Collision Response and Contact Handling for Articulated Body Dynamics

Sachin Patil

October 29, 2007

- Collision detection and generating appropriate collision responses is an integral part of any rigid body simulator
- Contacts are primarily of two kinds:
  - Colliding contacts : $\mathbf{v}_{rel} < -\epsilon$
    - Bodies "bounce" off each other, coefficient of restitution governs bounciness
    - Instantaneous change in velocity
    - Requires restart of the solver
  - Resting contacts : $-\epsilon < \mathbf{v}_{rel} < \epsilon$
    - Gradual contact forces prevent interpenetration
    - No instantaneous changes in velocity
    - Examples include sliding, rolling, stacking etc.

Colliding contacts (Impacts)

- These occur when two rigid bodies interpenetrate in a given time step of the simulation
- Two approaches to resolve this:
  - Force driven methods
    - Penalty (based on penetration distance) restoring force is applied to the bodies
    - Easier, but slow objects react in a 'slow' fashion to collision
  - Impulse driven methods
    - Apply impulses to colliding objects at point of collision causing instantaneous change in velocities
    - For frictionless bodies, direction is the same as the normal direction at point of contact

Continuous contacts

- Any continuous contact is a constraint
- Two approaches to resolve such contacts
  - Impulse based methods
    - Inherently local in nature
    - Faster and simpler
    - No explicit contact constraints
    - Cannot guarantee stability after resolution
  - Constraint based methods
    - Inherently global in nature
    - Much more stable than local, impulse based methods

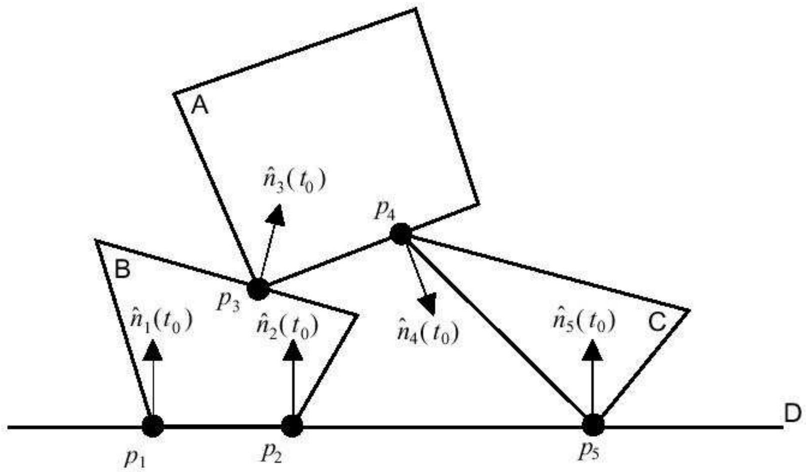

Figure 1: Resting Contact

- Apply normal forces $f_{cp}^{i}\hat{\mathbf{n}}_{cp}^{i}$ at each contact point
- Solve all contact forces simultaneously (since they influence each other) which yields a linear system
- All forces subject to three conditions (outlined in the next slide)

Resting contact response

- Interpenetration must be prevented : $\mathbf{a}_{cp} \geq 0$
- Forces can only be repulsive (to prevent further interpenetration) : $\mathbf{f}_{cp} \geq 0$
- Forces should equal zero when the bodies in contact start to separate : $\mathbf{a}_{cp}^T \mathbf{f}_{cp} = 0$
- Normal accelerations depend linearly on the normal forces : $\mathbf{a}_{cp} = \mathbf{A}\mathbf{f}_{cp} + \mathbf{b}$
- This is a Linear Complementarity Problem (LCP)
  - This formulation can also be used with impulse based methods

Linear Complementarity Problem (LCP)

- Need to solve a quadratic program to solve for the forces $\mathbf{f}_{cp}$
  - Generally solving a LCP is an NP-complete problem
- There are two kinds of methods to solve such problems:
  - Pivoting algorithms
    - Use a finite number of steps and require the recursive solution of systems of linear equations
    - Do not provide useful intermediate result
    - Theoretically faster than iterative methods
  - Iterative methods
    - Converge to the solution
    - Can be interrupted in the middle to yield a valid result
    - Gracefully deal with singular systems

Linear Complementarity Problem (LCP)

- More details on how to compute the $\mathbf{A}$ and $\mathbf{b}$ matrices and the LCP formulation for resolving resting contacts available in:
  - *Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies*, D.Baraff, Computer Graphics Proceedings, 1989
- More details on how to solve LCP problems available in:
  - Dantzig's algorithm : *Fast contact force computation for non-penetrating rigid bodies*, D.Baraff, Computer Graphics Proceedings, 1994
  - Lemke's algorithm : *Lemke's algorithm, the hammer in your toolbox*, C.Hecker, 2004, http://www.d6.com/users/checker/dynamics.htm
- Incorporate friction in previous formulation for realistic looking results

Overview of forward dynamics methods

- Previously we came across the following methods to compute forward dynamics for an articulated body
  - Composite Rigid Body algorithm
  - Lagrange Multipliers method
  - Featherstone's $\mathbb{O}(n)$ Articulated Body Method and $\mathbb{O}(lg(n))$ Divide-and-Conquer method
- This lecture will use Featherstone's ABM as the primary forward dynamics algorithm for illustrating concepts in collision response and contact handling

Forward dynamics methods

- Traditionally two kinds of approaches adopted towards solving this problem:
  - Maximal coordinates
    - For m links, there are $6m$ state variables where each link is a rigid object
    - Constraints used to keep the links together
    - Straightforward extension of simple rigid body dynamics
  - Reduced (generalized) coordinates
    - The number of state variables is the same as the number of joints in the articulation hierarchy
    - In spite of an intuitive representation, the equations of motion turn out to be much more complicated

