POLITECNICO DI MILANO

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**

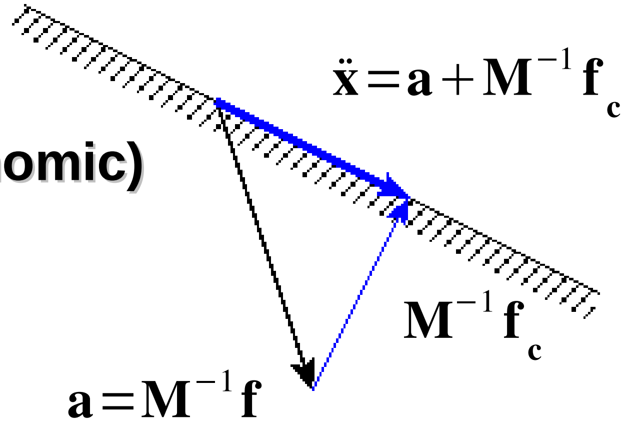
$$\phi(\mathbf{x}, t) = 0$$

- **non-holonomic (not integrable to holonomic)**

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

usually

$$\mathbf{A}(\mathbf{x}, t)\dot{\mathbf{x}} = \mathbf{b}(\mathbf{x}, t)$$



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

Multibody dynamics

Minimal set:

$$\mathbf{x} = \mathbf{x}(\mathbf{q}, t)$$

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:

$$\delta\left(\lambda\cdot\phi\right)=\delta\,\lambda\cdot\phi+\delta\,\mathbf{x}\cdot\phi_{/\mathbf{x}}^{\mathbf{T}}\,\lambda$$

$$\delta\left(\mu\cdot\psi\right)=\delta\,\mu\cdot\psi+\delta\,\mathbf{x}\cdot\psi_{/\dot{\mathbf{x}}}^{\mathbf{T}}\,\mu$$

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**

$$\begin{bmatrix}\mathbf{M} & \phi_{/\mathbf{x}}^{\mathbf{T}}\\ \phi_{/\mathbf{x}} & 0\end{bmatrix}\begin{Bmatrix}\ddot{\mathbf{x}}\\ \lambda\end{Bmatrix}=\begin{Bmatrix}\mathbf{f}\\ \mathbf{b}'\end{Bmatrix}$$

Multibody dynamics

Constraint equations written "*as is*":

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

problem becomes differential algebraic (DAE); issues:

- **needs specific care to be solved: (nearly) L-stable integration, i.e.**
  - **unconditionally stable, and**
  - $\Delta \mathbf{x}_{k+1} \to 0$ **for** $\Delta t \to \infty$
- **the constraint equation implies the additional constraints**

$$\boxed{\begin{aligned}\dot{\phi}(\mathbf{x}, t) &= 0\\ \ddot{\phi}(\mathbf{x}, t) &= 0\end{aligned}}$$

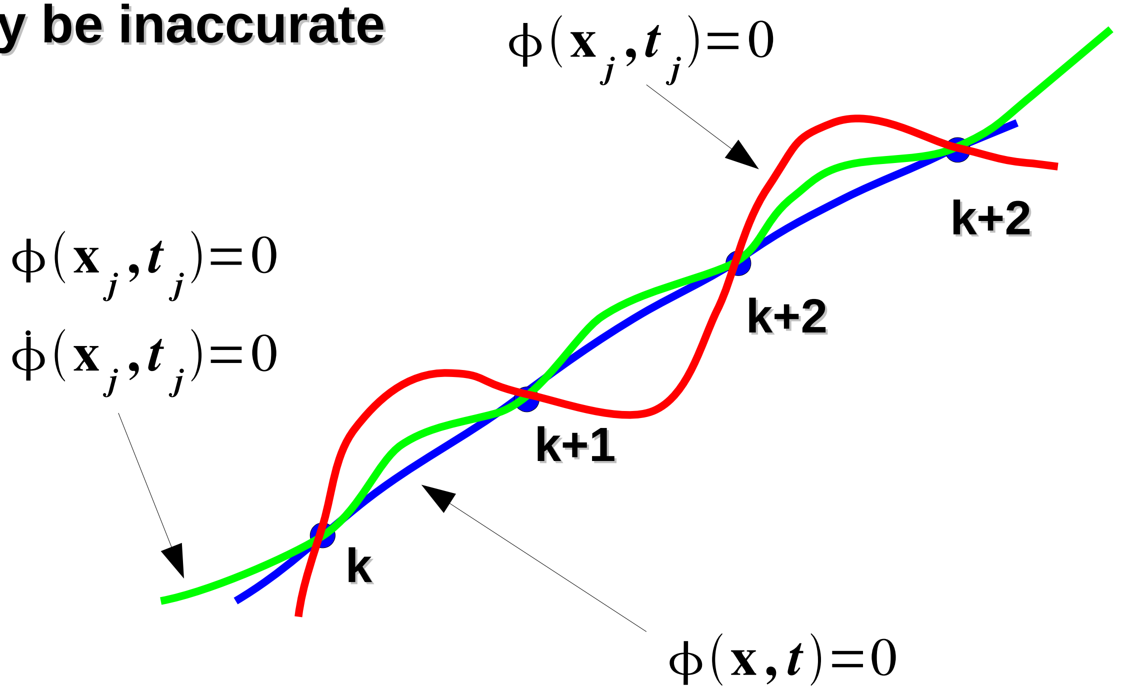
but they are not explicitly enforced:

<u>may need constraint stabilization</u> (Gear et al.)

