UNIVERSITY
OF TRENTO

Symbolic Computation Methods for the Numerical Solution of Dynamic Systems Described by Differential-Algebraic Equations

PhD thesis presentation by *Davide Stocco*
Supervisors: *Enrico Bertolazzi* & *Francesco Biral*

August 1, 2024

Outline

Why do we need simulation?

It is used to

- *predict* the behavior
- *design* and *optimize*
- accelerate development and testing
- risk-free testing environment
- *reduce* development costs

Simulations' complexity is increasing!

Higher **fidelity**, more **complexity**, more ...

- components
- interactions
- computational resources
- computation time

Why do we need simulation?

It is used to

- *predict* the behavior
- *design* and *optimize*
- accelerate development and testing
- risk-free testing environment
- *reduce* development costs

Simulations' complexity is increasing!

Higher **fidelity**, more **complexity**, more ...

- components
- interactions
- computational resources
- computation time

Why do we need simulation?

It is used to

- *predict* the behavior
- *design* and *optimize*
- accelerate development and testing
- risk-free testing environment
- *reduce* development costs

Simulations' complexity is increasing!

Higher **fidelity**, more **complexity**, more ...

- components
- interactions
- computational resources
- computation time

Introduction and Motivation

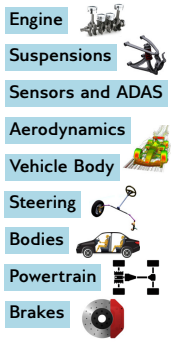
Dynamic Systems as a Collection of Subsystems

- Complex systems are divided into **subsystems**
  - mechanical
  - electrical
  - thermal
  - hydraulic
  - control
  - …
- Subsystems are modeled by **equations**
  - *algebraic*
  - ordinary *differential*
  - *differential-algebraic*
  - other types
- The **challenges** are
  - Speed
  - Accuracy
  - Efficiency

Introduction and Motivation

Dynamic Systems as a Collection of Subsystems

- Complex systems are divided into **subsystems**
  - mechanical
  - electrical
  - thermal
  - hydraulic
  - control
  - …
- Subsystems are modeled by **equations**
  - *algebraic*
  - ordinary *differential*
  - *differential-algebraic*
  - other types
- The **challenges** are
  - Speed
  - Accuracy
  - Efficiency



Introduction and Motivation

Dynamic Systems as a Collection of Subsystems

- Complex systems are divided into **subsystems**
  - mechanical
  - electrical
  - thermal
  - hydraulic
  - control
  - …
- Subsystems are modeled by **equations**
  - *algebraic*
  - ordinary *differential*
  - *differential-algebraic*
  - other types
- The **challenges** are

  Speed **Accuracy** **Efficiency**

The **solution** of these equations can be

- **Analytical**
  - deep understanding of the problem
  - rare and difficult to achieve
  - low computational resources
- **Numerical**
  - well-established and widely used
  - parametricity loss
  - high computational resources
- **Mixed analytical/numerical**
  - *ad hoc*, for specific cases
  - balance between the two approaches
  - reduced computational resources

The **solution** of these equations can be

- **Analytical**
  - deep understanding of the problem
  - rare and difficult to achieve
  - low computational resources
- **Numerical**
  - well-established and widely used
  - parametricity loss
  - high computational resources
- **Mixed analytical/numerical**
  - *ad hoc*, for specific cases
  - balance between the two approaches
  - reduced computational resources

Introduction and Motivation

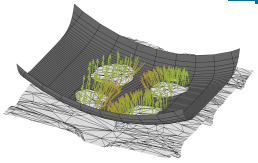
Solution Approaches

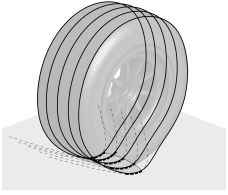
The **solution** of these equations can be

- **Analytical**
  - deep understanding of the problem
  - rare and difficult to achieve
  - low computational resources
- **Numerical**
  - well-established and widely used
  - parametricity loss
  - high computational resources
- **Mixed analytical/numerical**
  - *ad hoc*, for specific cases
  - balance between the two approaches
  - reduced computational resources

Tire-road interaction (A1-2)

$$\mathbf{P} = \frac{1}{V}\int_{\mathcal{V}} \text{proj}(\mathbf{x}, \partial\mathcal{G})\,\mathrm{d}\mathbf{x}$$
$$\hat{\mathbf{n}} = \int_{\mathcal{V}} \mathbf{d}(\mathbf{x}, \mathbf{h}_y)\,\mathrm{d}\mathbf{x}$$



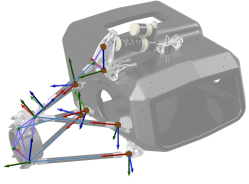


Tire model (A3)

$$\mathbf{F}(F_z, \sigma_x, \sigma_y, \varphi, \gamma, p, T, t)$$
$$\mathbf{M}(F_z, \sigma_x, \sigma_y, \varphi, \gamma, p, T, t)$$

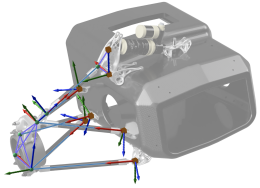
Suspension System (A4)

Macro-element FE model
Modified MB-DAE system



Introduction and Motivation

A Representative Example: FSAE Double Wishbone Suspension



Flexibility of components

Macro-element FE model

$$\begin{bmatrix} \mathbf{K}_{ff,72\times72}(\mathbb{R}) & \mathbf{K}_{fs,72\times66}(\mathbb{R}) \\ \mathbf{K}_{sf,66\times72}(\mathbb{R}) & \mathbf{K}_{ss,66\times66}(\mathbb{R}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{f,72}(\mathbb{R}) \\ \mathbf{d}_{s,66}(\mathbb{R}) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{f,72}(\mathbb{R}) \\ \mathbf{f}_{s,66}(\mathbb{R}) \end{bmatrix}$$

$$\xrightarrow[\text{factorization of } \mathbf{K}_{ff}]{\text{symbolic matrix}} \begin{aligned} \mathbf{d}_f &= \mathbf{K}_{ff}^{-1}\left(\mathbf{f}_f - \mathbf{K}_{fs}\mathbf{d}_s\right) \\ \mathbf{f}_s &= \mathbf{K}_{sf}\mathbf{d}_f + \mathbf{K}_{ss}\mathbf{d}_s \end{aligned}$$

Dynamics of the system

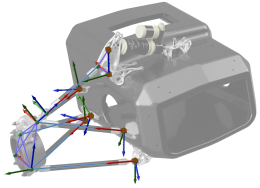
Modified MB-DAE system

$$\begin{cases} \mathbf{y} = \mathbf{x} + \mathbf{d} \\ \mathbf{M}(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{r}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{d}, \dot{\mathbf{d}}) = \mathbf{f}(t) \\ \boldsymbol{\Phi}_{\mathbf{x}}(\mathbf{x})^{\top}\boldsymbol{\lambda} = \mathbf{K}_c(\mathbf{x})\mathbf{d} + \mathbf{C}_c(\mathbf{x})\dot{\mathbf{d}} + \mathbf{b}(\mathbf{x}, \dot{\mathbf{x}}, t) \\ \boldsymbol{\Phi}(\mathbf{x}) = \mathbf{0} \end{cases}$$

This problem requires more knowledge and tools!

Introduction and Motivation

A Representative Example: FSAE Double Wishbone Suspension



Flexibility of components

Macro-element FE model

$$\begin{bmatrix} \mathbf{K}_{ff,72\times72}(\mathbb{R}) & \mathbf{K}_{fs,72\times66}(\mathbb{R}) \\ \mathbf{K}_{sf,66\times72}(\mathbb{R}) & \mathbf{K}_{ss,66\times66}(\mathbb{R}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{f,72}(\mathbb{R}) \\ \mathbf{d}_{s,66}(\mathbb{R}) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{f,72}(\mathbb{R}) \\ \mathbf{f}_{s,66}(\mathbb{R}) \end{bmatrix}$$

$$\xrightarrow[\text{factorization of } \mathbf{K}_{ff}]{\text{symbolic matrix}} \quad \begin{aligned} \mathbf{d}_f &= \mathbf{K}_{ff}^{-1}\left(\mathbf{f}_f - \mathbf{K}_{fs}\mathbf{d}_s\right) \\ \mathbf{f}_s &= \mathbf{K}_{sf}\mathbf{d}_f + \mathbf{K}_{ss}\mathbf{d}_s \end{aligned}$$

Dynamics of the system

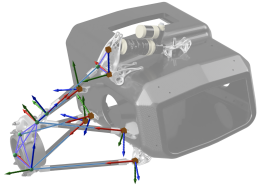
Modified MB-DAE system

$$\begin{cases} \mathbf{y} = \mathbf{x} + \mathbf{d} \\ \mathbf{M}(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{r}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{d}, \dot{\mathbf{d}}) = \mathbf{f}(t) \\ \mathbf{\Phi}_{\mathbf{x}}(\mathbf{x})^{\top}\boldsymbol{\lambda} = \mathbf{K}_c(\mathbf{x})\mathbf{d} + \mathbf{C}_c(\mathbf{x})\dot{\mathbf{d}} + \mathbf{b}(\mathbf{x}, \dot{\mathbf{x}}, t) \\ \mathbf{\Phi}(\mathbf{x}) = \mathbf{0} \end{cases}$$

This problem requires more knowledge and tools!

Introduction and Motivation

A Representative Example: FSAE Double Wishbone Suspension



Flexibility of components

Macro-element FE model

$$\begin{bmatrix} \mathbf{K}_{ff,72\times72}(\mathbb{R}) & \mathbf{K}_{fs,72\times66}(\mathbb{R}) \\ \mathbf{K}_{sf,66\times72}(\mathbb{R}) & \mathbf{K}_{ss,66\times66}(\mathbb{R}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{f,72}(\mathbb{R}) \\ \mathbf{d}_{s,66}(\mathbb{R}) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{f,72}(\mathbb{R}) \\ \mathbf{f}_{s,66}(\mathbb{R}) \end{bmatrix}$$

$$\xrightarrow[\text{factorization of } \mathbf{K}_{ff}]{\text{symbolic matrix}} \quad \begin{aligned} \mathbf{d}_f &= \mathbf{K}_{ff}^{-1}\left(\mathbf{f}_f - \mathbf{K}_{fs}\mathbf{d}_s\right) \\ \mathbf{f}_s &= \mathbf{K}_{sf}\mathbf{d}_f + \mathbf{K}_{ss}\mathbf{d}_s \end{aligned}$$

Dynamics of the system

Modified MB-DAE system

$$\begin{cases} \mathbf{y} = \mathbf{x} + \mathbf{d} \\ \mathbf{M}(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{r}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{d}, \dot{\mathbf{d}}) = \mathbf{f}(t) \\ \mathbf{\Phi}_{\mathbf{x}}(\mathbf{x})^{\top}\boldsymbol{\lambda} = \mathbf{K}_c(\mathbf{x})\mathbf{d} + \mathbf{C}_c(\mathbf{x})\dot{\mathbf{d}} + \mathbf{b}(\mathbf{x}, \dot{\mathbf{x}}, t) \\ \mathbf{\Phi}(\mathbf{x}) = \mathbf{0} \end{cases}$$

This problem requires more knowledge and tools!

Introduction and Motivation

Dynamic Systems Described by ODEs and DAEs

- Let us take a step back to generic **dynamic systems**

ODEs	DAEs
Easy to initialize	Hard to initialize
Strong theoretical foundation	Complex theoretical foundation
Efficient numerical methods	Require *ad hoc* numerical methods
Some models do not fit	State-of-the-art in modeling

- ODEs have **well-enstablished** theoretical foundation, DAEs do not
- Existing tools for DAEs are not as **mature** and **efficient** as ODEs tools
- From an engineering perspective, this is a **problem** (⏳$🏎️😡)

Therefore, the question is

How can we improve the state-of-the-art in the solution of DAEs?

Introduction and Motivation

Dynamic Systems Described by ODEs and DAEs

- Let us take a step back to generic **dynamic systems**

ODEs	DAEs
Easy to initialize	Hard to initialize
Strong theoretical foundation	Complex theoretical foundation
Efficient numerical methods	Require *ad hoc* numerical methods
Some models do not fit	State-of-the-art in modeling

- ODEs have **well-enstablished** theoretical foundation, DAEs do not
- Existing tools for DAEs are not as **mature** and **efficient** as ODEs tools
- From an engineering perspective, this is a **problem** (⏳$🏎️☹️)

Therefore, the question is

How can we improve the state-of-the-art in the solution of DAEs?

Outline

Differential-Algebraic Equations

A Brief Introduction

DAEs are

1. a **generalization** of algebraic equations and ODEs

$$\textbf{DAEs}: \quad \mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{0} \text{ with } \mathbf{F}_{\mathbf{x}'} \text{ singular}$$

$$\textbf{Algebraic}: \quad \mathbf{F}(\mathbf{x}, t) = \mathbf{0} \qquad \textbf{ODEs}: \quad \mathbf{x}' = \mathbf{f}(\mathbf{x}, t)$$

2. composed of
   - differential equations describe the system's **dynamics**
   - algebraic equations constrain the system to a **manifold**

ODE model

$$\begin{cases} \theta' = \omega \\ \omega' = -\dfrac{g}{\ell}\sin(\theta) \end{cases}$$

DAE model

$$\begin{cases} x' = u \\ y' = v \\ u' = -2x\lambda \\ v' = -2y\lambda - gm \\ 0 = x^2 + y^2 - \ell^2 \end{cases}$$

DAEs are

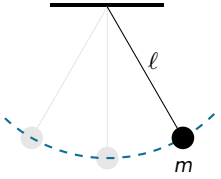
1. a **generalization** of algebraic equations and ODEs

DAEs
$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{0}$ with $\mathbf{F}_{\mathbf{x}'}$ singular

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

ODEs $\mathbf{x}' = \mathbf{f}(\mathbf{x}, t)$

2. composed of
   - differential equations describe the system's **dynamics**
   - algebraic equations constrain the system to a **manifold**



ODE model
$$\begin{cases} \theta' = \omega \\ \omega' = -\frac{g}{\ell}\sin(\theta) \end{cases}$$

DAE model
$$\begin{cases} x' = u \\ y' = v \\ u' = -2x\lambda \\ v' = -2y\lambda - gm \\ 0 = x^2 + y^2 - \ell^2 \end{cases}$$

Solution of Differential-Algebraic Equations

DAEs are not ODEs

Is the solution of DAEs a straightforward extension of ODEs?

