Multibody System Dynamics: MBDyn Overview

Pierangelo Masarati  <masarati@aero.polimi.it>
Politecnico di Milano
Dipartimento di Ingegneria Aerospaziale
Outline

• Multibody dynamics
• Software architectures
• Problems
• Arbitrary motion description
• Deformable components
• Solving the problem
• Extracting useful information
• Examples of multibody modeling with MBDyn
• Future development
• Documentation and support
Multibody dynamics

Basic equations:

\[ \mathbf{M}(\mathbf{x}) \ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t) \]

- mechanics of unconstrained system of bodies
- subjected to configuration-dependent loads

Can be obtained from many (equivalent!) approaches:

- *Newton-Euler*: linear/angular equilibrium of each body
- *d’Alembert-Lagrange*: virtual work of active forces/moments
- *Gauss, Hertz, Hamilton, ...*: variational principles
Multibody dynamics

Constrained system: kinematic constraints

- **holonomic**
  \[ \phi(x, t) = 0 \]

- **non-holonomic (not integrable to holonomic)**
  \[ \psi(x, \dot{x}, t) = 0 \]

usually

\[ A(x, t) \dot{x} = b(x, t) \quad \text{a} = M^{-1} f \]

- algebraic relationship between kinematic variables
- explicitly dependent on time: rheonomic
- scleronomous otherwise
Multibody dynamics

Minimal set:

\[ x = x(q, t) \]

usually, this relationship:

- is not known in advance, or
- cannot be easily made explicit with respect to \( q \)

Coordinate partitioning is required, e.g.:

- direct elimination from derivative of constraint equation
- QR or similar decomposition

Results in Maggi-Kane equations and similar approaches

Small system is obtained by expensive numerical reduction
(unless topology knowledge can be exploited)
Multibody dynamics

Redundant set: $\delta (\lambda \cdot \phi) = \delta \lambda \cdot \phi + \delta x \cdot \Phi^T /_x \lambda$

$\delta (\mu \cdot \psi) = \delta \mu \cdot \psi + \delta x \cdot \Psi^T /_x \mu$

By Lagrange multipliers:

- dynamics of constrained system using physical coordinates
- constraint reactions applied to equations of motion
- algebraic constraints explicitly added to the system

- Multiple bodies with few actual dofs:
  - system size nearly doubles

- Multiple bodies with few constraints:
  - system size not significantly altered

- Sparsity is almost preserved

\[
\begin{bmatrix}
M & \Phi^T /_x \\
\Phi /_x & 0
\end{bmatrix}
\begin{bmatrix}
\ddot{x} \\
\lambda
\end{bmatrix} =
\begin{bmatrix}
f \\
b'
\end{bmatrix}
\]
Multibody dynamics

Constraint equations written “as is”:

\[ \phi(x, t) = 0 \]

Problem becomes differential algebraic (DAE); issues:

- needs specific care to be solved: (nearly) L-stable integration, i.e.
  - unconditionally stable, and
  - \( \Delta x_{k+1} \rightarrow 0 \) for \( \Delta t \rightarrow \infty \)

- the constraint equation implies the additional constraints

\[
\begin{align*}
\phi(x, t) &= 0 \\
\ddot{\phi}(x, t) &= 0
\end{align*}
\]

but they are not explicitly enforced:
may need constraint stabilization (Gear et al.)
Multibody dynamics

Constraint equations:
• x is correct
• derivatives may be inaccurate
• multipliers may be inaccurate

\[ \Phi(x_j, t_j) = 0 \]
\[ \dot{\Phi}(x_j, t_j) = 0 \]
\[ \Phi(x, t) = 0 \]
Multibody dynamics

Alternative: constraint equations differentiated to second order:

$$\phi /_x \ddot{x} = b'$$

Problem remains ordinary differential (ODE);
- can be solved by conditionally stable algorithms
- the constraint equation does not imply the original constraints

$$\phi(x,t) = 0$$
$$\dot{\phi}(x,t) = 0$$

Definitely needs constraint stabilization!