Forward dynamics methods

- Approaches
  - Maximal coordinates
    - Operate in Cartesian coordinates
    - Hard to evaluate joint angles and velocities
    - Hard to enforce joint limits
    - Hard to apply internal joint torques
    - Numerical error(drift) may separate the links
  - Reduced coordinates
    - Joint space more intuitive for complex articulated bodies such as robots or humanoids
    - Fewer DOFs and constraints
- We will focus on the reduced coordinate approach in this lecture

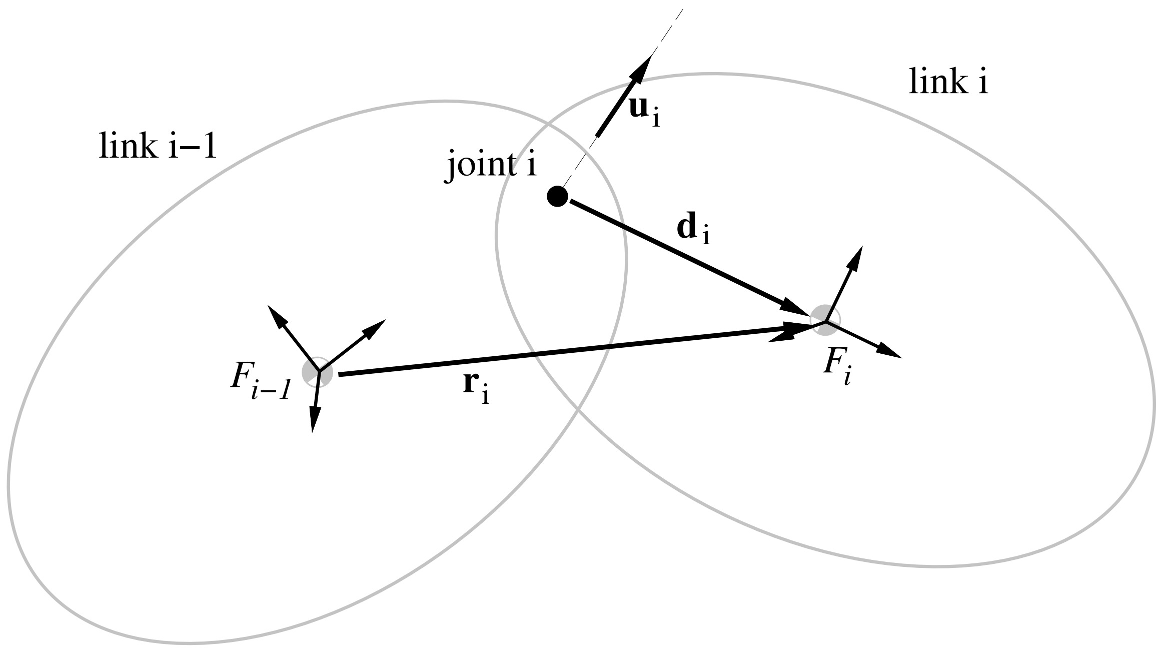

Figure 2: Quantities involved in forward ABD computation

Notation

- Link variables
  - $\mathbf{v}_i$: Linear velocity of link i
  - $\mathbf{a}_i$: Linear acceleration of link i
  - $\boldsymbol{\omega}_i$: Angular velocity of link i
  - $\boldsymbol{\alpha}_i$: Angular acceleration of link i
  - m_i : Mass of link i
  - $\mathbf{M}_i$: Mass matrix of link i
  - $\mathbf{I}_i$: Inertia tensor of link i
- Joint variables
  - q_i : Scalar angle of joint i
  - $\dot{q}_i$: Scalar velocity of joint i
  - $\ddot{q}_i$: Scalar acceleration of joint i

Spatial Algebra Notation

- $\hat{\mathbf{v}}_i$: Spatial velocity of link i given by $\begin{bmatrix} \boldsymbol{\omega}_i \\ \mathbf{v}_i \end{bmatrix}$
- $\hat{\mathbf{a}}_i$: Spatial acceleration of link i given by $\begin{bmatrix} \boldsymbol{\alpha}_i \\ \mathbf{a}_i \end{bmatrix}$
- $\hat{\mathbf{s}}_i$: Spatial joint axis of joint i given by $\begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_i \times \mathbf{d}_i \end{bmatrix}$
- $\hat{\mathbf{c}}_i$: Spatial Coriolis force of joint i
- $\hat{\mathbf{f}}_i$: Spatial force acting on link i given by $\begin{bmatrix} \mathbf{f}_i \\ \boldsymbol{\tau}_i \end{bmatrix}$

Spatial Algebra Notation

- $\hat{\mathbf{I}}_i$: Spatial isolated inertia of link i defined by $\begin{bmatrix} 0 & \mathbf{M}_i \\ \mathbf{I}_i & 0 \end{bmatrix}$
- $\hat{\mathbf{Z}}_i$: Spatial isolated zero-acceleration force of link i given by $\begin{bmatrix} -m_i\mathbf{g} \\ \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i \end{bmatrix}$
- $\hat{\mathbf{I}}_i^A$: Spatial articulated inertia of link i
- $\hat{\mathbf{Z}}_i^A$: Spatial articulated zero-acceleration force of link i
- ${}_G\hat{\mathbf{X}}_F$: Spatial transformation from frame F to frame G given by $\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ -\tilde{r}\mathbf{R} & \mathbf{R} \end{bmatrix}$

- *ComputeSerialLinkVelocities* (revolute):

