ANALYSIS OF LOAD PATTERNS IN RUBBER COMPONENTS FOR VEHICLES

Jerome Merel

formerly at Hutchinson Corporate Research Center

Israël Wander

Apex Technologies

Pierangelo Masarati, Marco Morandini

Dipartimento di Ingegneria Aerospaziale,

Politecnico di Milano

Multibody Dynamics 2007

Milano, June 25–28 2007

POLITECNICO DI MILANO

Outline

Outline

Motivation

- Multibody Dynamics started as formalism for mechanics of rigid-body mechanisms

Motivation

- Multibody Dynamics started as formalism for mechanics of rigid-body mechanisms
- Today: fully developed industrial-grade computational tool

Motivation

- Multibody Dynamics started as formalism for mechanics of rigid-body mechanisms
- Today: fully developed industrial-grade computational tool
- Blend of exact, arbitrary kinematics & nonlinear finite elements

Motivation

- Multibody Dynamics started as formalism for mechanics of rigid-body mechanisms
- Today: fully developed industrial-grade computational tool
- Blend of exact, arbitrary kinematics & nonlinear finite elements
- Ideal playground for multidisciplinary problems

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

$\longrightarrow$ nonlinear finite elements

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

  $\longrightarrow$ nonlinear finite elements

- The market requires to extend analysis capabilities to the entire mechanical system, in order to design and validate single components

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

  $\longrightarrow$ nonlinear finite elements

- The market requires to extend analysis capabilities to the entire mechanical system, in order to design and validate single components

  $\longrightarrow$ multibody dynamics

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

  $\longrightarrow$ nonlinear finite elements

- The market requires to extend analysis capabilities to the entire mechanical system, in order to design and validate single components

  $\longrightarrow$ multibody dynamics

- Hutchinson traditionally developed specialized finite element analysis capabilities in-house

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

  $\longrightarrow$ nonlinear finite elements

- The market requires to extend analysis capabilities to the entire mechanical system, in order to design and validate single components

  $\longrightarrow$ multibody dynamics

- Hutchinson traditionally developed specialized finite element analysis capabilities in-house
- Industrial requirement: preserve in-house analysis capabilities for multibody analysis as well; solution:

Motivation

- Hutchinson, as a worldwide leader in manufacturing of rubber components, has a long tradition in accurate and detailed modeling of their products

  $\longrightarrow$ nonlinear finite elements

- The market requires to extend analysis capabilities to the entire mechanical system, in order to design and validate single components

  $\longrightarrow$ multibody dynamics

- Hutchinson traditionally developed specialized finite element analysis capabilities in-house
- Industrial requirement: preserve in-house analysis capabilities for multibody analysis as well; solution:

  $\longrightarrow$ use free software

  as an alternative to "reinventing the wheel"

Outline

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)
  - Aircraft landing gear analysis

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)
  - Aircraft landing gear analysis
  - Robotics and mechatronics, including real-time simulation

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)
  - Aircraft landing gear analysis
  - Robotics and mechatronics, including real-time simulation
  - Automotive

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3^{rd} parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)
  - Aircraft landing gear analysis
  - Robotics and mechatronics, including real-time simulation
  - Automotive
  - Wind turbines

Software Description

- *MBDyn*: general-purpose MultiBody Dynamics software developed at Politecnico di Milano since the early '90s
- mainly applied to rotorcraft dynamics and aeroservoelasticity, but it is currently exploited (at Politecnico di Milano and by 3rd parties) in projects involving
  - Rotorcraft dynamics (helicopters, tiltrotors)
  - Aircraft landing gear analysis
  - Robotics and mechatronics, including real-time simulation
  - Automotive
  - Wind turbines
  - Biomechanics

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money,

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well,

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

the rights are not restricted

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

   the rights are not restricted

3. modify the source code

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

   the rights are not restricted

3. modify the source code
4. distribute modified sources, for free or for money,

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

   the rights are not restricted

3. modify the source code
4. distribute modified sources, for free or for money, provided the modified source code is distributed as well, and the original rights are not restricted,

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

   the rights are not restricted
3. modify the source code
4. distribute modified sources, for free or for money, provided the modified source code is distributed as well, and the original rights are not restricted, but rather

   extend to modifications

Software Description

MBDyn is free software (GPL). The user is granted the rights to:

1. access the source code
2. distribute the software, for free or for money, provided the source code is distributed as well, and

   the rights are not restricted
3. modify the source code
4. distribute modified sources, for free or for money, provided the modified source code is distributed as well, and the original rights are not restricted, but rather

   extend to modifications

Further information at

http://www.aero.polimi.it/~mbdyn/

Outline

Connectivity

Mechanical systems are modeled as nodes, that provide *shared* equations and degrees of freedom, connected by elements, that contribute to *shared* equations and optionally provide *private* ones.

Connectivity

Mechanical systems are modeled as nodes, that provide *shared* equations and degrees of freedom, connected by elements, that contribute to *shared* equations and optionally provide *private* ones.

Relevant elements can be

Connectivity

Mechanical systems are modeled as nodes, that provide *shared* equations and degrees of freedom, connected by elements, that contribute to *shared* equations and optionally provide *private* ones.

Relevant elements can be

- rigid bodies
  $\longrightarrow$ contribute inertia properties to nodes

Connectivity

Mechanical systems are modeled as nodes, that provide *shared* equations and degrees of freedom, connected by elements, that contribute to *shared* equations and optionally provide *private* ones.

