D [Distance] [Cntd.]

- Step 1. GCon $\Phi^D$ reflects the fact that motion is such that the distance between point $P$ on body $i$ and point $Q$ on body $j$ assumes a specified value greater than zero.

- Step 2. GCon attributes (quantities required to properly define the GCon above):
  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\mathbf{s}_i^P$ of the point $P$.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\mathbf{s}_j^Q$ of the point $Q$.
  - The prescribed value that the distance between the two points assumes. This prescribed value does not depend on GCs, but might depend on time. The prescribed value is specified through the function $f(t)$.

    * Most often, $f(t) = C^2 > 0$, which defines a kinematic constraint (the power 2 emphasizes that the constant function assumes a positive value).

    * If $f(t)$ actually depends on time, this leads to $\Phi^D$ being a driving (rheonomic) constraint.

- Step 3. The D-ACE asserts that:

$$\Phi^D(i, \mathbf{s}_i^P, j, \mathbf{s}_j^Q, f(t)) = d_{ij}^T \mathbf{d}_{ij} - f(t) = (\mathbf{r}_j + \mathbf{A}_j \mathbf{s}_j^Q - \mathbf{r}_i - \mathbf{A}_i \mathbf{s}_i^P)^T (\mathbf{r}_j + \mathbf{A}_j \mathbf{s}_j^Q - \mathbf{r}_i - \mathbf{A}_i \mathbf{s}_i^P) - f(t) = 0$$
Basic GCon: D [Distance] [Cntd.]

- Step 4. The Jacobian for D: discussed next lecture

- Step 5. The $\nu$ term that enters the right-hand side of the velocity equation assumes the form:

  $$\nu^D = -\frac{\partial \Phi^D}{\partial t} = \frac{\partial f}{\partial t}$$

- Step 6. The $\gamma$ term that enters the right-hand side of the acceleration equation assumes the form:

  $$\gamma^D = -2(\dot{\mathbf{r}}_j - \dot{\mathbf{r}}_i)^T (\ddot{\mathbf{r}}_j - \ddot{\mathbf{r}}_i) + 2[\mathbf{s}_j^Q]^T \ddot{\omega}_j \vec{\omega}_j \mathbf{s}_j^Q + 2[\mathbf{s}_i^P]^T \ddot{\omega}_i \vec{\omega}_i \mathbf{s}_i^P - 4[\mathbf{s}_j^Q]^T \ddot{\omega}_j \mathbf{A}_j^T \mathbf{A}_i \ddot{\omega}_i \mathbf{s}_i^P$$

  $$+ 4(\dot{\mathbf{r}}_j - \dot{\mathbf{r}}_i)^T (\mathbf{A}_i \ddot{\omega}_i \mathbf{s}_i^P - \mathbf{A}_j \ddot{\omega}_j \mathbf{s}_j^Q) - 2d_{ij} (\mathbf{A}_j \ddot{\omega}_j \mathbf{s}_j^Q - \mathbf{A}_i \ddot{\omega}_i \mathbf{s}_i^P) + \frac{\partial^2 f}{\partial t^2}$$

- Note: GCon-D imposes one ACE and as such it removes one DOF
Basic GCon: CD [Coordinate Difference] (Cntd.)

Points $P$ and $Q$ entering $\Phi^{CD}$

Body $i$

Body $j$

Points $P$ and $Q$ entering $\Phi^{CD}$
Basic GCon: CD

[Cntd.]

- Step 1. GCon $\Phi^{CD}$ reflects the fact that motion is such that the difference between the $x$ (or $y$ or $z$) coordinate of point $P$ on body $i$ and the $x$ (or $y$ or $z$) coordinate of point $Q$ on body $j$ assumes a specified value.

- Step 2. GCon attributes:
  - The coordinate $c$ of interest: $c \in \{i,j,k\}$
  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\bar{s}^i_P$ of the point $P$.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\bar{s}^Q_j$ of the point $Q$.
  - The prescribed value that the coordinate difference assumes. This prescribed value does not depend on GCs, but might depend on time. The prescribed value is specified through the function $f(t)$.

  * If $f(t) = \text{const.}$, $\Phi^{CD}$ defines a kinematic constraint. Otherwise, it defines a driving (rheonomic) constraint.

  * In many cases the second body $j$ ends up being the ground. In this case, by convention, $j = 0$ (the G-RF is attached to body 0).

- Step 3. The CD-ACE asserts that:

$$\Phi^{CD}(c, i, \bar{s}^i_P, j, \bar{s}^Q_j, f(t)) = c^T \mathbf{d}_{ij} - f(t) = c^T(\mathbf{r}_j + \Lambda_j \bar{s}^Q_j - \mathbf{r}_i - \Lambda_i \bar{s}^P_i) - f(t) = 0$$
Basic GCon: CD

[Cntd.]

- Step 4. The Jacobian for CD: discussed next lecture

- Step 5. The $\nu$ term that enters the right-hand side of the velocity equation assumes the form:

$$\nu^D = -\frac{\partial \Phi^D}{\partial t} = \frac{\partial f}{\partial t}$$

- Step 6. The $\gamma$ term that enters the right-hand side of the acceleration equation assumes the form:

$$\gamma^{CD} = \mathbf{c}^T (\mathbf{A}_i \ddot{\mathbf{w}}_i \mathbf{s}_i^P - \mathbf{A}_j \ddot{\mathbf{w}}_j \mathbf{s}_j^Q) + \frac{\partial^2 f}{\partial t^2}$$

- Note: GCon-CD imposes one ACE and as such it removes one DOF
Intermediate GCon: $\mathbb{B} 1$

[Perpendicular 1]

\[ \Phi_{\perp 1} : c_{j} \perp \mathcal{P}(a_{i}, b_{i}) \]
Intermediate GCon: B 1

[Cntd.]

- Step 1. GCon $\Phi^{-1}$ reflects the fact that motion is such that a vector $c_j$ on body $j$ is perpendicular on a plane of body $i$. This plane is defined by specifying two noncolinear vectors $a_i$ and $b_i$ that are contained in that plane. Another way to state GCon $\Phi^{-1}$ is to say that $c_j$ is parallel to the normal of the said plane. This GCon is built using GCon-DP1 twice. As such, it introduces two ACEs and therefore removes two DOFs.

- Step 2. GCon $\Phi^{-1}$ has the following attributes:
  - Body $i$ and the associated L-RF$_i$. The vectors $\bar{a}_i$ and $\bar{b}_i$.
  - Body $j$ and the associated L-RF$_j$. The vector $\bar{c}_j$.