$\boldsymbol{\omega}_{\mathbf{0}}, \mathbf{v}_0, \boldsymbol{\alpha}_0, \mathbf{a}_0 \leftarrow 0$

for i = 1 to n

$\mathbf{R} \leftarrow$ rotation matrix from F_{i-1} to F_i

$\mathbf{r} \leftarrow$ radius vector from F_{i-1} to F_i (in F_i coordinates)

$\boldsymbol{\omega}_i \leftarrow \mathbf{R}\boldsymbol{\omega}_{i-1} + \dot{q}_i \mathbf{u}_i$

$\mathbf{v}_i \leftarrow \mathbf{R}\mathbf{v}_i + \boldsymbol{\omega}_i \times \mathbf{r} + \dot{q}_i(\mathbf{u}_i \times \mathbf{d}_i)$

- *InitSerialLinks* (revolute):

for i = 1 to n

$$\hat{\mathbf{Z}}_i^A \leftarrow \begin{bmatrix} -m_i\mathbf{g} \\ \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i \end{bmatrix}$$

$$\hat{\mathbf{I}}_i^A \leftarrow \begin{bmatrix} 0 & M_i \\ \mathbf{I}_i & 0 \end{bmatrix}$$

$$\boldsymbol{\nu}_i \leftarrow \dot{q}_i\mathbf{u}_i$$

$$\hat{\mathbf{c}}_i \leftarrow \begin{bmatrix} \boldsymbol{\omega}_{i-1} \times \boldsymbol{\nu}_i \\ \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_i) + 2\boldsymbol{\omega}_{i-1} \times (\boldsymbol{\nu}_i \times \mathbf{d}_i) + \boldsymbol{\nu}_i \times (\boldsymbol{\nu}_i \times \mathbf{d}_i) \end{bmatrix}$$

The main ABM routine

Call *ComputeSerialLinkVelocities*

Call *InitSerialLinks*

for i = n to 2

$$\hat{\mathbf{I}}^A_{i-1} \leftarrow \hat{\mathbf{I}}^A_{i-1} + {}_{i-1}\hat{\mathbf{X}}_i \left[\hat{\mathbf{I}}^A_i - \frac{\hat{\mathbf{I}}^A_i \hat{\mathbf{s}}_i \hat{\mathbf{s}}'_i \hat{\mathbf{I}}^A_i}{\hat{\mathbf{s}}'_i \hat{\mathbf{I}}^A_i \hat{\mathbf{s}}_i} \right] {}_i\hat{\mathbf{X}}_{i-1}$$

$$\hat{\mathbf{Z}}^A_{i-1} \leftarrow \hat{\mathbf{Z}}^A_{i-1} + {}_{i-1}\hat{\mathbf{X}}_{i-1} \left[\hat{\mathbf{Z}}^A_i + \hat{\mathbf{I}}^A_i \hat{\mathbf{c}}_i + \frac{\hat{\mathbf{I}}^A_i \hat{\mathbf{s}}_i [Q_i - \hat{\mathbf{s}}'_i (\hat{\mathbf{Z}}^A_i + \hat{\mathbf{I}}^A_{i-1} \hat{\mathbf{c}}_i)]}{\hat{\mathbf{s}}'_i \hat{\mathbf{I}}^A_i \hat{\mathbf{s}}_i} \right]$$

$$\hat{\mathbf{a}}_0 \leftarrow \hat{\mathbf{0}}$$

for i = 1 to n

$$\ddot{q}_i = \frac{Q_i - \hat{\mathbf{s}}'_i \hat{\mathbf{I}}^A_i \, {}_i\hat{\mathbf{X}}_{i-1} \hat{\mathbf{a}}_{i-1} - \hat{\mathbf{s}}'_i (\hat{\mathbf{Z}}^A_i + \hat{\mathbf{I}}^A_i \hat{\mathbf{c}}_i)}{\hat{\mathbf{s}}'_i \hat{\mathbf{I}}^A_i \hat{\mathbf{s}}_i}$$

$$\hat{\mathbf{a}}_i \leftarrow {}_i\hat{\mathbf{X}}_{i-1} \hat{\mathbf{a}}_{i-1} + \hat{\mathbf{c}}_i + \ddot{q}_i \hat{\mathbf{s}}_i$$

Featherstone's ABM

- Excellent derivation of the original ABM (which does not assume a strong background in mechanics) in Chapter 4 of Mirtich's thesis
  - *Impulse-based Dynamic Simulation of Rigid Body Systems*, B.Mirtich, Ph.D. thesis, University of California at Berkeley, 1996
- Tutorial on Spatial Vector Algebra on Prof. Featherstone's web page
  - *Short course on Spatial Vector Algebra*, R.Featherstone, http://users.rsise.anu.edu.au/ roy/spatial/index.html, 2004

Impulse-based response for resolving impacts

- Impulse based collision response scheme for resolving impacts (proposed by B.Mirtich)
- Useful for computing collision response when one or more links of the articulated body collide with objects in the environment
- Computing impulse response is simpler than computing the effect of an applied force
  - Joint accelerations are a nonlinear function of forces applied to the articulated body
  - Collision impulse dominates all other forces and all nonlinear terms drop out of the dynamics equations

