Index Reduction for Nonlinear Differential-Algebraic Equations by Combinatorial Relaxation

Taihei OKI
University of Tokyo

Shonan Meeting 125 @ Hayama, Kanagawa, Japan
June 28, 2018

SHONAN MEETING
NII
東京大学
THE UNIVERSITY OF TOKYO

Contents

1 Differential-Algebraic Equations

2 Combinatorial Relaxation

3 Proposed Algorithm

Contents

1 Differential-Algebraic Equations

2 Combinatorial Relaxation

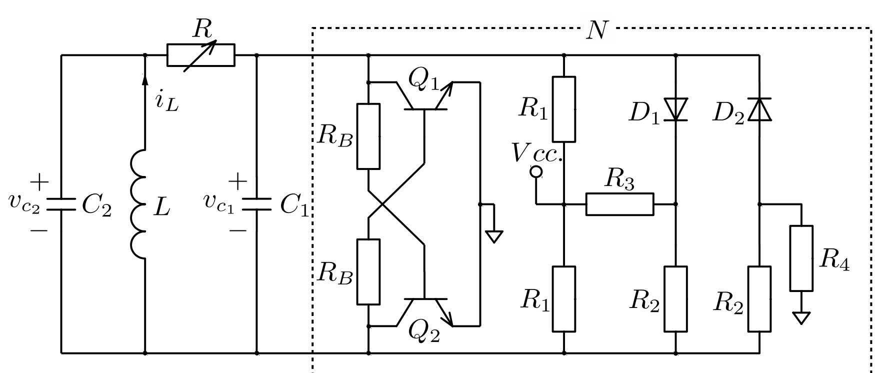
3 Proposed Algorithm

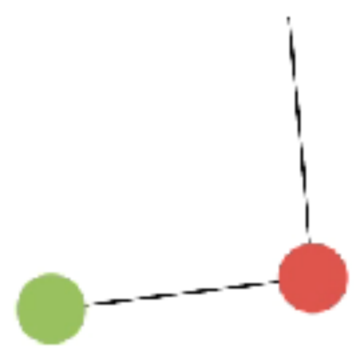
Simulation of Dynamical Systems

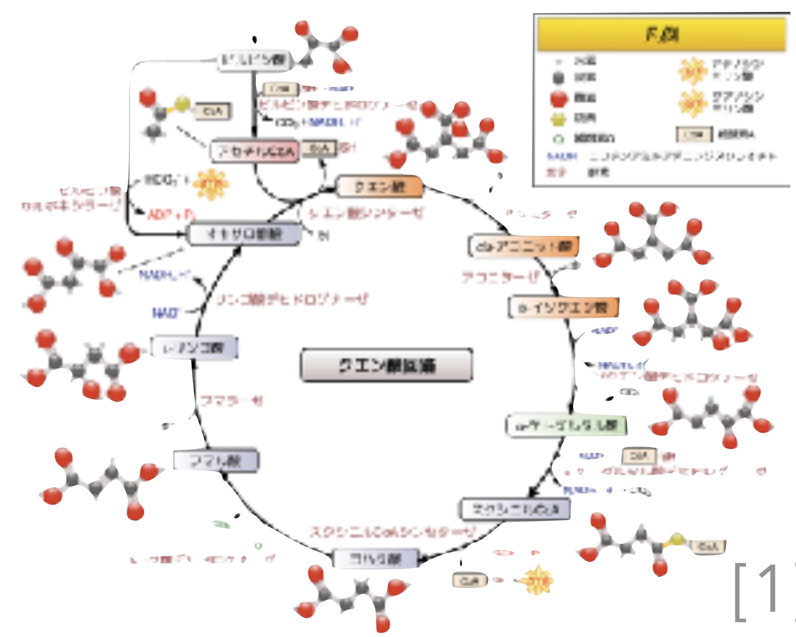
electrical network

mechanical system

chemical reaction plant







[1]

dynamical systems are modeled by differential equations

[1] https://ja.wikipedia.org/wiki/%E3%82%AF%E3%82%A8%E3%83%B3%E9%85%B8%E5%9B%9E%E8%B7%AF#/media/File:Citric_acid_cycle_with_aconitate_2_ja.svg

Ordinary Differential Equations (ODEs)

$x\colon \mathbb{R} \to \mathbb{R}^n$: unknown function

Ordinary Differential Equation (ODE)

$$\dot{x}(t) = \varphi(t, x(t))$$

$\varphi\colon \mathbb{R}^{1+n} \to \mathbb{R}^n$

- numerical methods are well-studied
  The (explicit/implicit) Euler method, the Lunge–Kutta method, etc.