- Step 3. The $\perp$1-ACE asserts that:
  $$\Phi^{-1}(i, \bar{a}_i, \bar{b}_i, j, \bar{c}_j) = \begin{bmatrix} \Phi^{DP1}(i, \bar{a}_i, j, \bar{c}_j, 0) \\ \Phi^{DP1}(i, \bar{b}_i, j, \bar{c}_j, 0) \end{bmatrix} = \begin{bmatrix} \bar{a}_i^T A_i^T A_j \bar{c}_j \\ \bar{b}_i^T A_i^T A_j \bar{c}_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Steps 4, 5, and 6: see discussion for GCon DP1.
Intermediate GCon: $B_2$

[Perpendicular 2]

\[ \Phi_{\perp 2}: \mathbf{d}_{ij} \perp \mathcal{P}(\mathbf{a}_i, \mathbf{b}_i) \]
Intermediate GCon: B 2
[Perpendicular 2][Cntd.]

- Step 1. GCon $\Phi^{\perp 2}$ reflects the fact that motion is such that a vector $\overrightarrow{P_i Q_j}$ from body $i$ to body $j$ remains perpendicular to a plane defined by two vectors $\vec{a}_i$ and $\vec{b}_i$. The GCon is derived by applying twice the basic GCon-DP2. As such, this GCon leads to two ACEs and it removes two DOFs.

- Step 2. GCon $\perp 2$ has the following attributes:
  - Body $i$ and the associated L-RF$_i$. On that body we need to know (1) the location $s^P_i$ of the point $P_i$, (2) the algebraic vector $\vec{a}_i$, and (3) the algebraic vector $\vec{b}_i$.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $s^Q_j$ of the point $Q_j$.

- Step 3. The $\perp 2$-ACE asserts that:

$$\Phi^{\perp 2}(i, \vec{a}_i, \vec{b}_i, s^P_i, j, s^Q_j) = \begin{bmatrix} \Phi^{DP2}(i, \vec{a}_i, s^P_i, j, s^Q_j, 0) \\ \Phi^{DP2}(i, \vec{b}_i, s^P_i, j, s^Q_j, 0) \end{bmatrix} = \begin{bmatrix} \vec{a}_i^T A_i^T d_{ij} \\ \vec{b}_i^T A_i^T d_{ij} \end{bmatrix} = 0$$

- Steps 4, 5, and 6: see discussion for GCon D2.
High Level GCon: SJ [Spherical Joint]

\[ \Phi^{SJ} = \begin{bmatrix} \Phi^{CD}(i, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \\ \Phi^{CD}(j, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \\ \Phi^{CD}(k, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \end{bmatrix} \]
High Level GCon: Spherical Joint

• Step 1. GCon $\Phi^{SJ}$ reflects the fact that motion is such that point $P$ on body $i$ and point $Q$ on body $j$ coincide at all times. This is equivalent to saying the the difference between the $x$, $y$, and $z$ coordinates of these two points is zero. The GCon therefore is implemented by a set of three GCon-CD and as such it removes three DOFs.

• Step 2. The GCon has the following attributes (inherited from $\Phi^{CD}$):
  – Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\bar{s}_i^P$ of the point $P$.
  – Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\bar{s}_j^Q$ of the point $Q$.

• Step 3. The SJ-ACE asserts that:

$$\Phi^{SJ}(i, \bar{s}_i^P, j, \bar{s}_j^Q) = \begin{bmatrix} \Phi^{CD}(i, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \\ \Phi^{CD}(j, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \\ \Phi^{CD}(k, i, \bar{s}_i^P, j, \bar{s}_j^Q, 0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

• Steps 4, 5, and 6: draw on the corresponding steps for $\Phi^{CD}$.
High Level GCon: CJ [Cylindrical Joint]
High Level GCon: CJ

- Step 1. GCon $\Phi^{CJ}$ reflects the motion associated with a cylindrical joint, which geometrically requires that two vectors on two bodies should stay collinear at all times. According to the figure, if body $j$ is fixed, body $i$ can slide and rotate around a specified axis and therefore has two DOFs.

- Step 2. The rotation/sliding axis is defined on body $j$ by $\bar{c}_j$. On body $i$, it is defined by the normal to the plane defined by two vectors $\bar{a}_i$ and $\bar{b}_i$. Additionally, consider point $Q_j$ as being the origin of $\bar{c}_j$. This point together with $P_i$ defines the axis of the cylindrical joint. Without any loss of generality, $P_i$ will be considered the origin of $\bar{a}_i$ and $\bar{b}_i$. The GCon has the following attributes (inherited from $\Phi^{1.1}$ and $\Phi^{1.2}$):
  
  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\bar{s}_i^P$ of the point $P$, and the directions $\bar{a}_i$ and $\bar{b}_i$.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\bar{s}_j^Q$, the origin of the vector $\bar{c}_j$.

- Step 3. The CJ-ACE asserts that:

$$\Phi^{CJ}(i, \bar{s}_i^P, \bar{a}_i, \bar{b}_i, j, \bar{s}_j^Q, \bar{c}_j) = \begin{bmatrix} \Phi^{1.1}(i, \bar{a}_i, \bar{b}_i, j, \bar{c}_j) \\ \Phi^{1.2}(i, \bar{a}_i, \bar{b}_i, \bar{s}_i^P, j, \bar{s}_j^Q) \end{bmatrix} = \begin{bmatrix} 0_2 \\ 0_2 \end{bmatrix}$$

- Steps 4, 5, and 6: draw on the corresponding steps for $\Phi^{1.1}$ and $\Phi^{1.2}$. 
High Level GCon: TJ
[Translational Joint]
High Level GCon: TJ

- Step 1. GCon $\Phi^{TJ}$ reflects the motion associated with a translational joint, which geometrically is similar to a cylindrical joint with the caveat that the rotational degree of freedom of the latter is suppressed. According to the figure, if body $j$ is fixed, body $i$ can slide up and down. Therefore, the joint allows one DOF of relative motion.

