GraSMech course 2009-2010

Computer-aided analysis of rigid and flexible multibody systems

Multibody system
Assembly
Paul Fisette, UCL
Olivier Verlinden, UMONS

Contents

- Relative coordinates :
  - Pseudo-rotation constraints
  - Coordinate partitioning – assembling aspects
  - Driving constraints
- Absolute coordinates :
  - Assembling techniques

Closed-loop MBS

The « cutting » procedure
Closed-loop MBS
1. Cut of a body: the most general case

\[ h_i(q_i, t) \triangleq x'(q_i, t) - x''(q_i, t) = 0 \]
\[ R(q_i, t) - E = 0 \]

Loop closure: pseudo-rotation constraints

\[ h_i(q_i, t) \triangleq \dot{x}'(q_i, t) - \ddot{x}'(q_i, t) = 0 \]
\[ R(q_i, t) - E = 0 \]

Loop closure: pseudo-rotation constraints

Rotation constraints: formulation

\[ R(q_i, t) - E = 0 \]

The 9 rotation constraints (5.51) can be rewritten in matrix form

\[ H(q_i, t) \triangleq E - R(q_i, t) = 0 \]

and their time derivatives are then given by (see equation 2.62)

\[ \dot{H} = R \hat{R} \]

\[ H = [ R_{11}, R_{12}, R_{13} ] \]
\[ R = [ R_{21}, R_{22}, R_{23} ] \]

we can write these derivatives as

\[ \dot{R}_{ij} = \hat{R}_i R_{j0} \]

or equivalently as

\[ \dot{R}_{ij} = -R_{0i} \hat{R}_j \quad \text{for} \ i, j = 1, 2, 3 \]
Loop closure: pseudo-rotation constraints

Rotation constraints: formulation

Let's choose a subset of 3 independent constraints (general 3D rotation):

\[ \begin{bmatrix} \dot{\mathbf{R}}(q_t) \end{bmatrix} = \begin{bmatrix} \mathbf{B}(q_t) \end{bmatrix} \lambda \]

with corresponded sub-matrix \( \mathbf{D}_c \):

\[ \mathbf{D}_c(q_t) = \begin{bmatrix} \mathbf{R}_{22}(q_t) & -\mathbf{R}_{12}(q_t) & 0 \\ -\mathbf{R}_{21}(q_t) & \mathbf{R}_{11}(q_t) & 0 \\ 0 & 0 & \mathbf{R}_{33}(q_t) \end{bmatrix} \]

If all the constraints are satisfied:

\[ \begin{bmatrix} \dot{\mathbf{R}}(q_t) \end{bmatrix} + \mathbf{C}(q_t, q_t) = \mathbf{Q} + \mathbf{B}^T(q_t) \lambda \]

\[ \mathbf{J}_c(q_t) = \mathbf{D}_c(q_t) \mathbf{B}(q_t) \]

Coordinate partitioning method can be used to reduce the system (ODE).
Loop closure: pseudo-rotation constraints

Elimination of the dependent coordinates:

\[ v^{i+1} = v^i - (R_{i+1})^T \lambda^i \]

Convergence in the neighborhood of the exact solution \( v^0 \)?

\[ \lambda^i \text{ can be expanded around } v^0: \]

\[ h_0(v; v^0) = D_0(v, v^0, t) R_0(v, v^0, t) (v^i - v^0) = R_0(v, v^0, t) (v^i - v^0) \]

where \( D_0(v, v^0, t) = B \)

\[ v^{i+1} = v^0 \] at the first order (\( \Rightarrow \) error of second order)

Pseudo-gradient

Loop closure: pseudo-rotation constraints

\[ M(q, t) v + c(q, t) = Q + B^T(q, t) \lambda \]

\[ h(v, t) = 0 \]