- Specialized **numerical techniques** are required
- **Reformulation** of the DAE system is typically required

Two main approaches

↘

Index reduction

Direct discretization

L. Petzold. "Differential/Algebraic Equations are not ODE's". In: *SIAM Journal on Scientific and Statistical Computing* 3.3 (Sept. 1982), pp. 367–384. ISSN: 2168-3417

Solution of Differential-Algebraic Equations

DAEs are not ODEs

Is the solution of DAEs a straightforward extension of ODEs?

- Specialized **numerical techniques** are required
- **Reformulation** of the DAE system is typically required

Two main approaches

Index reduction + ODE methods

Direct discretization

L. Petzold. "Differential/Algebraic Equations are not ODE's". In: *SIAM Journal on Scientific and Statistical Computing* 3.3 (Sept. 1982), pp. 367–384. ISSN: 2168-3417

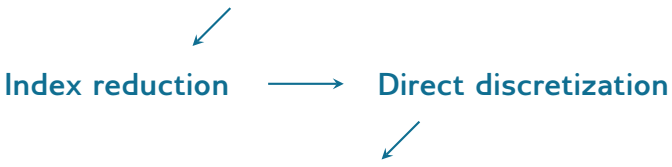
Solution of Differential-Algebraic Equations

DAEs are not ODEs

Is the solution of DAEs a straightforward extension of ODEs?

- Specialized **numerical techniques** are required
- **Reformulation** of the DAE system is typically required

Two main approaches



L. Petzold. "Differential/Algebraic Equations are not ODE's". In: *SIAM Journal on Scientific and Statistical Computing* 3.3 (Sept. 1982), pp. 367–384. ISSN: 2168-3417

- Approximate $\mathbf{x}'$ through a **discretization formula**
  - finite difference scheme
  - polynomial quadrature

Implicit Runge-Kutta

$$\mathbf{F}\left(\mathbf{x}_k + h_k \sum_{j=1}^{s} a_{ij}\mathbf{K}_j, \mathbf{K}_i, t_k + c_i h_k\right) = \mathbf{0}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \sum_{i=1}^{s} b_i \mathbf{K}_i$$

for $i = 1, \dots, s$, $k = 0, 1, \dots, m$

(a_{ij}) non-singular + h_k small enough $\rightarrow$ we can solve for $\mathbf{K}_i$ at each m-th step

⚠ Some **order of convergence** can be lost on some components depending on the **index**

C. Gear. "Simultaneous Numerical Solution of DAEs". In: *IEEE Transactions on Circuit Theory* 18.1 (1971), pp. 89–95. ISSN: 0018-9324

Solution of Differential-Algebraic Equations

Direct Discretization

- Approximate $\mathbf{x}'$ through a **discretization formula**
  - finite difference scheme
  - polynomial quadrature

Implicit Runge-Kutta

$$\mathbf{F}\left(\mathbf{x}_k + h_k \sum_{j=1}^{s} a_{ij}\mathbf{K}_j, \mathbf{K}_i, t_k + c_i h_k\right) = \mathbf{0}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k \sum_{i=1}^{s} b_i \mathbf{K}_i$$

for $i = 1, \dots, s$, $k = 0, 1, \dots, m$

(a_{ij}) non-singular + h_k small enough $\rightarrow$ we can solve for $\mathbf{K}_i$ at each m-th step

⚠ Some **order of convergence** can be lost on some components depending on the **index**

C. Gear. "Simultaneous Numerical Solution of DAEs". In: *IEEE Transactions on Circuit Theory* 18.1 (1971), pp. 89–95. ISSN: 0018-9324

Solution of Differential-Algebraic Equations

The Index . . . Or better Yet, The Indices

The index is a "**measure of difficulty**" in the solution of the DAEs

- Different approaches in classifying such difficulties led to different **index concepts**

  differential structural tractability
  geometrical perturbation strangeness

- Some indices **coincide** under proper regularity conditions

The Differential Index

It is the minimum number of differentiations required to transform a DAE system into its underlying ODE system.

V. Mehrmann. "Index Concepts for DAEs". In: *Encyclopedia of Applied and Computational Mathematics*. Springer, 2015, pp. 676–681. isbn: 978-3540705291

The index is a "**measure of difficulty**" in the solution of the DAEs

- Different approaches in classifying such difficulties led to different **index concepts**

  differential structural tractability
  geometrical perturbation strangeness

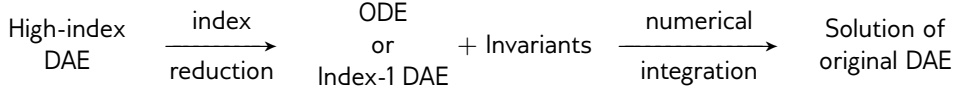
- Some indices **coincide** under proper regularity conditions

The Differential Index

It is the minimum number of differentiations required to transform a DAE system into its underlying ODE system.

V. Mehrmann. "Index Concepts for DAEs". In: *Encyclopedia of Applied and Computational Mathematics*. Springer, 2015, pp. 676–681. isbn: 978-3540705291

Index Reduction in the Solution Process



Index reduction is performed

- by expressions **manipulation** and **differentiation**
- through **symbolic computation** or **numerically** (with *automatic differentiation*)

No simple recipe exists!

Structural methods Projector-based analysis

How do we reduce the index?

Exploit an idea common to many index concepts!

1. **Separate** the differential and algebraic equations
2. **Differentiate** the algebraic equations
3. **Repeat** 1–2 until necessary (index-1 DAE or ODE is reached)

Easy to say, not so easy to do ...

Numerically	Symbolically
Numerical linear algebra	Symbolic linear algebra
Automatic differentiation	Symbolic differentiation
Track of the separators chain	Just manipulate the equations

V. Mehrmann. "Index Concepts for DAEs". In: *Encyclopedia of Applied and Computational Mathematics*. Springer, 2015, pp. 676–681. isbn: 978-3540705291

How do we reduce the index?

Exploit an idea common to many index concepts!

1. **Separate** the differential and algebraic equations
2. **Differentiate** the algebraic equations
3. **Repeat** 1–2 until necessary (index-1 DAE or ODE is reached)

Easy to say, not so easy to do ...

Numerically	Symbolically
Numerical linear algebra	Symbolic linear algebra
Automatic differentiation	Symbolic differentiation
Track of the separators chain	Just manipulate the equations

V. Mehrmann. "Index Concepts for DAEs". In: *Encyclopedia of Applied and Computational Mathematics*. Springer, 2015, pp. 676–681. isbn: 978-3540705291

How can we improve the state-of-the-art?

Exploit the **equation separation** concept in a **full-symbolic** framework!

- ? **No** a priori knowledge of the system structure
- **Basic** use of symbolic computation and linear algebra

Differentiation

$$\frac{\mathrm{d}}{\mathrm{d}x}$$

Factorization

LU
Fraction-Free LU

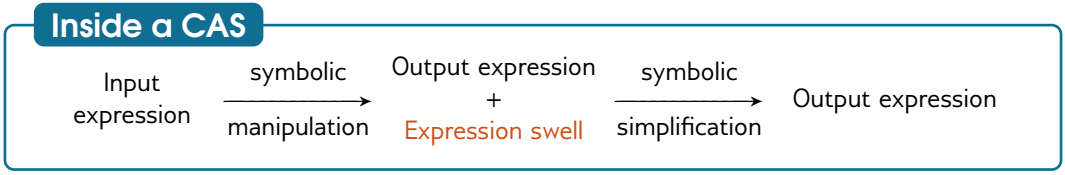
The **software tools**

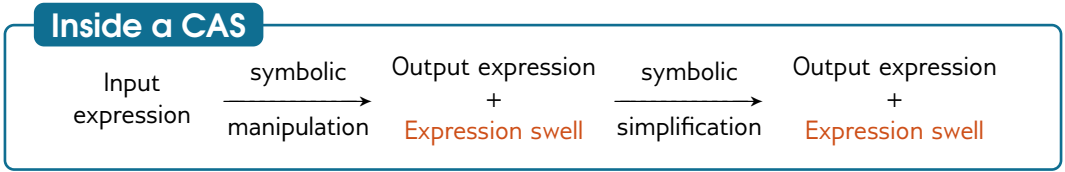
- **Maple®** for symbolic manipulation
- **Matlab®** for numerical evaluation

Outline

Symbolic Computation Essentials

Expression Swell





CAS are sensitive to expression swell!

High memory usage Slow computation

Two types of expression swell

Inherent

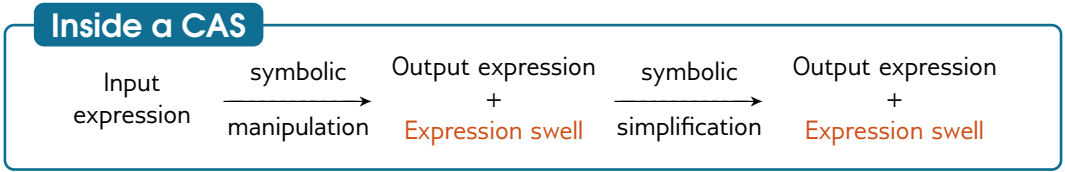
No applicable simplification **rule**

Require changes in the problem **formulation**

Intermediate

Depends on CAS's simplification **capabilities**

Intermediate growth, **en route** to a simple result



CAS are sensitive to expression swell!

High memory usage Slow computation

Two types of expression swell

Inherent

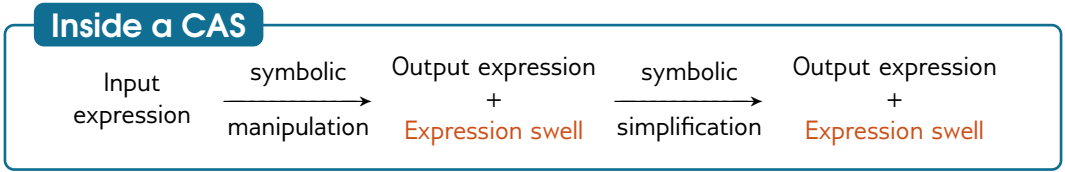
No applicable simplification **rule**

Require changes in the problem **formulation**

Intermediate

Depends on CAS's simplification **capabilities**

Intermediate growth, **en route** to a simple result



CAS are sensitive to expression swell!

High memory usage Slow computation

Two types of expression swell

Inherent

No applicable simplification **rule**

Require changes in the problem **formulation**

Intermediate

Depends on CAS's simplification **capabilities**

Intermediate growth, **en route** to a simple result

Large Expression Management

Hierarchical Representation

We progressively **hide** chuncks of expressions

$$2xy(1+y) \cdot f(x,y) + 5z(3a+z) \cdot g(y) + 3c(xy+z) \cdot h(x,z)$$
$$v_1 \cdot f(x,y) + v_2 \cdot g(y) + v_3 \cdot h(x,z)$$
$$v_4 + v_5$$

to store them in separate variables

$$v_1 = 2x(1+y)$$
$$v_2 = 5z(3a+z)$$
$$v_3 = 3c(xy+z)$$
$$v_4 = v_1 \cdot f(x,y)$$
$$v_5 = v_2 \cdot g(y) + v_3 \cdot h(x,z)$$

Large Expression Management

Hierarchical Representation

We progressively **hide** chuncks of expressions

$$\underline{2xy(1+y)} \cdot f(x,y) + \underline{5z(3a+z)} \cdot g(y) + \underline{3c(xy+z)} \cdot h(x,z)$$
$$\underline{v_1 \cdot f(x,y)} + \underline{v_2 \cdot g(y) + v_3 \cdot h(x,z)}$$
$$v_4 + v_5$$

to store them in separate variables

$$v_1 = 2x(1+y)$$
$$v_2 = 5z(3a+z)$$
$$v_3 = 3c(xy+z)$$
$$v_4 = v_1 \cdot f(x,y)$$
$$v_5 = v_2 \cdot g(y) + v_3 \cdot h(x,z)$$

We progressively **hide** chuncks of expressions

$$\underline{2xy(1+y)} \cdot f(x,y) + \underline{5z(3a+z)} \cdot g(y) + \underline{3c(xy+z)} \cdot h(x,z)$$
$$\underline{v_1 \cdot f(x,y)} + \underline{v_2 \cdot g(y) + v_3 \cdot h(x,z)}$$
$$v_4 + v_5$$

to store them in separate variables

$$\begin{aligned} v_1 &= 2x(1+y) \\ v_2 &= 5z(3a+z) \\ v_3 &= 3c(xy+z) \\ v_4 &= v_1 \cdot f(x,y) \\ v_5 &= v_2 \cdot g(y) + v_3 \cdot h(x,z) \end{aligned}$$

Hierarchical representation of expressions is performed through **veiling variables**

$$\mathbf{f}(\mathbf{x}) \xrightarrow[\text{variables}]{\text{veiling}} \mathbf{f}(\mathbf{x}, \mathbf{v}) \qquad \text{where} \qquad \mathbf{v}(\mathbf{x}) = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(v_1, \mathbf{x}) \\ \vdots \\ v_n(v_1, \dots, v_{n-1}, \mathbf{x}) \end{bmatrix}$$

LEM is made of two actions

1. **veiled** large expressions
2. **unveil** to recover the original form

Expression complexity remains but the CAS can not see it! 🙈

Hierarchical representation of expressions is performed through **veiling variables**

$$\mathbf{f}(\mathbf{x}) \xrightarrow[\text{variables}]{\text{veiling}} \mathbf{f}(\mathbf{x}, \mathbf{v}) \qquad \text{where} \qquad \mathbf{v}(\mathbf{x}) = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(v_1, \mathbf{x}) \\ \vdots \\ v_n(v_1, \dots, v_{n-1}, \mathbf{x}) \end{bmatrix}$$

LEM is made of two actions

1. **veiled** large expressions
2. **unveil** to recover the original form

Expression complexity remains but the CAS can not see it! 🙈

Hierarchical representation of expressions is performed through **veiling variables**

$$\mathbf{f}(\mathbf{x}) \xrightarrow[\text{variables}]{\text{veiling}} \mathbf{f}(\mathbf{x}, \mathbf{v}) \qquad \text{where} \qquad \mathbf{v}(\mathbf{x}) = \begin{bmatrix} v_1(\mathbf{x}) \\ v_2(v_1, \mathbf{x}) \\ \vdots \\ v_n(v_1, \dots, v_{n-1}, \mathbf{x}) \end{bmatrix}$$

LEM is made of two actions

1. **veiled** large expressions
2. **unveil** to recover the original form

Expression complexity remains but the CAS can not see it! 🙈

Easy to implement, but ...