- Step 2. The sliding axis is defined on body $j$ by $\bar{c}_j$. On body $i$, it is defined by the normal to the plane defined by two vectors $\bar{a}_i$ and $\bar{b}_i$. Additionally, consider the points $P_i$ and $Q_j$ as defining the axis of translation. Since the relative rotation about the translation direction is suppressed, a vector $\mathbf{a}_j$ on body $j$ parallel to $\mathcal{P}(\mathbf{a}_i, \mathbf{b}_i)$ enters the definition of the joint. Without any loss of generality, $P_i$ can be considered the origin of $\mathbf{a}_i$, $\mathbf{b}_i$, while $Q_j$ the origin of $\mathbf{a}_j$ and $\mathbf{c}_j$. In the end, the GCon has the following attributes (inherited from $\Phi^{CJ}$ and $\Phi^{DP1}$):

  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\bar{s}_i^P$ of the point $P$, and the directions $\bar{a}_i$ and $\bar{b}_i$, both with origin at $P$

  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\bar{s}_j^Q$, the origin of the vectors $\bar{a}_j$ and $\bar{c}_j$.

- Step 3. The TJ-ACE asserts that:

$$
\Phi^{TJ}(i, \bar{s}_i^P, \bar{a}_i, \bar{b}_i, j, \bar{s}_j^Q, \bar{a}_j, \bar{c}_j) = \begin{bmatrix}
\Phi^{CJ}(i, \bar{s}_i^P, \bar{a}_i, \bar{b}_i, j, \bar{s}_j^Q, \bar{c}_j) \\
\Phi^{DP1}(i, \bar{a}_i, j, \bar{a}_j, \text{const.})
\end{bmatrix} = \begin{bmatrix}
0_4 \\
0
\end{bmatrix}
$$

- Steps 4, 5, and 6: draw on the corresponding steps for $\Phi^{DP1}$ and $\Phi^{CJ}$. 

High Level GCon: RJ [Revolute Joint]
High Level GCon: RJ

- Step 1. GCon $\Phi^{RJ}$ reflects the motion associated with a revolute joint (also called hinge or rotational joint). According to the figure, if body $i$ is fixed, body $j$ can rotated around the hinge axis. Note that body $j$ has one DOF relative to body $i$.

- Step 2. The joint is defined as follows: first, note that this is very similar to a spherical joint, in that two points $P_i$ and $Q_j$ coincide at all times. Additionally, a vector $c_j$ of origin $Q_j$ remains perpendicular on a plane $\mathcal{P}(a_i, b_i)$. Without loss of generality, we can assume that $P_i$ is the origin of $a_i$ and $b_i$. The GCon has the following attributes (inherited from $\Phi^{\perp 1}$ and $\Phi^{SJ}$):

  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\bar{s}_i^P$ of the point $P$, and $\bar{a}_i$ and $\bar{b}_i$. The normal to the plane $\mathcal{P}(a_i, b_i)$ defines the direction of the axis of rotation on body $i$.

  Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\bar{s}_j^Q$ of the point $Q$, and the direction $\bar{c}_j$. The latter defines the direction of the axis of rotation on body $j$.

- Step 3. The RJ-ACE asserts that:

$$
\Phi^{RJ}(i, s_i^P, a_i, b_i, j, s_j^Q, c_j) = \begin{bmatrix}
\Phi^{SJ}(i, \bar{s}_i^P, j, \bar{s}_j^Q) \\
\Phi^{\perp 1}(i, \bar{a}_i, \bar{b}_i, j, \bar{c}_j)
\end{bmatrix} = \begin{bmatrix}
0_3 \\
0_2
\end{bmatrix}
$$

- Steps 4, 5, and 6: draw on the corresponding steps for $\Phi^{SJ}$ and $\Phi^{\perp 1}$. 
High Level GCon: UJ [Universal Joint]

$$\Phi^{UJ} = \begin{bmatrix} \Phi^{SJ}(i, \bar{s}_i^P, j, \bar{s}_j^Q) \\ \Phi^{DP1}(i, \bar{a}_i, j, \bar{a}_j, 0) \end{bmatrix}$$

Figure 9.4.15  Singular behavior of universal joint.
High Level GCon: UJ [Universal Joint]

- Step 1. GCon $\Phi^{UJ}$ reflects the motion associated with a universal joint. According to the figure, if body $i$ is rotated, body $j$ starts rotated as well. The joint is defined as follows: first, note that this is very similar to a spherical joint, in that points $P_i$ and $Q_j$ coincide at all times. However, we need to capture that a rotation of $i$ induces a rotation of $j$. This is enforced by a requirement that vectors $\mathbf{a}_i$ and $\mathbf{a}_j$, which together define the cross of the joint, should stay at all times orthogonal. Note when body $i$ is held fixed, body $j$ can undertake two rotations around each of the axes $\mathbf{a}_i$ and $\mathbf{a}_j$ (the cross) of the UJ. Therefore, body $j$ has two DOFs relative to body $i$.

- Step 2. The GCon has the following attributes (inherited from $\Phi^{DP1}$ and $\Phi^{SJ}$):
  - Body $i$ and the associated L-RF$_i$. On that body we need to know the location $\mathbf{s}^P_i$ of the point $P$, and the direction $\mathbf{a}_i$, which defines half of the cross.
  - Body $j$ and the associated L-RF$_j$. On that body, we need to know the location $\mathbf{s}^Q_j$ of the point $Q$, and the direction $\mathbf{a}_j$, which defines the other half of the cross.

$$\Phi^{UJ}(i, \mathbf{s}^P_i, \mathbf{a}_i, j, \mathbf{s}^Q_j, \mathbf{a}_j) = \begin{bmatrix}
\Phi^{CD}(i, i, \mathbf{s}^P_i, j, \mathbf{s}^Q_j, 0) \\
\Phi^{CD}(j, i, \mathbf{s}^P_i, j, \mathbf{s}^Q_j, 0) \\
\Phi^{CD}(k, i, \mathbf{s}^P_i, j, \mathbf{s}^Q_j, 0) \\
\Phi^{DP1}(i, \mathbf{a}_i, j, \mathbf{a}_j, 0)
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}$$

- Steps 4, 5, and 6: draw on the corresponding steps for $\Phi^{CD}$ and $\Phi^{DP1}$. 
GCon’s: Concluding Remarks

- All the basic geometric constraints are scalar conditions
- If you want to implement your own Kinematics solver, with only four basic constraints you can cover 80% of the most used GCons
- The approach outlined for defining various GCons is not unique
  - Don’t have to use the four building blocks, you can have each GCon be defined in its own specific way
    - If you do this, you lose generality, hard to write software to generate a library of GCons
- Unfinished business: We still have to produce the Jacobian matrices for the four basic geometric constraints
Kinematics Analysis: Comments on the Three Stages