Multibody dynamics

Constraint equations:

- **x is correct**
- **derivatives may be inaccurate**
- **multipliers may be inaccurate**



Multibody dynamics

Alternative: constraint equations differentiated to second order:

$$\phi_{/\mathbf{x}} \ddot{\mathbf{x}} = \mathbf{b}'$$

problem remains ordinary differential (ODE);

- **can be solved by conditionally stable algorithms**
- **the constraint equation does not imply the original constraints**

$$\boxed{\begin{aligned} \phi(\mathbf{x}, t) &= 0 \\ \dot{\phi}(\mathbf{x}, t) &= 0 \end{aligned}}$$

<u>definitely needs constraint stabilization!</u>

common technique: Baumgarte

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

(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

Problems

Equations of motion: for each node (purely geometrical entity),

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

$$\begin{aligned} \mathbf{M}\,\dot{\mathbf{x}} &= \mathbf{p} \\ \dot{\mathbf{p}} &= \mathbf{f}\left(\mathbf{x},\dot{\mathbf{x}},t\right) \end{aligned}$$

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

Constrained equations in differential-algebraic form:

$$\begin{aligned} \mathbf{M}\,\dot{\mathbf{x}} &= \mathbf{p} \\ \dot{\mathbf{p}} + \phi_{/\mathbf{x}}^{\mathrm{T}}\lambda &= \mathbf{f}\left(\mathbf{x},\dot{\mathbf{x}},t\right) \\ \phi\left(\mathbf{x},t\right) &= 0 \end{aligned}$$

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”

Problems

$\mathbf{K}'=\mathbf{I}$: Moore-Penrose pseudo-inverse

Experimental inverse dynamics problem

- **inverse kinematics:**

same matrix!

$$\begin{bmatrix} \mathbf{K}' & \phi_{/\mathbf{x}}^T \\ \phi_{/\mathbf{x}} & 0 \end{bmatrix} \begin{Bmatrix} \Delta \mathbf{x} \\ \Delta \lambda_0 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -\phi(\mathbf{x}, t) \end{Bmatrix}$$

nonlinear

$$\begin{bmatrix} \mathbf{K}' & \phi_{/\mathbf{x}}^T \\ \phi_{/\mathbf{x}} & 0 \end{bmatrix} \begin{Bmatrix} \Delta \dot{\mathbf{x}} \\ \Delta \lambda_1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ \mathbf{b}(\mathbf{x}, t) \end{Bmatrix}$$

linear

$$\begin{bmatrix} \mathbf{K}' & \phi_{/\mathbf{x}}^T \\ \phi_{/\mathbf{x}} & 0 \end{bmatrix} \begin{Bmatrix} \Delta \ddot{\mathbf{x}} \\ \Delta \lambda_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ \mathbf{b}'(\mathbf{x}, \dot{\mathbf{x}}, t) \end{Bmatrix}$$

linear

- **the RHS contains the desired motion and its derivatives**
- **the (regularized) static analysis provides the kinematic inversion**

$$\phi_{/\mathbf{x}}^T \Delta \lambda = \mathbf{f}'(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, t)$$

linear

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:**

$$\mathbf{R} = \mathbf{R}(\mathbf{g})$$

- **orthonormality:**

$$\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1}$$

- **derivative:**

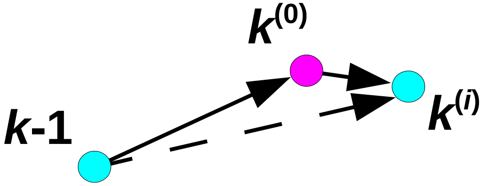
$$\dot{\mathbf{R}}\,\mathbf{R}^{\mathrm{T}} = \omega \times = (\mathbf{G}(\mathbf{g})\,\dot{\mathbf{g}}) \times$$

- **incremental approach from step *k* to *k*+1 to eliminate the orientation parameters singularity issue (increments are necessarily small for accuracy):**

$$\mathbf{R}_{k} = \mathbf{R}(\mathbf{g}_{k})\,\mathbf{R}_{k-1}$$

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:**

$$\mathbf{R}_{\mathbf{k}}^{(i)} = \mathbf{R}\left(\mathbf{g}_{\Delta}^{(i)}\right)\mathbf{R}_{k}^{(0)}$$

- **Derivative:**

$$\dot{\mathbf{R}}_{k}^{(i)}\left(\mathbf{R}_{k}^{(i)}\right)^{\mathrm{T}} = \omega_{k}^{(i)}\times = \left(\mathbf{R}\left(\mathbf{g}_{\Delta}^{(i)}\right)\omega_{k}^{(0)}\right)\times + \left(\mathbf{G}\left(\mathbf{g}_{\Delta}^{(i)}\right)\dot{\mathbf{g}}_{\Delta}^{(i)}\right)\times$$

Arbitrary motion description

- **Incremental orientation from previous step:**
  - **Orientation parameters order of magnitude:**

$$\mathbf{g} \sim \mathbf{O}\left(\|\omega\| \Delta \mathbf{t}\right)$$

- **Incremental orientation from prediction (as in MBDyn):**
  - **Orientation parameters order of magnitude:**

$$\mathbf{g}_{\Delta} \sim \mathbf{O}\left(\Delta \mathbf{t}^{n+1}\right)$$

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:**

$$\mathbf{R}\left(\mathbf{g}_{\Delta}\right) \approx \mathbf{I}$$

(only in Jacobian)

$$\mathbf{G}\left(\mathbf{g}_{\Delta}\right) \approx \mathbf{I}$$

$$\dot{\mathbf{G}}\left(\mathbf{g}_{\Delta}\right) \approx 0$$

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):

$$\theta = \mathbf{ax}\left(\exp^{-1}\left(\mathbf{R}_{\mathbf{a}}^{\mathbf{T}}\mathbf{R}_{\mathbf{b}}\right)\right)$$