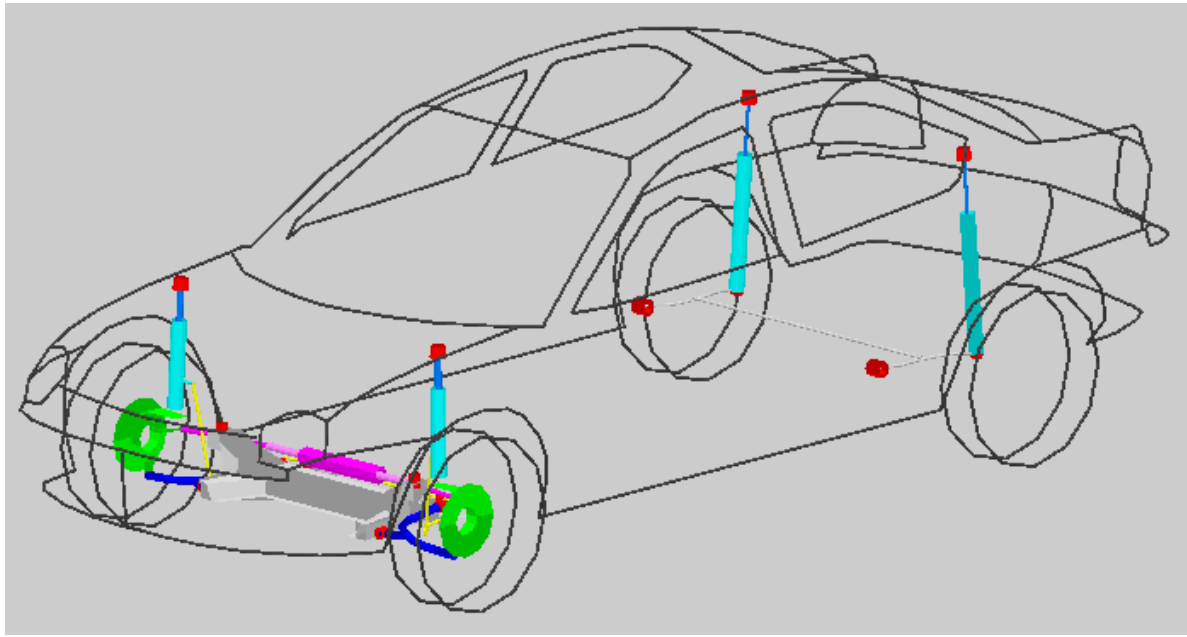
Relevant elements can be

- rigid bodies
  $\longrightarrow$ contribute inertia properties to nodes
- joints
  $\longrightarrow$ add private algebraic equations that constrain nodes
  $\longrightarrow$ contribute constraint reactions to shared equations

Connectivity

Selected test application: car suspensions model

a wide variety of deformable components model rubber parts



Connectivity

Deformable components:

Connectivity

Deformable components:

- nodes exchange configuration-dependent forces and moments

Connectivity

Deformable components:

- nodes exchange configuration-dependent forces and moments
- separation between constitutive model...

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}, \ldots\right)$$

Connectivity

Deformable components:

- nodes exchange configuration-dependent forces and moments
- separation between constitutive model...

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}, \ldots\right)$$

...and connectivity

$$\begin{aligned}
\mathbf{u} &= \mathbf{u}\left(\mathbf{u}_1, \mathbf{u}_2\right) \\
\mathbf{F}_1 &= \mathbf{T}_1\left(\mathbf{u}_1, \mathbf{u}_2\right)\mathbf{F} \\
\mathbf{F}_2 &= \mathbf{T}_2\left(\mathbf{u}_1, \mathbf{u}_2\right)\mathbf{F}
\end{aligned}$$

Connectivity

Deformable components:

- nodes exchange configuration-dependent forces and moments
- separation between constitutive model...

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}, \ldots\right)$$

...and connectivity

$$\begin{aligned}\mathbf{u} &= \mathbf{u}\left(\mathbf{u}_1, \mathbf{u}_2\right)\\ \mathbf{F}_1 &= \mathbf{T}_1\left(\mathbf{u}_1, \mathbf{u}_2\right)\mathbf{F}\\ \mathbf{F}_2 &= \mathbf{T}_2\left(\mathbf{u}_1, \mathbf{u}_2\right)\mathbf{F}\end{aligned}$$

- Connectivity and constitutive models development is decoupled; provided an adequately expressive API is designed, they mutually benefit from each other.

Connectivity

Deformable components implemented in MBDyn:

Connectivity

Deformable components implemented in MBDyn:

- 1D component: rod
- 3D component: angular spring
- 3D component: linear spring
- 6D component: linear & angular spring
- 6D component: geometrically "exact", composite ready beam
- Component Mode Synthesis (CMS) element

Connectivity

Deformable components implemented in MBDyn:

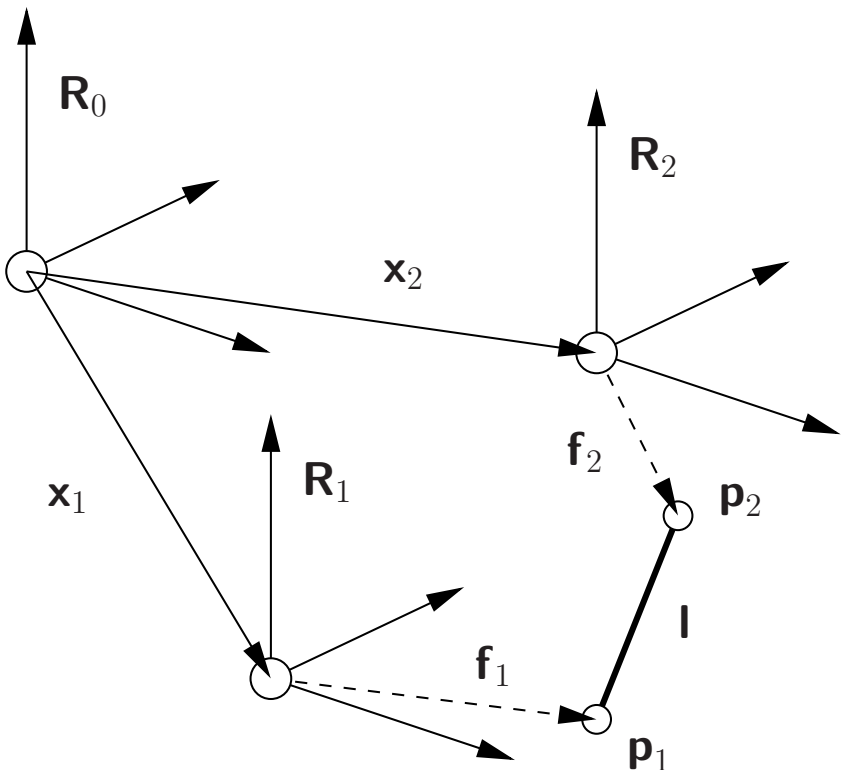
- 1D component: rod
- 3D component: angular spring
- 3D component: linear spring
- 6D component: linear & angular spring
- 6D component: geometrically "exact", composite ready beam
- Component Mode Synthesis (CMS) element

Connectivity

Deformable components implemented in MBDyn:

- 1D component: rod
- 3D component: angular spring
- 3D component: linear spring
- 6D component: linear & angular spring
- 6D component: geometrically "exact", composite ready beam
- Component Mode Synthesis (CMS) element

1D Component: Rod



- connects two points $\mathbf{p}_1$, $\mathbf{p}_2$
- optionally offset from the respective nodes:

$$\mathbf{p}_1 = \mathbf{x}_1 + \mathbf{f}_1$$
$$\mathbf{p}_2 = \mathbf{x}_2 + \mathbf{f}_2$$