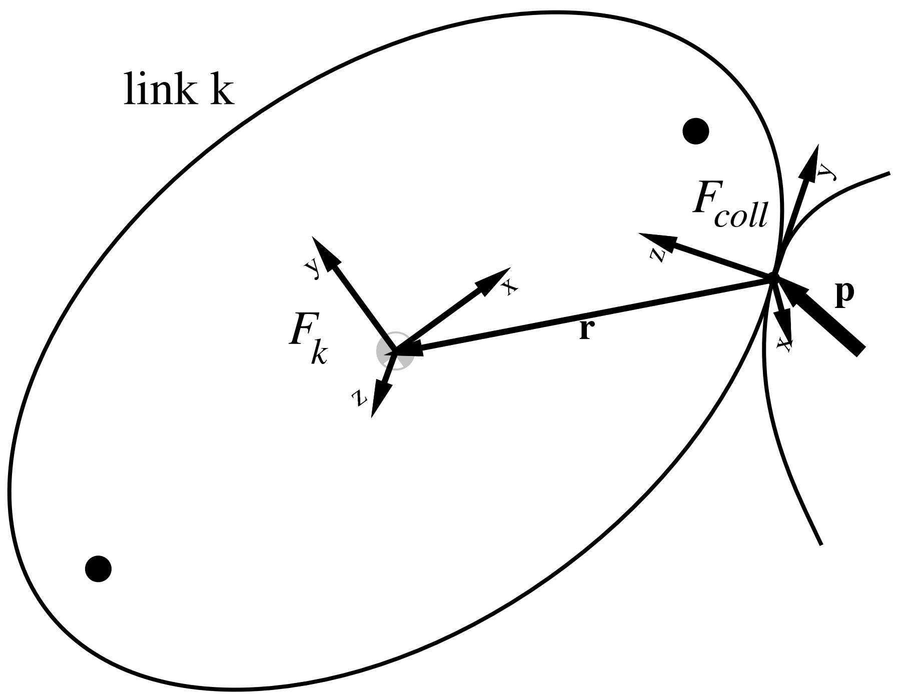

Figure 3: Impulse imparted to the colliding link

- Steps involved in computing the impulse based collision response:

  1. Update the articulated inertias of all links in the articulated body
  2. Apply a test impulse to the colliding link
  3. Compute the impulse response of the body along the path from the colliding link to the base
  4. Compute the collision matrix K by applying test impulses of known magnitude and measuring their effects
  5. Use K and collision integration to compute the collision impulse
  6. Propagate the collision impulse to the base of the body
  7. Propagate the resulting changes in velocity throughout the body

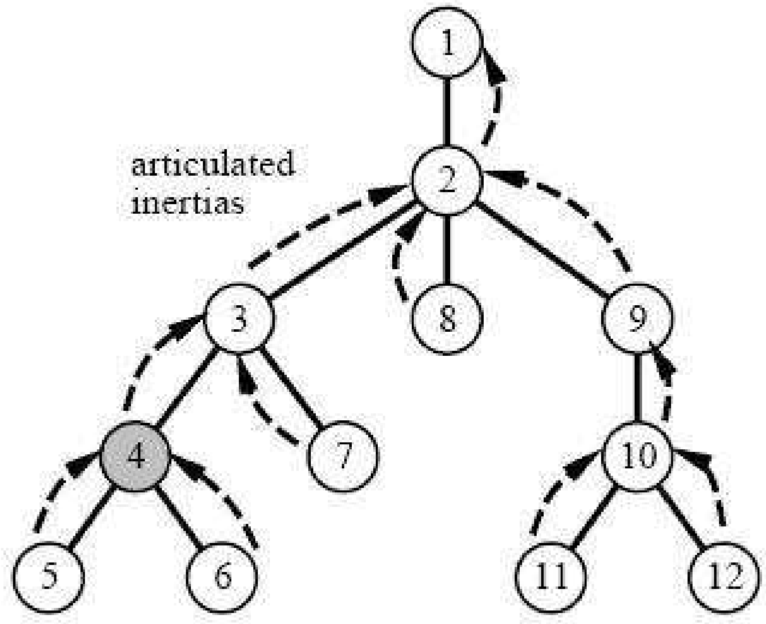

Figure 4: Update the articulated inertias of all links in the articulated body

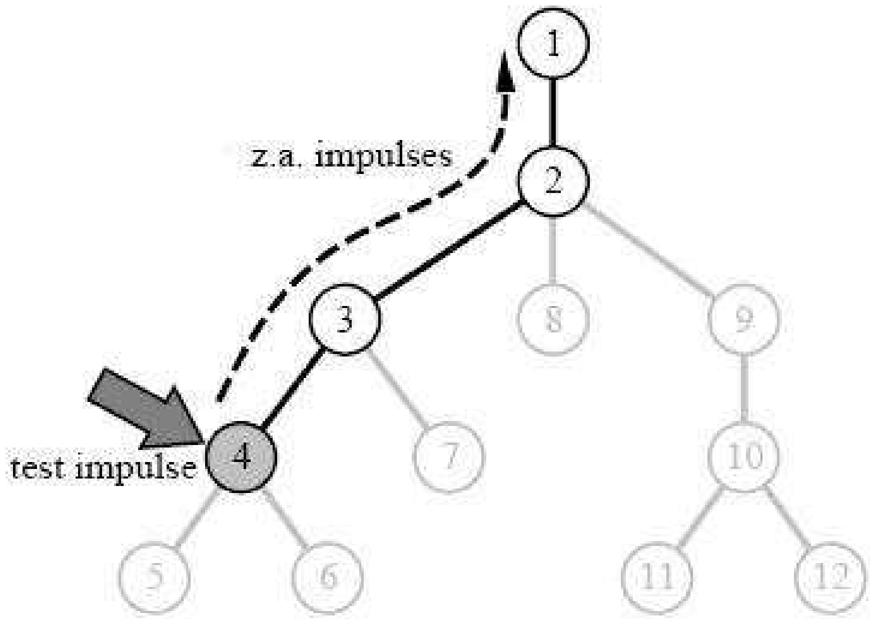

Figure 5: Compute the impulse response of the body along the path from the colliding link to the base