Common technique: Baumgarte

$$\phi /_x \ddot{x} = b' - 2 \alpha \dot{\phi} - \beta^2 \phi$$

(Violation governed by asymptotically stable linear differential eq.)
Software architectures

- **Monolithic:**
  - user prepares specific model using built-in library elements
  - general-purpose solver swallows model and spits results

- **Library:**
  - user writes specific solver using library elements
    - usually needs programming skills; the solver must be compiled
  - specific solver solves the problem and spits results

- **Symbolic manipulators:**
  - user writes equations
  - symbolic manipulation engine solves equations and spits results

- **Modelica (and Modelica-like):**
  - user prepares model using a modeling language and libs
  - general-purpose interpreter generates specific solver
  - specific solver solves the problem and spits results
Software architectures

Free software examples (surely there are more):

- **Monolithic:**
  - MBDyn

- **Library:**
  - DynaMechs (C++)
  - Mbs3d (requires Matlab)
  - Open Dynamics Engine (ODE) (C++)

- **Symbolic manipulators:**
  - 3D_MEC
  - EasyDyn (MuPad)
  - RoboTran (requires Matlab)

- **Modelica:**
  - OpenModelica?

*non-free counterparts omitted*

*frequently architectures overlap*
Software architectures

- MBDyn is monolithic
- Input consists in a text file
- The input syntax allows some flexibility, e.g.:
  - math expressions evaluation
  - variables definition
  - “rigorous” syntax checking, but free style, indentation, ...
- Relevant portions of the code are modular and can be extended by:
  - writing run-time loadable modules
  - hacking the code (it's free, all in all!)
- There is no built-in pre-post processing facility
- Help in this area is warmly appreciated!
  - MBDyn output can be translated into EasyAnim
  - there is an independent, partial customization based on Blender
Equations of motion: for each node (purely geometrical entity),

- Newton-Euler, written as first-order system of equations:

\[
\begin{align*}
M \dot{x} &= p \\
\dot{p} &= f(x, \dot{x}, t)
\end{align*}
\]

- Momentum and momenta moment instead of pseudo-velocities
- allows multiple contributions to inertia of a single node

Constrained equations in differential-algebraic form:

\[
\begin{align*}
M \dot{x} &= p \\
\dot{p} + \phi^T_x \lambda &= f(x, \dot{x}, t) \\
\phi(x, t) &= 0
\end{align*}
\]
Problems

- **Fundamental problem:**
  - Integration of Initial Value Problem (IVP) in time

- **Static analysis as degeneration of IVP dynamic analysis:**
  - momentum and momenta moment definitions omitted
  - only gravity is considered
  - system determination only provided by kinematic constraints and deformable components

- **Kinematic analysis as degeneration of IVP dynamic analysis:**
  - inertia elements omitted
  - system determination only provided by kinematic constraints
  - deformable components can act as “regularization”
Experimental inverse dynamics problem

- **Inverse kinematics:**
  \[ \begin{bmatrix} \mathbf{K}' \ \mathbf{K}' \mathbf{T} \\ \mathbf{K}' \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{f}(\mathbf{x}, t) \end{bmatrix} \]

- **the RHS contains the desired motion and its derivatives**
- **the (regularized) static analysis provides the kinematic inversion**
  \[ \mathbf{\phi}_x^\mathbf{T} \mathbf{\Delta \lambda} = \mathbf{f}'(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t) \]
Problems

- Experimental direct eigenanalysis
  - issues with constraints formulation (mainly with rotations)
  - issues with equations implementation (matrices not available)

- Relative coordinate frame dynamics
  - imposed frame motion: modifications only to RHS inertia elems
  - instrumental for many helicopter rotor dynamics problems
Arbitrary motion description

- Mechanical degrees of freedom:
  - structural node positions in the absolute reference frame
  - structural node orientation with respect to the absolute frame

- Kinematics is always written with respect to the absolute frame

- Newton-Euler equations are written in the absolute frame
  - moment equilibrium (Euler) equations are written with respect to the respective (moving) node