$$\mathbf{m} = \mathbf{R}(\xi\theta)\tilde{\mathbf{m}}(\theta)$$

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

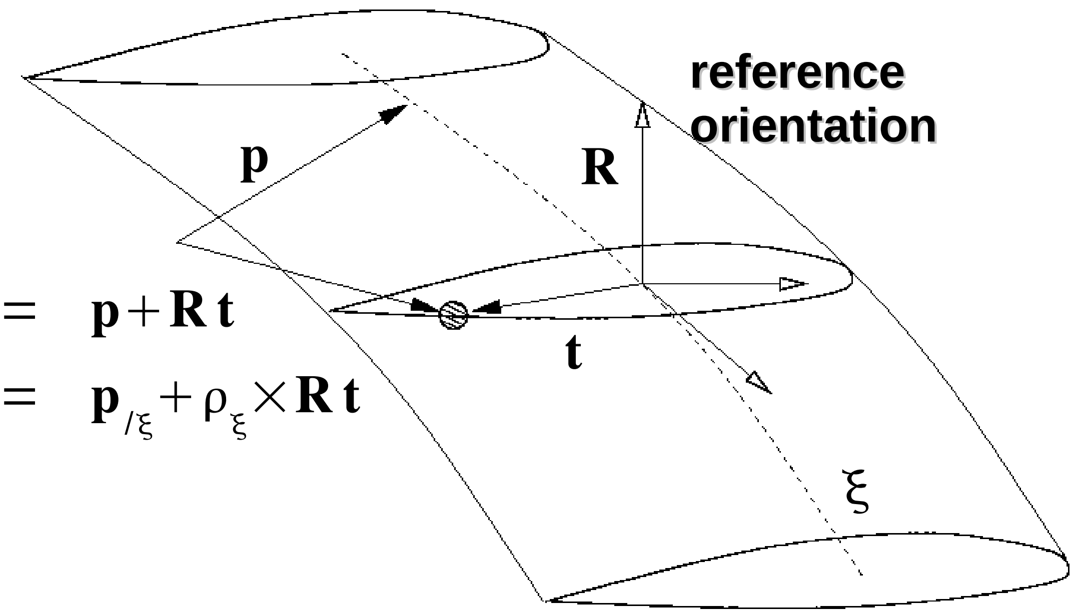
- **Intrinsic form (invariant:** $\xi = 1/2$ **):**
  - **Constitutive properties referred to a floating reference frame**
  - **Intrinsically handles geometrical nonlinearity related to rotations**
  - **Correctly captures bending-torsion buckling behavior**
  - **Essential for anisotropic deformable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

Deformable components

Intrinsic, composite-ready beam

- **Topology:**
  - **1D reference line p, 1D reference structure R**
  - **2D section characterization**

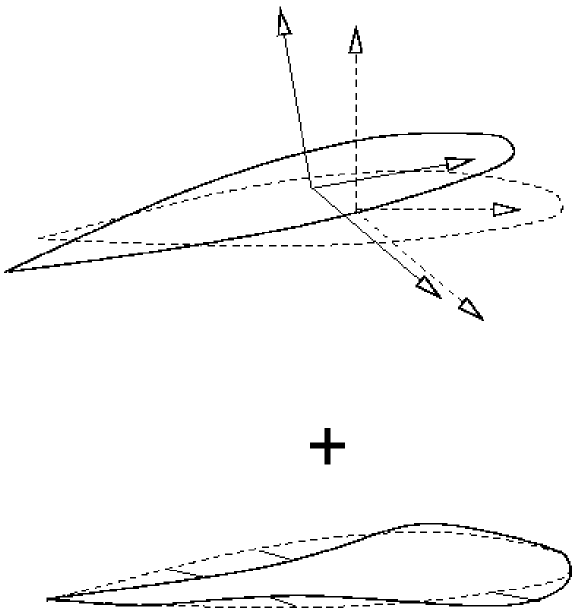
reference line



$$\mathbf{x} = \mathbf{p} + \mathbf{R}\,\mathbf{t}$$

$$\mathbf{x}_{/\xi} = \mathbf{p}_{/\xi} + \rho_{\xi} \times \mathbf{R}\,\mathbf{t}$$

reference motion: $\mathbf{p}, \mathbf{R}$



warping

Deformable components

Intrinsic, composite-ready beams

- **strain measure:**

$$\begin{aligned}\nu &= \mathbf{R}^T \mathbf{p}_{/\xi} - \mathbf{R}_0^T \mathbf{p}_{0/\xi} \\ \kappa &= \mathbf{R}^T \rho_{\xi} - \mathbf{R}_0^T \rho_{\xi 0}\end{aligned}$$

- **equilibrium (from VWP):**

$$\begin{aligned}\mathbf{f}_{/\xi} &= \tau \\ \mathbf{m}_{/\xi} + \mathbf{p}_{/\xi} \times \mathbf{f} &= \mu\end{aligned}$$