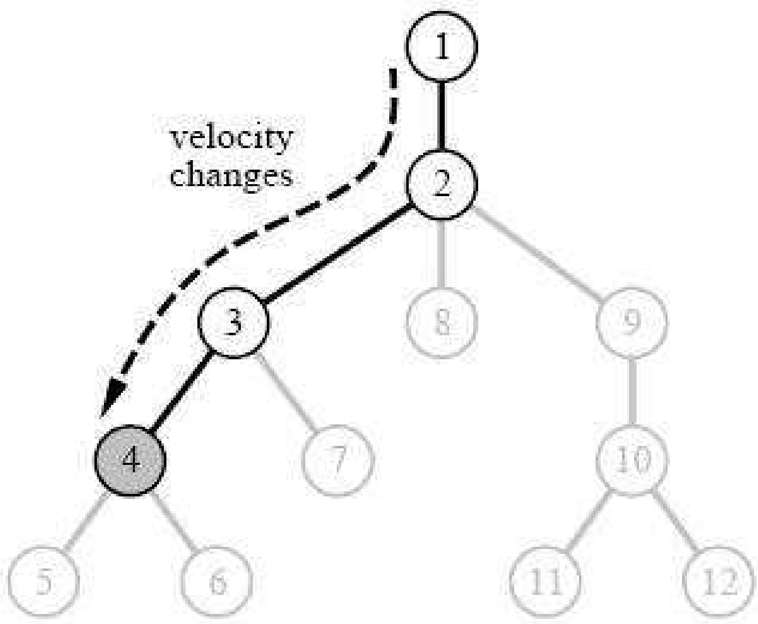

Figure 6: Compute velocity response of the body to the applied test impulse

- Steps 1,2,3: *ImpulseResponse*

$\hat{\mathbf{Y}}_k^A \leftarrow -\hat{\mathbf{p}}_{test}$

$i \leftarrow k$

while link i has a parent

$h \leftarrow$ index of parent of link i

$$\hat{\mathbf{Y}}_h^A = {}_h\hat{\mathbf{X}}_i \left[\mathbf{1} - \frac{\hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i \hat{\mathbf{s}}_i'}{\hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i} \right] \hat{\mathbf{Y}}_i^A$$

$i \leftarrow h$

$\hat{\mathbf{a}}_0 \leftarrow \hat{\mathbf{0}}$

$h \leftarrow 0$

repeat

$i \leftarrow$ index of child of link h on path to link k

$$\Delta \dot{q}_i = -\frac{\hat{\mathbf{s}}_i'}{\hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i} \left[\hat{\mathbf{I}}_i^A \, {}_i\hat{\mathbf{X}}_h \Delta \hat{\mathbf{v}}_h + \hat{\mathbf{Y}}_i^A \right]$$

$$\Delta \hat{\mathbf{v}}_i = {}_i\hat{\mathbf{X}}_h \Delta \hat{\mathbf{v}}_h + \Delta \dot{q}_i \hat{\mathbf{s}}_i$$

$h \leftarrow i$

until $i = k$

- Step 4: *ComputeMultibodyKi*
  Compute the articulated inertias ($\hat{\mathbf{I}}_i^A$) for all links of the articulated body

  Compute ${}_k\hat{\mathbf{X}}_{coll}$ and ${}_{coll}\hat{\mathbf{X}}_k$ from $\mathbf{R}$ and $\mathbf{r}$

  for i =1 to 3

  $$\hat{\mathbf{p}}_{coll} \leftarrow {}_k\hat{\mathbf{X}}_{coll} \begin{bmatrix} \mathbf{e}_i \\ \mathbf{0} \end{bmatrix}$$

  call ImpulseResponse to compute $\Delta\hat{\mathbf{v}}_k$

  $\Delta\phi_i \leftarrow$ lower 3×1 component of ${}_{coll}\hat{\mathbf{X}}_k \Delta\hat{\mathbf{v}}_k$

  $$\mathbf{K}_i \leftarrow \left[\begin{array}{c|c|c} \Delta\phi_1 & \Delta\phi_2 & \Delta\phi_3 \end{array} \right]$$