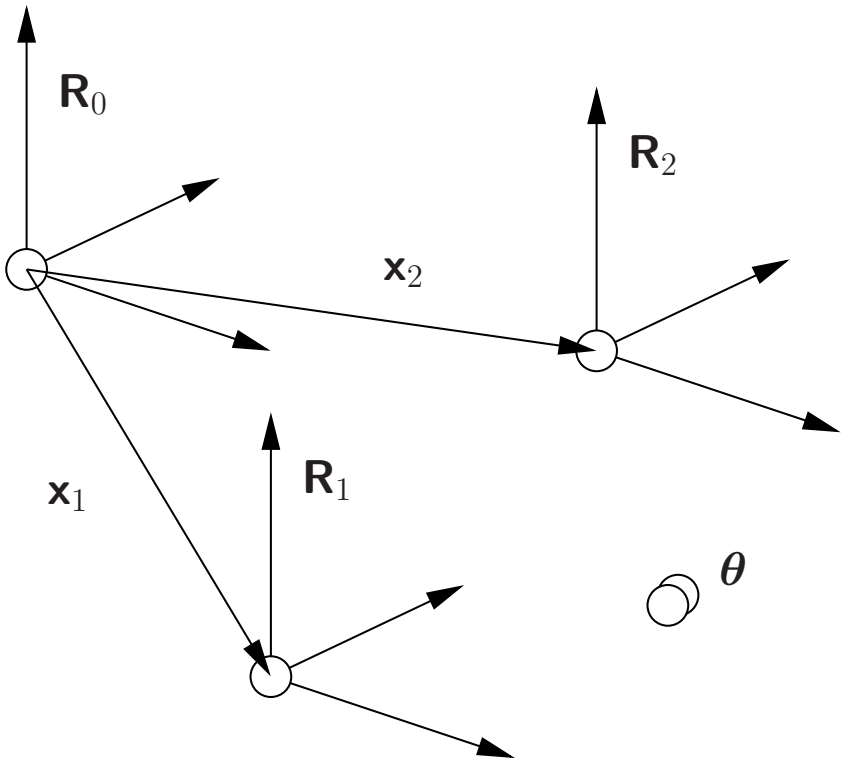
- offsets are rigidly connected to nodes

$$\mathbf{f}_1 = \mathbf{R}_1 \bar{\mathbf{f}}_1$$
$$\mathbf{f}_2 = \mathbf{R}_2 \bar{\mathbf{f}}_2$$

- straining related to distance between points $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$\mathbf{l} = \mathbf{p}_2 - \mathbf{p}_1 \qquad \varepsilon = \frac{\sqrt{\mathbf{l}^T \mathbf{l}}}{l_0} - 1$$

3D Component: Angular Spring



- connects two nodes $\mathbf{x}_1$, $\mathbf{x}_2$
- straining related to perturbation θ of relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- the joint has no location in space (it is typically paired to a spherical hinge or other relative position constraint)

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \mathrm{ax}\left(\exp^{-1}(\mathbf{R})\right)$

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \mathrm{ax}\left(\exp^{-1}(\mathbf{R})\right)$
- relative angular velocity $\boldsymbol{\omega} = \mathbf{R}_1^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \mathrm{ax}\left(\exp^{-1}(\mathbf{R})\right)$
- relative angular velocity $\boldsymbol{\omega} = \mathbf{R}_1^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$, referred to node 1

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \text{ax}\left(\exp^{-1}(\mathbf{R})\right)$
- relative angular velocity $\boldsymbol{\omega} = \mathbf{R}_1^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$, referred to node 1
- contribution to virtual work $\delta\mathcal{L} = -\boldsymbol{\theta}_\delta^T \overline{\mathbf{M}}$

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \mathrm{ax}\left(\exp^{-1}(\mathbf{R})\right)$
- relative angular velocity $\boldsymbol{\omega} = \mathbf{R}_1^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$, referred to node 1
- contribution to virtual work $\delta\mathcal{L} = -\boldsymbol{\theta}_\delta^T \overline{\mathbf{M}}$
- relative orientation virtual perturbation $\boldsymbol{\theta}_\delta = \mathbf{R}_1^T (\boldsymbol{\theta}_{2\delta} - \boldsymbol{\theta}_{1\delta})$

3D Component: Angular Spring

- relative orientation matrix $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$
- $\longrightarrow$ implies that the component constitutive properties are intrinsically referred to node 1
- relative orientation vector $\boldsymbol{\theta} = \mathrm{ax}\left(\exp^{-1}(\mathbf{R})\right)$
- relative angular velocity $\boldsymbol{\omega} = \mathbf{R}_1^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \boldsymbol{\omega})$, referred to node 1
- contribution to virtual work $\delta\mathcal{L} = -\boldsymbol{\theta}_\delta^T \overline{\mathbf{M}}$
- relative orientation virtual perturbation $\boldsymbol{\theta}_\delta = \mathbf{R}_1^T (\boldsymbol{\theta}_{2\delta} - \boldsymbol{\theta}_{1\delta})$
- contributions to node equilibrium

$$\mathbf{M}_1 = \mathbf{M}$$
$$\mathbf{M}_2 = -\mathbf{M}$$

with $\mathbf{M} = \mathbf{R}_1 \overline{\mathbf{M}}$

3D Component: Angular Spring

Pros:

- the formulation is straightforward

3D Component: Angular Spring

Pros:

- the formulation is straightforward

Cons:

- the model is biased towards one node
- as a consequence, formulating any constitutive law but isotropic may not be straightforward

3D Component: Angular Spring

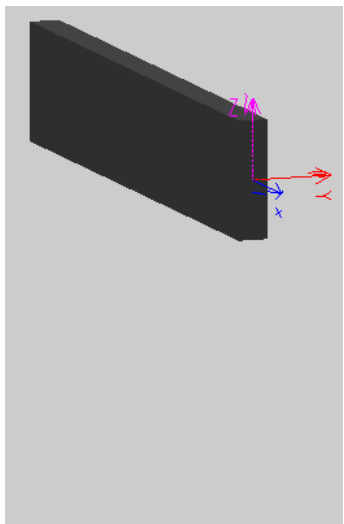
Pros:

- the formulation is straightforward

Cons:

- the model is biased towards one node
- as a consequence, formulating any constitutive law but isotropic may not be straightforward

With linear anisotropic constitutive law, if connectivity is reversed:



3D Component: Angular Spring

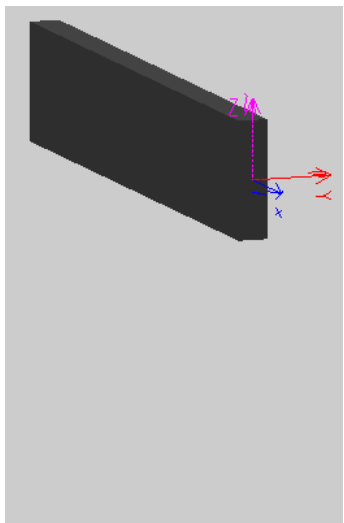
Pros:

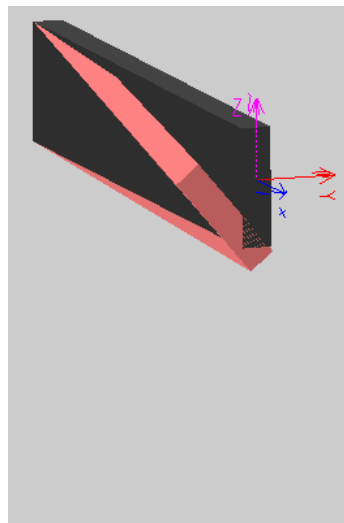
- the formulation is straightforward

Cons:

- the model is biased towards one node
- as a consequence, formulating any constitutive law but isotropic may not be straightforward