- higher-order ODEs can be converted
  into first-order ODEs by introducing new variables

Algebraic Equations

$x\colon\ \mathbb{R} \to \mathbb{R}^n$: unknown function

Ordinary Differential Equation (ODE)

$$\dot{x}(t) = \varphi(t, x(t))$$

Algebraic Equation

$$G(t, x(t)) = 0$$

$G\colon\ \mathbb{R}^{1+n} \to \mathbb{R}^n$

- numerical methods are well-studied
  The Newton–Raphson method, etc.

Differential-Algebraic Equations (DAEs)

$x \colon \mathbb{R} \to \mathbb{R}^n$: unknown function

Ordinary Differential Equation (ODE)

$$\dot{x}(t) = \varphi(t, x(t))$$

Algebraic Equation

$$G(t, x(t)) = 0$$

[Gear '71]

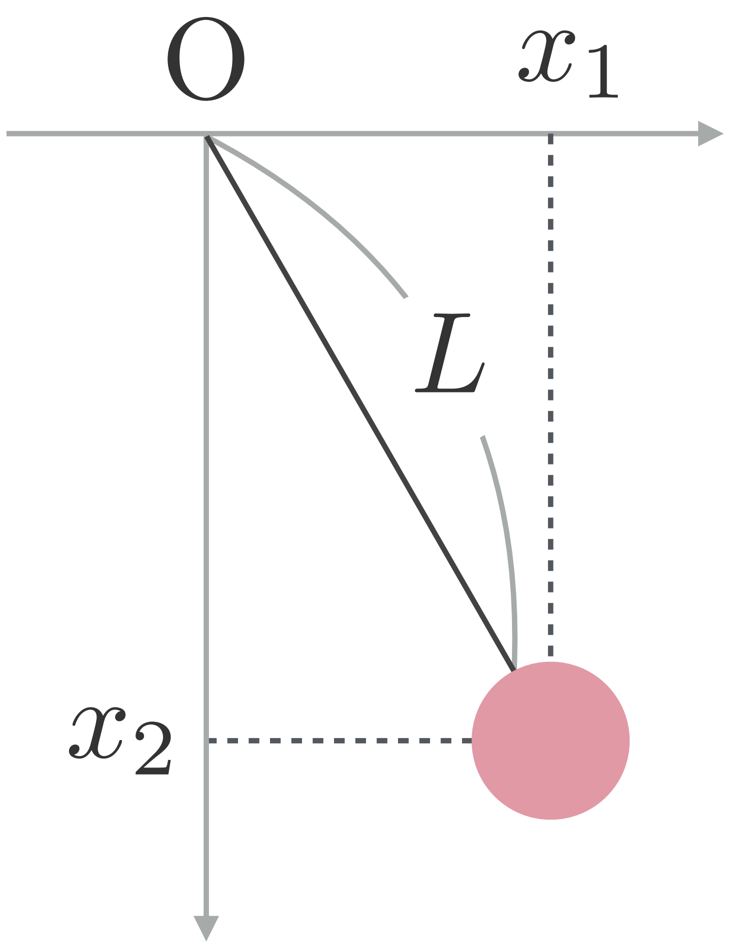
Differential-Algebraic Equation (DAE)

$F \colon \mathbb{R}^{1+n(l+1)} \to \mathbb{R}^n$

$$F(t, x(t), \dot{x}(t), \ddot{x}(t), \ldots, x^{(l)}(t)) = 0$$

DAE = **ODE** + **Algebraic Equation**

Example | Simple Pendulum



unknown functions: $x_1(t), x_2(t), x_3(t)$

$$\begin{cases} \ddot{x}_1 + x_1 x_3 = 0 \\ \ddot{x}_2 + x_2 x_3 - g = 0 \\ \text{ODEs} \\ x_1^2 + x_2^2 - L^2 = 0 \\ \text{algebraic equation} \end{cases}$$

… Can we solve DAEs numerically?