How to measure symbolic expression complexity?

- **Length in characters**
  Same expression can have different lengths
- **Directed acyclic graph**
  Measures the number of nodes and edges
- **Computational cost**
  Insensible to internal representation

What is the right level of expression complexity?

- **Trial and error**
  No general rule, depends on the software
  The least to make computations go on

Easy to implement, but ...

How to measure symbolic expression complexity?

- **Length in characters**
  Same expression can have different lengths
- **Directed acyclic graph**
  Measures the number of nodes and edges
- **Computational cost**
  Insensible to internal representation

What is the right level of expression complexity?

- **Trial and error**
  No general rule, depends on the software
  The least to make computations go on

Maple® factorizations are sensitive to expression swell

- LU and **Fraction-Free LU (FFLU)** factorizations
  - preserve **sparsity** and **fill-in**
  - perform **minimal algebraic manipulations**
  - allow for custom **pivoting strategies**
- However ...
  1. output's numerical stability is not guaranteed
     How to ensure numerical stability?
  2. expressions grow unpredictably during manipulation
     How to manage large symbolic expressions?

Maple® factorizations are sensitive to expression swell

- **LU** and **Fraction-Free LU (FFLU)** factorizations
  - preserve **sparsity** and **fill-in**
  - perform **minimal algebraic manipulations**
  - allow for custom **pivoting strategies**
- However ...
  1. output's numerical stability is not guaranteed
     - **How to ensure numerical stability?**
  2. expressions grow unpredictably during manipulation
     - **How to manage large symbolic expressions?**

Maple® factorizations are sensitive to expression swell

- **LU** and **Fraction-Free LU (FFLU)** factorizations
  - preserve **sparsity** and **fill-in**
  - perform **minimal algebraic manipulations**
  - allow for custom **pivoting strategies**
- However ...
  1. output's numerical stability is not guaranteed
     How to ensure numerical stability?
  2. expressions grow unpredictably during manipulation
     How to manage large symbolic expressions?

How to ensure numerical stability?

A good symbolic **pivoting strategy**

1. selects the **minimum-degree** entry
   Limit fill-in and expression swell
2. if numeric, selects the **biggest** entry
   Improve numerical stability
3. selects **least complicated** entry
   Limit expression swell
4. looks for the **best choice**
   Full-pivoting strategy

```
procedure SymbolicPivoting(A, k)
  d^r, d^c ← ComputeDegrees(A)
  for i from k to m do
    for j from k to n do
      D_ij ← ∞
      if A_ij ≠ 0 then D_ij ← d_i^r max(0, d_j^c − 1) + d_j^c max(0, d_i^r − 1)
    end for
  end for
  P ← Sort(D)
  q, l ← 0, 0 and p, p_c, p_n ← ∞, ∞, ∞
  for all (i, j) in P do
    if p_c ≠ ∞ and D_ij > D_ql then break
    t ← A_ij
    if Signature(t) = 0 then continue
    t ← Simplify(t) and t_c ← ExpressionComplexity(t) and t_n ← ∞
    if t is numeric then t_n ← max(1, abs(t))
    if t_c < p_c or (t_c = p_c and t_n > p_n) then
      q, l ← i, j and p, p_c, p_n ← t, t_c, t_n
    end if
  end for
  return p, q, l
end procedure
```

How to ensure numerical stability?

A good symbolic **pivoting strategy**

1. selects the **minimum-degree** entry
   Limit fill-in and expression swell
2. if numeric, selects the **biggest** entry
   Improve numerical stability
3. selects **least complicated** entry
   Limit expression swell
4. looks for the **best choice**
   Full-pivoting strategy

procedure SymbolicPivoting(**A**, k)
  $\mathbf{d}^r$, $\mathbf{d}^c \leftarrow$ ComputeDegrees(**A**)
  for i **from** k **to** m **do**
    for j **from** k **to** n **do**
      $D_{ij} \leftarrow \infty$
      if $A_{ij} \neq 0$ **then** $D_{ij} \leftarrow d_i^r \max(0,\ d_j^c - 1) + d_j^c \max(0,\ d_i^r - 1)$
    end for
  end for
  $\mathcal{P} \leftarrow$ Sort(**D**)
  q, $l \leftarrow$ 0, 0 and p, p_c, $p_n \leftarrow \infty$, ∞, ∞
  for all (i, j) **in** $\mathcal{P}$ **do**
    if $p_c \neq \infty$ **and** $D_{ij} > D_{ql}$ **then break**
    $t \leftarrow A_{ij}$
    if Signature(t) $= 0$ **then continue**
    $t \leftarrow$ Simplify(t) and $t_c \leftarrow$ ExpressionComplexity(t) and $t_n \leftarrow \infty$
    if t is numeric **then** $t_n \leftarrow \max(1, \text{abs}(t))$
    if $t_c < p_c$ **or** ($t_c = p_c$ **and** $t_n > p_n$) **then**
      q, $l \leftarrow i$, j and p, p_c, $p_n \leftarrow t$, t_c, t_n
    end if
  end for
  return p, q, l
end procedure

How to ensure numerical stability?

A good symbolic **pivoting strategy**

1. selects the **minimum-degree** entry
   Limit fill-in and expression swell
2. if numeric, selects the **biggest** entry
   Improve numerical stability
3. selects **least complicated** entry
   Limit expression swell
4. looks for the **best choice**
   Full-pivoting strategy

```
procedure SymbolicPivoting(A, k)
  d^r, d^c ← ComputeDegrees(A)
  for i from k to m do
    for j from k to n do
      D_ij ← ∞
      if A_ij ≠ 0 then D_ij ← d_i^r max(0, d_j^c − 1) + d_j^c max(0, d_i^r − 1)
    end for
  end for
  P ← Sort(D)
  q, l ← 0, 0 and p, p_c, p_n ← ∞, ∞, ∞
  for all (i, j) in P do
    if p_c ≠ ∞ and D_ij > D_ql then break
    t ← A_ij
    if Signature(t) = 0 then continue
    t ← Simplify(t) and t_c ← ExpressionComplexity(t) and t_n ← ∞
    if t is numeric then t_n ← max(1, abs(t))
    if t_c < p_c or (t_c = p_c and t_n > p_n) then
      q, l ← i, j and p, p_c, p_n ← t, t_c, t_n
    end if
  end for
  return p, q, l
end procedure
```

How to ensure numerical stability?

A good symbolic **pivoting strategy**

1. selects the **minimum-degree** entry
   Limit fill-in and expression swell
2. if numeric, selects the **biggest** entry
   Improve numerical stability
3. selects **least complicated** entry
   Limit expression swell
4. looks for the **best choice**
   Full-pivoting strategy

```
procedure SymbolicPivoting(A, k)
  d^r, d^c ← ComputeDegrees(A)
  for i from k to m do
    for j from k to n do
      D_ij ← ∞
      if A_ij ≠ 0 then D_ij ← d_i^r max(0, d_j^c − 1) + d_j^c max(0, d_i^r − 1)
    end for
  end for
  P ← Sort(D)
  q, l ← 0, 0 and p, p_c, p_n ← ∞, ∞, ∞
  for all (i, j) in P do
    if p_c ≠ ∞ and D_ij > D_ql then break
    t ← A_ij
    if Signature(t) = 0 then continue
    t ← Simplify(t) and t_c ← ExpressionComplexity(t) and t_n ← ∞
    if t is numeric then t_n ← max(1, abs(t))
    if t_c < p_c or (t_c = p_c and t_n > p_n) then
      q, l ← i, j and p, p_c, p_n ← t, t_c, t_n
    end if
  end for
  return p, q, l
end procedure
```

How to manage large symbolic expressions?

Veils are introduced during ...

- **Factorization step**
  - Gaussian elimination
  - Schur complement
- **Solution step**
  - forward substitution
  - backward substitution

```
procedure BuildLU(A, k)
  M ← A
  rnk ← min(m, n)
  for k from 1 to rnk do
    p, q, l ← SymbolicPivoting(M, k)
    if p = 0 then
      rnk ← k − 1 and break
    end if
    r_k, c_k ← q, l
    M ← SwapRowsCols(M, k, q, l)
    for i from k + 1 to m do
      M_kk ← Veil(M_kk)
      M_ik ← Veil(Normalizer(M_ik / M_kk))
      for j from k + 1 to n do
        M_ij ← Veil(Normalizer(M_ij − M_ik M_kj))
      end for
    end for
  end for
  P, Q ← PermutationMatrices(r, c)
  L, U ← LowerUpperMatrices(M)
  return L, U, P, Q, r, c, rnk
end procedure
```

```
procedure SolveLU(L, U, P, Q, b)
  y ← Pb
  m, n ← Size(L)
  for i from 2 to m do
    y_i ← Veil(y_i − Σ_{j=1}^{i−1} L_ij y_j)
  end for
  x_n ← Veil(y_n / U_nn)
  for i from n − 1 to 1 do
    x_i ← Veil(y_i − Σ_{j=i+1}^{n} U_ij x_j)
    x_i ← Veil(x_i / U_ii)
  end for
  x ← Q^⊤ x
end procedure
```

How to manage large symbolic expressions?

Veils are introduced during ...

- Factorization step
  - Gaussian elimination
  - Schur complement
- **Solution step**
  - forward substitution
  - backward substitution

```
procedure BuildLU(A, k)
  M ← A
  rnk ← min(m, n)
  for k from 1 to rnk do
    p, q, l ← SymbolicPivoting(M, k)
    if p = 0 then
      rnk ← k − 1 and break
    end if
    r_k, c_k ← q, l
    M ← SwapRowsCols(M, k, q, l)
    for i from k + 1 to m do
      M_kk ← Veil(M_kk)
      M_ik ← Veil(Normalizer(M_ik / M_kk))
      for j from k + 1 to n do
        M_ij ← Veil(Normalizer(M_ij − M_ik M_kj))
      end for
    end for
  end for
  P, Q ← PermutationMatrices(r, c)
  L, U ← LowerUpperMatrices(M)
  return L, U, P, Q, r, c, rnk
end procedure
```

```
procedure SolveLU(L, U, P, Q, b)
  y ← Pb
  m, n ← Size(L)
  for i from 2 to m do
    y_i ← Veil(y_i − Σ_{j=1}^{i−1} L_ij y_j)
  end for
  x_n ← Veil(y_n / U_nn)
  for i from n − 1 to 1 do
    x_i ← Veil(y_i − Σ_{j=i+1}^{n} U_ij x_j)
    x_i ← Veil(x_i / U_ii)
  end for
  x ← Q^⊤ x
end procedure
```

Outline

Index Reduction Algorithm

Separation of Differential and Algebraic Equations

1. Consider the **generic** DAEs system

$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

2. **Separate** the equations with the cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{0} \end{bmatrix} \mathbf{x}' = \begin{bmatrix} \mathbf{g}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) \end{bmatrix} \quad \text{with} \quad \begin{aligned} \mathbf{E}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{A}(\mathbf{x}, t) \\ \mathbf{g}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) &= \mathbf{K}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \end{aligned}$$

Cokernel Computation

The cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$ are calculated using matrix factorization

Index Reduction Algorithm

Separation of Differential and Algebraic Equations

1. Consider the **generic** DAEs system

$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

2. **Separate** the equations with the cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{0} \end{bmatrix} \mathbf{x}' = \begin{bmatrix} \mathbf{g}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) \end{bmatrix} \quad \text{with} \quad \begin{aligned} \mathbf{E}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{A}(\mathbf{x}, t) \\ \mathbf{g}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) &= \mathbf{K}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \end{aligned}$$

Cokernel Computation

The cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$ are calculated using matrix factorization