With linear anisotropic constitutive law, if connectivity is reversed:





3D Component: Angular Spring

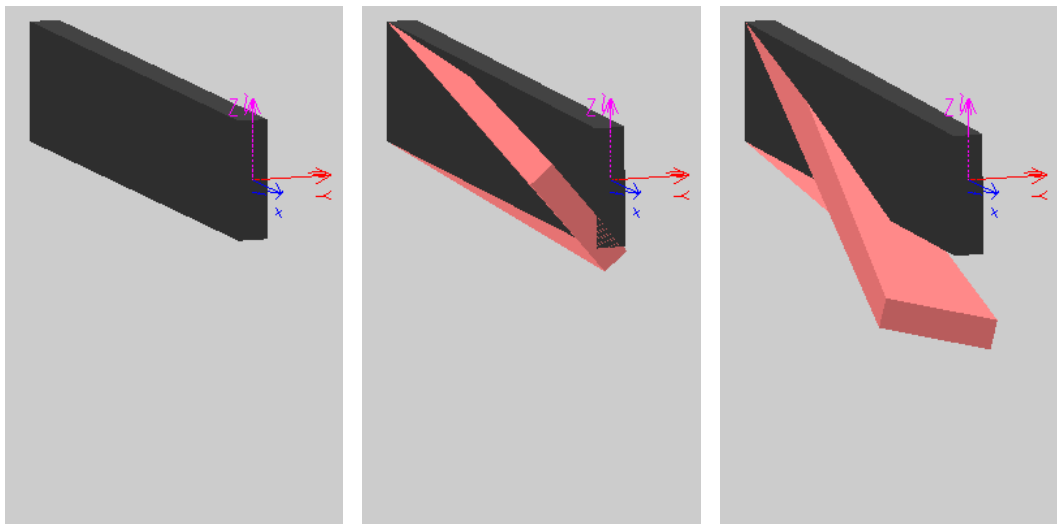
Pros:

- the formulation is straightforward

Cons:

- the model is biased towards one node
- as a consequence, formulating any constitutive law but isotropic may not be straightforward

With linear anisotropic constitutive law, if connectivity is reversed:



3D Component: Angular Spring

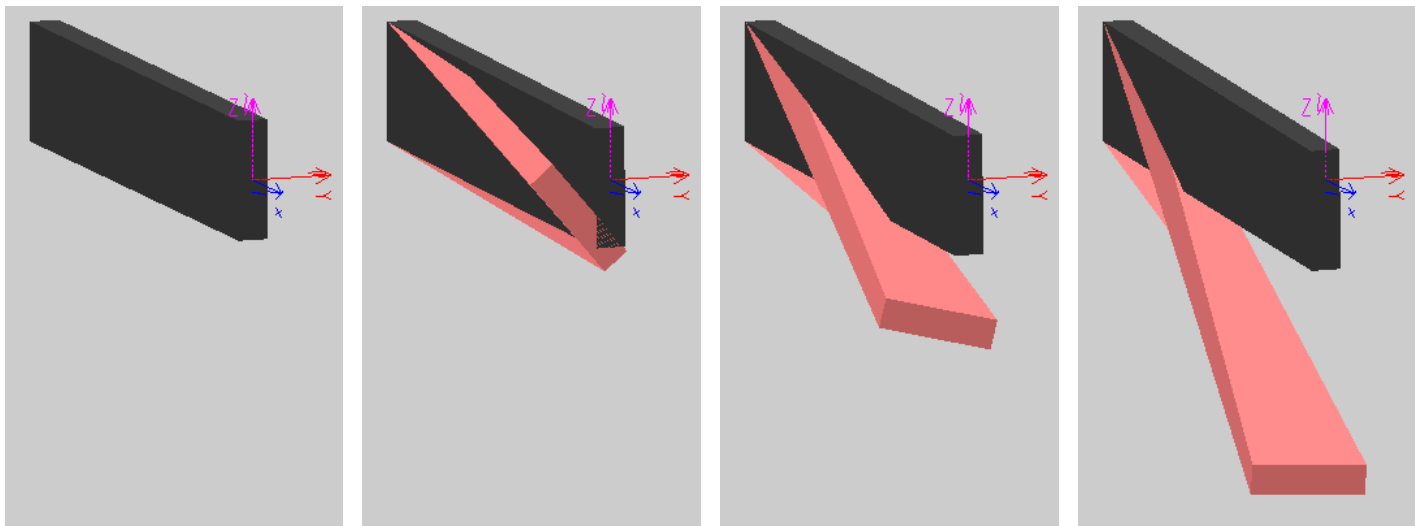
Pros:

- the formulation is straightforward

Cons:

- the model is biased towards one node
- as a consequence, formulating any constitutive law but isotropic may not be straightforward

With linear anisotropic constitutive law, if connectivity is reversed:



3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity
- in all cases, linear (visco)elastic properties were used

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity
- in all cases, linear (visco)elastic properties were used
- reversing the order of the nodes always solved the problem

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity
- in all cases, linear (visco)elastic properties were used
- reversing the order of the nodes always solved the problem

Rationale:

- the behavior of the component is intrinsically independent from the ordering of the connectivity

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity
- in all cases, linear (visco)elastic properties were used
- reversing the order of the nodes always solved the problem

Rationale:

- the behavior of the component is intrinsically independent from the ordering of the connectivity
- if the connectivity formulation depends on its ordering, the constitutive properties need to take care of invariance

3D Component: "Invariant" Angular Spring

From now on, the previous angular spring is termed "attached"

- the "attached" formulation issue quickly showed up when using orthotropic bushings in the car suspension model
- it took a while to find out that unstable oscillations were caused by geometrical nonlinearity in the bushings connectivity
- in all cases, linear (visco)elastic properties were used
- reversing the order of the nodes always solved the problem

Rationale:

- the behavior of the component is intrinsically independent from the ordering of the connectivity
- if the connectivity formulation depends on its ordering, the constitutive properties need to take care of invariance
- otherwise, connectivity must take care of invariance itself

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$,

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$
- perturbation of intermediate orientation $\tilde{\boldsymbol{\theta}}_\delta = \left(\mathbf{I} + \tilde{\mathbf{R}}\right)^{-1} \boldsymbol{\theta}_\delta$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$
- perturbation of intermediate orientation $\tilde{\boldsymbol{\theta}}_\delta = \left(\mathbf{I} + \tilde{\mathbf{R}}\right)^{-1} \boldsymbol{\theta}_\delta$
- absolute mid-rotation orientation $\hat{\mathbf{R}} = \mathbf{R}_1\tilde{\mathbf{R}} = \mathbf{R}_2\tilde{\mathbf{R}}^T$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$
- perturbation of intermediate orientation $\tilde{\boldsymbol{\theta}}_\delta = \left(\mathbf{I} + \tilde{\mathbf{R}}\right)^{-1} \boldsymbol{\theta}_\delta$
- absolute mid-rotation orientation $\hat{\mathbf{R}} = \mathbf{R}_1\tilde{\mathbf{R}} = \mathbf{R}_2\tilde{\mathbf{R}}^T$
- relative angular velocity $\overline{\boldsymbol{\omega}} = \hat{\mathbf{R}}^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$
- perturbation of intermediate orientation $\tilde{\boldsymbol{\theta}}_\delta = \left(\mathbf{I} + \tilde{\mathbf{R}}\right)^{-1} \boldsymbol{\theta}_\delta$
- absolute mid-rotation orientation $\hat{\mathbf{R}} = \mathbf{R}_1\tilde{\mathbf{R}} = \mathbf{R}_2\tilde{\mathbf{R}}^T$
- relative angular velocity $\overline{\boldsymbol{\omega}} = \hat{\mathbf{R}}^T (\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment is now $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \overline{\boldsymbol{\omega}})$