Differentiation Index

differentiation index
[Campbell–Gear '95]

the number of times one have to differentiate a DAE to get an ODE

DAE

$$\begin{cases} \dot{x}_1^2 + x_1^2 - 1 = 0 \\ \dot{x}_1 + \dot{x}_2 = 0 \end{cases}$$

solve for $\dot{x}_1$ and $\dot{x}_2$

ODE

$$\begin{cases} \dot{x}_1 = \sqrt{1 - x_1^2} \\ \dot{x}_2 = -\sqrt{1 - x_1^2} \end{cases}$$

index = **0**

Differentiation Index

differentiation index
[Campbell–Gear '95]

the number of times one have to differentiate a DAE to get an ODE

DAE

$$\begin{cases} \dot{x}_1^2 + x_1^2 - 1 = 0 \\ \dot{x}_1 + x_2 = 0 \end{cases}$$

I have no $\dot{x}_2$!

Differentiation Index

differentiation index
[Campbell–Gear '95]

the number of times one have to differentiate a DAE to get an ODE

DAE

$$\begin{cases} \dot{x}_1^2 + x_1^2 - 1 = 0 \\ \dot{x}_1 + x_2 = 0 \end{cases}$$

solve for $\dot{x}_1$ and x_2 →

not an ODE

$$\begin{cases} \dot{x}_1 = \sqrt{1 - x_1^2} \\ x_2 = -\sqrt{1 - x_1^2} \end{cases}$$

differentiate the 2nd eq. →

ODE

$$\begin{cases} \dot{x}_1 = \sqrt{1 - x_1^2} \\ \dot{x}_2 = \frac{2x_1 \dot{x}_1}{\sqrt{1 - x_1^2}} = 2x_1 \end{cases}$$