Index Reduction Algorithm

Separation of Differential and Algebraic Equations

1. Consider the **generic** DAEs system

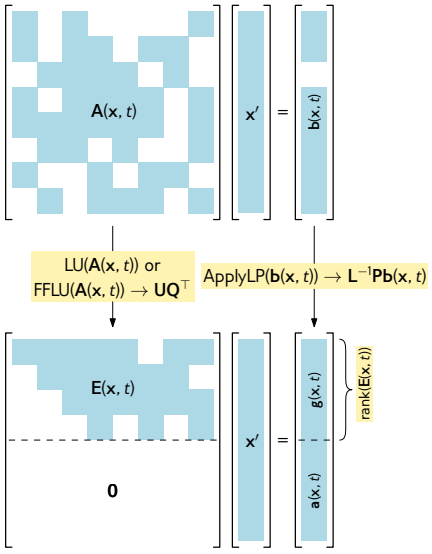
$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

2. **Separate** the equations with the cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{0} \end{bmatrix} \mathbf{x}' = \begin{bmatrix} \mathbf{g}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) \end{bmatrix} \quad \text{with} \quad \begin{aligned} \mathbf{E}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{A}(\mathbf{x}, t) \\ \mathbf{g}(\mathbf{x}, t) &= \mathbf{N}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \\ \mathbf{a}(\mathbf{x}, t) &= \mathbf{K}(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t) \end{aligned}$$

Cokernel Computation

The cokernel $\mathbf{K}(\mathbf{x}, t)$ and its orthogonal complement $\mathbf{N}(\mathbf{x}, t)$ of $\mathbf{A}(\mathbf{x}, t)$ are calculated using matrix factorization



Index Reduction Algorithm

Differentiation of Algebraic Equations

3. Update the **invariants** as $\mathbf{h}(\mathbf{x}, t) = \mathbf{h}(\mathbf{x}, t) \cup \mathbf{a}(\mathbf{x}, t)$
4. **Differentiate** the algebraic equations $\mathbf{a}(\mathbf{x}, t)$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{a}(\mathbf{x}, t) = \mathbf{E}_\mathbf{a}(\mathbf{x}, t)\mathbf{x}' - \mathbf{g}_\mathbf{a}(\mathbf{x}, t)$$

5. The **index reduced** system of DAEs takes the form

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{E}_\mathbf{a}(\mathbf{x}, t) \end{bmatrix} \mathbf{x}' = \begin{bmatrix} g(\mathbf{x}, t) \\ \mathbf{g}_\mathbf{a}(\mathbf{x}, t) \end{bmatrix}$$
$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

A Sequential Algorithm

Apply 1–6 repeatedly until $\mathbf{A}(\mathbf{x}, t)$ is non-singular

Index Reduction Algorithm

Differentiation of Algebraic Equations

3. Update the **invariants** as $\mathbf{h}(\mathbf{x}, t) = \mathbf{h}(\mathbf{x}, t) \cup \mathbf{a}(\mathbf{x}, t)$
4. **Differentiate** the algebraic equations $\mathbf{a}(\mathbf{x}, t)$

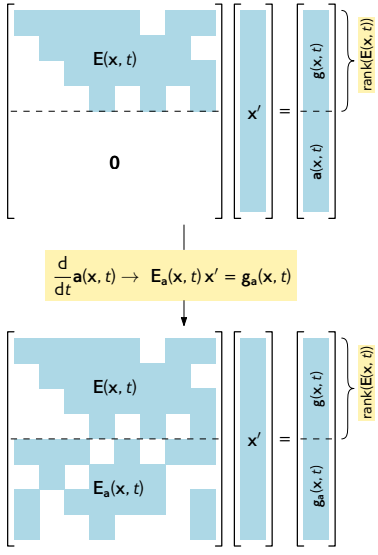
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{a}(\mathbf{x}, t) = \mathbf{E}_\mathbf{a}(\mathbf{x}, t)\mathbf{x}' - \mathbf{g}_\mathbf{a}(\mathbf{x}, t)$$

5. The **index reduced** system of DAEs takes the form

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{E}_\mathbf{a}(\mathbf{x}, t) \end{bmatrix} \mathbf{x}' = \begin{bmatrix} g(\mathbf{x}, t) \\ g_\mathbf{a}(\mathbf{x}, t) \end{bmatrix}$$
$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

A Sequential Algorithm

Apply 1–6 repeatedly until $\mathbf{A}(\mathbf{x}, t)$ is non-singular



Index Reduction Algorithm

Differentiation of Algebraic Equations

3. Update the **invariants** as $\mathbf{h}(\mathbf{x}, t) = \mathbf{h}(\mathbf{x}, t) \cup \mathbf{a}(\mathbf{x}, t)$
4. **Differentiate** the algebraic equations $\mathbf{a}(\mathbf{x}, t)$

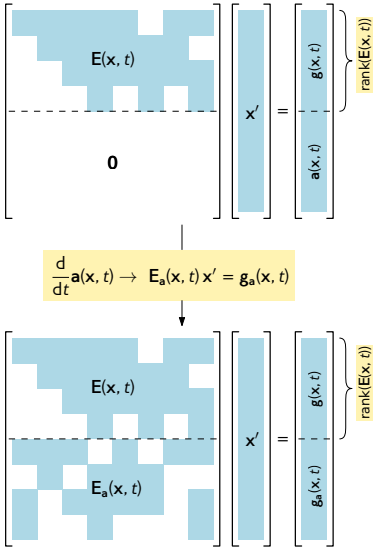
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{a}(\mathbf{x}, t) = \mathbf{E}_\mathbf{a}(\mathbf{x}, t)\mathbf{x}' - \mathbf{g}_\mathbf{a}(\mathbf{x}, t)$$

5. The **index reduced** system of DAEs takes the form

$$\begin{bmatrix} \mathbf{E}(\mathbf{x}, t) \\ \mathbf{E}_\mathbf{a}(\mathbf{x}, t) \end{bmatrix} \mathbf{x}' = \begin{bmatrix} \mathbf{g}(\mathbf{x}, t) \\ \mathbf{g}_\mathbf{a}(\mathbf{x}, t) \end{bmatrix}$$
$$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = \mathbf{A}(\mathbf{x}, t)\mathbf{x}' - \mathbf{b}(\mathbf{x}, t) = \mathbf{0}$$

A Sequential Algorithm

Apply 1–6 repeatedly until $\mathbf{A}(\mathbf{x}, t)$ is non-singular

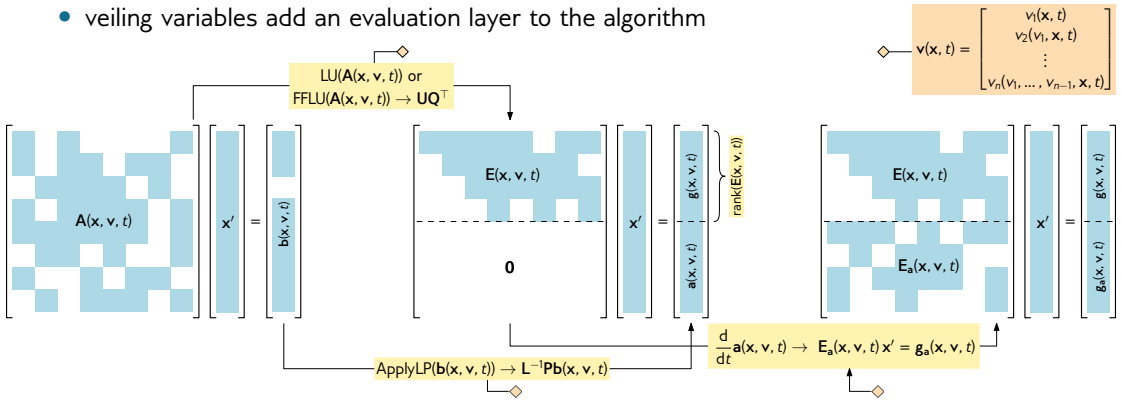


Index Reduction Algorithm

Including Veiling Variables

The algorithm can be extended to include **LEM**

- veiling variables are stored in the list $\mathbf{v}(\mathbf{x}, t)$
- equations are also function of $\mathbf{v}(\mathbf{x}, t)$
- veiling variables add an evaluation layer to the algorithm



Now we can code-generate the reduced DAE system!

- **Differential part**

$$\begin{aligned} \mathbf{F}(\mathbf{x}, \mathbf{x}', \mathbf{v}, t) &= \mathbf{0} && \text{implicit system class} \\ \mathbf{A}(\mathbf{x}, \mathbf{v}, t)\mathbf{x}' &= \mathbf{b}(\mathbf{x}, \mathbf{v}, t) && \text{semi-explicit system class} \\ \mathbf{x}' &= \mathbf{f}(\mathbf{x}, \mathbf{v}, t) && \text{explicit system class} \end{aligned}$$

- Invariants

$$\mathbf{h}(\mathbf{x}, \mathbf{v}, t) = \begin{bmatrix} \mathbf{h}_i(\mathbf{x}, \mathbf{v}, t) \\ \mathbf{h}_u(\mathbf{x}, \mathbf{v}, t) \end{bmatrix} = \mathbf{0}$$

hidden constraints
optional user-defined invariants

- Veiling variables

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} v_1(\mathbf{x}, t) \\ v_2(v_1, \mathbf{x}, t) \\ \vdots \\ v_n(v_1, \ldots, v_{n-1}, \mathbf{x}, t) \end{bmatrix}$$

Now we can code-generate the reduced DAE system!

- **Differential part**

$$\begin{aligned} \mathbf{F}(\mathbf{x}, \mathbf{x}', \mathbf{v}, t) &= \mathbf{0} && \text{implicit system class} \\ \mathbf{A}(\mathbf{x}, \mathbf{v}, t)\mathbf{x}' &= \mathbf{b}(\mathbf{x}, \mathbf{v}, t) && \text{semi-explicit system class} \\ \mathbf{x}' &= \mathbf{f}(\mathbf{x}, \mathbf{v}, t) && \text{explicit system class} \end{aligned}$$

- **Invariants**

$$\mathbf{h}(\mathbf{x}, \mathbf{v}, t) = \begin{bmatrix} \mathbf{h}_i(\mathbf{x}, \mathbf{v}, t) \\ \mathbf{h}_u(\mathbf{x}, \mathbf{v}, t) \end{bmatrix} = \mathbf{0}$$

hidden constraints
optional user-defined invariants

- Veiling variables

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} v_1(\mathbf{x}, t) \\ v_2(v_1, \mathbf{x}, t) \\ \vdots \\ v_n(v_1, \ldots, v_{n-1}, \mathbf{x}, t) \end{bmatrix}$$

Now we can code-generate the reduced DAE system!

- **Differential part**

$$\begin{aligned} \mathbf{F}(\mathbf{x}, \mathbf{x}', \mathbf{v}, t) &= \mathbf{0} && \text{implicit system class} \\ \mathbf{A}(\mathbf{x}, \mathbf{v}, t)\mathbf{x}' &= \mathbf{b}(\mathbf{x}, \mathbf{v}, t) && \text{semi-explicit system class} \\ \mathbf{x}' &= \mathbf{f}(\mathbf{x}, \mathbf{v}, t) && \text{explicit system class} \end{aligned}$$

- **Invariants**

$$\mathbf{h}(\mathbf{x}, \mathbf{v}, t) = \begin{bmatrix} \mathbf{h}_i(\mathbf{x}, \mathbf{v}, t) \\ \mathbf{h}_u(\mathbf{x}, \mathbf{v}, t) \end{bmatrix} = \mathbf{0}$$

hidden constraints
optional user-defined invariants

- **Veiling variables**

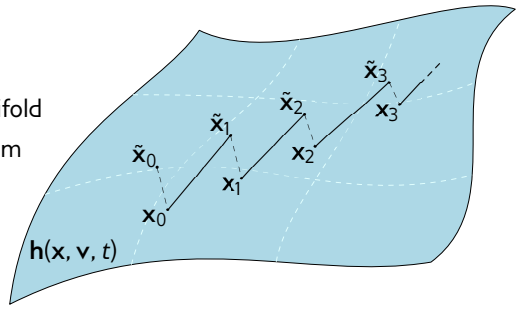
$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} v_1(\mathbf{x}, t) \\ v_2(v_1, \mathbf{x}, t) \\ \vdots \\ v_n(v_1, \dots, v_{n-1}, \mathbf{x}, t) \end{bmatrix}$$

»» Now we can integrate the system in Matlab®!