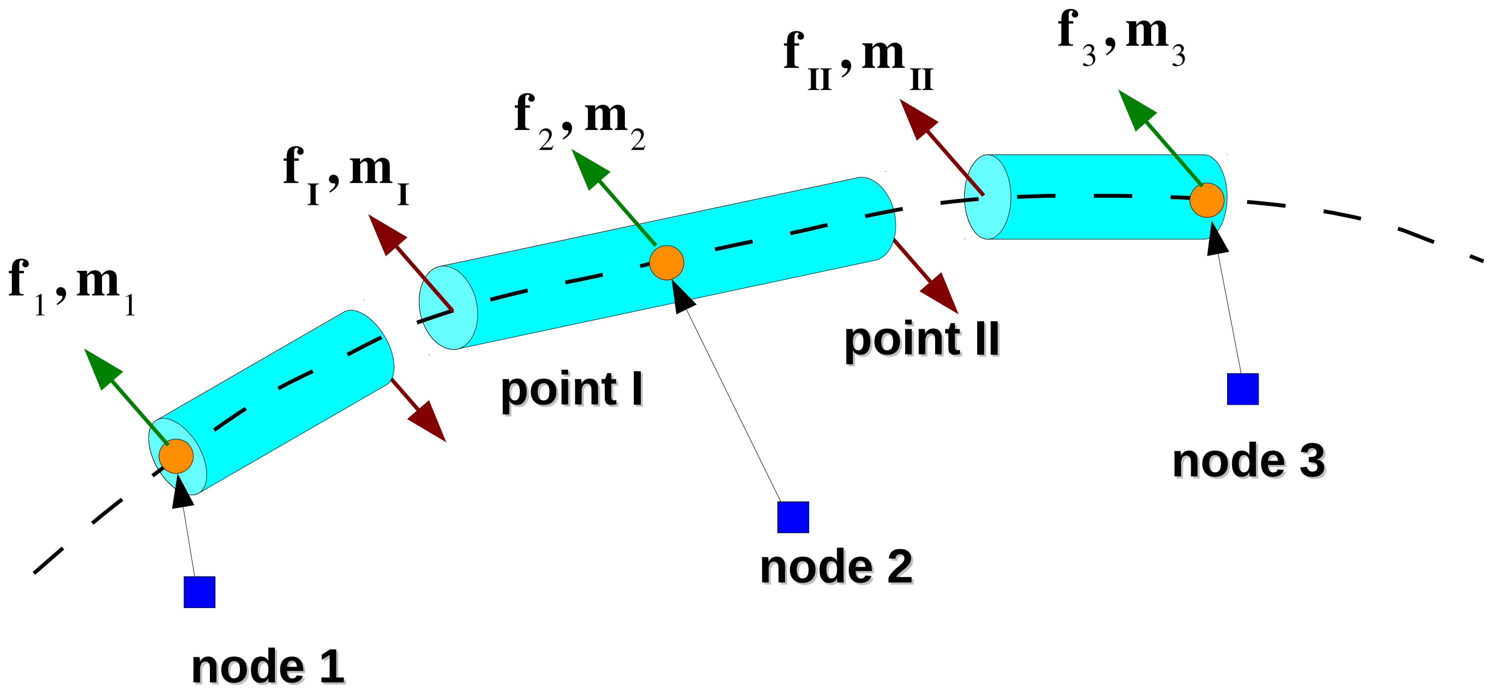
- **constitutive properties:**

$$\begin{aligned}\mathbf{f} &= \mathbf{f}(\nu, \kappa) \\ \mathbf{m} &= \mathbf{m}(\nu, \kappa)\end{aligned}$$

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**

$$\mathbf{y}_k = \mathbf{a}_1 \mathbf{y}_{k-1} + \mathbf{a}_2 \mathbf{y}_{k-2} + \Delta t \left(\mathbf{b}_0 \dot{\mathbf{y}}_k + \mathbf{b}_1 \dot{\mathbf{y}}_{k-1} + \mathbf{b}_2 \dot{\mathbf{y}}_{k-2} \right)$$

- **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:**

$$\dot{\mathbf{y}}_k^{(0)} = (\mathbf{m}_1 \mathbf{y}_{k-1} + \mathbf{m}_2 \mathbf{y}_{k-2})/\Delta t + \mathbf{n}_1 \dot{\mathbf{y}}_{k-1} + \mathbf{n}_2 \dot{\mathbf{y}}_{k-2}$$

$$\mathbf{y}_k^{(0)} = \mathbf{a}_1 \mathbf{y}_{k-1} + \mathbf{a}_2 \mathbf{y}_{k-2} + \Delta t (\mathbf{b}_0 \dot{\mathbf{y}}_k^{(0)} + \mathbf{b}_1 \dot{\mathbf{y}}_{k-1} + \mathbf{b}_2 \dot{\mathbf{y}}_{k-2})$$

- **Correction iteration:**

$$\mathbf{f}_{/\dot{\mathbf{y}}} \Delta \dot{\mathbf{y}}^{(i)} + \mathbf{f}_{/\mathbf{y}} \Delta \mathbf{y}^{(i)} = -\mathbf{f}(\dot{\mathbf{y}}_k^{(i-1)}, \mathbf{y}_k^{(i-1)}, t_k)$$

but

$$\Delta \mathbf{y}^{(i)} = \Delta t\, \mathbf{b}_0 \Delta \dot{\mathbf{y}}^{(i)}$$

the problem becomes algebraic

$$(\mathbf{f}_{/\dot{\mathbf{y}}} + \Delta t\, \mathbf{b}_0 \mathbf{f}_{/\mathbf{y}}) \Delta \dot{\mathbf{y}}^{(i)} = -\mathbf{f}(\dot{\mathbf{y}}_k^{(i-1)}, \mathbf{y}_k^{(i-1)}, t_k)$$

$$\dot{\mathbf{y}}_k^{(i)} = \dot{\mathbf{y}}_k^{(i-1)} + \Delta \dot{\mathbf{y}}^{(i)}$$

$$\mathbf{y}_k^{(i)} = \mathbf{y}_k^{(i-1)} + \Delta t\, \mathbf{b}_0 \Delta \dot{\mathbf{y}}^{(i)}$$

Solving the problem

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:**

$$\phi(\mathbf{x}_{0,} \boldsymbol{t}_0) = 0$$

$$\dot{\phi}(\mathbf{x}_{0,} \boldsymbol{t}_0) = 0$$

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

$$\begin{aligned} \mathbf{K}'(\mathbf{x} - \mathbf{x}_0) + \phi_{/\mathbf{x}}^{\mathbf{T}} \lambda' &= \mathbf{f}' \\ \mathbf{C}'(\dot{\mathbf{x}} - \dot{\mathbf{x}}_0) + \phi_{/\mathbf{x}}^{\mathbf{T}} \mu' &= \dot{\mathbf{f}}' \\ \phi(\mathbf{x}, \boldsymbol{t}_0) &= 0 \\ \dot{\phi}(\mathbf{x}, \boldsymbol{t}_0) &= 0 \end{aligned}$$

Solving the problem

Solution initialization (so-called “derivatives”)

- **explicit problem:**

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$$

- **implicit problem:**

$$0 = \mathbf{f}(\mathbf{y}, \dot{\mathbf{y}}, t)$$

- **modified correction phase to initialize solution:**

$$\left(\mathbf{f}_{/\dot{\mathbf{y}}} + c\,\mathbf{f}_{/\mathbf{y}}\right) \Delta\dot{\mathbf{y}}^{(i)} = -\mathbf{f}\left(\dot{\mathbf{y}}_0^{(i-1)}, \mathbf{y}_0, t_0\right)$$

$$\dot{\mathbf{y}}_0^{(i)} = \dot{\mathbf{y}}_0^{(i-1)} + \Delta\dot{\mathbf{y}}^{(i)}$$

$$\mathbf{y}_0 = \mathbf{y}_0$$

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

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:**

$$\mathbf{X}^{\mathrm{T}}\left(\in \mathbb{R}^{n\times N}\right)=\mathbf{U}\,\Sigma\,\mathbf{V}^{\mathrm{T}}$$

- **The singular values allow to determine the *m* most relevant signals**

$$\mathbf{X}^{\mathrm{T}}_{1:m,n}=\mathbf{U}_{n,1:m}\,\Sigma_{1:m,1:m}\,\mathbf{V}^{\mathrm{T}}_{N,1:m}$$