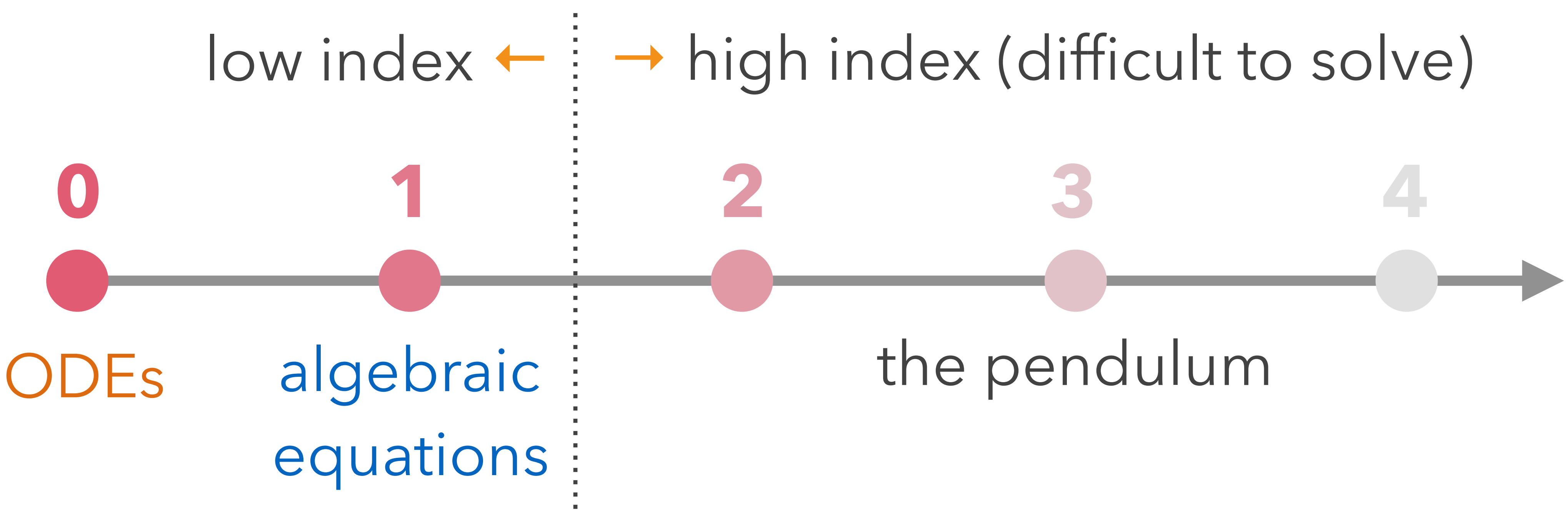
index = **1**

Differentiation Index

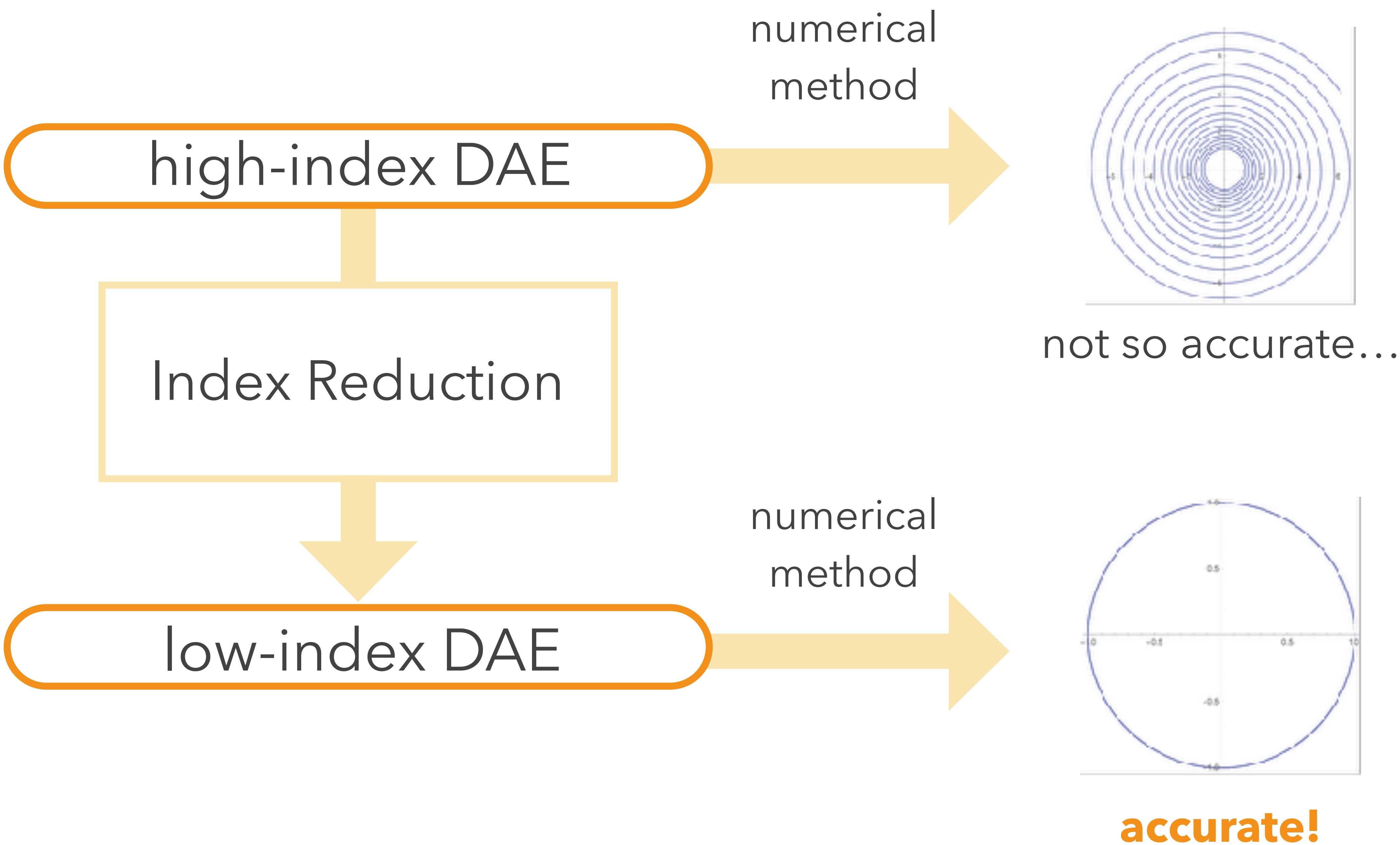
differentiation index
[Campbell–Gear '95]

the number of times one have to differentiate a DAE to get an ODE

Indeed, the difficulty in numerically solving a DAE is measured by its **index**.