- The three stages of Kinematics Analysis: **position** analysis, **velocity** analysis, and **acceleration** analysis they each follow *very* similar recipes for finding for each body of the mechanism its position, velocity, and acceleration, respectively

- **ALL STAGES RELY ON THE CONCEPT OF JACOBIAN MATRIX:**
  - $\Phi_q$ – the partial derivative of the constraints wrt the generalized coordinates

- **ALL STAGES REQUIRE THE SOLUTION OF A SYSTEM OF EQUATIONS**
  
  $$\Phi_q \ x = b$$

“When You Come to a Fork in the Road, Take It!”
Yogi Berra
Before we get started…

● Last Time:
  ● Geometric Constraints
    ● Covered four Basic and one Intermediate GCon’s
    ● Recall that more complex GCon’s are obtained by combining the four basic ones: DP1, DP2, D, CD

● Today:
  ● Finish Geometric Constraints
  ● Talk about their partial derivatives

● HW5 – due on Feb. 25
  ● Posted online

● Reading: Example 9.4.7
The $p - \omega$ Fork

“When You Come to a Fork in the Road, Take It!”
Yogi Berra
The $\mathbf{p} - \omega$ Fork

- This could have been equally well called the $\mathbf{p} - \bar{\omega}$ fork

- The fork:
  
  - When carrying out Velocity Analysis, should you try to solve a linear system for $\mathbf{p}$ and recover $\bar{\omega}$, or the other way around?
    
    * Recall that $\bar{\omega} = 2 \mathbf{G} \dot{\mathbf{p}}$, but equally well, $\dot{\mathbf{p}} = \frac{1}{2} \mathbf{G}^T \bar{\omega}$

  - Similarly, when carrying out Acceleration Analysis, should you try to solve a linear system for $\ddot{\mathbf{p}}$ and recover $\dot{\bar{\omega}}$, or the other way around?

- Recall that when solving for the position, you must stick with $\mathbf{p}$ when solving the nonlinear system $\Phi(\mathbf{q}, t) = 0_m$
The $\mathbf{p} - \omega$ Fork

- Why do we have a fork, after all?
  - Focus discussion on the Velocity Analysis, same arguments carry over to Acceleration Analysis discussion

- Fork shows up do to the fact that $\dot{\Phi}(\mathbf{q}, t)$ is a linear function of $\dot{\mathbf{p}}$ if I stick with having the Euler Parameters as unknowns, but it is also a linear function of $\dot{\omega}$ (or $\omega$), if I decide to make the angular velocity the unknown in the Velocity Analysis problem

- Specifically, using Ilaug book’s notation, we can express the time derivative $\dot{\Phi}(\mathbf{q}, t)$ in one of the following two ways (recall that $\mathbf{q} = \left[ \begin{array}{c} \mathbf{r} \\ \mathbf{p} \end{array} \right]_{nc}$):

$$\frac{d\Phi(\mathbf{r}, \mathbf{p}, t)}{dt} = \Phi_r \dot{\mathbf{r}} + \Phi_p \dot{\mathbf{p}} + \Phi_t = \left[ \Phi_r \quad \Phi_p \right] \left[ \begin{array}{c} \dot{\mathbf{r}} \\ \dot{\mathbf{p}} \end{array} \right] + \Phi_t = 0_{7nb} \Rightarrow \Phi_q \dot{\mathbf{q}} = \nu_{7nb}$$

$$\frac{d\Phi(\mathbf{r}, \mathbf{p}, t)}{dt} = \Phi_r \dot{\mathbf{r}} + \bar{\Pi} \dot{\omega} + \Phi_t = \left[ \Phi_r \quad \bar{\Pi} \right] \left[ \begin{array}{c} \dot{\mathbf{r}} \\ \dot{\omega} \end{array} \right] + \Phi_t = 0_{6nb} \Rightarrow \bar{\mathbf{J}} \left[ \begin{array}{c} \dot{\mathbf{r}} \\ \dot{\omega} \end{array} \right] = \nu_{6nb}$$

- Conclusion: depending on which path we take, to carry out Velocity Analysis, we need $\Phi_r$ and $\Phi_p$, or $\Phi_r$ and $\bar{\Pi}$
The $p - \omega$ Fork  [Cntd.]

- Note that we use a different notation than used in Haug’s book (table 9.4.1, pp. 357). Therein, the quantity $\bar{\Pi}$ is called $\Phi_{\bar{\pi}}$. In this class we reserve a subscript to denote a partial derivative with respect to that variable.
  - In this case, (i) $\bar{\pi}$ is not a variable, and (ii) $\Phi_{\bar{\pi}}$ is not a partial derivative.
  - Specifically, by definition, $\Pi$ is the coefficient matrix that multiplies $\bar{\omega}$ in the expression of the time derivative $\Phi(q, t)$

- Note that there is the $J = [\Phi_f \ \Pi]$ matrix that can be used to compute directly $\omega$ instead of $\bar{\omega}$:

$$\frac{d\Phi(r, p, t)}{dt} = \Phi_r \dot{r} + \Pi \omega + \Phi_t = [\Phi_f \ \Pi] \begin{bmatrix} \dot{r} \\ \omega \end{bmatrix} + \Phi_t = 0_{6nb} \Rightarrow J \begin{bmatrix} \dot{r} \\ \omega \end{bmatrix} = \nu_{6nb}$$

- Finally, note that for the Acceleration Analysis, one ends up with the same scenario, where the problem can be formulated in one of the following forms

$$\Phi_q \ddot{q} = \gamma_{7nb} \quad \text{or} \quad \bar{J} \begin{bmatrix} \dddot{r} \\ \dddot{\omega} \end{bmatrix} = \gamma_{6nb} \quad \text{or} \quad J \begin{bmatrix} \dddot{r} \\ \dddot{\omega} \end{bmatrix} = \gamma_{6nb}$$
The $\Phi_p$ versus $\bar{\Pi}$ Issue

- $\Phi_r$, $\Phi_p$, and $\bar{\Pi}$ are needed for both Kinematics and Dynamics Analysis

- The drill is as follows: first compute the partial derivatives $\Phi_r$ and $\Phi_p$ for the four basic constraints: DP1, DP2, D, CD
  
  - Recall that all the other intermediate and higher level GCon’s are obtained by stacking together partials of DP1, DP2, D, CD. Therefore, all we need is $\Phi_r$, $\Phi_p$, and $\bar{\Pi}$ for these four basic GCon building blocks
  
  - Note that all basic GCcon’s depend on the generalized coordinates of two bodies: $i$, and $j$. Therefore, when taking the partial derivative with respect to $r = \begin{bmatrix} r_1 \\ \vdots \\ r_{nb} \end{bmatrix}$ or $p = \begin{bmatrix} p_1 \\ \vdots \\ p_{nb} \end{bmatrix}$, at most two sets of non-zero entries are obtained in $\Phi_r$ and $\Phi_p$, respectively.