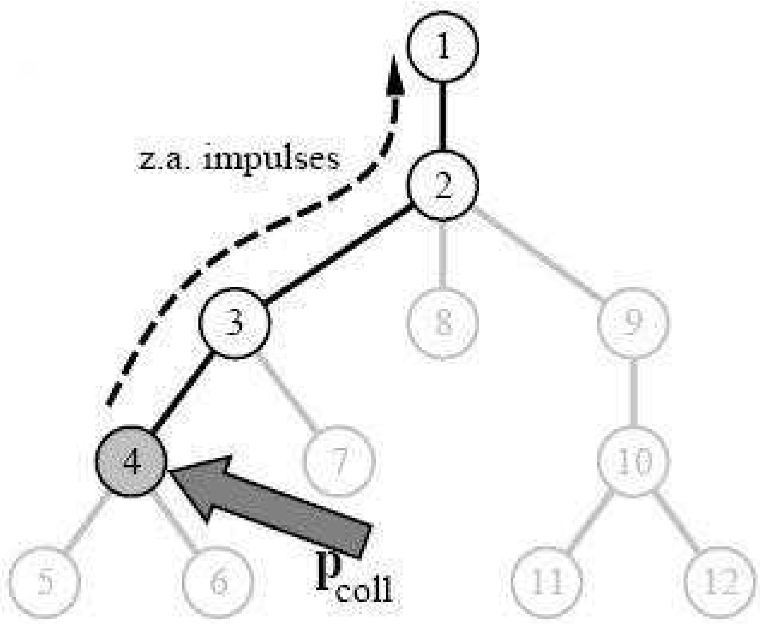

Figure 7: Propagate the collision impulse to the base of the body

Impulse-based collision response

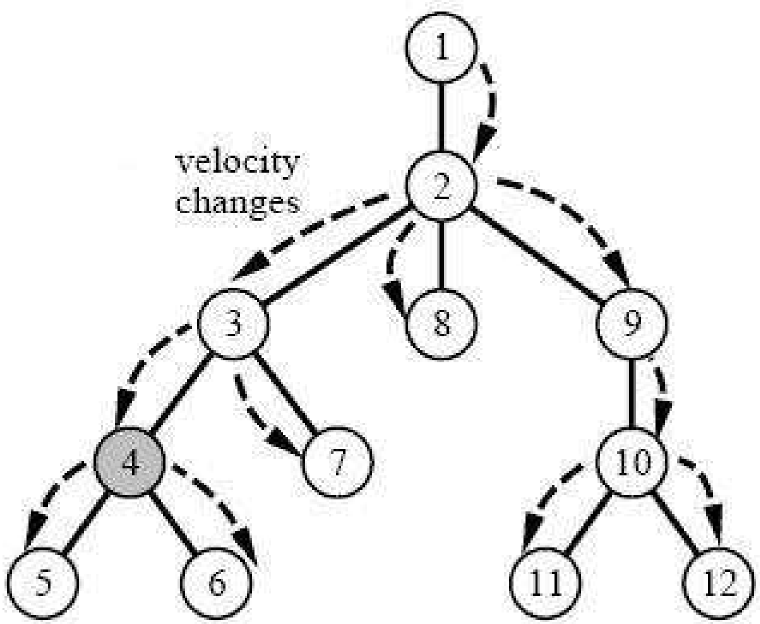

Figure 8: Propagate the resulting changes in velocity throughout the body

- Steps 6,7: *PropagateImpulse*

$\hat{\mathbf{Y}}_k^A \leftarrow -\hat{\mathbf{p}}_{coll}$
$i \leftarrow k$
while link i has a parent
$h \leftarrow$ index of parent of link i
$$\hat{\mathbf{Y}}_h^A = {}_h\hat{\mathbf{X}}_i \left[\mathbf{1} - \frac{\hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i \hat{\mathbf{s}}_i'}{\hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i} \right] \hat{\mathbf{Y}}_i^A$$
$i \leftarrow h$
for all links i not on path from base link to link k, $\hat{\mathbf{Y}}_i^A \leftarrow \hat{\mathbf{0}}$

$\hat{\mathbf{a}}_0 \leftarrow \hat{\mathbf{0}}$
for $i = 1$ to n
$h \leftarrow$ index of link inboard to joint i
$$\Delta \dot{q}_i = -\frac{\hat{\mathbf{s}}_i'}{\hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i} \left[\hat{\mathbf{I}}_i^A {}_i\hat{\mathbf{X}}_h \Delta \hat{\mathbf{v}}_h + \hat{\mathbf{Y}}_i^A \right]$$
$$\Delta \hat{\mathbf{v}}_i = {}_i\hat{\mathbf{X}}_h \Delta \hat{\mathbf{v}}_h + \Delta \dot{q}_i \hat{\mathbf{s}}_i$$