- **Note that**

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}=\mathbf{U}\,\Sigma^{2}\,\mathbf{U}^{\mathrm{T}} \qquad \mathbf{X}\,\mathbf{X}^{\mathrm{T}}=\mathbf{V}\,\Sigma^{2}\,\mathbf{V}^{\mathrm{T}}$$

$$\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}=\Sigma\,\mathbf{V}^{\mathrm{T}} \qquad \mathbf{X}^{\mathrm{T}}\mathbf{V}=\mathbf{U}\,\Sigma$$

- **This allows to efficiently compute the singular values and the POMs**

Extracting useful information

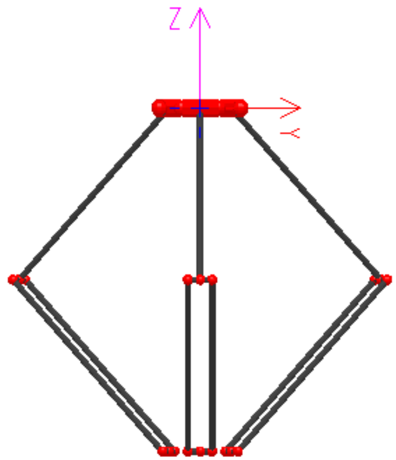
- The POMs can be used to identify a transition matrix

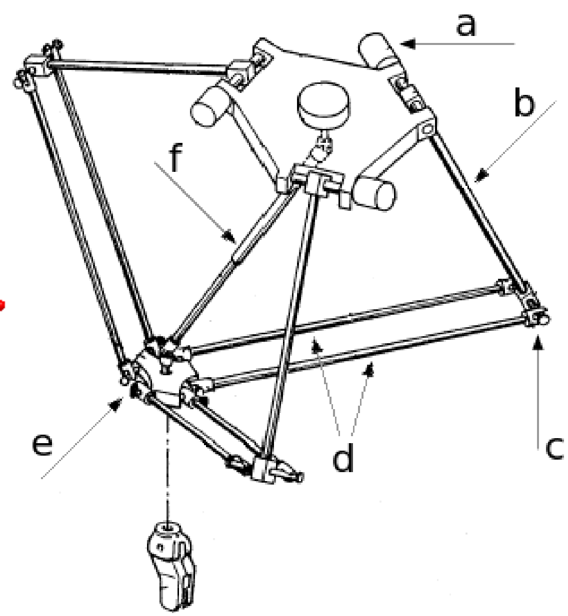
$$\mathbf{X}^{(k+1)} = \Phi\, \mathbf{X}^{(k)}$$

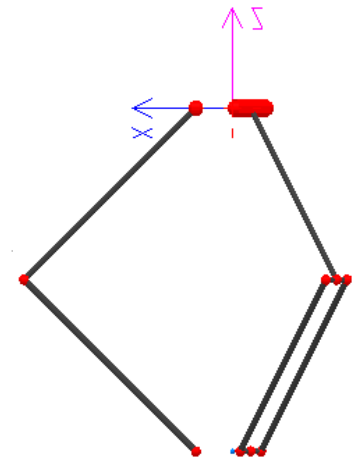
- If X contains the free response of the system, the transition matrix allows to estimate the relevant eigenvalues (AR model)
- More sophisticated system identification techniques can be used (model order reduction is an open research field)
- A technique based on covariance estimates from time histories has been recently proposed; works for:
  - free response
  - forced response
  - unmeasured forced response

Examples of multibody modeling with MBDyn

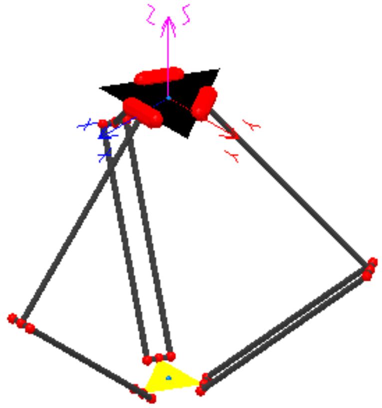
Robotics:
delta robot

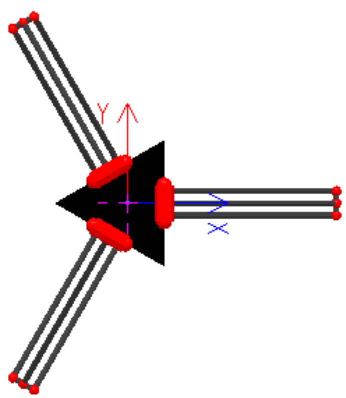






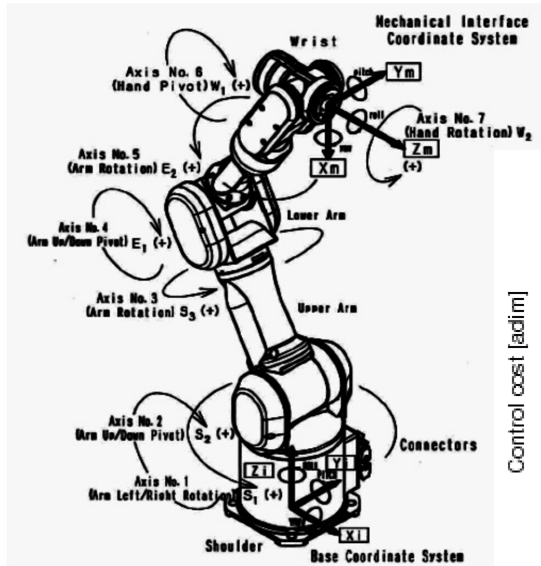
inverse dynamics
for computed
torque control