Index Reduction



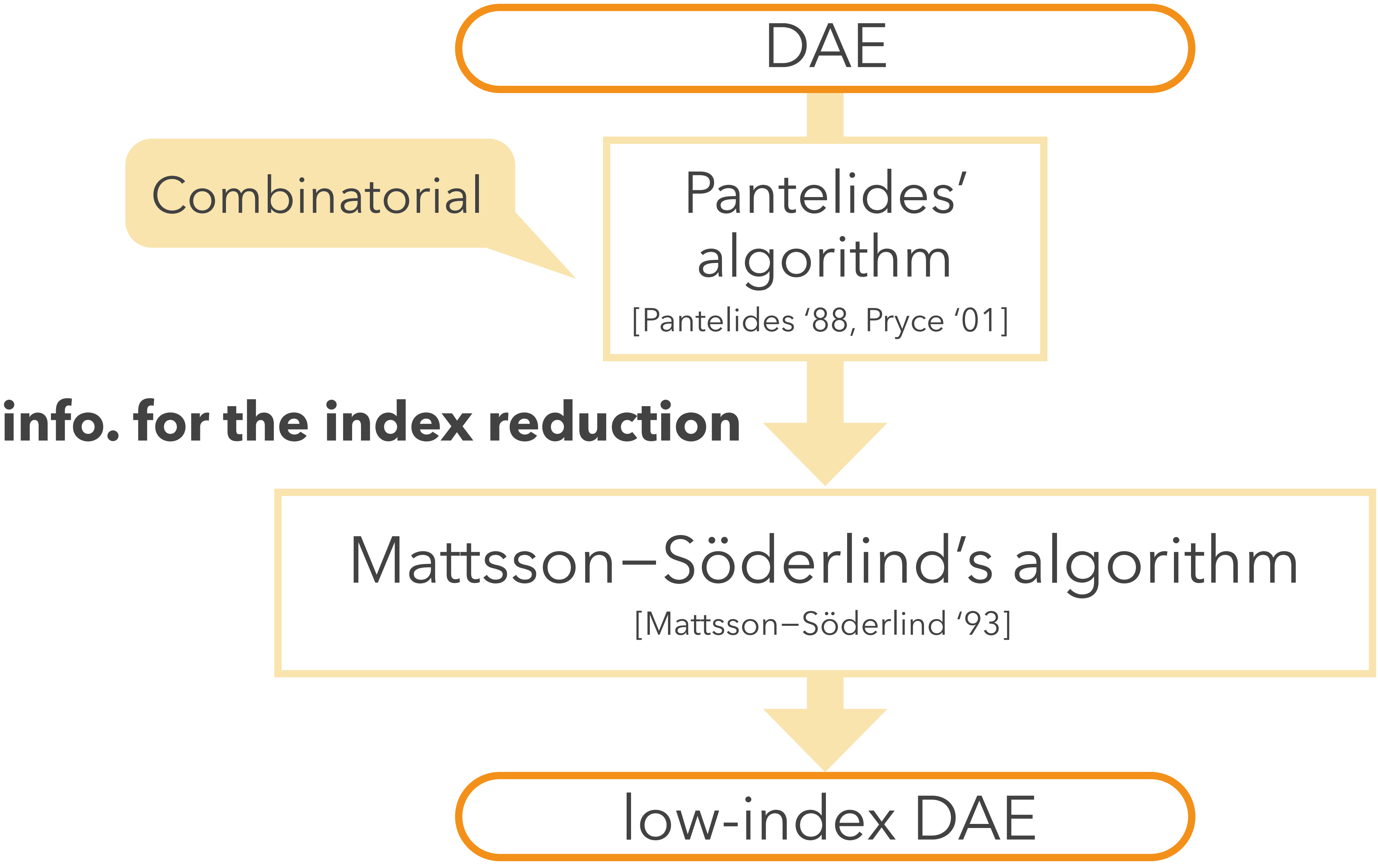
Contents

1 Differential-Algebraic Equations

2 Combinatorial Relaxation

3 Proposed Algorithm

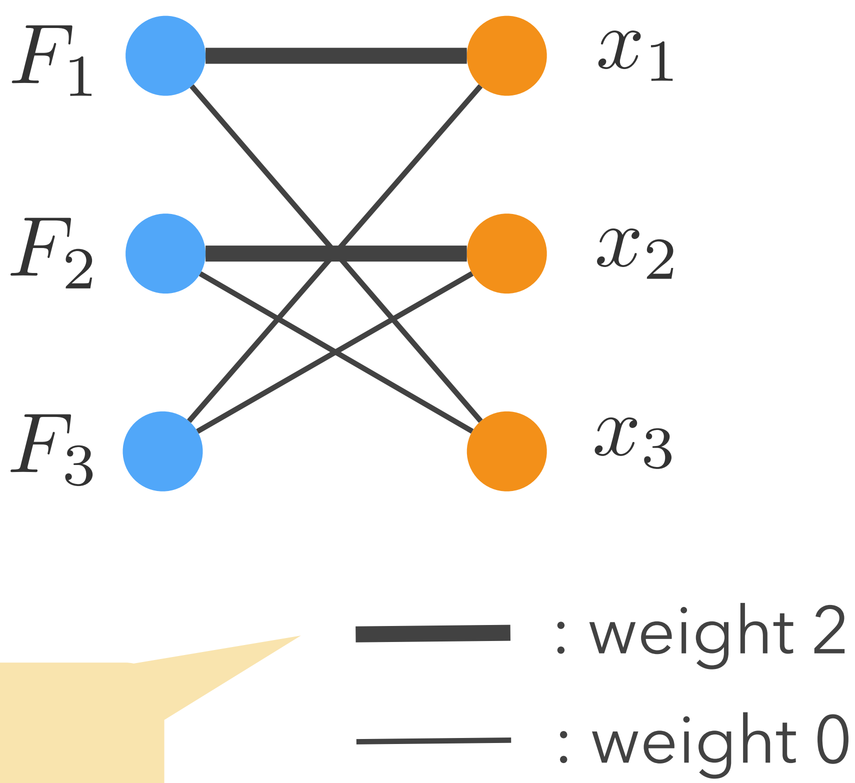
Mattsson–Söderlind's Index Reduction Algorithm



Pantelides' Algorithm

Construct a bipartite graph