Projection is performed

- during the **numerical integration**
- to **enforce** the solution $\mathbf{x}$ onto the $\mathbf{h}(\mathbf{x}, \mathbf{v}, t)$ manifold
- by solving the **constrained minimization** problem

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{x} - \tilde{\mathbf{x}})^2 \\ \text{subject to} \quad & \mathbf{h}(\mathbf{x}, \mathbf{v}, t) = \mathbf{0} \end{aligned}$$



Find $\mathbf{x}$ with minimal distance from $\tilde{\mathbf{x}}$ that satisfies the invariants $\mathbf{h}(\mathbf{x}, \mathbf{v}, t)$

Symbolic-Numerical Validation

The Problem

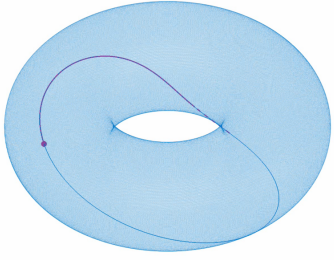
A **particle** moving over a **torus** surface

$$
\begin{cases}
x_1' = u_1 \\
x_2' = u_2 \\
x_3' = u_3 \\
u_1' = u_3 \cos(t) - x_3 \sin(t) - u_2 + 2cx_1\lambda \\
u_2' = u_3 \sin(t) + x_3 \cos(t) + u_1 + 2cx_2\lambda \\
u_3' = x_3 + 2x_3\lambda \\
\rho^2 = x_1^2 + x_2^2 + x_3^2 - 2r(x_1^2 + x_2^2)^{\frac{1}{2}} + r^2
\end{cases}
$$

with $c = 1 - r/(x_1^2 + x_2^2)^{\frac{1}{2}}$, $\rho = 5$, $r = 10$, and ICs $\mathbf{x}_0 = [15, 0, 0, 0, 15, -5, \lambda]^\top$

Exact Solution

$$
\mathbf{x}_{\text{exact}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (\rho\cos(2\pi - t) + r)\cos(t) \\ (\rho\cos(2\pi - t) + r)\sin(t) \\ \rho\sin(2\pi - t) \end{bmatrix}
$$



S. L. Campbell and E. Moore. “Constraint preserving integrators for general nonlinear higher index DAEs”. In: *Numerische Mathematik* 69.4 (Feb. 1995), pp. 383–399. ISSN: 0945-3245

Symbolic-Numerical Validation

Index Reduction

LU Factorization

Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 47\,\mathrm{f} + 30\,\mathrm{m} + 23\,\mathrm{a}$		$\mathbf{h}(\mathbf{x}, t) = 0$
Reduction step	$\mathbf{E}(\mathbf{x}, t)$	$\mathbf{g}(\mathbf{x}, t)$	$\mathbf{a}(\mathbf{x}, t)$
Index-3 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$7\,\mathrm{f} + 10\,\mathrm{m} + 6\,\mathrm{a}$
Index-2 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$22\,\mathrm{f} + 20\,\mathrm{m} + 8\,\mathrm{a}$
Index-1 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$68\,\mathrm{f} + 72\,\mathrm{m} + 33\,\mathrm{a}$
Index-0 DAEs	$388\,\mathrm{f} + 424\,\mathrm{m} + 180\,\mathrm{a}$	$79\,\mathrm{f} + 77\,\mathrm{m} + 26\,\mathrm{a}$	0
Reduced DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 258\,\mathrm{f} + 239\,\mathrm{m} + 109\,\mathrm{a}$		$\mathbf{h}(\mathbf{x}, t) = 97\,\mathrm{f} + 102\,\mathrm{m} + 47\,\mathrm{a}$

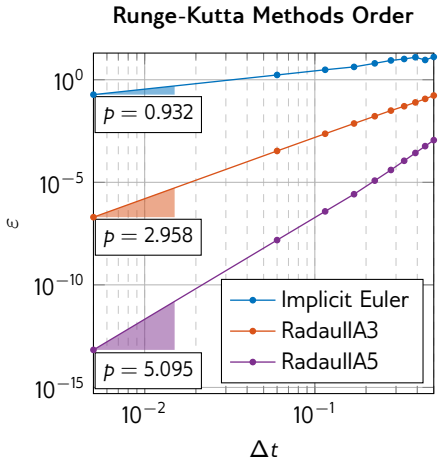
FFLU Factorization

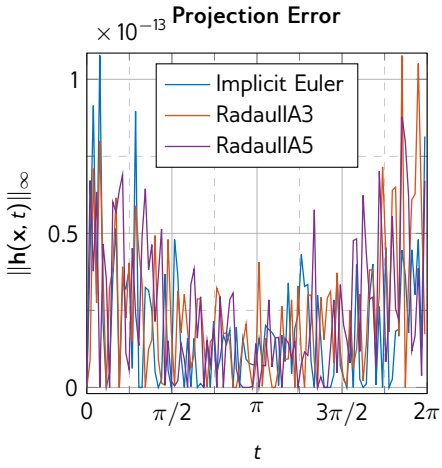
Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 47\,\mathrm{f} + 30\,\mathrm{m} + 23\,\mathrm{a}$		$\mathbf{h}(\mathbf{x}, t) = 0$
Reduction step	$\mathbf{E}(\mathbf{x}, t)$	$\mathbf{g}(\mathbf{x}, t)$	$\mathbf{a}(\mathbf{x}, t)$
Index-3 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$7\,\mathrm{f} + 10\,\mathrm{m} + 6\,\mathrm{a}$
Index-2 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$26\,\mathrm{f} + 23\,\mathrm{m} + 8\,\mathrm{a}$
Index-1 DAEs	0	$39\,\mathrm{f} + 36\,\mathrm{m} + 13\,\mathrm{a}$	$68\,\mathrm{f} + 72\,\mathrm{m} + 33\,\mathrm{a}$
Index-0 DAEs	$388\,\mathrm{f} + 424\,\mathrm{m} + 180\,\mathrm{a}$	$79\,\mathrm{f} + 77\,\mathrm{m} + 26\,\mathrm{a}$	0
Reduced DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 258\,\mathrm{f} + 239\,\mathrm{m} + 109\,\mathrm{a}$		$\mathbf{h}(\mathbf{x}, t) = 101\,\mathrm{f} + 105\,\mathrm{m} + 47\,\mathrm{a}$

Legend: f = functions, a = additions, m = multiplications, and d = divisions.

Symbolic-Numerical Validation

Order and Error Analysis



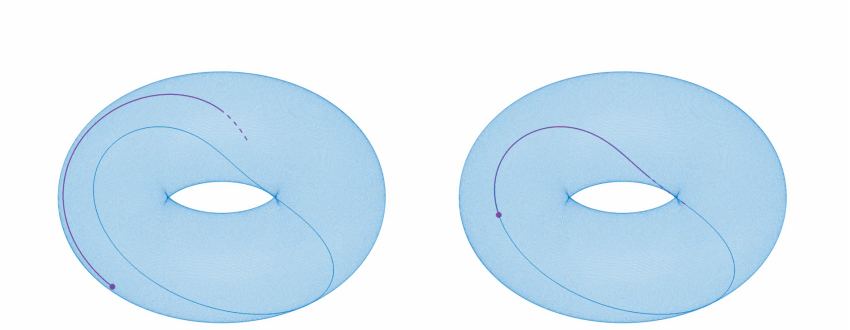


Symbolic-Numerical Validation

Solution Visualization

RadauIIA5 $\Delta t = 0.05\,\mathbf{s}$ $t \in [0, 200\pi]\,\mathbf{s}$

No projection **With projection**



Outline

Application Fields and Performance

Bechmark Problems

Multi-body dynamics

1. Car-axis (index-3)
2. Flexible slider-crank (index-3)
3. Double-wishbone suspension (index-3)

Trajectory prescribed path control problems

4. Space shuttle initial stage reentry (index-3)
5. Space shuttle final stage reentry (index-2)
6. Robotic arm control (index-5)

Electric circuits

7. Eight-nodes transistor-amplifier (index-3)
8. Electric ring modulator (index-1)
9. Cascaded differential amplifiers (up to index-100)

Generic DAE systems

10. Particle motion (index-3);
11. 2-pendula (index-3)
12. 3-pendula (index-5)
13. 4-pendula (index-9)
14. 5-pendula (index-11)

Most of the problems are taken from:

F. Mazzia and C. Magherini. *Test set for initial value problem solvers.* Tech. rep. Department of Mathematics, University of Bari, 2008

K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in DAEs.* SIAM, Jan. 1995. isbn: 978-1611971224

Application Fields and Performance

Benchmarking the Index Reduction Algorithm

The **rules**

- ⧗ 100 s time limit for symbolic simplification
- ∞ unlimited time for numerical integration
- ✓ if you can integrate the problem, you win

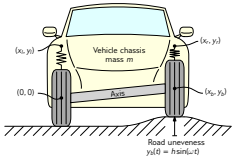
The **competitors**

- Maple® `dsolve` (undisclosed)
- Matlab® `reduceDAEIndex` (Pantelides)
- Matlab® `reduceDAEToODE` (Gaussian elim.)
- Mathematica® `NDSolve` (Pantelides)
- **Maple® + Matlab® Proposed**

Solver	Problems													
	MBD			TPPC			El. Circuits			Generic DAEs				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Maple® `dsolve`	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
Matlab® `NDSolve`	✓	✓	–	–	–	–	–	–	–	–	✗	✓	–	–
Matlab® `reduceDAEIndex`	✓	✓	–	–	–	–	–	–	–	–	✗	✓	–	–
Mathematica® `reduceDAEToODE`	✓	✓	–	–	–	–	–	–	–	–	✗	✓	–	–
Maple® + Matlab® Proposed	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓

* Incomplete testing results.

D. Schwarz Estévez, R. Lamour, and R. März. “Singularities of the Robotic Arm DAE”. In: *DAEs Forum*. Springer, 2020, pp. 433–480. isbn: 978-3030539054



LU performs better than FFLU

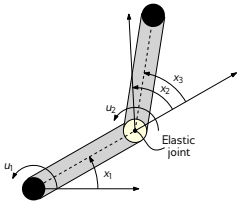
Car-Axis (Index-3) – LU Factorization

Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 108\,\mathrm{f} + 131\,\mathrm{m} + 56\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, t) = 0$	
Reduction step	$\mathbf{E}(\mathbf{x}, t)$	$\mathbf{g}(\mathbf{x}, t)$	$\mathbf{a}(\mathbf{x}, t)$
Index-3 DAEs	12 m	94 f + 145 m + 54 a	14 f + 16 m + 10 a
Index-2 DAEs	12 m	94 f + 145 m + 54 a	26 f + 45 m + 15 a
Index-1 DAEs	12 m	94 f + 145 m + 54 a	136 f + 4 d + 261 m + 95 a
Index-0 DAEs	1060 f + 38 d + 1901 m + 717 a	431 f + 8 d + 842 m + 268 a	0
Reduced DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 896\,\mathrm{f} + 4\,\mathrm{d} + 1202\,\mathrm{m} + 546\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, t) = 176\,\mathrm{f} + 4\,\mathrm{d} + 322\,\mathrm{m} + 120\,\mathrm{a}$	

Car-Axis (Index-3) – FFLU Factorization

Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 108\,\mathrm{f} + 131\,\mathrm{m} + 56\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, t) = 0$	
Reduction step	$\mathbf{E}(\mathbf{x}, t)$	$\mathbf{g}(\mathbf{x}, t)$	$\mathbf{a}(\mathbf{x}, t)$
Index-3 DAEs	0	94 f + 8 d + 150 m + 54 a	14 f + 21 m + 10 a
Index-2 DAEs	0	94 f + 8 d + 154 m + 54 a	26 f + 1 d + 44 m + 15 a
Index-1 DAEs	0	94 f + 8 d + 155 m + 54 a	136 f + 6 d + 4 d + 261 m + 95 a
Index-0 DAEs	1066 f + 55 d + 1888 m + 717 a	431 f + 18 d + 851 m + 268 a	0
Reduced DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 1549\,\mathrm{f} + 73\,\mathrm{d} + 2765\,\mathrm{m} + 1011\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, t) = 176\,\mathrm{f} + 7\,\mathrm{d} + 326\,\mathrm{m} + 120\,\mathrm{a}$	

And when strong expression swell arise ...



Robotic Arm (Index-5)

Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = 125\,\mathrm{f} + 19\,\mathrm{d} + 56\,\mathrm{m} + 64\,\mathrm{a} \quad \mathbf{h}(\mathbf{x}, t) = 0$		
Reduction step	$\mathbf{E}(\mathbf{x}, t)$	$\mathbf{g}(\mathbf{x}, t)$	$\mathbf{a}(\mathbf{x}, t)$
Index-5 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$16\,\mathrm{f} + 12\,\mathrm{a}$
Index-4 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$24\,\mathrm{f} + 6\,\mathrm{m} + 14\,\mathrm{a}$
Index-3 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$162\,\mathrm{f} + 2\,\mathrm{d} + 138\,\mathrm{m} + 114\,\mathrm{a}$
Index-2 DAEs	$14\,\mathrm{f} + 2\,\mathrm{d} + 6\,\mathrm{m} + 6\,\mathrm{a}$	$372\,\mathrm{f} + 4\,\mathrm{d} + 375\,\mathrm{m} + 253\,\mathrm{a}$	$972\,\mathrm{f} + 1\,\mathrm{d} + 1062\,\mathrm{m} + 770\,\mathrm{a}$
Index-1 DAEs	$14\,\mathrm{f} + 2\,\mathrm{d} + 6\,\mathrm{m} + 6\,\mathrm{a}$	$372\,\mathrm{f} + 4\,\mathrm{d} + 375\,\mathrm{m} + 253\,\mathrm{a}$	$\star(6.5\,\mathrm{f} + 5.6\,\mathrm{m} + 1.8\,\mathrm{a})\cdot 10^6 + 4\,\mathrm{d}$
Index-0 DAEs	$\star(8.3\,\mathrm{f} + 7.1\,\mathrm{m} + 2.3\,\mathrm{a})\cdot 10^7 + 58\,\mathrm{d}$	$(2.4\,\mathrm{f} + 2.0\,\mathrm{m} + 0.9\,\mathrm{a})\cdot 10^6 + 8\,\mathrm{d}$	0
Reduced DAEs	$\star\mathbf{F}(\mathbf{x}, \mathbf{x}', t) = (8.6\,\mathrm{f} + 7.3\,\mathrm{m} + 2.4\,\mathrm{a})\cdot 10^7 + 66\,\mathrm{d} \quad \star\mathbf{h}(\mathbf{x}, t) = (6.5\,\mathrm{f} + 5.6\,\mathrm{m} + 1.8\,\mathrm{a})\cdot 10^6 + 7\,\mathrm{d}$		