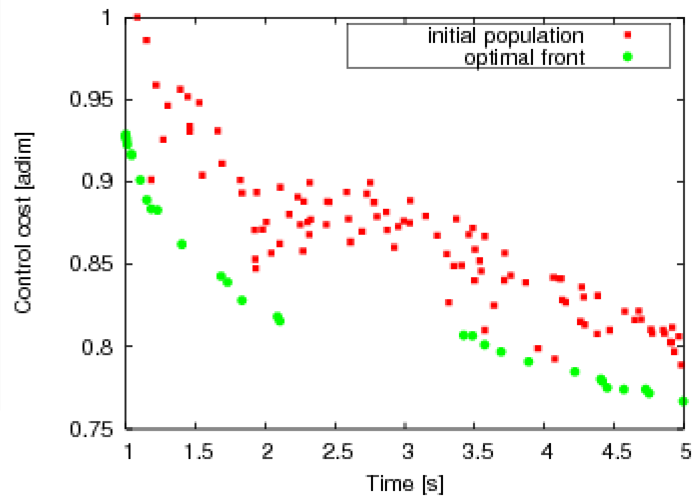


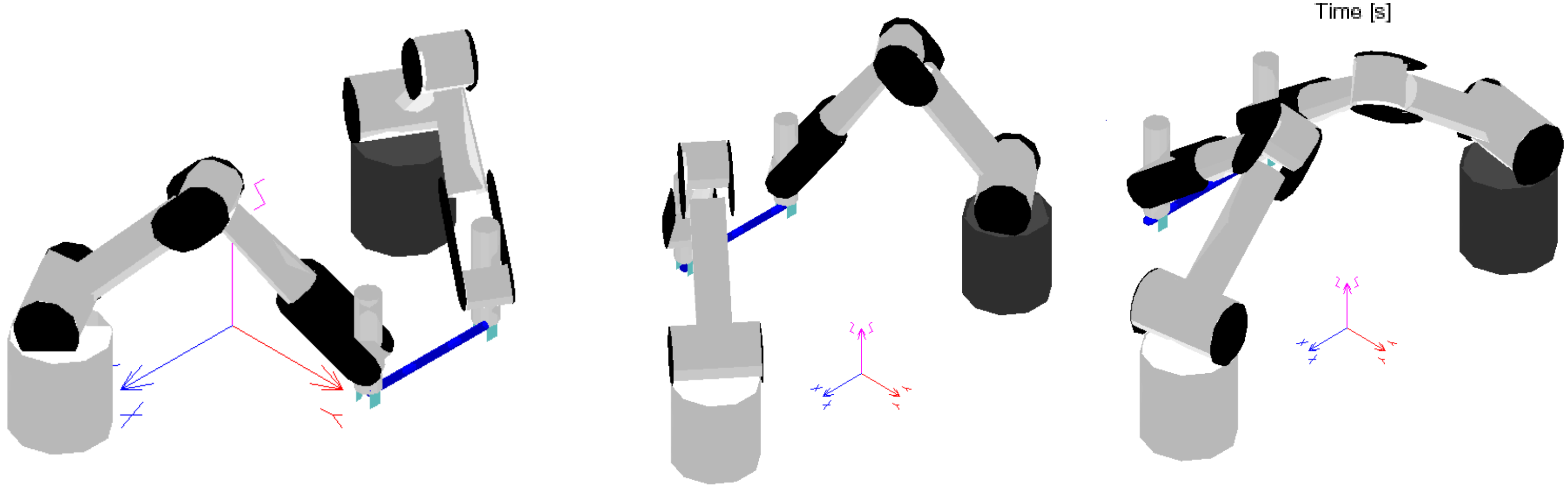


Examples of multibody modeling with MBDyn

Robotics: PA-10 inverse kinematics with path optimization of cooperating robots

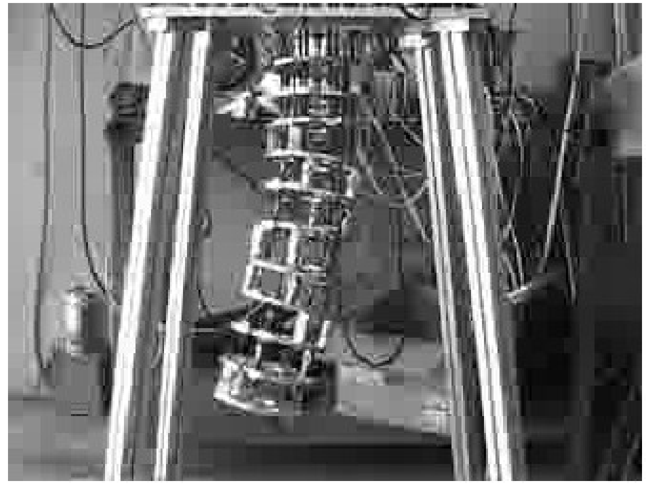


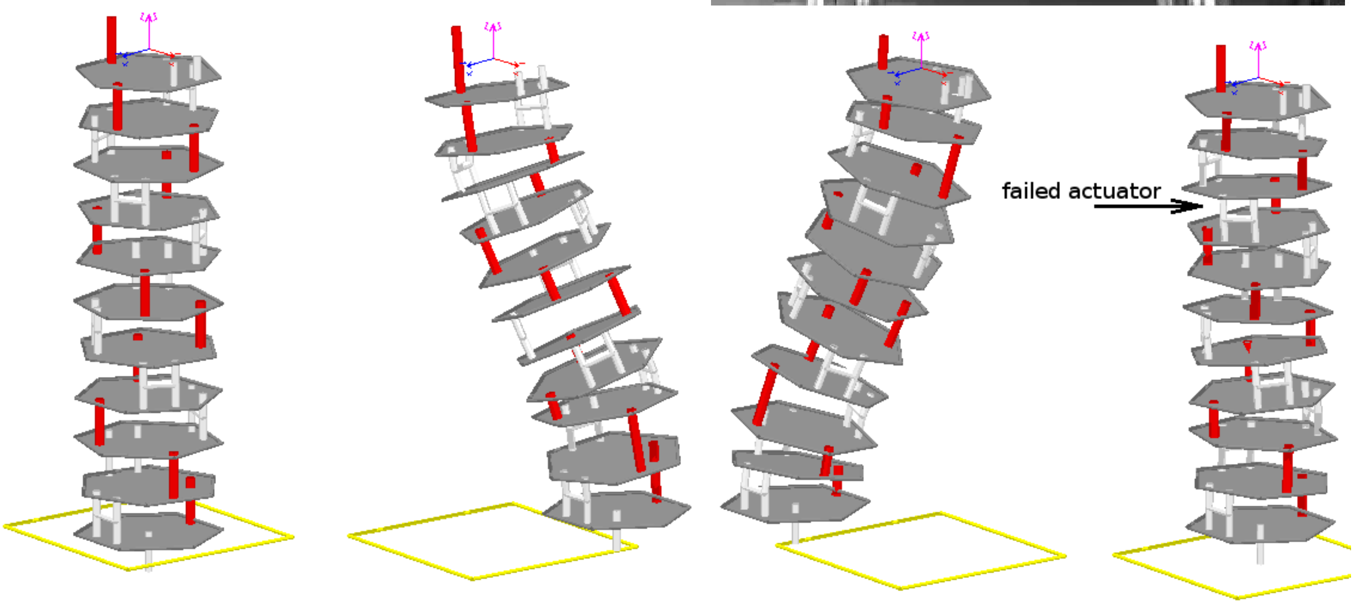