representing the occurrence of variables in equations

$$\begin{cases} F_1 : \ddot{x}_1 + x_1 x_3 = 0 \\ F_2 : \ddot{x}_2 + x_2 x_3 - g = 0 \\ F_3 : x_1^2 + x_2^2 - L^2 = 0 \end{cases}$$



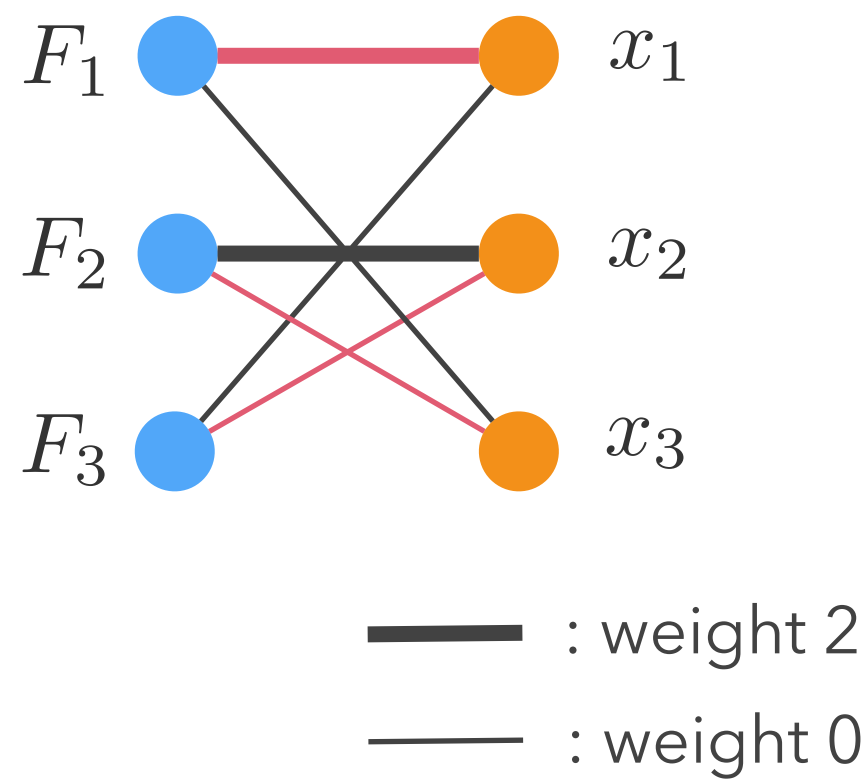
the weight $c_{i,j}$ of each edge (i, j)
represents the order of the differentiation