\( \text{!!! danger: the complete set } h(q) \text{ are not necessarily satisfied} \):

\[ h(q) = \left( \begin{array}{c} h_{11}(q, t) \\ h_{22}(q, t) \\ h_{33}(q, t) \end{array} \right) = 0 \]

because:

\[ h_{i}(q, t) \triangleq \left( \begin{array}{c} h_{i1}(t) \\ h_{i2}(t) \\ h_{i3}(t) \end{array} \right) = \left( \begin{array}{c} -R_{i3} \\ R_{i2} \\ -R_{i1} \end{array} \right) \]

is only a subset of the full set of constraints

Loop closure: pseudo-rotation constraints

because:

\[ h_{i}(q, t) \triangleq \left( \begin{array}{c} h_{i1}(t) \\ h_{i2}(t) \\ h_{i3}(t) \end{array} \right) = \left( \begin{array}{c} -R_{i3} \\ R_{i2} \\ -R_{i1} \end{array} \right) \iff R_{i,3,k} = +1 \text{ or } -1 \]

Undesirable solutions (since \( R \) orthonormal and \( \det(R) = 1 \)):

\[ R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \text{ or } \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

First undesirable solution, denoted \( v^1 \):

\[ D_0(q, t) = D_0(q, t) B(q, t) \]

\[ D_0(q, v^0, t) = \begin{pmatrix} R_{0,1} & -R_{0,2} & 0 \\ R_{0,2} & R_{0,3} & 0 \\ 0 & 0 & 1 \end{pmatrix} \]
Elimination of the dependent coordinates:

\[ \gamma_{k+1} = \gamma_k - (B_k)^{-1} \delta \]

in the neighborhood of \( \gamma_k, h_k \), can be approximated by:

\[ h_k(u, \gamma_k, t) = D_k(u, \gamma_k, t) B_k(u, \gamma_k, t) \left( \gamma_k - \gamma'_k \right) \]

\[ \gamma_{k+1} = \gamma_k - (B_k)^{-1} D_k(u, \gamma_k, t) B_k(u, \gamma_k, t) \left( \gamma_k - \gamma'_k \right) \]

which has the form:

\[ \gamma_{k+1} = \gamma_k - (I(u, \gamma_k, t) \delta) \]

where:

\[ \gamma_k = \begin{cases} \text{epsilon value is 1 = divergence} \\ \end{cases} \]

---

**Conclusion 1:**

\[ M(q,t) \dot{\omega}(q,t) + \xi(q,t) = Q + B^T(q,t) \lambda \]

\[ h_k(q,t) = 0 \]

Pseudo-rotation constraints

Pseudo-gradient

\[ \gamma_{k+1} = \gamma_k - (B_k)^{-1} h_k \]

with:

\[ \omega = B(q,t) \dot{q} + h(q,t) \]

---

**Conclusion 2:**

\[ \Delta F_{\text{constraint}} = \sum \xi_i \Delta q_i = \Delta q^T Q \]

\( \lambda_i \) in the present case, given by:

\[ \Delta F_{\text{constraint}} = \Delta q^T \lambda \]

and, when the loop is closed, reduces to:

\[ \Delta F_{\text{constraint}} = \Delta q^T B^T(q,t) \lambda \]

\[ \Delta F_{\text{constraint}} = \delta \omega^T \lambda \]

\[ \Delta F_{\text{constraint}} = I \Delta (\omega - \omega') = \Delta \omega^T L \]

\[ \Delta \omega = L^T \lambda \]

---

**Conclusion 2:**

\[ \Delta F_{\text{constraint}} = \sum \xi_i \Delta q_i = \Delta q^T Q \]

\( \lambda_i \) in the present case, given by:

\[ \Delta F_{\text{constraint}} = \Delta q^T \lambda \]

and, when the loop is closed, reduces to:

\[ \Delta F_{\text{constraint}} = \Delta q^T B^T(q,t) \lambda \]

\[ \Delta F_{\text{constraint}} = \delta \omega^T \lambda \]

\[ \Delta F_{\text{constraint}} = I \Delta (\omega - \omega') = \Delta \omega^T L \]

\[ \Delta \omega = L^T \lambda \]
Contents

- Relative coordinates:
  - Pseudo-rotation constraints
  - Coordinate partitioning – assembling aspects
  - Driving constraints
- Absolute coordinates
  - Assembling techniques

Coordinate Partitioning - method

Principle

\[ q = \begin{pmatrix} a \\ e \end{pmatrix}, \quad J = \begin{pmatrix} J_a & J_e \end{pmatrix} \]