- At the end, we will also look into how to compute $\bar{\Pi}$ (these are provided in Haug’s book)
**[Short Detour]**

**Computing** $[\mathbf{A}(\mathbf{p})\bar{\mathbf{a}}]_{\mathbf{p}}$

- Note that,

$$
[\mathbf{A}(\mathbf{p})\bar{\mathbf{a}}]_{\mathbf{p}} = \begin{bmatrix} (e_0^2 - e^T e) \bar{a} + 2 (ee^T + e_0 \bar{e}) \bar{a} \end{bmatrix}_{\mathbf{p}}
= \begin{bmatrix} 2e_0 \bar{a} + 2\bar{e}a & -2\bar{e}a^T + 2e^T \bar{a}I_3 + 2ea^T - 2e_0 \bar{a} \end{bmatrix}
= 2 \begin{bmatrix} (e_0 I_3 + \bar{e}) \bar{a} & ea^T - (e_0 I_3 + \bar{e}) \bar{a} \end{bmatrix}
$$

- The following identities were used to obtain the result above:

$$(e^T e \bar{a})_{e} = (\bar{e}e^T e)_{e} = \bar{a}(e^T e)_{e} = 2\bar{e}a^T$$ \& \quad (ee^T \bar{a})_{e} = (e(\bar{a}^T e))_{e} = ea^T + (\bar{a}^T e) I_3

- We define a matrix $\mathbf{B} \in \mathbb{R}^{3 \times 4}$ given two vectors $\mathbf{p} \in \mathbb{R}^4$ and $\bar{\mathbf{a}} \in \mathbb{R}^3$ as

$$
\mathbf{B}(\mathbf{p}, \bar{\mathbf{a}}) \equiv 2 \begin{bmatrix} (e_0 I_3 + \bar{e}) \bar{a} & ea^T - (e_0 I_3 + \bar{e}) \bar{a} \end{bmatrix}
$$

- Then, the partial of interest is obtained as

$$
[\mathbf{A}(\mathbf{p})\bar{\mathbf{a}}]_{\mathbf{p}} = \mathbf{a}_\mathbf{p} = \mathbf{B}(\mathbf{p}, \bar{\mathbf{a}})
$$
“In China if you are one in a million – there are 1,300 other people just like you.”
Bill Gates
Before we get started...

- **Last Time:**
  - Finished Geometric Constraints
  - Started discussion about their partial derivatives

- **Today:**
  - Discussion of assignment
  - Finish partial derivatives
  - Discuss computation of $\Pi$
  - Quick remarks on Position Analysis + Newton Raphson
  - Start the dynamics problem

- **HW6** – due on March 4, posted online today
  - Heads up: next week’s HW7 will ask you to define your ME751 Final Project

- **Student Feedback:** posted online on class website
  - Only two students provided feedback
Today’s Assigned HW

● First in a series of assignments that each contains a MATLAB component
  ● For first one you’ll have to implement computational support for two basic GCon’s

● You will be coding over the next four or five HWs a series of MATLAB functions that in the end should form a cohesive 3D Simulation Engine for Kinematics and Dynamics Analysis

● On Th, March 4, Hammad will give a 15 minute presentation of the steps you need to take in order to visualize the motion of your model
  ● Using Ogre, an open source 3D Graphics Engine (http://www.ogre3d.org/)

● The task of putting together 3D Simulation Engine can become your Final Project
3D Simulation Engine: Comments

- At least three A grades will be assigned in conjunction with 3D Simulation Engine:
  - Category Speed: The student with the fastest code on a benchmark problem gets an automatic A grade in ME751
  - Category Functionality: The student with the implementation with the richest functionality will get an automatic A grade
    - Functionality: defined as the number of GCon’s supported, analysis modes implemented (Kinematics, Dynamics, Equilibrium, Inverse Dynamics, etc.), integration method supported, etc.
  - Category Pre-Processing: The student with the best way to define a multi-body model (preferably GUI-based model definition) will get an automatic A grade
    - Might or might not be offered
  - Category Post-Processing: The student with the best way to visualize simulation results will get an automatic A grade
3D Simulation Engine: Comments

- The code of each of you will be made available online at the course website (under the “Students’ Code” link)
  - Privacy concerns: we’ll use code names to protect your identity
    - Please send me your code name (Examples: jamesbond, batman, robinhood, etc.)
      - If you don’t provide a code name I’ll assign one and email to you (you don’t want this to happen)
      - Use the following email message “ME751 code name: jamesbond”. Don’t include anything in the body of the email since I’m not going to read it

- The TA will not debug your code. He’ll only try to validate it using a set of different input data

- If you did mess up the code for week 2 (you’ll know if this is the case once you get your HW score back), when working on week 3 assignment it might be wise to look on the class website at other people’s code and recycle it
  - NOTE: For this approach to work, I will not accept late HW
3D Simulation Engine: Comments

- When you email your code; that is, when you submit your homework, zip all your MATLAB files and email to me
  - Include *all* your MATLAB files; i.e., Week 3 will include your week 2 and week 1 files, etc.
  - Use the following email subject: “ME751: 3D Simulation Engine” and don’t include anything *except* a zip file. Any questions you might have, email them separately

- The way the TA will check your code will be by running a MATLAB file called “simEngine3D.mat”
  - In other words, make sure you run your code by executing this MATLAB file, which in turn calls other MATLAB files that you implement
  - Why? Because we don’t want to learn how to run the program of each and every one of you. This will enforce a unique entry point to all 3D Simulation Engine developed in ME751

- This is a multi-week project – please make sure you **USE COMMENTS** heavily in your code. Otherwise, in week 4 you’ll struggle to understand what you did in week 1
Basic GCon DP1: $\Phi^{DP1}_r$ and $\Phi^{DP1}_p$

- Recall that
  \[ \Phi^{DP1}(i, \tilde{a}_i, j, \tilde{a}_j, f(t)) = \tilde{a}_i^T A_i^T A_j \tilde{a}_j - f(t) = a_i^T a_j - f(t) = 0 \]