3D Component: "Invariant" Angular Spring

- relative orientation $\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2$ remains the same
- constitutive properties referred to mid-rotation $\tilde{\boldsymbol{\theta}} = \frac{1}{2}\boldsymbol{\theta}$, such that intermediate relative orientation matrix $\tilde{\mathbf{R}} = \exp\left(\tilde{\boldsymbol{\theta}} \times\right)$ yields $\tilde{\mathbf{R}}\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^2 = \mathbf{R}$
- perturbation of intermediate orientation $\tilde{\boldsymbol{\theta}}_\delta = \left(\mathbf{I} + \tilde{\mathbf{R}}\right)^{-1} \boldsymbol{\theta}_\delta$
- absolute mid-rotation orientation $\hat{\mathbf{R}} = \mathbf{R}_1\tilde{\mathbf{R}} = \mathbf{R}_2\tilde{\mathbf{R}}^T$
- relative angular velocity $\overline{\boldsymbol{\omega}} = \hat{\mathbf{R}}^T(\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1)$
- internal moment is now $\overline{\mathbf{M}} = \overline{\mathbf{M}}(\boldsymbol{\theta}, \overline{\boldsymbol{\omega}})$
- internal moment in the absolute frame $\mathbf{M} = \hat{\mathbf{R}}\overline{\mathbf{M}}$

3D Component: "Invariant" Angular Spring

Pros:

- the model is no longer biased towards one node
- simpler constitutive laws may be formulated

3D Component: "Invariant" Angular Spring

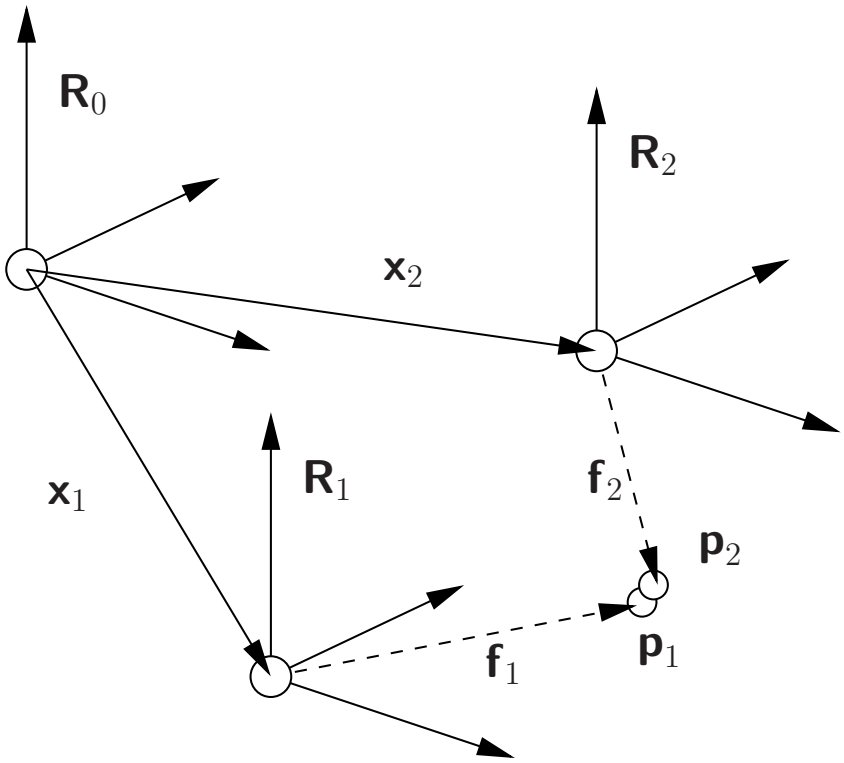
Pros:

- the model is no longer biased towards one node
- simpler constitutive laws may be formulated

Cons:

- the formulation is less straightforward
- little bit more computationally expensive

3D Component: Linear Spring



- Same as rod, but...
- straining related to distance between points $\varepsilon = \mathbf{p}_2 - \mathbf{p}_1$
- the joint does not react pure relative rotation (pin; usually paired to relative orientation constraint)

3D Component: Linear Spring

- same issue about "attached" vs. "invariant" formulation

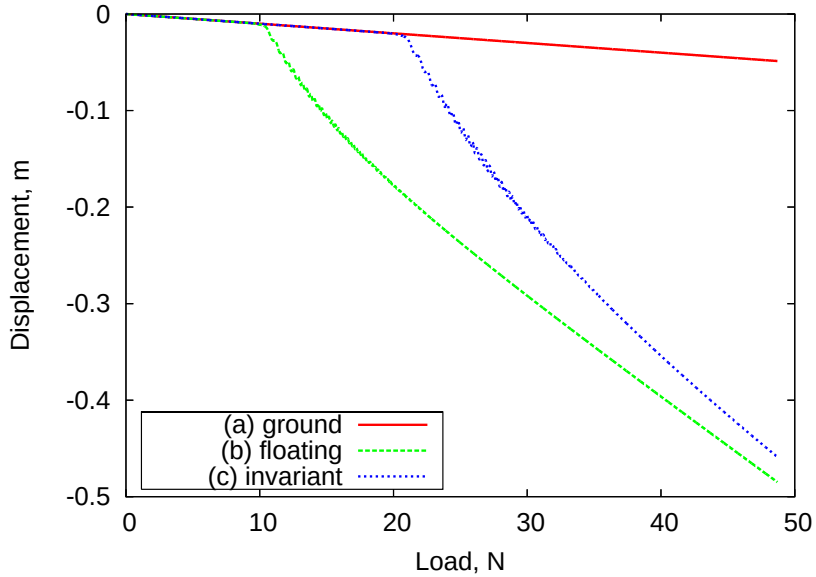
3D Component: Linear Spring

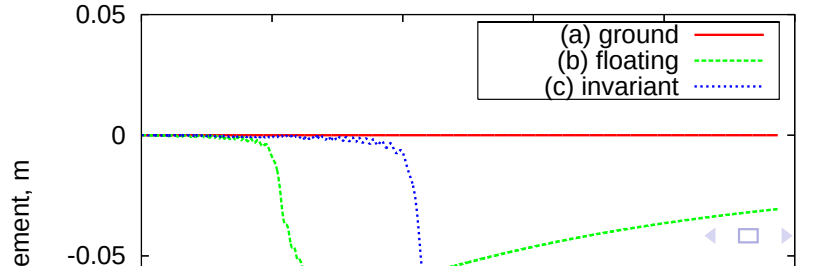
- same issue about "attached" vs. "invariant" formulation
- formulation of invariant case even less straightforward (not presented here, but fully developed and implemented in the software)

3D Component: Linear Spring

- same issue about "attached" vs. "invariant" formulation
- formulation of invariant case even less straightforward (not presented here, but fully developed and implemented in the software)

Same linear anisotropic constitutive law, transverse shear case





6D Components

- linear & angular spring

6D Components

- linear & angular spring
  - models fully coupled bushings

6D Components

- linear & angular spring
  - models fully coupled bushings
  - formulation fully developed, but...