Pantelides' Algorithm

Find a maximum weight perfect matching

PRIMAL

$$
\begin{aligned}
\max \quad & \sum_{i \in R} \sum_{j \in C} c_{i,j} \xi_{i,j} \\
\text{s.t.} \quad & \sum_{j \in C} \xi_{i,j} = 1, \quad (i \in R) \\
& \sum_{i \in R} \xi_{i,j} = 1, \quad (j \in C) \\
& \xi_{i,j} \geq 0 \quad (i \in R,\ j \in C)
\end{aligned}
$$

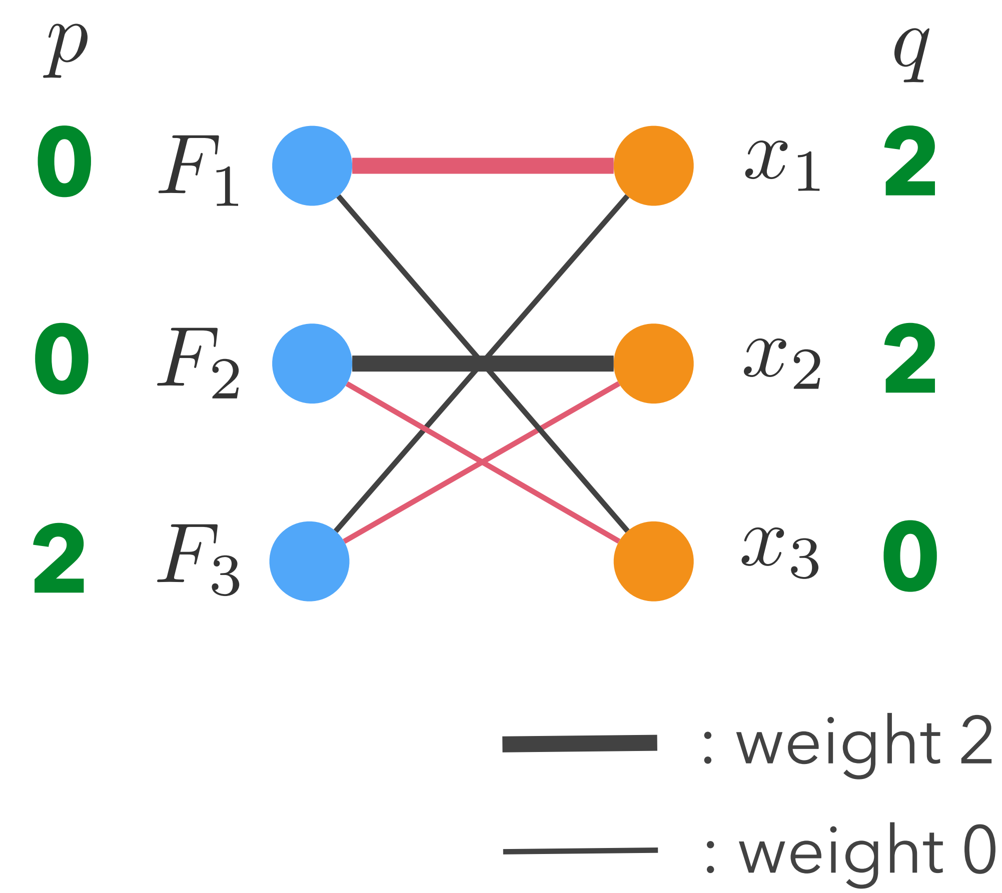


Pantelides' Algorithm

Find a maximum weight perfect matching
and its dual optimal solution (p, q)