- Then, it follows that
  \[ \frac{\partial \Phi^{DP1}}{\partial r_i} = 0_{1 \times 3} \quad \frac{\partial \Phi^{DP1}}{\partial p_i} = a_j^T B(p_i, \tilde{a}_i) \]
  \[ \frac{\partial \Phi^{DP1}}{\partial r_j} = 0_{1 \times 3} \quad \frac{\partial \Phi^{DP1}}{\partial p_j} = a_i^T B(p_j, \tilde{a}_j) \]

- Putting it all together (note that $\Phi^{DP1}_q \in \mathbb{R}^{1 \times 7nb}$),
  \[ \Phi^{DP1}_q = \begin{bmatrix} 0_{1 \times 3} & \cdots & 0_{1 \times 3} & \cdots & 0_{1 \times 4} & \frac{\partial \Phi^{DP1}}{\partial p_i} & 0_{1 \times 4} & \cdots & 0_{1 \times 4} & \frac{\partial \Phi^{DP1}}{\partial p_j} & 0_{1 \times 4} & \cdots & 0_{1 \times 4} \end{bmatrix} \]
  \[ \text{Partials with} \quad \text{Partials with} \]
  \[ \text{respect to } r \quad \text{respect to } p \]

  \[ = \begin{bmatrix} 0_{1 \times 3} & \cdots & 0_{1 \times 3} & \cdots & 0_{1 \times 3} & \cdots & 0_{1 \times 4} & a_j^T B(p_i, \tilde{a}_i) & 0_{1 \times 4} & \cdots & 0_{1 \times 4} & a_i^T B(p_j, \tilde{a}_j) & 0_{1 \times 4} & \cdots & 0_{1 \times 4} \end{bmatrix} \]

  \[ \begin{array}{cccccc}
  \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
  \text{Body} & \text{Body} & \text{Body} & \text{Body} & \text{Body} & \text{Body} \\
  1, r & i, r & j, r & i-1, p & i+1, p & j-1, p \\
  \end{array} \]
Computing $[d_{ij}]_q$

- Recall that

$$d_{ij} = r_j + A_j \bar{s}_j^Q - r_i - A_i \bar{s}_i^P = r_j + s_j^Q - r_i - s_i^P$$

- It follows that

$$[d_{ij}]_{q_i,q_j} = [-I_3, -(s_i^P)_{p_i}, I_3, (s_j^Q)_{p_j}]$$

$$= [-I_3, -B(p_i, \bar{s}_i^P), I_3, B(p_j, \bar{s}_j^Q)]$$
Basic GCon DP2: $\Phi_r^{DP2}$ and $\Phi_p^{DP1}$

- Recall that

$$\Phi_{q_i,q_j}^{DP2}(i, \tilde{a}_i, \tilde{s}_i^P, j, \tilde{s}_j^Q, f(t)) = a_i^T A_i^T d_{ij} - f(t) = a_i^T d_{ij} - f(t) = 0$$

- It follows that

$$\Phi_{q_i,q_j}^{DP2}(a_i, d_{ij}) = a_i^T (d_{ij})_{q_i,q_j} + d_{ij}^T (a_i)_{q_i,q_j}$$

$$= a_i^T \begin{bmatrix} -I_3 & -B(p_i, \tilde{s}_i^P) & I_3 & B(p_j, \tilde{s}_j^Q) \end{bmatrix} + d_{ij}^T \begin{bmatrix} 0 & B(p_i, \tilde{a}_i) & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_i^T d_{ij} B(p_i, \tilde{a}_i) - a_i^T B(p_i, \tilde{s}_i^P) & a_i^T & a_i^T B(p_j, \tilde{s}_j^Q) \end{bmatrix}$$
Basic GCon D: $\Phi^D_r$ and $\Phi^D_p$

- Recall that the GCon-D assumes the expression

$$\Phi^D(i, \bar{s}^P_i, j, \bar{s}^Q_j, f(t)) = d_{ij}^T d_{ij} - f(t) = 0$$

- It follows that

$$\Phi^D_{q_i, q_j} = (d_{ij}^T d_{ij})_{q_i, q_j} = 2d_{ij}^T [d_{ij}]_{q_i, q_j}$$

$$= 2d_{ij}^T \begin{bmatrix} -I_3 & -B(p_i, \bar{s}^P_i) & I_3 & B(p_j, \bar{s}^Q_j) \end{bmatrix}$$
Basic GCon CD: $\Phi_r^{CD}$ and $\Phi_p^{CD}$

• Recall that the GCon-CD assumes the expression

$$\Phi^{CD}(c, i, \bar{s}_i^P, j, \bar{s}_j^Q, f(t)) = c^T d_{ij} - f(t) = 0$$

• It follows that

$$\Phi_{q_i, q_j}^{CD} = (c^T d_{ij})_{q_i, q_j} = c^T [d_{ij}]_{q_i, q_j}$$

$$= c^T \begin{bmatrix} -I_3 & -B(p_i, \bar{s}_i^P) & I_3 & B(p_j, \bar{s}_j^Q) \end{bmatrix}$$

$$= \begin{bmatrix} -c^T & -c^T B(p_i, \bar{s}_i^P) & c^T & c^T B(p_j, \bar{s}_j^Q) \end{bmatrix}$$
End Computing Partial Derivatives $\Phi_r$ and $\Phi_p$

Start Computing $\Pi$. 
Computing $\vec{\Pi}$

- Recall that in last lecture we commented on some notation used in Haug’s book (table 9.4.1, pp. 357; Professor Haug provided last week a handout that addresses these issues and I will post on the class website under link ”Chapter 9 Supplement”). We concluded that:

  - *By definition*, $\vec{\Pi}$ is the coefficient matrix that multiplies $\vec{\omega}$ in the expression of the time derivative $\dot{\Phi}(q, t)$

  - The $\Phi_{\vec{\pi}}$ of the book is denoted in this class by $\vec{\Pi}$

  - $\vec{\pi}$ is not a variable

  - $\Phi_{\vec{\pi}}$ is not a partial derivative

- The matrix $\vec{R} = [\Phi_r \quad \vec{\Pi}]$ was introduced and used to get the velocities $\dot{r}$ and $\vec{\omega}$ (instead of $\dot{r}$ and $\dot{\vec{p}}$):

\[
\frac{d\Phi(r, p, t)}{dt} = \Phi_r \dot{r} + \vec{\Pi} \vec{\omega} + \Phi_t = [\Phi_r \quad \vec{\Pi}] \begin{bmatrix} \dot{r} \\ \vec{\omega} \end{bmatrix} + \Phi_t = 0_{6nb} \Rightarrow \vec{R} \begin{bmatrix} \dot{r} \\ \vec{\omega} \end{bmatrix} = \nu_{6nb}
\]