Application Fields and Performance

Expression Swell Mitigation

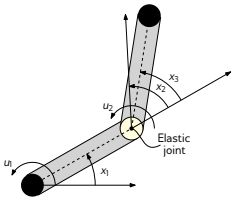
. . . hierarchical representation does the job

Robotic Arm (Index-5)

Original DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', \mathbf{v}, t) = 125\,\mathrm{f} + 19\,\mathrm{d} + 56\,\mathrm{m} + 64\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, \mathbf{v}, t) = 0$	$\mathbf{v}(\mathbf{x}, t) = 0$
Reduction step	$\mathbf{E}(\mathbf{x}, \mathbf{v}, t)$	$\mathbf{g}(\mathbf{x}, \mathbf{v}, t)$	$\mathbf{a}(\mathbf{x}, \mathbf{v}, t)$
Index-5 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$16\,\mathrm{f} + 12\,\mathrm{a}$
Index-4 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$24\,\mathrm{f} + 6\,\mathrm{m} + 14\,\mathrm{a}$
Index-3 DAEs	0	$66\,\mathrm{f} + 3\,\mathrm{d} + 50\,\mathrm{m} + 35\,\mathrm{a}$	$162\,\mathrm{f} + 2\,\mathrm{d} + 138\,\mathrm{m} + 114\,\mathrm{a}$
Index-2 DAEs	$14\,\mathrm{f} + 2\,\mathrm{d} + 6\,\mathrm{m} + 6\,\mathrm{a}$	$66\,\mathrm{f} + 1\,\mathrm{v} + 3\,\mathrm{d} + 51\,\mathrm{m} + 35\,\mathrm{a}$	$1\,\mathrm{m} + 1\,\mathrm{v}$
Index-1 DAEs	$2\,\mathrm{v} + 1\,\mathrm{a}$	$66\,\mathrm{f} + 1\,\mathrm{v} + 3\,\mathrm{d} + 51\,\mathrm{m} + 35\,\mathrm{a}$	$9\,\mathrm{f} + 4\,\mathrm{v} + 2\,\mathrm{d} + 8\,\mathrm{m} + 5\,\mathrm{a}$
Index-0 DAEs	$7\,\mathrm{v} + 1\,\mathrm{d} + 2\,\mathrm{m} + 2\,\mathrm{a}$	$66\,\mathrm{f} + 2\,\mathrm{v} + 3\,\mathrm{d} + 52\,\mathrm{m} + 35\,\mathrm{a}$	0
Reduced DAEs	$\mathbf{F}(\mathbf{x}, \mathbf{x}', \mathbf{v}, t) = 90\,\mathrm{f} + 9\,\mathrm{v} + 4\,\mathrm{d} + 63\,\mathrm{m} + 48\,\mathrm{a}$	$\mathbf{h}(\mathbf{x}, \mathbf{v}, t) = 202\,\mathrm{f} + 5\,\mathrm{v} + 4\,\mathrm{d} + 141\,\mathrm{m} + 130\,\mathrm{a}$	

Hierarchical representation details (29 veils)

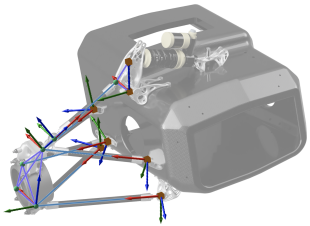
Reduction step	$\mathbf{v}(\mathbf{x}, t)$
Index-5 DAEs	0
Index-4 DAEs	0
Index-3 DAEs	0
Index-2 DAEs	$1278\,\mathrm{f} + 3\,\mathrm{v} + 6\,\mathrm{d} + 1319\,\mathrm{m} + 918\,\mathrm{a}$
Index-1 DAEs	$8401\,\mathrm{f} + 20\,\mathrm{v} + 24\,\mathrm{d} + 9451\,\mathrm{m} + 6095\,\mathrm{a}$
Index-0 DAEs	$37010\,\mathrm{f} + 558\,\mathrm{v} + 56\,\mathrm{d} + 45087\,\mathrm{m} + 28665\,\mathrm{a}$
Reduced DAEs	$\mathbf{v}(\mathbf{x}, t) = 37010\,\mathrm{f} + 558\,\mathrm{v} + 56\,\mathrm{d} + 45087\,\mathrm{m} + 28665\,\mathrm{a}$

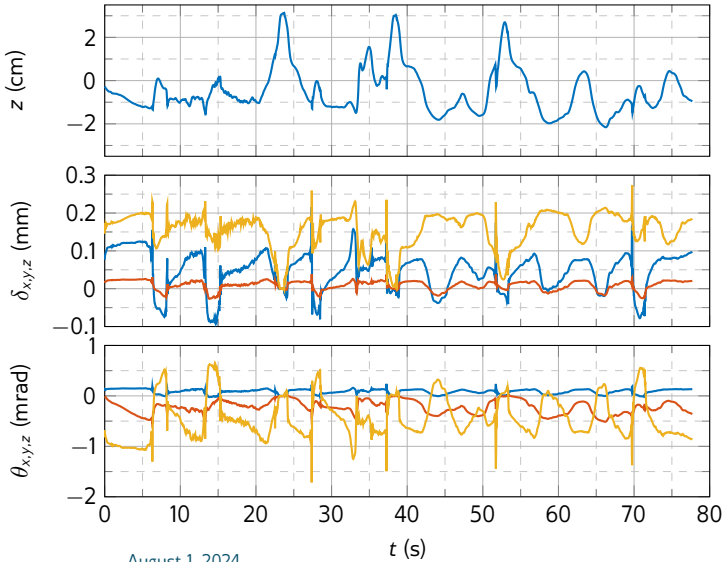


Application Fields and Performance

Numerical Stability

Numerical stability is preserved but not guaranteed





Outline

Conclusions

PhD Period

What are the main takeaways?

- **Index reduction of DAEs**
  - basic linear algebra operations
  - heavy use of symbolic computation
  - capabilities and limitations
- **Side and related work**
  - geometry library
  - tire-road interaction
  - tire model
  - symbolic analysis of structures (DSM) $\rightarrow$ *The reason that lead to the birth of LAST!*
- **Software tools**

Technical Aspects

1. Detection of **linear index-1 variables**, similar to the MBD case

$$\begin{cases} \mathbf{q}' = \mathbf{p} \\ \mathbf{p}' = \dot{\mathbf{p}} \end{cases} \quad \text{where} \quad \begin{bmatrix} \mathbf{M}(\mathbf{q},\mathbf{p},t) & -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t)^{\top} \\ -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{q},\mathbf{p},t) \\ \dot{\boldsymbol{\Phi}}_{\mathbf{q}}(\mathbf{q},t)\mathbf{p} + \boldsymbol{\Phi}_{tt}(\mathbf{q},t) \end{bmatrix}$$

2. **System augmentation** with "veil" equations
   - complicated pivots become "veil" equations
   - 1s in the diagonal, complicated terms on the RHS

Implementation Aspects

1. **Open-source** CAS
   - what is happening inside?
   - why is it not working on this example?
2. Development of a **C++ library** for enhanced integration speed and precision

Technical Aspects

1. Detection of **linear index-1 variables**, similar to the MBD case

$$\begin{cases} \mathbf{q}' = \mathbf{p} \\ \mathbf{p}' = \dot{\mathbf{p}} \end{cases} \quad \text{where} \quad \begin{bmatrix} \mathbf{M}(\mathbf{q},\mathbf{p},t) & -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t)^{\top} \\ -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{q},\mathbf{p},t) \\ \dot{\boldsymbol{\Phi}}_{\mathbf{q}}(\mathbf{q},t)\mathbf{p} + \boldsymbol{\Phi}_{tt}(\mathbf{q},t) \end{bmatrix}$$

2. **System augmentation** with "veil" equations
   - complicated pivots become "veil" equations
   - 1s in the diagonal, complicated terms on the RHS

Implementation Aspects

1. **Open-source** CAS
   - what is happening inside?
   - why is it not working on this example?
2. Development of a **C++ library** for enhanced integration speed and precision

Possible Improvements

Index Reduction Algorithm

Technical Aspects

1. Detection of **linear index-1 variables**, similar to the MBD case

$$\begin{cases} \mathbf{q}' = \mathbf{p} \\ \mathbf{p}' = \dot{\mathbf{p}} \end{cases} \quad \text{where} \quad \begin{bmatrix} \mathbf{M}(\mathbf{q},\mathbf{p},t) & -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t)^{\top} \\ -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{q},\mathbf{p},t) \\ \dot{\boldsymbol{\Phi}}_{\mathbf{q}}(\mathbf{q},t)\mathbf{p} + \boldsymbol{\Phi}_{tt}(\mathbf{q},t) \end{bmatrix}$$

2. **System augmentation** with "veil" equations
   - complicated pivots become "veil" equations
   - 1s in the diagonal, complicated terms on the RHS

Implementation Aspects

1. **Open-source** CAS
   - what is happening inside?
   - why is it not working on this example?
2. Development of a **C++ library** for enhanced integration speed and precision

Possible Improvements

Index Reduction Algorithm

Technical Aspects

1. Detection of **linear index-1 variables**, similar to the MBD case

$$\begin{cases} \mathbf{q}' = \mathbf{p} \\ \mathbf{p}' = \dot{\mathbf{p}} \end{cases} \quad \text{where} \quad \begin{bmatrix} \mathbf{M}(\mathbf{q},\mathbf{p},t) & -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t)^{\top} \\ -\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q},t) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{q},\mathbf{p},t) \\ \dot{\boldsymbol{\Phi}}_{\mathbf{q}}(\mathbf{q},t)\mathbf{p} + \boldsymbol{\Phi}_{tt}(\mathbf{q},t) \end{bmatrix}$$

2. **System augmentation** with "veil" equations
   - complicated pivots become "veil" equations
   - 1s in the diagonal, complicated terms on the RHS

Implementation Aspects

1. **Open-source** CAS
   - what is happening inside?
   - why is it not working on this example?
2. Development of a **C++ library** for enhanced integration speed and precision

Thank you for your attention!

Questions?

Projector-based Analysis

Tractability Index

- **Projector-based analysis** deals with properly stated leading term DAEs of the form

$$\mathbf{F}(\mathbf{d}(\mathbf{x},t)',\mathbf{x},t)=\mathbf{0}$$

- If we define the matrix sequence

$$\begin{aligned}&\mathbf{G}_0=\mathbf{A}\mathbf{D} \quad \text{and} \quad \mathbf{B}_0=\mathbf{B}\\ &\mathbf{G}_{i+1}=\mathbf{G}_i+\mathbf{B}_i\mathbf{Q}_i\\ &\mathbf{B}_{i+1}=\left(\mathbf{B}_i-\mathbf{G}_{i+1}\mathbf{D}^-\tfrac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{D}\mathbf{P}_i\ldots\mathbf{P}_{i+1}\mathbf{D}^-\right)\mathbf{D}\mathbf{P}_i\ldots\mathbf{P}_{i-1}\right)\mathbf{P}_i\end{aligned}$$

with

$$\begin{aligned}\mathbf{D}&:=\mathbf{D}(\mathbf{x},t)=\mathbf{d}_{\mathbf{x}}(\mathbf{x},t)\\ \mathbf{A}&:=\mathbf{A}(\mathbf{y},\mathbf{x},t)=\mathbf{F}_{\mathbf{y}}(\mathbf{y},\mathbf{x},t)\\ \mathbf{B}&:=\mathbf{B}(\mathbf{y},\mathbf{x},t)=\mathbf{F}_{\mathbf{x}}(\mathbf{y},\mathbf{x},t)\end{aligned}$$

and where $\mathbf{P}_i=\mathbf{I}-\mathbf{Q}_i$, $\quad\mathbf{Q}_i\mathbf{Q}_j=\mathbf{0}$, $\quad\mathbf{D}^-\mathbf{D}=\mathbf{P}_0$

- The **tractability index** first integer μ where the matrix $\mathbf{G}_\mu$ becomes non-singular

The difficult part is to compute the projectors $\mathbf{Q}_i$

Projector-based Analysis

Nnullspace Projectors

- The **nullspace projector** $\mathbf{Q}_i$ is found such that

$$\operatorname{im}(\mathbf{Q}_i) = \ker(\mathbf{G}_i) \qquad (\mathbf{G}_i\mathbf{Q}_i = \mathbf{0})$$

- In R. Lamour, R. März, and C. Tischendorf. *DAEs: A Projector Based Analysis*. Springer, 2013. isbn: 978-3642275555, ch.7, p.400 is claimed that:

  For problems of limited dimension a formula manipulation system like Mathematica® *or* Maple® *can be used to compute a basis [of* $\ker(\mathbf{Q})$*]. The command in* Mathematica® *is* `NullSpace[G]` and in Maple® `nullspace(G)`.
  However, to provide a basis of the nullspace of a given matrix one usually has to carry out a factorization, for instance a singular value decomposition (SVD).