6D Components

- linear & angular spring
  - models fully coupled bushings
  - formulation fully developed, but...
  - ...only partially implemented, essentially because of limited usefulness so far

6D Components

- linear & angular spring
  - models fully coupled bushings
  - formulation fully developed, but...
  - ...only partially implemented, essentially because of limited usefulness so far
- kinematically "exact", composite ready beam

6D Components

- linear & angular spring
  - models fully coupled bushings
  - formulation fully developed, but...
  - ...only partially implemented, essentially because of limited usefulness so far
- kinematically "exact", composite ready beam
  - detailed description outside the scope of this work

6D Components

- linear & angular spring
  - models fully coupled bushings
  - formulation fully developed, but...
  - ...only partially implemented, essentially because of limited usefulness so far
- kinematically "exact", composite ready beam
  - detailed description outside the scope of this work
  - see Ghiringhelli et al., AIAA Journal, 2000

Outline

Constitutive Laws

- define the input/output relationship required by deformable components

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}\right)$$

Constitutive Laws

- define the input/output relationship required by deformable components

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}\right)$$

- task separation allows to exploit similar constitutive laws in different contexts, without bothering about connectivity

$$\delta\mathbf{F} = \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\delta\mathbf{u} + \frac{\partial \mathbf{F}}{\partial \dot{\mathbf{u}}}\delta\dot{\mathbf{u}}$$

Constitutive Laws

- define the input/output relationship required by deformable components

$$\mathbf{F} = \mathbf{F}\left(\mathbf{u}, \dot{\mathbf{u}}\right)$$

- task separation allows to exploit similar constitutive laws in different contexts, without bothering about connectivity

$$\delta\mathbf{F} = \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\delta\mathbf{u} + \frac{\partial \mathbf{F}}{\partial \dot{\mathbf{u}}}\delta\dot{\mathbf{u}}$$

- this aspect is emphasized in the implementation exploiting C++ *templates* for different dimensionalities (1D, 3D, 6D)

$$\delta\mathbf{F}_n = \underbrace{\frac{\partial \mathbf{F}_n}{\partial \mathbf{u}_n}}_{n\times n}\delta\mathbf{u}_n + \underbrace{\frac{\partial \mathbf{F}_n}{\partial \dot{\mathbf{u}}_n}}_{n\times n}\delta\dot{\mathbf{u}}_n$$

Constitutive Laws

Use:

Constitutive Laws

Use:

- call `Update()` at each iteration

Constitutive Laws

Use:

- call `Update()` at each iteration
- subsequent calls to `F()`, `FDE()`, `FDEPrime()` allow to access the force and its partial derivatives (e.g. to compute the residual or the contribution to the Jacobian matrix)

Constitutive Laws

Use:

- call `Update()` at each iteration
- subsequent calls to `F()`, `FDE()`, `FDEPrime()` allow to access the force and its partial derivatives (e.g. to compute the residual or the contribution to the Jacobian matrix)
- as soon as `AfterConvergence()` is called, the solution converged, and the final state can be consolidated, if needed

Constitutive Laws

Use:

- call `Update()` at each iteration
- subsequent calls to `F()`, `FDE()`, `FDEPrime()` allow to access the force and its partial derivatives (e.g. to compute the residual or the contribution to the Jacobian matrix)
- as soon as `AfterConvergence()` is called, the solution converged, and the final state can be consolidated, if needed
- the paper contains examples of C++ meta-code for constitutive laws, including isotropic and orthotropic templates

Constitutive Laws

Use:

- call `Update()` at each iteration
- subsequent calls to `F()`, `FDE()`, `FDEPrime()` allow to access the force and its partial derivatives (e.g. to compute the residual or the contribution to the Jacobian matrix)
- as soon as `AfterConvergence()` is called, the solution converged, and the final state can be consolidated, if needed
- the paper contains examples of C++ meta-code for constitutive laws, including isotropic and orthotropic templates
- or, refer to `mbdyn/base/constltp*` files in the source code

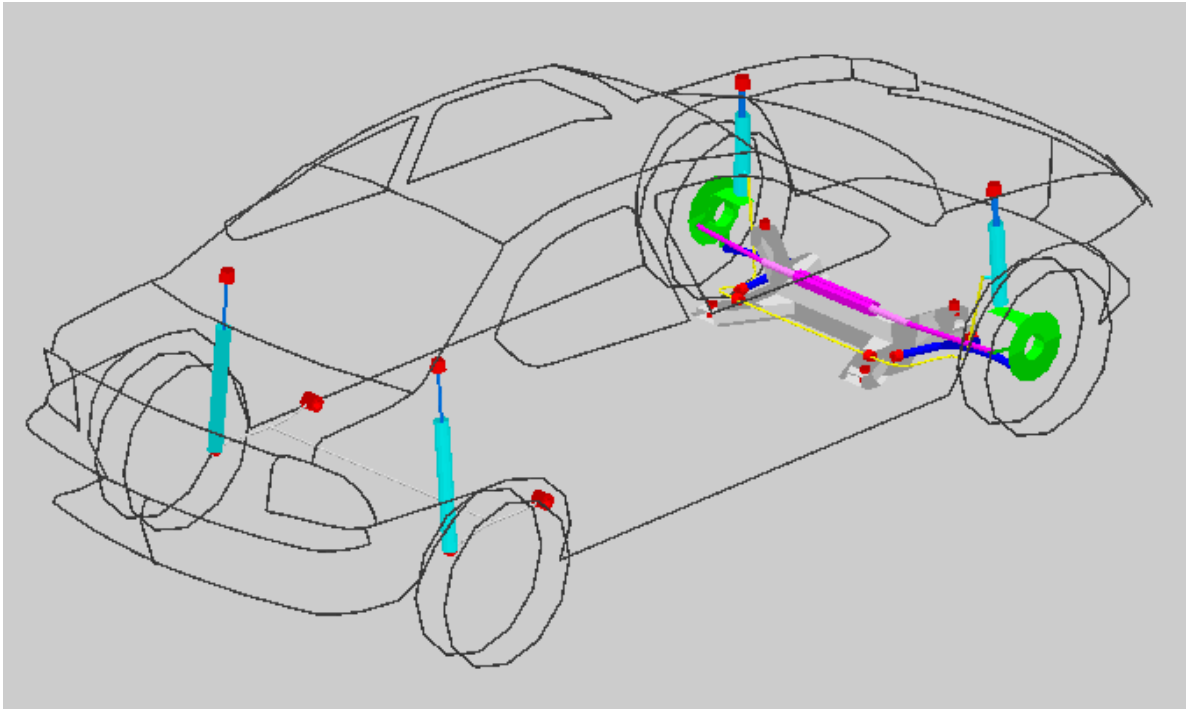
Outline

Application: Car Suspension Model

- the deformable components have been applied to the analysis of a car suspension model