Exact resolution of the constraints

Position: \[ \dot{q}^{t+1} = q^t - (J_e)^{-1} b \]

Velocity: \[ \dot{\dot{q}} = B_a \dot{v} - (J_a)^{-1} \frac{\partial h}{\partial q} \]

with \[ B_a \equiv -(J_a)^{-1} J_e \]

Acceleration: \[ \ddot{b} = B_a \dot{b} - (J_a)^{-1} b \]

with \[ b \equiv \begin{pmatrix} \dot{q} + \frac{\partial h}{\partial q} \end{pmatrix} \]

\[ \begin{pmatrix} M_{aa} & M_{a}\quad \quad M_{eo} \\ M_{ea} & M_{ee} \quad \quad M_{eo} \end{pmatrix} \begin{pmatrix} \dot{q} \\ \dot{b} \end{pmatrix} + \begin{pmatrix} \alpha_a \\ \alpha_e \end{pmatrix} = \begin{pmatrix} \dot{q}_o \\ \dot{b}_o \end{pmatrix} + \begin{pmatrix} J_a^T \quad J_e^T \end{pmatrix} \lambda \]

Coordinate Partitioning - method

\[ M_{aa} + B_a B_o + B_{ao}^T M_{eo} + B_{ao}^T M_{ee} B_{oa} \] \[ b - (M_{ao} + B_{ao}^T M_{eo} - (J_e)^{-1} b \] \[ + c_e + B_{ao}^T c_o - Q_s + B_{ao}^T Q \]

Finally:

\[ M_{eo}(\eta, \tau) + c_e(\eta, \tau, \tau) = \dot{Q}_{o,e} \]

\[ ODE ! \]
Coordinate Partitioning - assembling

What choice \( \{u, v\} \)

\[ v = \begin{pmatrix} u \\ v \end{pmatrix} \quad J = \begin{pmatrix} J_u & J_v \end{pmatrix} \]

Position level:

\[ h(v) = h(v_{old}) + \frac{\partial h}{\partial v}(v - v_{old}) + 2 \text{ dof} \]

\[ v - v_{old} = (\frac{\partial h}{\partial v})^{-1} h(v) \]

20 years of MBS simulation without any closure problem!

Velocity level:

\[ (J_u J_v) \begin{pmatrix} \frac{\partial h}{\partial v} \end{pmatrix} = 0 \quad \implies \quad J_u \dot{v} = b \quad \text{with} \quad b = -J_u \ddot{u} + \ldots \]

\[ \dot{v} = B_{u,v} \dot{u} - (J_u)^{-1} \begin{pmatrix} \frac{\partial h}{\partial v} \end{pmatrix} \quad \text{with} \quad B_{u,v} = -(J_u)^{-1} J_v \]

Coordinate Partitioning - assembling

How to choose \( \{u, v\} \)

1. User choice:

2. Numerical choice:

3. Compromise between 1. and 2.

4. A symbolic choice:

- \( J_u \) matrix « structure »

Coordinate Partitioning - assembling

How to choose \( \{u, v\} \)

Numerical choice:

\[ \dot{v}^{(N+1)} = \dot{v}^{(N)} - (J_u)^{-1} A \]

\[ \dot{v} = B_{u,v} \dot{u} - (J_u)^{-1} \begin{pmatrix} \frac{\partial h}{\partial v} \end{pmatrix} \quad \text{with} \quad B_{u,v} = -(J_u)^{-1} J_v \]

\[ h(q) = 0.5 x(q)^2 - L \]

\[ J = \begin{pmatrix} \frac{\partial h}{\partial u} & \frac{\partial h}{\partial v} \end{pmatrix} \]

Newton/Raphson
Coordinate Partitioning - assembling

How to choose \( \{u, v\} \)

Numerical choice:

\[
\begin{align*}
\text{Newton/Raphson} \\
\text{Coordinate Partitioning - assembling}
\end{align*}
\]

\[
\begin{pmatrix} 0 & x \end{pmatrix} = J
\]

\[
J = \begin{pmatrix} J_u & J_v \end{pmatrix}
\]

\[
\begin{pmatrix} u \choose v \end{pmatrix}
\]

Numerical choice (for 3D closed-systems)

A robust method (among others)

=> \( J \) matrix : LU factorization with full pivoting

\[
\begin{pmatrix} J_u \end{pmatrix}
\]

\[
\begin{pmatrix} n \choose x \end{pmatrix}
\]

Yes!
Coordinate Partitioning - assembling

**Compromise between « user » and « numerical »**

=> Partial J matrix : LU factorization with full pivoting

A posteriori verification :

=> Rank of partial J = rank of full J

---

Coordinate Partitioning - assembling

Redundant constraints ?