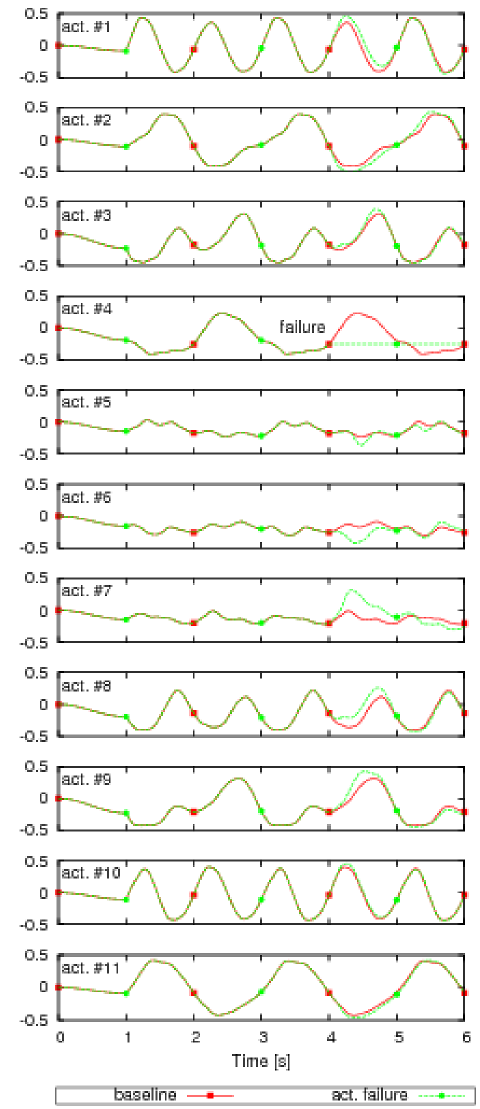


Examples of multibody modeling with MBDyn

Robotics:
biomimetic robot real-time motion planning by inverse kinematics with fault detection







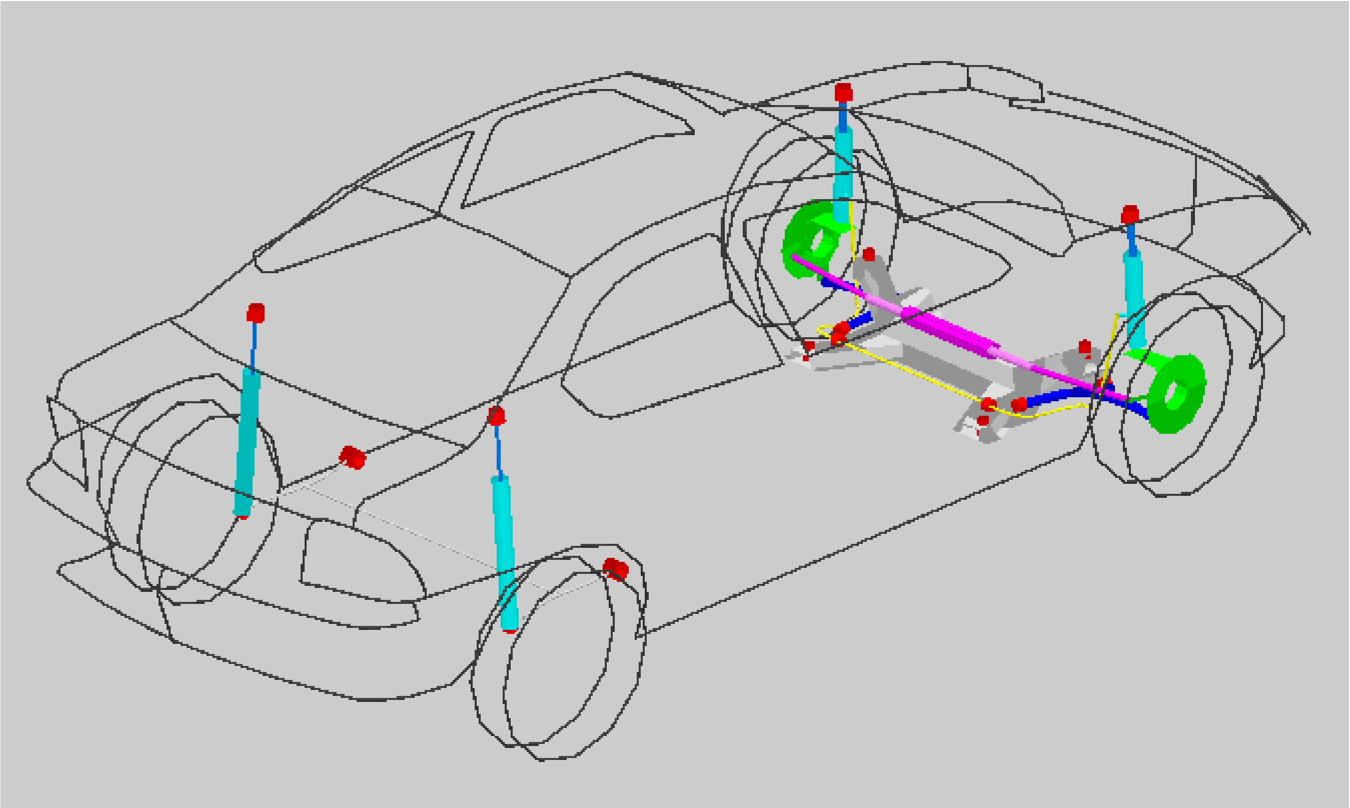
Examples of multibody modeling with MBDyn