Application: Car Suspension Model

- the deformable components have been applied to the analysis of a car suspension model
- the model consists in the full front and rear suspension system of a generic car (not representative of a specific vehicle)



Application: Car Suspension Model

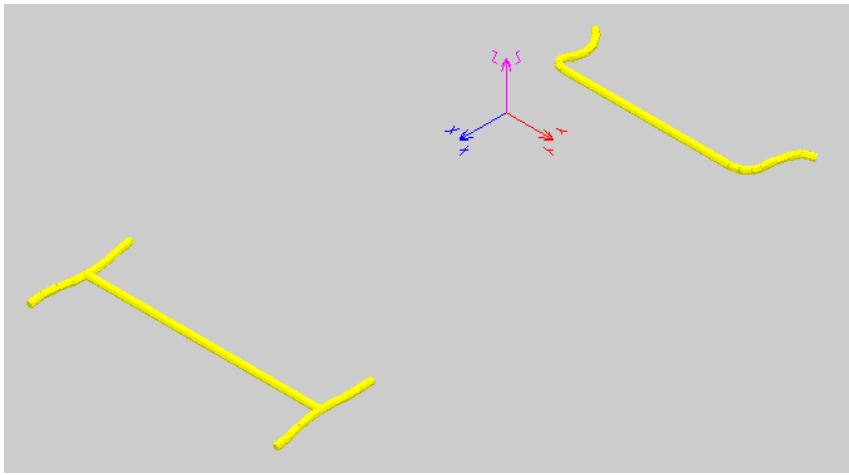
- the chassis is modeled as a rigid body

Application: Car Suspension Model

- the chassis is modeled as a rigid body
  (the use of a CMS model is foreseen for further validation)

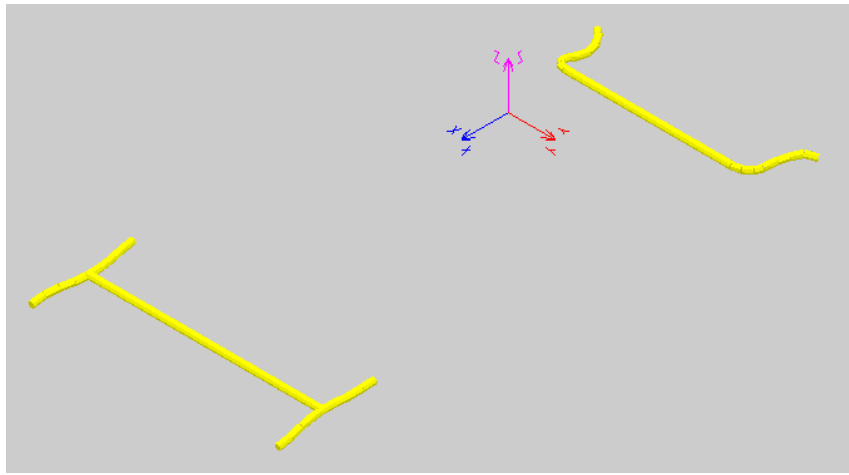
Application: Car Suspension Model

- the chassis is modeled as a rigid body
  (the use of a CMS model is foreseen for further validation)
- the front torsion bar and the rear twist beam are modeled by beam elements



Application: Car Suspension Model

- the chassis is modeled as a rigid body
  (the use of a CMS model is foreseen for further validation)
- the front torsion bar and the rear twist beam are modeled by beam elements



- the model consists in about 1300 equations, related to
  - about 100 structural nodes
  - about 60 nonlinear beam elements
  - more than 80 joints

Application: Car Suspension Model

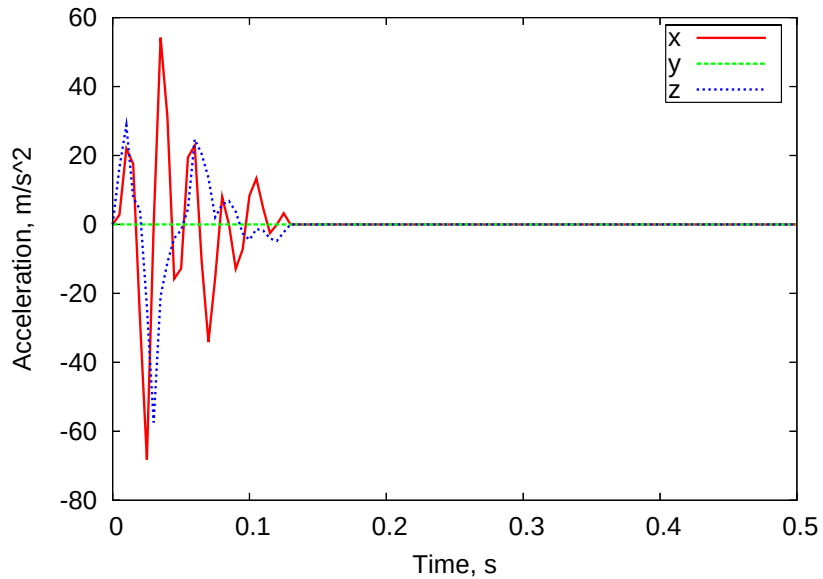
Typical analysis:

- consists in evaluating the loads in rubber components when the model is subjected to test rig excitation