- Special elements may introduce further approximations
  - e.g. Component Mode Synthesis (CMS) element
Arbitrary motion description

Orientation handling:
- **orientation variables**: Cayley-Gibbs-Rodrigues parameters
- **orientation matrix**:
  \[ R = R(g) \]
- **orthonormality**:
  \[ R^T = R^{-1} \]
- **derivative**:
  \[ \dot{R} R^T = \omega \times = (G(g) \dot{g}) \times \]
- **incremental approach from step** \( k \) **to** \( k+1 \) **to eliminate the orientation parameters singularity issue** (increments are necessarily small for accuracy):
  \[ R_k = R(g_k) R_{k-1} \]
Arbitrary motion description

- **Orientation handling:**
  - the actual orientation variables are the Cayley-Gibbs-Rodrigues parameters relative to the correction phase of each step
  - $k$: time step counter
  - $i$: correction iteration counter (0: predicted value)

- **Orientation matrix:**
  \[ R^{(i)}_k = R\left(g^{(i)}_\Delta\right) R^{(0)}_k \]

- **Derivative:**
  \[ \dot{R}^{(i)}_k (R^{(i)}_k)^T = \omega^{(i)}_k \times = (R\left(g^{(i)}_\Delta\right) \omega^{(0)}_k) \times + (G\left(g^{(i)}_\Delta\right) \dot{g}^{(i)}_\Delta) \times \]
Arbitrary motion description

• Incremental orientation from previous step:
  ▪ Orientation parameters order of magnitude:
    \[ g \sim O(\|\omega\| \Delta t) \]

• Incremental orientation from prediction (as in MBDyn):
  ▪ Orientation parameters order of magnitude:
    \[ g_\Delta \sim O(\Delta t^{n+1}) \]

where \( n \) is the min between the order of the predictor and of the integration method (MBDyn: 3 and 2, respectively, so \( n = 2 \))

▪ As a consequence:
  \[ R(g_\Delta) \approx I \]
  \[ G(g_\Delta) \approx I \]
  \[ \dot{G}(g_\Delta) \approx 0 \]

(only in Jacobian)
Deformable components

- **Lumped deformable components**
  - rod (1D)
  - linear, angular components (3D)
  - linear & angular component (6D)

- **Intrinsic, composite-ready Finite-Volume beam element**
  - arbitrary constitutive law
  - piezoelectric constitutive law
  - aerodynamic beam element

- **Intrinsic, composite-ready shell and membrane elements**

- **Component Mode Synthesis (CMS)**
  - attached to a floating frame (a node)
  - linear state-space representation of unsteady aerodynamics
Deformable components

Lumped deformable components (3D, 6D):

\[ \theta = ax \left( \exp^{-1} \left( R_a^T R_b \right) \right) \]

\[ m = R \left( \xi \theta \right) \hat{m} \left( \theta \right) \]

- **Attached form**: \( \xi = 0, \xi = 1 \)
  - Constitutive properties referred to either of the connected nodes

- **Intrinsic form (invariant: \( \xi = 1/2 \))**: 
  - Constitutive properties referred to a floating reference frame
  - Intrinsically handles geometrical nonlinearity related to rotations
  - Correctly captures bending-torsion buckling behavior
  - Essential for anisotropic deformable components
Deformable components

Intrinsic, composite-ready beam
- Topology:
  - 1D reference line $p$, 1D reference structure $R$
  - 2D section characterization

reference line

\[
x = p + R \tau
\]
\[
x_{/\xi} = p_{/\xi} + \rho_{/\xi} \times R \tau
\]

reference motion: $p, R$

warping
Deformable components

Intrinsic, composite-ready beams

- **strain measure:**
  \[ \nu = R^T p_{/\xi} - R_0^T p_{0/\xi} \]
  \[ \kappa = R^T \rho_{\xi} - R_0^T \rho_{\xi 0} \]

- **equilibrium (from VWP):**
  \[ f_{/\xi} = \tau \]
  \[ m_{/\xi} + p_{/\xi} \times f = \mu \]