1 or 0 dof ? => 1 !

Re-partitioning

Not necessary Not necessary Not necessary Not necessary

---

Coordinate Partitioning - assembling

**Newton/Raphson « limitation »**

No convergence

By experience : not a real problem
Possible reasons :

- Unfeasible rigid body mechanism (data mistake, ...)
- Very far initial conditions (3D visualization helps a lot)

Convergence, but !

Fortunately :

- Small time steps (1e-3 ... 1e-5)
- Very close initial conditions \( v_{ini} \)
Coordinate Partitioning - assembling

Newton/Raphson « limitation » ...⇒ remedy
Slow convergence ⇒ relaxation

Difficult convergence ⇒ dichotomic method coupled with N.R.

How to choose \((u, v)\)

1. User choice:

2. Numerical choice:

3. Compromise between 1. and 2.

State equations in terms of \(u\)

4. A symbolic choice:

\(J_v\) matrix « morphology »

A symbolic approach

- Selection of a subset of dependent coordinates \(v = f(u)\)
- Analysis of the Jacobian matrix
  - Numerical method: pivotal element search
  - Symbolic method: looking for a block triangular sub-matrix \(J_v\)

Cuts

Jacobian matrix

\[ J = \frac{\partial h}{\partial q} \]
Coordinate partitioning - assembling
A symbolic approach

This partition does not lead to a block triangular shape

Independent blocks

Bloc Triangular Factorization of $J_v$
A symbolic approach

- Constraint decoupling
- Block by block resolution of $h(q)$
- Solve several small problems rather than a big one

Contents
- Relative coordinates:
  - Pseudo-rotation constraints
  - Coordinate partitioning – assembling aspects
  - Driving constraints
- Absolute coordinates
  - Assembling techniques
Driving constraints: intro

- **Kinematic analysis**: analyse the behaviour of the mechanism while imposing the motion of parts of the system (e.g., robotics: impose the motion of the tool)

Crank rotation imposed

Driving “constraints”

**Principle**: Given a driving motion: \( q_a = f(t) \)

Equivalent to a constraint: \( h(q, t) \triangleq q_a - f(t) = 0 \)

**Examples**:
- Robot inverse kinematics
- Mechanism inverse dynamics

Driving “constraints”

Driving constraint:

- Defined at position, velocity and acceleration level:
  \[
  \ddot{q}^i = f(t) ; \quad \dot{q}^i = \dot{f}(t) ; \quad q^i = \ddot{f}(t)
  \]

- Trivial Jacobian: \( J = \begin{pmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 & 0 \end{pmatrix} \)

**Example**:
- Constant velocity motion: \( q^i = \omega t ; \quad \dot{q}^i = \omega ; \quad \ddot{q}^i = 0 \)
- Harmonic excitation: \( q^i = a \sin \omega t ; \quad \dot{q}^i = a \omega \cos \omega t ; \quad \ddot{q}^i = -a \omega^2 \sin \omega t \)
- Locked motion (particular case): \( q^i = \alpha ; \quad \dot{q}^i = 0 ; \quad \ddot{q}^i = 0 \)
Driving “constraints”

In a Coordinate partitioning approach : second partitioning

⇒ Driven or “forced” joints : « a priori independent » variables \( u \)
⇒ After reduction (nbr 1), second reduction (nbr 2)

for the driven constraints :

\[
\begin{bmatrix}
M_{uu}^{ref} & M_{uF}^{ref} \\
M_{Fu}^{ref} & M_{FF}^{ref}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}^l \\
\ddot{F}^l
\end{bmatrix}
= \begin{bmatrix}
C_{uu}^{ref} \\
C_{F}^{ref}
\end{bmatrix}
\begin{bmatrix}
0 \\
X^l
\end{bmatrix}
\]