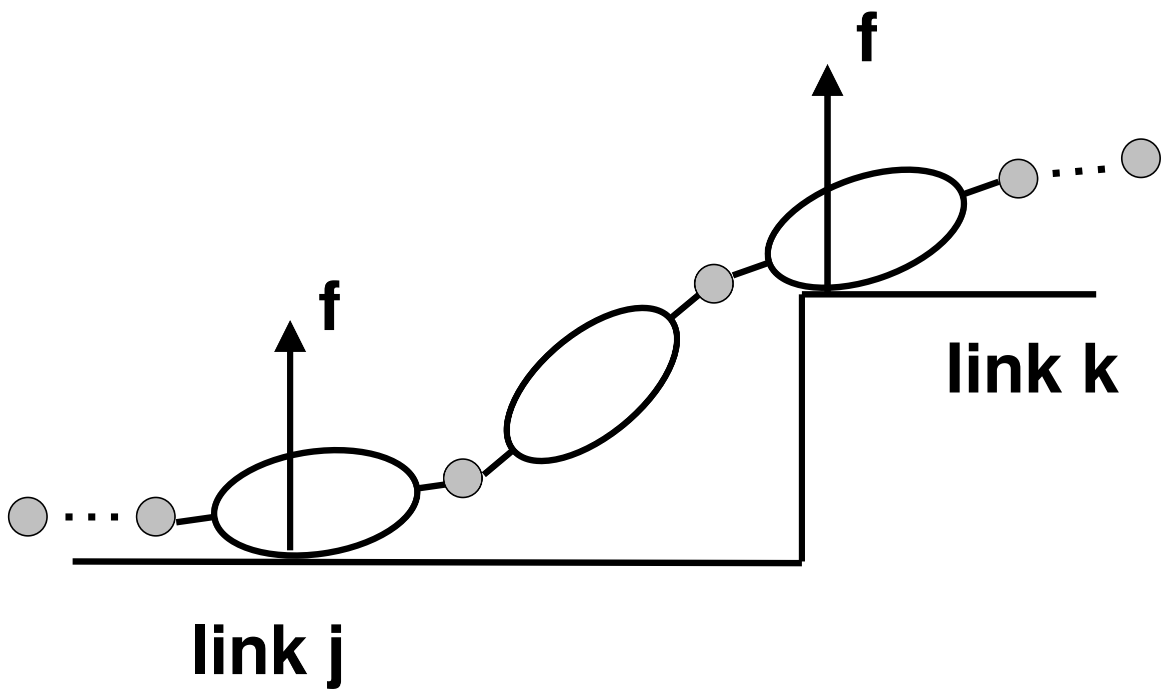

Figure 9: Articulated body in continuous contact with the environment

Constraint formulation

- Simulation of articulated bodies involves simulating interaction with the environment
- Constraints can be of various kinds:
  - Acceleration constraints : Setting the acceleration of a particular joint or link to a specified value
  - Impact constrains : Using impulses to instantaneously change the velocity at certain points on the articulated body
  - Unilateral constraints : Simultaneously satisfy complementary conditions arising from situations such as contact forces and enforcing joint limits
- Penalty forces could be applied whenever a constraint is violated, but for complex articulated structures, leads to instability
- We will focus on analytical methods for enforcing constraints

Acceleration constraints

- Common constraints in articulated body motion can be expressed in terms of body's accelerations
  - Example: A point on the body whose velocity is currently zero can be prevented from moving by setting its acceleration to zero
- Magnitude of joint and link accelerations is not a linear function of a force applied to the body
  - Joint accelerations depend on external force and other forces such as gravity, and because of velocity dependent Coriolis and centripetal forces, joint accelerations are not a linear function of the applied force

- For a given time instant, though, it can be proved that there exists a linear relationship between magnitude of a generalized force applied to the body and magnitude of acceleration of body's joints or links (by assuming that the Jacobian remains constant during that time instant)
  - $a^f = kf + a^0$
    - a^0 and a^f are the observed acceleration magnitudes before and after the force is applied
    - f is the magnitude of the force applied
    - k is a scalar constant
- a can represent the magnitude of acceleration $\ddot{\mathbf{q}}_i$ of any joint i or the magnitude of the linear acceleration, $\mathbf{a}_P$, of a point P on a link of the articulated body

- By extension, any linear function of the joint accelerations, $h(\ddot{q})$, following holds for some k:
  - $h(\ddot{q}^f) - h(\ddot{q}^0) = kf$, where $h(\ddot{q}^0)$ and $h(\ddot{q}^f)$ are vectors of joint accelerations before and after the force is applied
- Setting the acceleration of a joint or a point on a link to a specified value, a^d can be written as a linear constraint function $h(\ddot{\mathbf{q}}) = a - a^d$
- Goal: find an appropriate joint acceleration vector, $\ddot{\mathbf{q}}^c$, such that $h(\ddot{\mathbf{q}}^c) = 0$

Single acceleration constraint

- Consider the problem: find a generalized force τ^d that when applied to joint i will cause acceleration of joint i to become $\ddot{q}_i^d$
- Easy way to do this:
  - Call Featherstone's method to compute (magnitude) default acceleration of joint i,$\ddot{q}_i^0$
  - Apply test torque of known (non-zero) magnitude τ^t on joint i, call Featherstone's method again and record new acceleration $\ddot{q}_i^t$
  - Following relationship then holds: $\ddot{q}_i^t = k\tau^t + \ddot{q}_i^0$
  - Compute k from the above as: $k = \frac{\ddot{q}_i^t - \ddot{q}_i^0}{\tau^t}$
  - Compute desired torque $\tau^d = \tau^t \frac{\ddot{q}_i^d - \ddot{q}_i^0}{\ddot{q}_i^t - \ddot{q}_i^0}$
- Setting acceleration of a point on the link of the articulated body can be achieved in the same way

Multiple acceleration constraints

- Consider m constraints (on joints or points on links), such that the acceleration of i should become $\ddot{q}_i^d$
- Goal: Compute a set of m forces, f_i^d such that all m constraints are satisfied simultaneously
- Rewriting this as a system of equations:

$$
\begin{aligned}
\ddot{q}_1^d - \ddot{q}_1^0 &= k_{11} f_1^d + k_{12} f_2^d + \ldots + k_{1m} f_m^d \\
\ddot{q}_2^d - \ddot{q}_2^0 &= k_{21} f_1^d + k_{22} f_2^d + \ldots + k_{2m} f_m^d \\
&\vdots \\
\ddot{q}_m^d - \ddot{q}_m^0 &= k_{m1} f_1^d + k_{m2} f_2^d + \ldots + k_{mm} f_m^d
\end{aligned}
$$

Multiple acceleration constraints

- Solve by applying test forces f_i^t and computing the coefficients k_{ji} for each test force
- Compute the set of forces by solving the system of linear equations
- Complexity of the entire operation : $\mathbb{O}(nm + m^3)$
  - m is typically a lot smaller than n and hence the m^3 term does not dominate in practice
  - m Featherstone method calls to compute response to the m test forces can be computed in parallel

- Used to instantaneously change the velocity of the body when impact occurs
- Consider case where the magnitude of the velocity of a point P on the body (joint or point on a link) needs to change instantaneously from its current value v_P^- to a known value v_P^+
- During the small time interval δt when the impact occurs, magnitude of acceleration of P becomes: $a_P^I = \frac{(v_P^+ - v_P^-)}{\delta t}$
- Solve for impulsive force f_P using the approach suggested earlier and assuming δt to be 1 (w.l.o.g)
- Compute the new set of joint accelerations $\ddot{\mathbf{q}}^I$ using Featherstone's method
- New set of joint velocities $\dot{\mathbf{q}}^+$ can be computed using the relation: $\dot{\mathbf{q}}^+ = \dot{\mathbf{q}}^- + \ddot{\mathbf{q}}^I \delta t$