- **constitutive properties:**
  \[ f = f(\nu, \kappa) \]
  \[ m = m(\nu, \kappa) \]
Deformable components

Intrinsic, composite-ready beams: 3-node discretization
- Finite Volume approach: equilibrium of finite portions of beam
- Internal forces function of node kinematics thru constitutive laws
- Warping goes into constitutive properties computation
Solving the problem

Numerical integration

- implicit, (quasi-)L stable 2 step algorithm
  \[
  y_k = a_1 y_{k-1} + a_2 y_{k-2} + \Delta t \left( b_0 \dot{y}_k + b_1 \dot{y}_{k-1} + b_2 \dot{y}_{k-2} \right)
  \]

- tunable algorithmic dissipation: asymptotic spectral radius 1→0
  - asymptotic spectral radius = 0: 2nd order BDF
  - “optimal” dissipation: spectral radius ~ 0.6

- second-order accurate, with third-order accurate predictor
- variable time step
- not ideal for non-smooth problems (multi-step)
- different integrators can be used; new ones can be implemented
Solving the problem

- **Prediction:**
  \[ \dot{y}_k^{(0)} = \left( m_1 y_{k-1} + m_2 y_{k-2} \right) / \Delta t + n_1 \dot{y}_{k-1} + n_2 \dot{y}_{k-2} \]
  \[ y_k^{(0)} = a_1 y_{k-1} + a_2 y_{k-2} + \Delta t \left( b_0 \dot{y}_k^{(0)} + b_1 \dot{y}_{k-1} + b_2 \dot{y}_{k-2} \right) \]

- **Correction iteration:**
  \[ f_{/\dot{y}} \Delta \dot{y}^{(i)} + f_{/y} \Delta y^{(i)} = -f \left( \dot{y}_k^{(i-1)}, y_k^{(i-1)}, t_k \right) \]
  but \[ \Delta y^{(i)} = \Delta t b_0 \Delta \dot{y}^{(i)} \]

the problem becomes algebraic

\[ \left( f_{/\dot{y}} + \Delta t b_0 f_{/y} \right) \Delta \dot{y}^{(i)} = -f \left( \dot{y}_k^{(i-1)}, y_k^{(i-1)}, t_k \right) \]
\[ \dot{y}_k^{(i)} = \dot{y}_k^{(i-1)} + \Delta \dot{y}^{(i)} \]
\[ y_k^{(i)} = y_k^{(i-1)} + \Delta t b_0 \Delta \dot{y}^{(i)} \]
Model assembly

- model could be input incorrectly
- initial values of the state (position, velocity, reactions) are needed
- this might not be a trivial task
- initial state values must comply with constraints:

\[
\begin{align*}
\phi(x_0, t_0) &= 0 \\
\dot{\phi}(x_0, t_0) &= 0
\end{align*}
\]

- a dummy static nonlinear problem is solved (regularization):

\[
\begin{align*}
K'(x - x_0) + \phi^T_{/x} \lambda' &= f' \\
C'(\dot{x} - \dot{x}_0) + \phi^T_{/x} \mu' &= \dot{f}'
\end{align*}
\]

\[
\begin{align*}
\phi(x, t_0) &= 0 \\
\dot{\phi}(x, t_0) &= 0
\end{align*}
\]
Solving the problem

Solution initialization (so-called “derivatives”)

- **explicit problem:**
  \[ \dot{y} = f(y, t) \]

- **implicit problem:**
  \[ 0 = f(y, \dot{y}, t) \]

- **modified correction phase to initialize solution:**
  \[
  \left( f_{\dot{y}} + cf_{/y} \right) \Delta \dot{y}^{(i)} = -f(\dot{y}_0^{(i-1)}, y_0, t_0)
  \]
  \[ \dot{y}_0^{(i)} = \dot{y}_0^{(i-1)} + \Delta \dot{y}^{(i)} \]
  \[ y_0 = y_0 \]