Equations of motion ⇒ Direct Dynamics
(for time integration, equilibrium, modal analysis)

Driven-constraint equations ⇒ Inverse Dynamics
(reaction force, actuator force, …)

An “Academic” Example

Contents

- Relative coordinates :
  - Pseudo-rotation constraints
  - Coordinate partitioning – assembling aspects
  - Driving constraints
- Absolute coordinates
  - Assembling techniques
Assembly in absolute coordinates

- Problem of assembly: close the constraints from an approximate initial set of configuration parameters (critical for absolute coordinates)
  - often insufficient set of initial conditions => the problem is the most often transformed to an optimization process (find the closest configuration, or the less deformed,...)

Assembly – What’s the problem?

- The user gives a set of initial configuration parameters $q^0$ such that the constraints are often not verified
  \[ \phi(q^0) \neq 0 \]
- The purpose of assembly is to search for $q^a$ such that
  \[ \phi(q^a) = 0 \]
- $n_c$ constraints in $n_{cp}$ configuration parameters => some other constraints (or principles) must be added

Supplementary initial conditions

- Some more so-called initial conditions $\phi^d$ (driving constraints) are added, as:
  - the initial coordinate of a point
    \[ x_C(q^a) - C = 0 \text{ or } y_C(q^a) - C = 0 \text{ or } z_C(q^a) - C = 0 \]
  - an initial angle
    \[ \theta(q^a) - C = 0 \]
  - a perpendicularity condition in initial configuration
    \[ x_C(q^a) \cdot C = 0 \text{ or } y_C(q^a) \cdot C = 0 \text{ or } z_C(q^a) \cdot C = 0 \]
  - the initial value of one of the configuration parameters
    \[ q^a - C = 0 \]
  - the initial distance between two points
    \[ \sqrt{(x_C(q^a) - x_D(q^a))^2 + (y_C(q^a) - y_D(q^a))^2 + (z_C(q^a) - z_D(q^a))^2} - d = 0 \]
Assembly equations

- The total set of nonlinear equations to solve to get coherent initial configuration parameters \( \mathbf{q}^a \) is
  \[ \phi^* = \left( \begin{array}{c} \phi(q) \\ \phi'(q) \end{array} \right) = 0 \]
- The complete jacobian matrix of constraints is given by
  \[ \mathbf{B}^* = \begin{bmatrix} \mathbf{B} \\ \mathbf{B}^d \end{bmatrix} \quad \text{with} \quad B_{ij} = \frac{\partial \phi_j}{\partial \mathbf{q}_i}, \quad B_{ij}^d = \frac{\partial \phi_j^d}{\partial \mathbf{q}_i} \]
- Set of nonlinear equations \( \Rightarrow \) Why not Newton-Raphson?

Problems of assembly

- Insufficient number of constraints: less constraints than configuration parameters (infinity of solutions)
- Several solutions (which one to choose?)
- Or no solution at all (how to detect it?)

\( \Rightarrow \) Newton-Raphson not a good candidate in this case!

Minimization procedure

- The closest configuration is searched for by minimizing the cost function
  \[ \varphi = \sum_i \frac{1}{2} W_i (q_i - q_i^0)^2 + \frac{1}{2} (\mathbf{q} - \mathbf{q}^0)^T \mathbf{W} (\mathbf{q} - \mathbf{q}^0) \]
  with \( \mathbf{W} \) a diagonal matrix of weights, under the constraints \( \phi(q) = 0 \)
- The weights \( \mathbf{W} \) can be adapted to enforce the value of a configuration parameter
  Example: under ADAMS, if the initial coordinate of a body is specified as « exact », the corresponding weighting coefficient is very large \( (1E10) \)
Numerical solution