Unilateral constraints

- Allows one to impose restrictions on the direction of the allowable constraint force and the resulting acceleration
- Constraints arising from continuous contact are one such example
- Formulation similar to the LCP formulation discussed earlier for simple rigid bodies
- Solve the LCP system: $\mathbf{h}(\ddot{\mathbf{q}}^d) = \mathbf{K}\mathbf{f} + \mathbf{h}(\ddot{\mathbf{q}}^0)$ where
  - $h_i(\ddot{\mathbf{q}}) = a_{cp_i}(\ddot{\mathbf{q}}) - a^d_{cp_i}$
  - $\mathbf{f}$ is the vector of forces acting at all the contact points

- In Featherstone's ABM method, external forces act on the c.o.m of the links
- External forces acting on a point $P = (x, y, z)$ on a link i need to be expressed in the inertial frame F_i of the link before invoking the ABM method
- Use the Jacobian corresponding to point P to do this:
  - $\mathbf{J}_P(\mathbf{q}(t)) = \begin{pmatrix} \frac{\delta x}{\delta q_1} & \cdots & \frac{\delta x}{\delta q_n} \\ \frac{\delta y}{\delta q_1} & \cdots & \frac{\delta y}{\delta q_n} \\ \frac{\delta z}{\delta q_1} & \cdots & \frac{\delta z}{\delta q_n} \end{pmatrix}$
  - The external force $\mathbf{f}_{ext}$ can then be expressed as : $\mathbf{J}_P(\mathbf{q}(t))^T \mathbf{f}_{ext}$

- SimulationStep($\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{C}, h$)

while($h > 0$)

    $\mathbf{f}_C \leftarrow$ *ComputeConstraintForces*($\mathbf{C}$)

    $\ddot{\mathbf{q}}_t \leftarrow$ *ComputeAccelerations*($\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{f}_C$)

    $h_{try} \leftarrow h$; *repeat* = true;

    while(*repeat*)

        $[\mathbf{q}_{new}, \dot{\mathbf{q}}_{new}] \leftarrow$ *Integrate*($\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t, h_{try}$)

        $[C$, *needImpact*, *needBacktrack*$] \leftarrow$

            UpdateConstraints($\mathbf{q}_{new}, \dot{\mathbf{q}}_{new}$)

        if(*needBackTrack*) $h_{try} = h_{try} * 0.5$

        if(*needImpact*)

            $\dot{\mathbf{q}}_{new} \leftarrow$ *ResolveImpacts*$(\mathbf{C})$

            if(*success*)

                $h = h - h_{try}$; *repeat* = false

        $[\mathbf{q}_t, \dot{\mathbf{q}}_t] \leftarrow [\mathbf{q}_{new}, \dot{\mathbf{q}}_{new}]$

return $(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{C})$

Practical Physics for Articulated Bodies

- Further details for formulating collisions and contacts as constraints and solving them are available in:
  - *Practical Physics for Articulated Characters*, E.Kokkevis, Game Developers Conference Proceedings, 2004
- Provides a unified framework for ABD including additional details for incorporating friction in contact handling and enforcing joint limits

Adaptive ABD with contact handling and collision response

- Featherstone's ABD method for forward dynamics is replaced by the Divide-and-Conquer method
- Maintains a hierarchy of 'active' joints (based on a motion simplification metric) and 'rigidify' other joints
  - Further details available in: *Adaptive Dynamics of articulated bodies*, S.Redon, N.Galoppo, M.Lin, SIGGRAPH 2005
- Extends the forward dynamics algorithm to handle collisions and impacts

- Extends Mirtich's impulse based formulation to generate an appropriate response for dealing with impacts
- Reduces the amount of computation in the original formulation by propagating the effects of the collision impulse along the 'active' joint hierarchy (also referred to as the 'hybrid' tree) and treating remaining rigidified sections as single rigid links
- Uses a 'hybrid' Jacobian to map joint velocities to the spatial velocities of the rigidified links
  - $\mathbf{J}_P^{hybrid}(\mathbf{q}(t)) = \begin{pmatrix} \frac{\delta x}{\delta q_1} & \cdots & \frac{\delta x}{\delta q_m} \\ \frac{\delta y}{\delta q_1} & \cdots & \frac{\delta y}{\delta q_m} \\ \frac{\delta z}{\delta q_1} & \cdots & \frac{\delta z}{\delta q_m} \end{pmatrix}$
    where each q_i is the scalar angle of each active joint and m is the number of the active joints

Adaptive ABD with contact handling

- Uses Kokkevis' constraint formulation to enforce joint limits
- An impact constraint is used to force the velocity of a joint breaking its limit to zero
- An acceleration constraint is used to ensure that the joint limits are not violated in later timesteps
- Also uses the 'hybrid' tree consisting of 'active' joints to ensure a *sub-linear* running time collision resolution scheme

References

- *Robot Dynamics: Equations and algorithms*, R.Featherstone, D.Orin, IEEE International Conference on Robotics and Automation, 2000
- *Impulse-based Dynamic Simulation of Rigid Body Systems*, B.Mirtich, Ph.D. thesis, University of California at Berkeley, 1996
- *Practical Physics for Articulated Characters*, E.Kokkevis, Game Developers Conference Proceedings, 2004
- *Adaptive Dynamics with Efficient Contact Handling for Articulated Robots*, R.Gayle, M.C.Lin, D.Manocha, Robotic Science and Systems, 2006
- Slides from previous COMP768 and COMP790 course offerings, M.C.Lin