Projection and Consistent Initialization

Projection Scheme

- Same the **constrained minimization** problem

$$\underset{\mathbf{x}}{\text{minimize}} \quad \frac{1}{2}(\mathbf{x}-\tilde{\mathbf{x}})^2 \quad \text{subject to} \quad \mathbf{h}(\mathbf{x},\mathbf{v},t)=\mathbf{0}$$

- The **Lagrangian** of the minimization problem is

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = \frac{1}{2}(\mathbf{x}-\tilde{\mathbf{x}})^2 + \boldsymbol{\lambda}\cdot\mathbf{h}(\mathbf{x},\mathbf{v},t) \quad \rightarrow \quad \begin{cases} \mathbf{x}+\mathbf{J}\mathbf{h}_{\mathbf{x}}^{\top}\boldsymbol{\lambda}=\tilde{\mathbf{x}} \\ \mathbf{h}(\mathbf{x},\mathbf{v},t)=\mathbf{0} \end{cases}$$

- The **iterative method** is ...

$$\begin{bmatrix} \mathbf{I} & \mathbf{J}\mathbf{h}_{\mathbf{x}}^{\top} \\ \mathbf{J}\mathbf{h}_{\mathbf{x}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}-\mathbf{x} \\ -\mathbf{h}(\mathbf{x},\mathbf{v},t) \end{bmatrix} \quad \text{where the step is} \quad \mathbf{x}=\tilde{\mathbf{x}}+\delta\mathbf{x}$$

  ...derived from the Taylor expansion

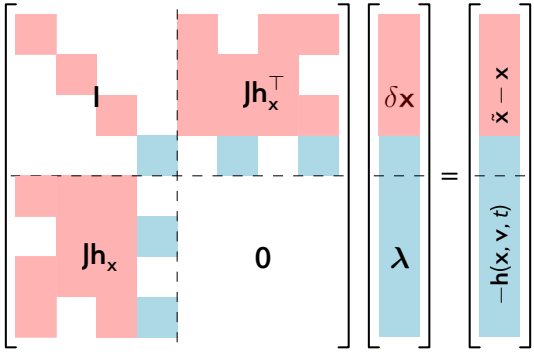
$$\begin{cases} \mathbf{x}+\delta\mathbf{x}+\mathbf{J}\mathbf{h}_{\mathbf{x}}^{\top}(\mathbf{x}+\delta\mathbf{x},\mathbf{v},t)\boldsymbol{\lambda}=\tilde{\mathbf{x}} \\ \mathbf{h}(\mathbf{x},\mathbf{v},t)+\mathbf{J}\mathbf{h}_{\mathbf{x}}(\mathbf{x},\mathbf{v},t)\delta\mathbf{x}+\mathcal{O}\left(\|\delta\mathbf{x}\|^2\right)=\mathbf{0} \end{cases}$$

Projection and Consistent Initialization

Two Problems, One Solution

If part of ICs are not available:

- We can use the **projection scheme** to calculate consistent ICs
- The **iterative method** is reduced as selecting only the unknown ICs and the invariants we want to be maintained



Outline

Full-Pivoting LU Factorization

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, with $m \geq n$, the full-pivoting LU decomposition is defined as the process of decomposing $\mathbf{A}$ into the product of

- a $\mathbf{L} \in \mathbb{R}^{m \times m}$ lower-triangular matrix with all diagonal entries equal to 1
- a $\mathbf{U} \in \mathbb{R}^{m \times n}$ upper-triangular matrix
- a $\mathbf{P} \in \mathbb{R}^{m \times m}$ and a $\mathbf{Q} \in \mathbb{R}^{n \times n}$ matrices for rows and columns permutation

such that $\mathbf{PAQ} = \mathbf{LU}$

Full-Pivoting FFLU Factorization

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, with $m \geq n$, the full-pivoting FFLU decomposition is defined as the process of decomposing $\mathbf{A}$ into the product of

- a lower-triangular matrix $\mathbf{L} \in \mathbb{R}^{m \times m}$ with all diagonal entries equal to 1;
- a diagonal matrix $\mathbf{D} \in \mathbb{R}^{m \times m}$
- an upper-triangular matrix $\mathbf{U} \in \mathbb{R}^{m \times n}$
- a $\mathbf{P} \in \mathbb{R}^{m \times m}$ and a $\mathbf{Q} \in \mathbb{R}^{n \times n}$ matrices for rows and columns permutation

such that $\mathbf{PDAQ} = \mathbf{LU}$

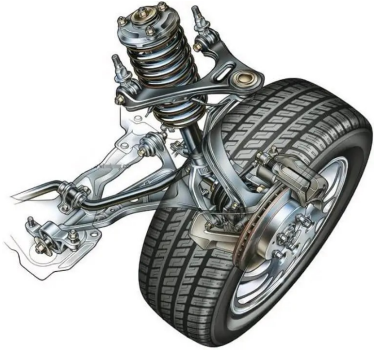
Outline

A Symbolic Approach to Structure Compliance

Practical Application on Vehicle Suspension Compliance Analysis

- The suspension system has to **guarantee** ...
  1. **comfort** to the passengersminimize vertical accelerations of the chassis
  2. **road holding** and **stability** of the vehicle
  3. control the **wheel attitude** with respect to the road surface

→ Flexibility affects the vehicle dynamics



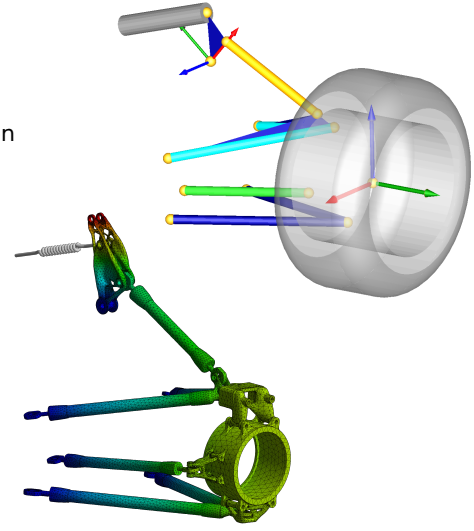
K&C analysis

Introduction

The **kinematics** and **compliance** of the suspension system are ...

1. the **motion** and **deformation** of the system
2. key aspects of the **design** of the system

Kinematics and compliance are what is needed to describe the suspension in a vehicle model



K&C analysis

State-of-the-art solutions

Methods	Real-Time	Accuracy	Parametric	Easy to Setup
FE methods	✗	✓	✗	✗
Map-based methods	✓	•	✗	✓
Flexible multibody	•	✓	•	✗
What we aim to …	✓	•	✓	✓

Legend: ✓ satisfied, • satisfied with limitations, and ✗ not satisfied.

Symbolic computation

Finite element method problem

- Symbolic computation can be applied to the **finite element (FE) method**.
- **Stiffness matrix**, **load vector**, and **boundary conditions** are defined symbolically.

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fs} \\ \mathbf{K}_{sf} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} c\mathbf{d}_f \\ \mathbf{d}_s \end{bmatrix} = \begin{bmatrix} c\mathbf{f}_f \\ \mathbf{f}_s \end{bmatrix} \xrightarrow[\text{method}]{\text{direct stiffness}} \begin{aligned} \mathbf{d}_f &= \mathbf{K}_{ff}^{-1} \left(\mathbf{f}_f - \mathbf{K}_{fs}\, \mathbf{d}_s \right) \\ \mathbf{f}_s &= \mathbf{K}_{sf}\, \mathbf{d}_f + \mathbf{K}_{ss}\, \mathbf{d}_s \end{aligned}$$

1. Possibility to **evaluate** the system **without** the need to "**reassemble**" the system
2. Parametric formulation
3. Symbolic solution of the structure deformations and reaction forces

$$\text{FE method} \xrightarrow[\text{computation}]{\text{symbolic}} \text{Symbolic FE model + A lot of flexibility!}$$

- We combined all these features in the Maple® TrussMe-Fem library (open-source)

- Symbolic matrix factorization is used for the computation of the linear system **solutions**

$$\mathbf{d}_f = \mathbf{K}_{ff}^{-1} \left(\mathbf{f}_f - \mathbf{K}_{fs} \, \mathbf{d}_s \right)$$

Problem

Maple®matrix factorizations have **limited capabilities**.

- We developed the `LAST` symbolic matrix factorization toolbox capable of ...
  1. extending Maple®factorization capabilities
  2. introducing **veiling variables** not to increase the expressions size (through `LEM` toolbox)
  3. performing efficient LU, Fraction-Free LU, QR, and Gauss-Jordan factorizations with limited **expresion swell**

Symbolic computation

Mixed symbolic/numerical solution

- **Symbolic** solution is not always possible due to
  1. capability of the symbolic kernel to handle **complicated** expressions
  2. **non-linear** or mathematically **unsolvable** problems
- Symbolic computation should be exploited as much as possible!

FE model $\xrightarrow[\text{computation}]{\text{symbolic}}$ **Stop** $\xrightarrow[\text{generation}]{\text{code}}$ **Efficient** numeric FE model solution

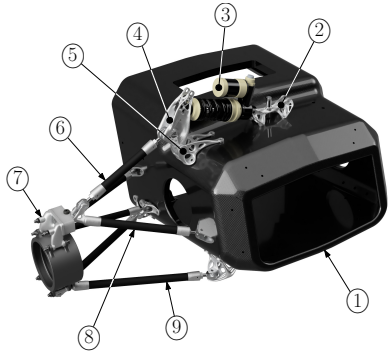
FE model $\xrightarrow[\text{computation}]{\text{symbolic}}$ **Symbolic solution** $\xrightarrow[\text{generation}]{\text{code}}$ **Fastest** numeric evaluation!

- K&C of suspensions are computed through a **mixed** symbolic/numerical approach

Suspensions symbolic modeling

Case study

The modeled suspension is a **double wishbone** suspension system of the **Formula SAE** vehicle of *E-Agle Trento Racing Team* (University of Trento)

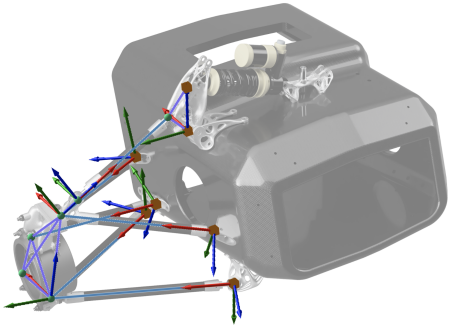


Components:

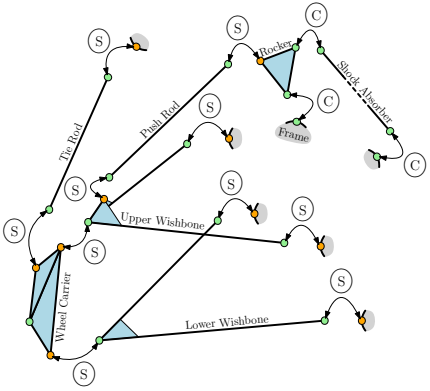
1. Carbon fiber cell
2. Shock absorber support
3. Shock absorber
4. Rocker
5. Rocker support
6. Push rod
7. Wheel carrier
8. Upper wishbone arm
9. Lower wishbone arm

Suspensions symbolic modeling

Rigid multibody & FE models

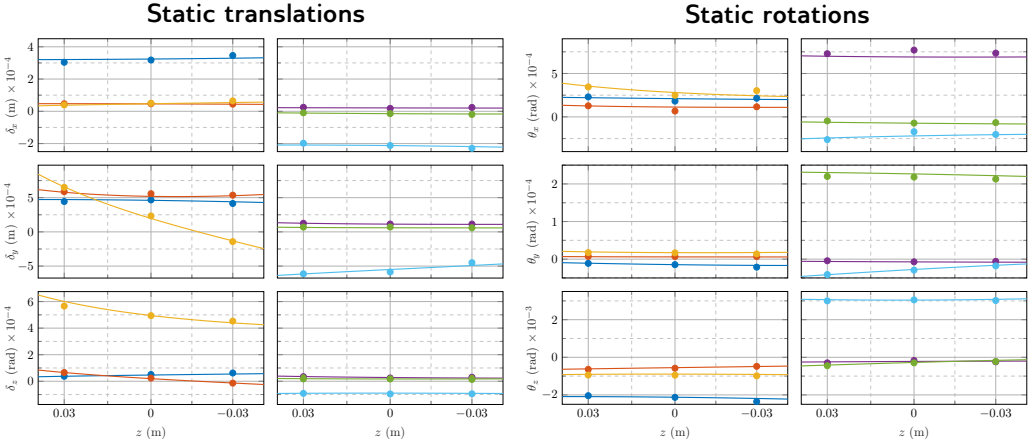


138 DoFs symbolic FE model



Simulation results

Static analysis



Legend: TrussMe-Fem solution (*solid lines*), Ansys® data (*bullets*).

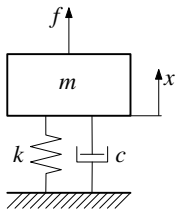
■ $F_x = 4000\,\mathrm{N}$, ■ $F_y = 4000\,\mathrm{N}$, ■ $F_z = 4000\,\mathrm{N}$, with $M_x = M_y = M_z = 0\,\mathrm{N\,m}$.

■ $M_x = 400\,\mathrm{N\,m}$, ■ $M_y = 400\,\mathrm{N\,m}$, ■ $M_z = 400\,\mathrm{N\,m}$, with $F_x = F_y = F_z = 0\,\mathrm{N}$

Suspensions symbolic modeling

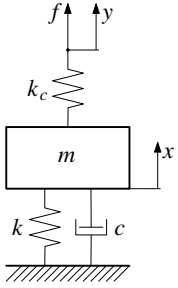
Modeling assumptions

No compliance



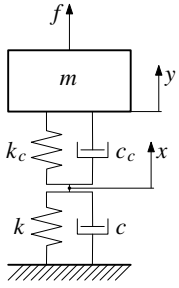
$$m\ddot{x} + c\dot{x} + kx = f$$

Quasi-static compliance



$$\begin{cases} y = x + d \\ k_c d = f \\ m\ddot{x} + c\dot{x} + kx = f \end{cases}$$

Dynamic compliance



$$\begin{cases} y = x + d \\ kx + c\dot{x} = k_c d + c_c \dot{d} \\ m\ddot{y} = k_c d + c_c \dot{d} + f \end{cases}$$

Suspensions symbolic modeling

Frequency response

The **frequency response** of the system is influenced by the **decoupling** strategy.