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

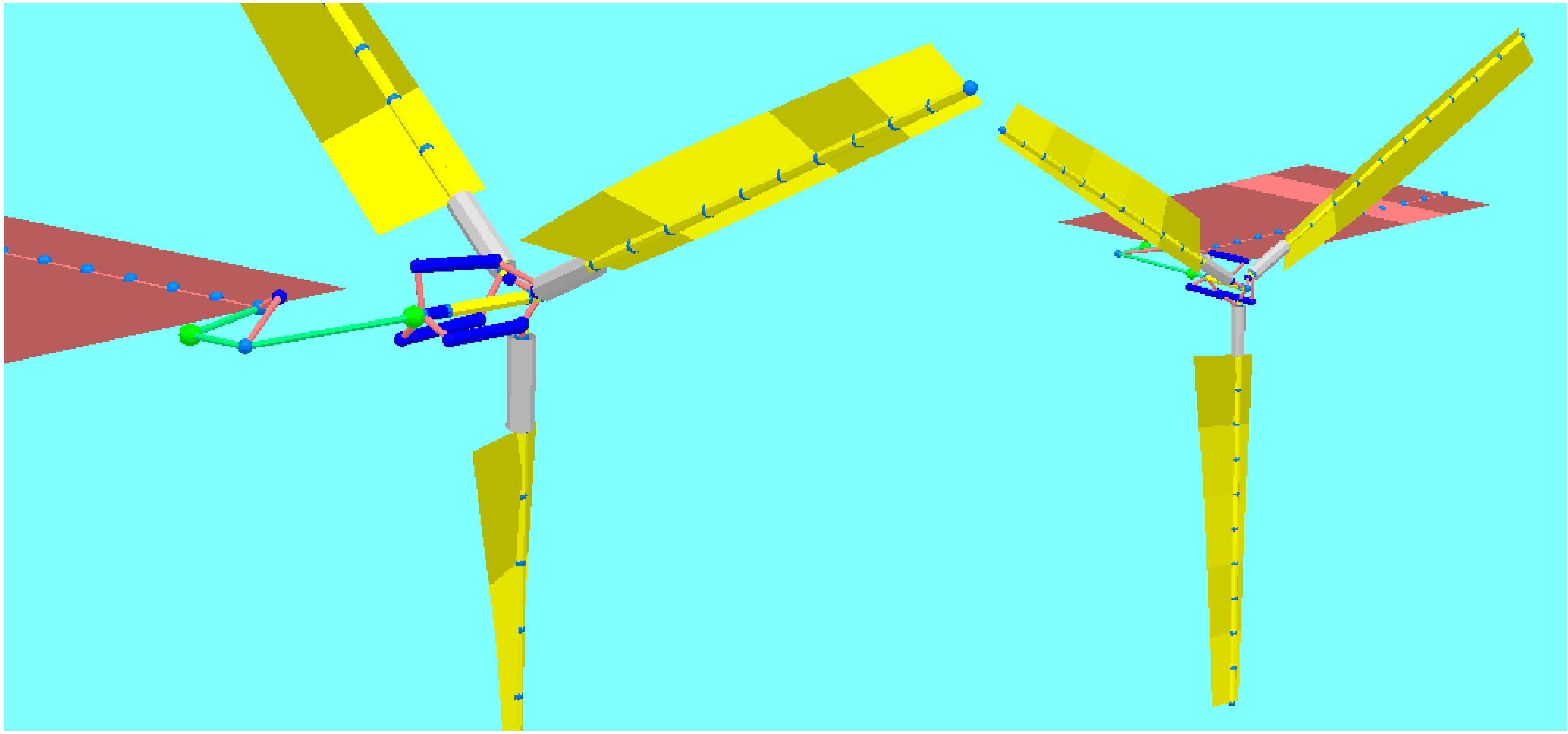
Automotive: mechanical modeling of suspensions
purpose: determine loads in rubber bushings and other components



Examples of multibody modeling with MBDyn

Rotorcraft dynamics and aeroservoelasticity:

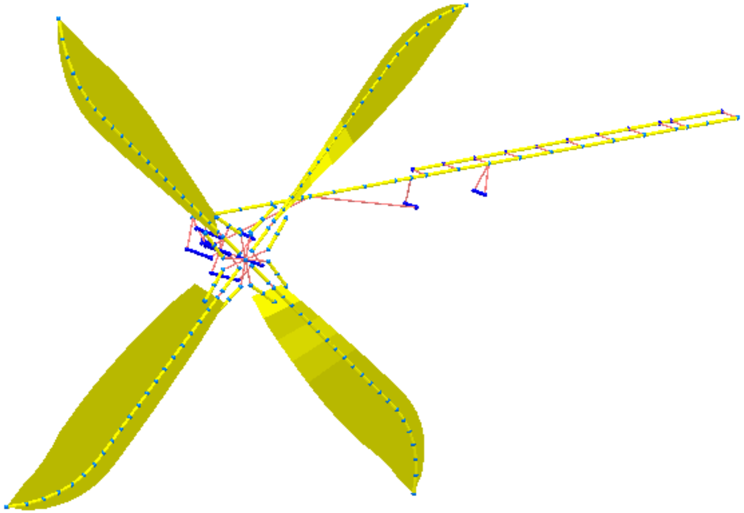
- **WRATS (NASA/Army) tiltrotor aeromechanics**

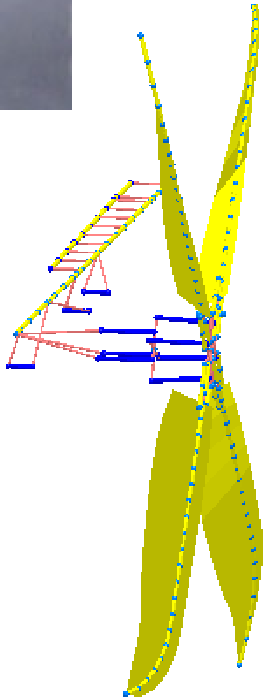


Examples of multibody modeling with MBDyn

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?