- convergence no longer quadratic, but saves lots of code duplication
- **Setting** \( c = 0 \) **might not work** (problem can be structurally singular)

\[ 0 = f(y, \dot{y}, t) \]
Extracting useful information

• Detailed analysis requires detailed models, but...
• Excessive details endanger the chance to extract useful information
• Proper Orthogonal Decomposition allows to extract information from redundant measures
• Consider a set of $N$ measurements $X$ for $n$ time steps; their SVD:
  \[ X^T \in \mathbb{R}^{n \times N} = U \Sigma V^T \]

• The singular values allow to determine the $m$ most relevant signals
  \[ X_{1:m,n}^T = U_{n,1:m} \Sigma_{1:m,1:m} V_{N,1:m}^T \]

• Note that
  \[ X^T X = U \Sigma^2 U^T \]
  \[ X X^T = V \Sigma^2 V^T \]
  \[ U^T X^T = \Sigma V^T \]
  \[ X^T V = U \Sigma \]

• This allows to efficiently compute the singular values and the POMs
Extracting useful information

- The POMs can be used to identify a transition matrix
  \[ X^{(k+1)} = \Phi X^{(k)} \]

- If \( X \) contains the free response of the system, the transition matrix allows to estimate the relevant eigenvalues (AR model)

- More sophisticated system identification techniques can be used (model order reduction is an open research field)

- A technique based on covariance estimates from time histories has been recently proposed; works for:
  - free response
  - forced response
  - unmeasured forced response
Examples of multibody modeling with MBDyn

Robotics:
- delta robot

inverse dynamics for computed torque control
Examples of multibody modeling with MBDyn

Robotics: PA-10
inverse kinematics
with path optimization
of cooperating robots
Examples of multibody modeling with MBDyn

Robotics:
- biomimetic robot
- real-time motion
- planning by inverse kinematics with fault detection
Examples of multibody modeling with MBDyn

Industrial processes:

- simulation of automotive components assembly (car brake pipe) to:
  - check stresses introduced during assembly
  - check loads on supports introduced during assembly
  - check interference with other parts during assembly
  - check interference with other parts during operation
- the model has been developed by a rubber manufacturer
- it is used for product design and certification
- it required the development of specific features for solution partitioning, which are now part of MBDyn
Examples of multibody modeling with MBDyn

Automotive: mechanical modeling of suspensions
purpose: determine loads in rubber bushings and other components
Examples of multibody modeling with MBDyn

Rotorcraft dynamics and aeroservoelasticity:
- WRATS (NASA/Army) tiltrotor aeromechanics
Examples of multibody modeling with MBDyn

Rotorcraft dynamics and aeroservoelasticity:

• ERICA (AgustaWestland) tiltrotor aeromechanics (ADYN, NICETRIP)
Future development

- **Multiscale handling of submodels with different dynamics**
  - aircraft flight mechanics (~1 to 5 Hz: very slow)
  - main rotor dynamics (~5 to 40 Hz: intermediate)
  - tail rotor dynamics (~25 to >100 Hz: fast)

- **Interfacing with different domains**
  - Fluid-structure (Lagrangian/Eulerian modeling of workflows)
  - structure-structure
  - active control of large deformable systems

- **Better abstraction/modularization of components/solution phases**
  - more freedom in model customization
  - tight integration into nonlinear structural analysis (Aster?)

- More...
Documentation and support

- **Theory manual:**
  - Incomplete; needs lots of work
- **User manual**
  - available and up to date
- **Tutorials**
  - available, but reportedly too simple; need work
- **Applications manual**
  - available, but only few applications so far
- **Installation manual**
  - available, incomplete and outdated (not critical)
- **Mailing lists**
  - available: announce, users, devel
  - the “users” list also serves as issue tracking provision
Documentation and support

- Another important item that is missing is an automated test suite
  - can be run automatically after building the software
  - allows to check build errors
  - allows to check regressions in new releases
  - serves as example of modeling functionalities

- The rest is underway (always a work in progress)

Given the nature of the project, contributions are always welcome!
Questions?