- no compliance

$$\frac{\ddot{Y}(s)}{F(s)} = \frac{s^2}{ms^2 + cs + k}$$

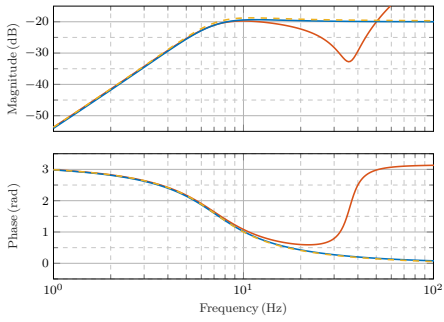
- quasi-static compliance

$$\frac{\ddot{Y}(s)}{F(s)} = \frac{s^2}{ms^2 + cs + k} + \frac{s^2}{k_c}$$

$$\omega_n = \sqrt{\frac{k}{m}} \qquad \omega_z = \sqrt{\frac{k + k_c}{m}}$$

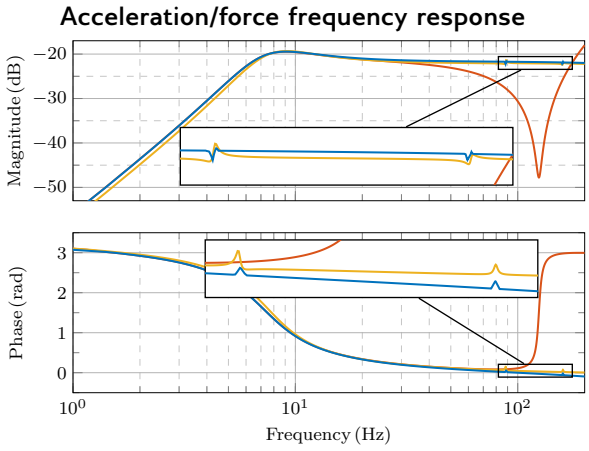
- dynamic compliance

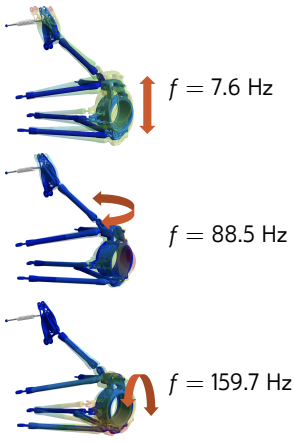
$$\frac{\ddot{Y}(s)}{F(s)} = \frac{(c+c_c)s^3 + (k+k_c)s^2}{m(c+c_c)s^3 + (c\,c_c + m(k+k_c))s^2 + (k_c\,c + k\,c_c)s + k\,k_c}$$



Simulation results

Frequency response

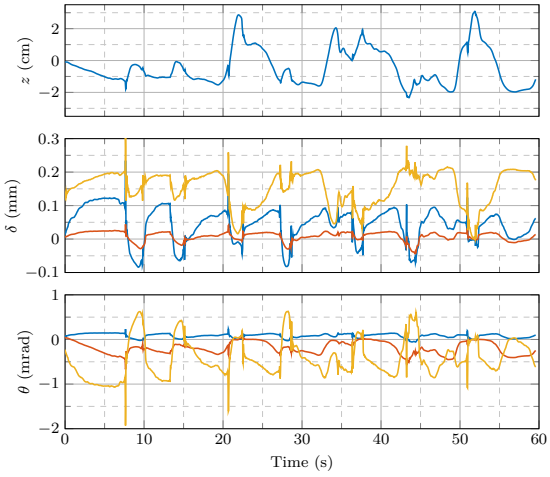




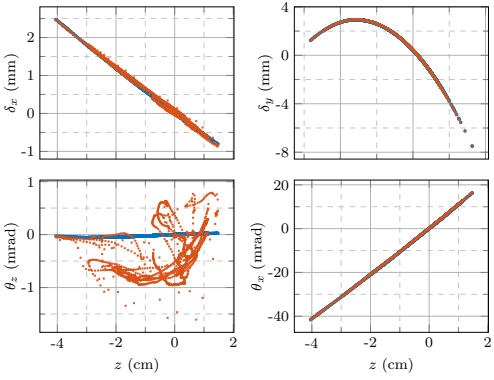
Legend: MB with compliance dynamic, MB and steady-state compliance, Ansys®Finite Element (FE) modal analysis.

Simulation results

Full Vehicle Simulation



Legend: *x*-, *y*-, *z*-component.



Legend: kinematic, total.

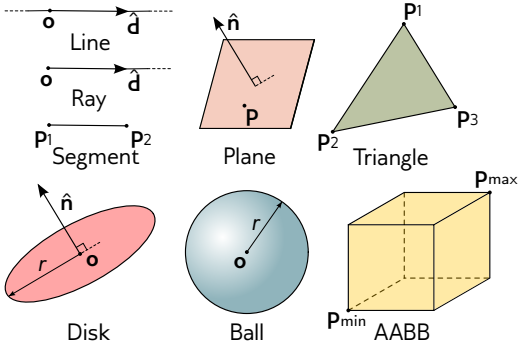
Simulation results

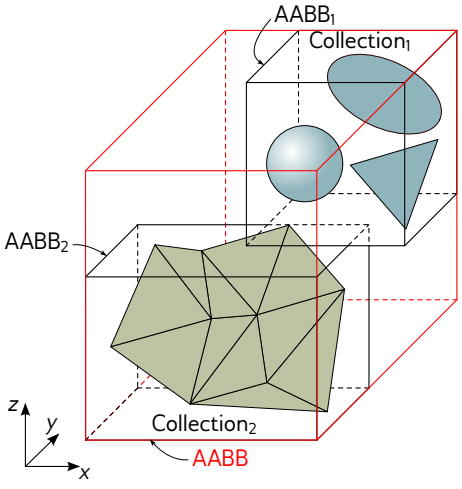
Timing

Compliance model	Computation time			
	μ (µs)	min (µs)	max (µs)	σ^2 (µs^2)
Compliance maps (linear)	40	4	48	1.2
Compliance maps (cubic)	193	84	880	42.9
Symbolic FEM	181	42	417	45.2

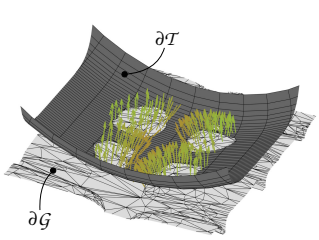
Outline

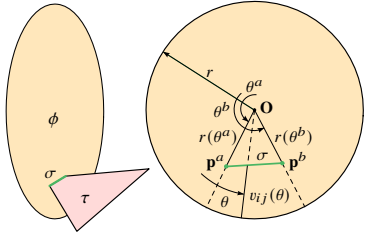
Geometry Library

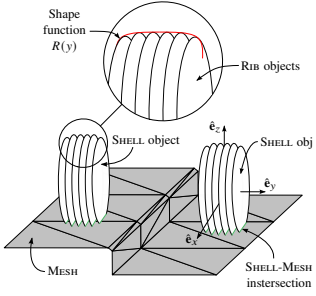


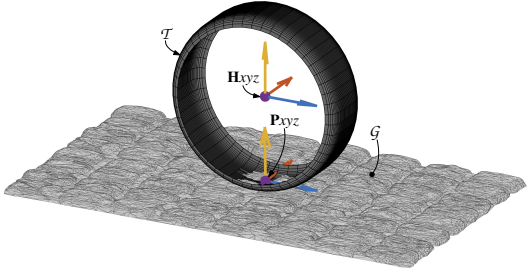


Tire-Road Interaction Model

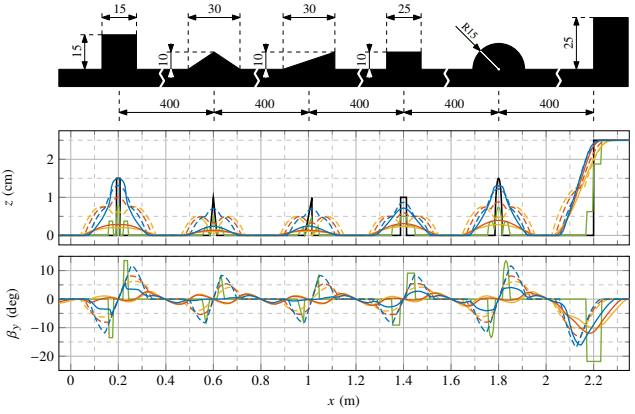


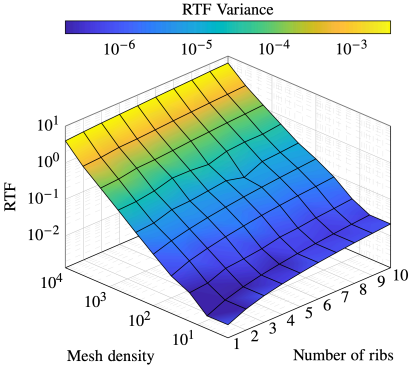




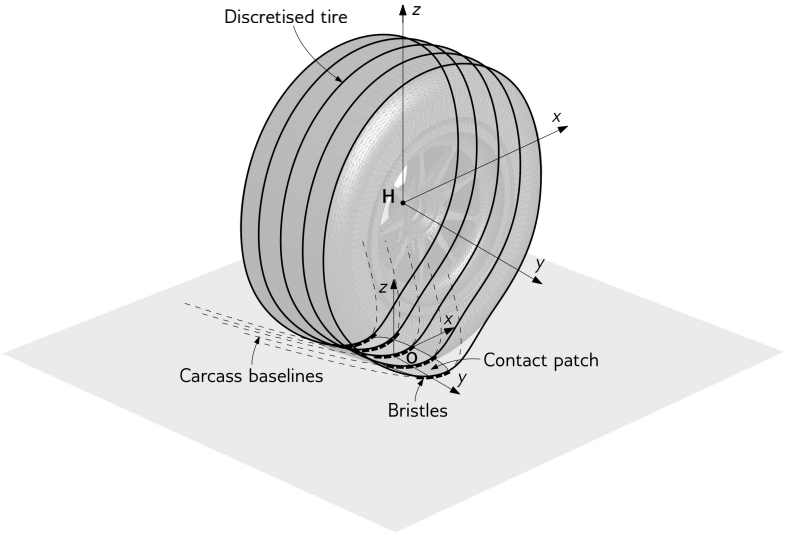


Tire-Road Interaction Model

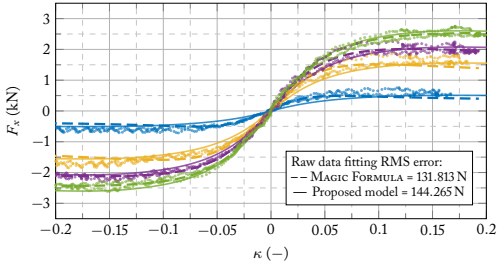


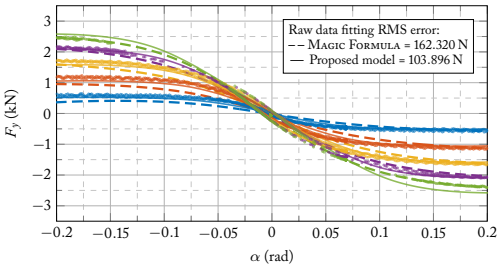


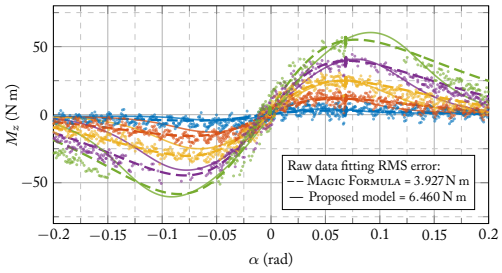
Brush Tire Model



Brush Tire Model







Brush Tire Model

Quasi-Newton Methods Performance with 5 Ribs

Method	Time (µs)				Success
	min	max	μ	σ^2	
PIM	518.0	7949.0	617.2	13953.2	0.0%
G1M	—	—	—	—	Overflow
G2M	23.5	1089.4	69.0	5252.0	99.5%
ENM	25.0	2025.0	81.5	11002.8	99.4%
BBM	—	—	—	—	0.0%
BGM	22.6	948.1	62.5	1973.0	99.9%
BCM	22.9	264.4	55.7	885.9	100.0%
D-PIM	507.0	5383.0	1151.9	568491	0.0%
D-G1M	—	—	—	—	Overflow
D-G2M	21.5	3195.5	80.4	40321.9	99.5%
D-ENM	27.0	5207.7	97.5	52752.6	99.5%
D-BBM	22.3	1284.0	76.9	5126.3	100.0%
D-BGM	21.3	3238.9	67.0	8142.9	99.9%
D-BCM	22.5	316.8	62.1	1615.1	100.0%

Brush Tire Model

Broyden Combined Method Performance

Ribs	Time (µs)				Success
	min	max	μ	σ^2	
1	12.1	89.0	19.4	20.6	100.0%
5	22.9	264.4	55.7	885.9	100.0%
10	106.9	361.8	167.3	1746.22	100.0%
15	158.0	523.5	249.4	3878.2	100.0%
20	210.1	693.3	332.9	7028.7	100.0%
25	262.7	858.4	414.0	10708.3	100.0%