- Note that a similar matrix $\vec{R} = [\Phi_r \quad \Pi]$ can be introduced, used to compute $\dot{r}$ and $\omega$:

\[
\frac{d\Phi(r, p, t)}{dt} = \Phi_r \dot{r} + \Pi \omega + \Phi_t = [\Phi_r \quad \Pi] \begin{bmatrix} \dot{r} \\ \omega \end{bmatrix} + \Phi_t = 0_{6nb} \Rightarrow \vec{R} \begin{bmatrix} \dot{r} \\ \omega \end{bmatrix} = \nu_{6nb}
\]
Basic GCon DP1: $\tilde{\Pi}_i^{DP1}$ and $\tilde{\Pi}_j^{DP1}$

- Recall that
  \[
  \Psi^{DP1}(i, \tilde{a}_i, j, \tilde{a}_j, f(t)) = \bar{a}_i^T A_i^T A_j \tilde{a}_j - f(t) = a_i^T a_j - f(t) = 0
  \]

- Then, it follows that
  \[
  \dot{\Psi}^{DP1}(i, \tilde{a}_i, j, \tilde{a}_j, f(t)) = \bar{a}_i^T A_i^T \dot{A}_j \tilde{a}_j + \bar{a}_j^T A_j^T \dot{A}_i \tilde{a}_i - \dot{f}(t)
  \]
  \[
  = -\bar{a}_i^T A_i^T A_j \tilde{a}_j \tilde{\omega}_j - \bar{a}_j^T A_j^T A_i \tilde{\omega}_i \tilde{a}_i
  \]

- Therefore, we end up with
  \[
  \tilde{\Pi}_i^{DP1} = -\bar{a}_j^T A_j^T A_i \tilde{a}_i \quad \text{AND} \quad \tilde{\Pi}_j^{DP1} = -\bar{a}_i^T A_i^T A_j \tilde{a}_j
  \]
Basic GCon DP2: $\bar{\Pi}^D_{i}P^2$ and $\bar{\Pi}^D_{j}P^2$

- Recall that

$$\Phi^{DP2}(i, \tilde{a}_i, \tilde{s}_i^P, j, \tilde{s}_j^Q, f(t)) = \tilde{a}_i^T A_i^T d_{ij} - f(t) = a_i^T d_{ij} - f(t) = 0$$

- It follows that

$$\dot{\Phi}^{DP2}(a_i, d_{ij}) = a_i^T \dot{d}_{ij} + d_{ij}^T \dot{a}_i - \dot{f}(t)$$

$$= \ldots$$

- Just like before, move $\bar{\omega}_i$ and $\bar{\omega}_j$ at the right end of the terms in which they show up to obtain that

$$\bar{\Pi}^{DP2}_{i} = a_i^T \tilde{s}_i^P - d_{ij}^T A_i \tilde{a}_i \hspace{1cm} \text{AND} \hspace{1cm} \bar{\Pi}^{DP2}_{j} = -a_i^T A_i^T A_j \tilde{s}_j^Q$$
Basic GCon D: $\overline{\Pi}_i^D$ and $\overline{\Pi}_j^D$

- Recall that the GCon-D assumes the expression

$$\Phi^D(i, s^P_i, j, s^Q_j, f(t)) = d_{ij}^T d_{ij} - f(t) = 0$$

- It follows that

$$\dot{\Phi}^D = 2d_{ij}^T \dot{d}_{ij} - f(t)$$

$$= \ldots$$

- Just like before, move $\bar{\omega}_i$ and $\bar{\omega}_j$ at the right end of the terms in which they show up to obtain that

$$\overline{\Pi}^D_i = 2d_{ij}^T A_i \tilde{s}_i^P \quad \text{AND} \quad \overline{\Pi}^D_j = -2d_{ij}^T A_j \tilde{s}_j^Q$$
Basic GCon CD: $\tilde{\Pi}_i^{CD}$ and $\tilde{\Pi}_j^{CD}$

- Recall that the GCon-D assumes the expression

$$\Phi^{CD}(c, i, \tilde{s}_i^P, j, \tilde{s}_j^Q, f(t)) = c^T d_{ij} - f(t) = 0$$

- It follows that

$$\dot{\Phi}^D = c^T \dot{d}_{ij} - \dot{f}(t)$$

$$= \ldots$$

- Just like before, move $\tilde{\omega}_i$ and $\tilde{\omega}_j$ at the right end of the terms in which they show up to obtain that

$$\tilde{\Pi}_i^D = c^T A_i \tilde{s}_i^P \quad \text{AND} \quad \tilde{\Pi}_j^D = -c^T A_j \tilde{s}_j^Q$$
Final Comments on the Content of $\Phi(q, t)$

- These final thoughts are motivated by the fact that when dealing with Euler Parameters we need to clarify what exactly we mean by $\Phi(q, t)$

- Up to this point, we said that

$$\Phi(q, t) = \begin{bmatrix} \Phi^K(q) \\ \Phi^D(q, t) \end{bmatrix}$$

- It turns out that this is painting a picture with too wide of a brush. Specifically, since I work with Euler Parameters I will also have to explicitly include in (q,t) the Euler Parameter normalization constraints

- We will make the following distinction from now on (two notation conventions):

  - We will denote by $\Phi^P$ the specific set of $nb$ Euler Parameter normalization constraints:

    $$\Phi^P(q) = \Phi^P(r, p) = \Phi^P(p) = \begin{bmatrix} p_1^T p_1 - 1.0 \\ \vdots \\ p_{nb}^T p_{nb} - 1.0 \end{bmatrix}$$

  - We will denote by $\Phi^K(q)$ the set of ACE that are associated with a GCon present in the system
Final Comments on the Content of $\Phi(q, t)$

- Subsequently, when carrying out Velocity Analysis, if solving for $\dot{\mathbf{r}}$ and $\dot{\mathbf{p}}$, you have to include $\Phi^P = 0_{nb}$ in the set of constraints $\Phi(q, t)$

  Justification: although not stemming from a GCon, the Euler Parameter normalization constraints are nonetheless constraints induced by the particular choice of generalized coordinates. It’s important to make this distinction between ACEs (a) stemming from the geometry of the motion (from GCon’s), and (b) induced by the choice of generalized coordinates we’ve decided to work with (normalization constraints for $\mathbf{p}$, in our case).