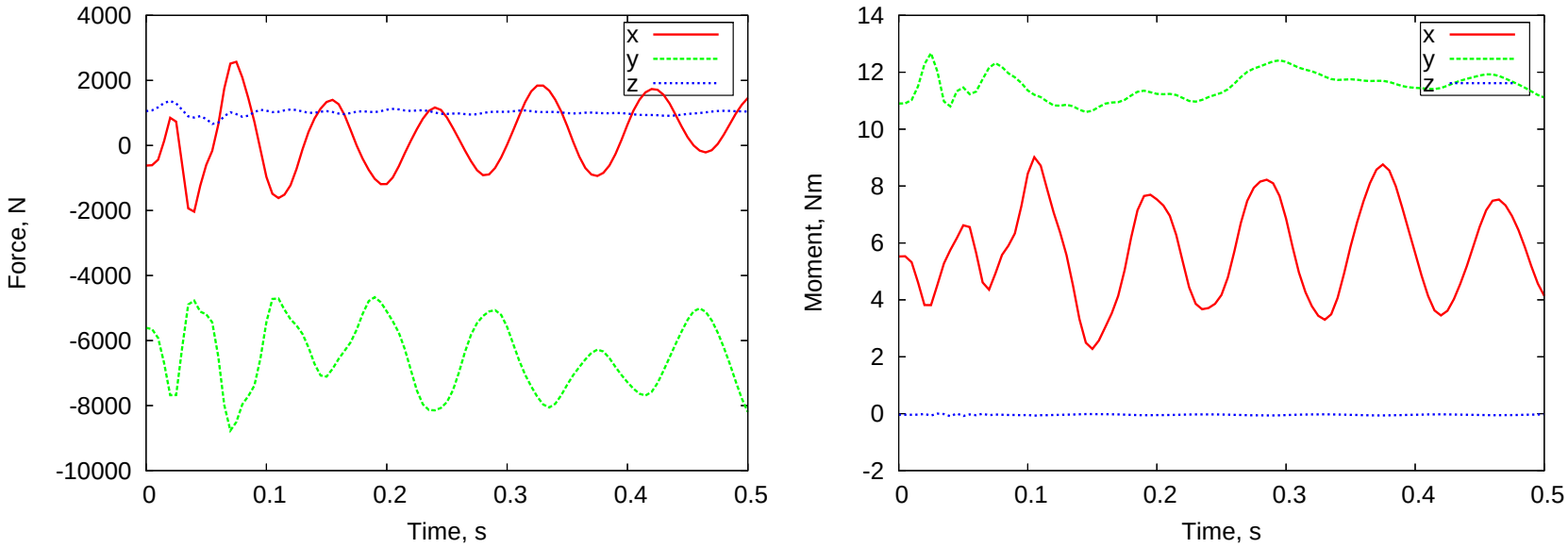
Application: Car Suspension Model

Typical analysis:

- consists in evaluating the loads in rubber components when the model is subjected to test rig excitation
- excitation pattern: acceleration imposed to front right axle

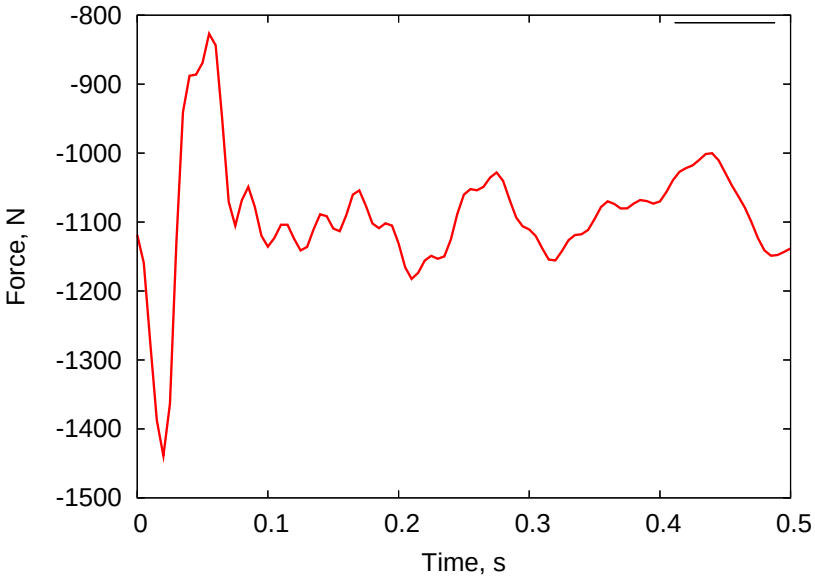


Application: Car Suspension Model

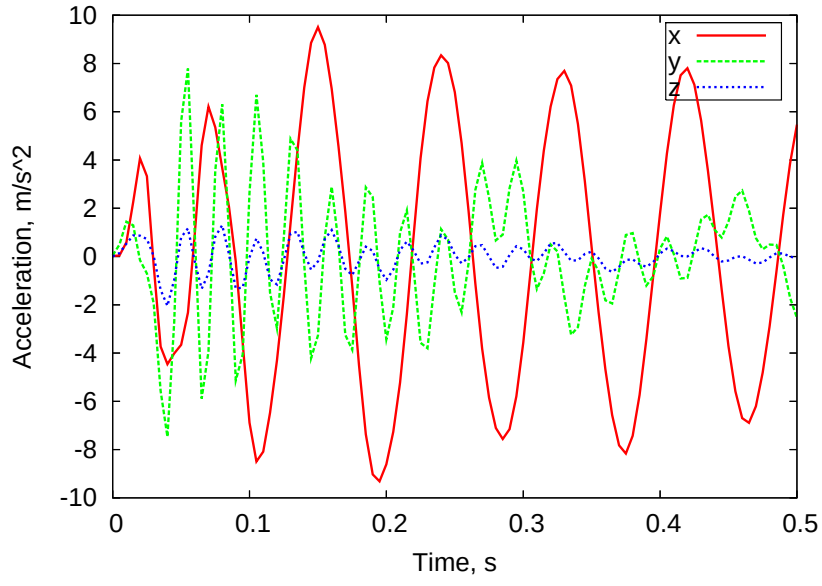


Force and moment in front right shock absorber top bushing

Application: Car Suspension Model



Force in front right shock absorber



CG acceleration

Outline

Conclusions & Acknowledgements

- The work illustrates versatile modeling of structural components for mechanics analysis of rubber components

Conclusions & Acknowledgements

- The work illustrates versatile modeling of structural components for mechanics analysis of rubber components
- Component behavior dependence on connectivity has been eliminated by invariant deformable components, without formulating connectivity-dependent constitutive properties

Conclusions & Acknowledgements

- The work illustrates versatile modeling of structural components for mechanics analysis of rubber components
- Component behavior dependence on connectivity has been eliminated by invariant deformable components, without formulating connectivity-dependent constitutive properties

- The features illustrated in this work will be distributed shortly, with the next release of the software (1.3.0)

Conclusions & Acknowledgements

- The work illustrates versatile modeling of structural components for mechanics analysis of rubber components
- Component behavior dependence on connectivity has been eliminated by invariant deformable components, without formulating connectivity-dependent constitutive properties
- The features illustrated in this work will be distributed shortly, with the next release of the software (1.3.0)
- MBDyn is considered by Hutchinson as a viable tool for industrial exploitation

Conclusions & Acknowledgements

- The work illustrates versatile modeling of structural components for mechanics analysis of rubber components
- Component behavior dependence on connectivity has been eliminated by invariant deformable components, without formulating connectivity-dependent constitutive properties

- The features illustrated in this work will be distributed shortly, with the next release of the software (1.3.0)

- MBDyn is considered by Hutchinson as a viable tool for industrial exploitation
- The authors acknowledge the support of Hutchinson SA R&D Centre and particularly of Dr. Daniel Benoualid in developing free multibody software.

ANALYSIS OF LOAD PATTERNS IN RUBBER COMPONENTS FOR VEHICLES

Jerome Merel, Israël Wander,
Pierangelo Masarati, Marco Morandini

Questions?