- The solutions $q^a$ are the roots of the system
  \[ \frac{\partial \phi}{\partial q} + B^T \cdot \lambda = W \cdot (q - q^0) + B^T \cdot \lambda = 0 \]
  \[ \phi^a(q^a) = 0 \]
  with $\lambda$ a set of Lagrange multipliers, commonly solved by Newton-Raphson
  \[ \left( \begin{array}{c} \frac{\partial \phi^a}{\partial q} \\ \frac{\partial \phi^a}{\partial \lambda} \end{array} \right) \approx \left( \begin{array}{c} \frac{\partial \phi^{a-1}}{\partial q} \\ \frac{\partial \phi^{a-1}}{\partial \lambda} \end{array} \right) - \left( \begin{array}{c|c} W & B^T \\ \hline B & 0 \end{array} \right)^{-1} \left( \begin{array}{c} W \cdot (q^{a-1} - q^0) + B^T \cdot \lambda^{a-1} \\ 0 \end{array} \right) \]

- The matrix of weights $W$ can be replaced by the mass matrix $M$
  $\Rightarrow$ The jacobian matrix is the same as the one built for integration

Assembly: flexible mechanism

- Elastic strain: $\epsilon_i = \frac{1}{2} \left( \frac{L_i^2 - l_{i0}^2}{l_{i0}^2} \right)$
  $\epsilon_i^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$

- Kinematically admissible configuration:
  $e(q) = \left[ \epsilon_1(q) \quad \epsilon_2(q) \quad \epsilon_3(q) \right]$

- Strain energy: $W(q) = \frac{1}{2} \sum_{i=1}^3 EA_i \epsilon_i^2 = \frac{1}{2} e(q) S e(q)$
  $S = \text{diag}(EA_i)$

Assembly: flexible mechanism

- Zero strain energy approach
  $\min_q W(q)$ s.t. $\Phi(q) = 0$

- Numerical solution:
  $B^T(q) S e(q) + B^T(q) \lambda = 0$
  $\Phi(q) = 0$
  $E = \frac{\delta \phi}{\delta q}, B = \frac{\delta \phi}{\delta \lambda}$

  $\Rightarrow$ Newton iterations
  - One or several solutions?
  - Convergence?
  - $\lambda \neq 0$: pre-stressed configuration
More robust procedure

- The function to minimize is written (E.J. Haug)
  \[ \varphi(q, r) = (q - q^0)^T \cdot (q - q^0) + r(\dot{q}^T \cdot \dot{q} + \ddot{q}^T \cdot \ddot{q}) \]
  with \( r \) a strictly positive factor
- The gradient is given by
  \[ \psi_i = \frac{\partial \varphi}{\partial q_i} = 2(q_i - q_i^0) + 2r \sum_j \frac{\partial \dot{q}_j}{\partial q_i} \dot{q}_j + 2r \sum_j \frac{\partial \ddot{q}_j}{\partial q_i} \ddot{q}_j \]
- The optimization is performed several times with increasing valued of \( r \) and stopped when
  \[ (\dot{q}^T \cdot \dot{q} + \ddot{q}^T \cdot \ddot{q}) < \epsilon \]
- The approach can detect the absence of solution (no evolution of the constraint error)

Assembly – conclusions

(absolute coordinates)

- The apparently simple problem of assembly is tricky:
  - one solution, several solutions or no solution at all
- supplementary constraints can be given
- advanced numerical procedures are needed, often based on optimization
  - find the configuration which is the closest to the initial one given by the user
  - find the less deformed configuration
  - ...

Appendix: Fletcher-Powell’s algorithm

1. Begin with estimate \( q^0 \) and \( H^0 = I \) as an estimate of the matrix of second derivatives of \( \varphi \).
2. At iteration \( i \), compute
   \[ a = -H^{-1} \cdot \psi(q^i) \]  
   \[ (1) \]
3. Find \( \alpha \) which minimizes \( \psi(q + \alpha a) \)
4. Compute
   \[ q^{i+1} = q^i + \alpha a \]  
   \[ H^{i+1} = H^i + A + C \]
   \[ (2) \]
   \[ (3) \]
with
   \[ \chi = \psi(q^{i+1}) - \psi(q^i) \]
   \[ \psi = \frac{1}{\chi^T \cdot H \cdot \chi} \]  
   \[ (4) \]
   \[ A = \left( \psi^2 \cdot \psi \right) \cdot \chi^T \]
   \[ (5) \]
   \[ C = -\left( \frac{1}{\chi^T \cdot H \cdot \chi} \right) H^T \cdot \chi \cdot \chi^T H \]  
   \[ (6) \]