- Note that for the Velocity and Acceleration right-hand side of the linear equations, you have:

  $$\nu^P = 0_{nb} \quad \& \quad \gamma^P = \begin{bmatrix} -2\mathbf{p}^T_1 \dot{p}_1 \\ \vdots \\ -2\mathbf{p}^T_{nb} \dot{p}_{nb} \end{bmatrix}$$

- Notice that you don’t have to be concerned with the Euler Parameter normalization constraints (their time derivative, that is) if you solve for $\dot{\mathbf{r}}$ and $\bar{\omega}$. Since you compute $\dot{\mathbf{p}}$ as $\dot{\mathbf{p}} - \frac{1}{2} \mathbf{G}^T \bar{\omega}$, the velocity constraints associated with the Euler Parameter normalization constraints are automatically satisfied (Problem 2 of today’s Homework).
Final Comments on the Content of $\Phi(q, t)$

- In order to clarify what we mean by $\Phi$, we’ll use the following notation (superscript $F$ stands for ‘Full’)

$$
\Phi(q, t) = \begin{bmatrix}
\Phi^K(q) \\
\Phi^D(q, t)
\end{bmatrix} = 0_{6nb}
$$

$$
\Phi^F(q, t) = \begin{bmatrix}
\Phi^K(q) \\
\Phi^D(q, t) \\
\Phi^P(p)
\end{bmatrix} = 0_{7nb}
$$

- In Kinematics, when carrying out Position Analysis, one must always solve $\Phi^F(q, t) = 0_{7nb}$; i.e., a nonlinear system of dimension $7nb$ (as long as we use Euler Parameters as generalized coordinates).

- When carrying out Velocity Analysis, one must solve either $\Phi^F_q \dot{q} = \nu^F$ (of dimension $7nb$), or

$$
\bar{R} \begin{bmatrix}
\dot{r} \\
\dot{\omega}
\end{bmatrix} = \nu \text{ (of dimension 6nb)}, \text{ or } R \begin{bmatrix}
\dot{r} \\
\dot{\omega}
\end{bmatrix} = \nu \text{ (of dimension 6nb)}.
$$

- When carrying out Acceleration Analysis, one must solve either $\Phi^F_{\dot{q}} \ddot{q} = \gamma^F$ (of dimension $7nb$), or

$$
\bar{R} \begin{bmatrix}
\ddot{r} \\
\ddot{\omega}
\end{bmatrix} = \gamma \text{ (of dimension 6nb)}, \text{ or } R \begin{bmatrix}
\ddot{r} \\
\ddot{\omega}
\end{bmatrix} = \gamma \text{ (of dimension 6nb)}.
$$
Solving a Nonlinear System

- The most important numerical algorithm to understand in Kinematics
- Relied upon heavily by ADAMS, used almost in all analysis modes
  - Kinematics
  - Dynamics
  - Equilibrium

- How does one go about finding the solution?

\[
\sqrt{x} - \sin x = 0 \\
\begin{cases}
x - e^y = 1 \\
\ln (1 + x) - \cos y = 0
\end{cases}
\]
Newton-Raphson Method

- Framework, for the one dimensional case:
  - A function $f(x)$ is given, $f: \mathbb{R} \rightarrow \mathbb{R}$. You are interested in finding the root $\alpha$ of the function $f$, or in other words, finding $\alpha$ that verifies the equation:
    $$f(x) = 0$$
  - The assumption is that $f$ is twice continuously differentiable

- The Newton-Raphson algorithm is an iterative algorithm that is implemented as follows:
  $$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}$$
  $$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}$$
  $$x^{(3)} = x^{(2)} - \frac{f(x^{(2)})}{f'(x^{(2)})}$$
  ...

- Note that an initial guess $x^{(0)}$ is needed.
- The iterative algorithms is stopped after taking a sufficient number of iterations that gradually get $x^{(k)}$ closer to the value $\alpha$ (this if nothing goes wrong...)
Newton-Raphson Method

Geometric Interpretation

![Graph showing the geometric interpretation of the Newton-Raphson method.](image)
The algorithm becomes

\[ q^{(1)} = q^{(0)} - [\Phi_q(q^{(0)})]^{-1} \Phi(q^{(0)}, t) \]

The Jacobian is defined as

\[ \Phi_q(q^{(0)}) = \frac{\partial \Phi}{\partial q} \bigg|_{q=q^{(0)}} \]
Putting things in perspective...

- Newton algorithm for nonlinear systems requires:
  - A starting point $q^{(0)}$ from where the solution starts being searched for
  - An iterative process in which the approximation of the solution is gradually improved:

\[
q^{(1)} = q^{(0)} - \left[ \Phi_q(q^{(0)}) \right]^{-1} \Phi(q^{(0)}, t)
\]

\[
q^{(2)} = q^{(1)} - \left[ \Phi_q(q^{(1)}) \right]^{-1} \Phi(q^{(1)}, t)
\]

\[
q^{(3)} = q^{(2)} - \left[ \Phi_q(q^{(2)}) \right]^{-1} \Phi(q^{(2)}, t)
\]

\[ \cdots \text{etc.} \]
Newton’s Method: Closing Remarks

- Can ever things go wrong with Newton’s method?

- Yes, there are at least three instances:
  1. Most commonly, the starting point is not close to the solution that you try to find and the iterative algorithm diverges (goes to infinity)
  2. Since a nonlinear system can have multiple solutions, the Newton algorithm finds a solution that is not the one sought (happens if you don’t choose the starting point right)
  3. The speed of convergence of the algorithm is not good (happens if the Jacobian is close to being singular (zero determinant) at the root, not that common)
Newton’s Method: Closing Remarks

- What can you do address these issues?
- You cannot do anything about 3 above, but can fix 1 and 2 provided you choose your starting point carefully.
- Newton’s method converges very fast (quadratically) if started close enough to the solution.
- To help Newton’s method in Position Analysis, you can take the starting point of the algorithm at time $t_k$ to be the value of $q$ from $t_{k-1}$ (that is, the very previous configuration of the mechanism).
- See the pptx file available on the class website for MATLAB code that implements the Newton-Raphson method implemented in conjunction with the Position Analysis stage.
End Kinematics

Start Dynamics