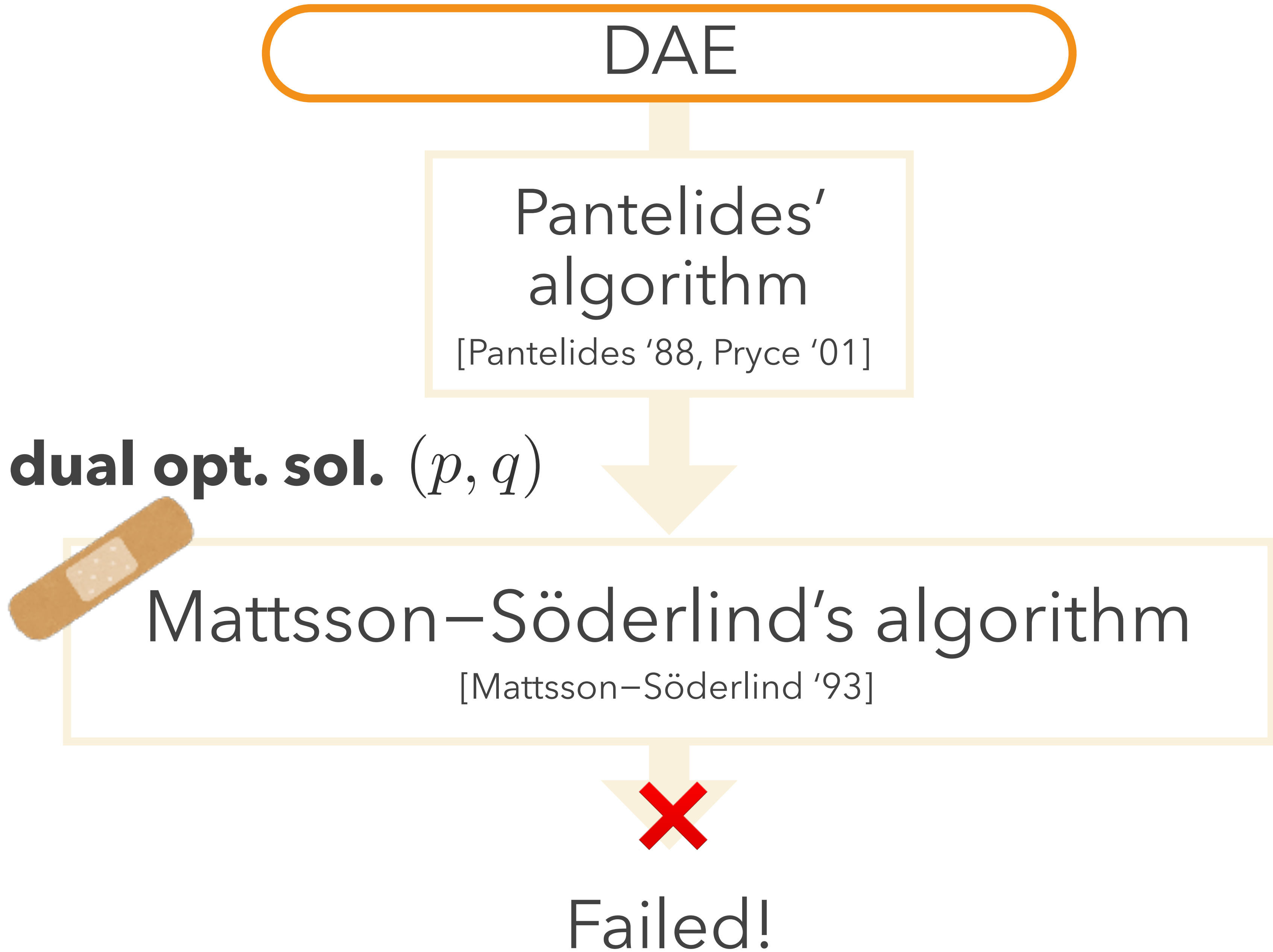
DUAL

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j \in C} q_j - \sum_{i \in R} p_i & \\
\text{s.t.} & q_j - p_i \geq c_{i,j}, & ((i,j) \in E(A)) \\
& p_i \in \mathbb{Z}_{\geq 0}, & (i \in R) \\
& q_j \in \mathbb{Z}_{\geq 0}. & (j \in C)
\end{array}
$$



→ the MS-alg. uses (p, q)

However...


DAE
Pantelides' algorithm
[Pantelides '88, Pryce '01]
dual opt. sol. (p, q)
Mattsson–Söderlind's algorithm
[Mattsson–Söderlind '93]
Failed!

Failure of the MS-algorithm

Why the MS-algorithm fails? Intuitively speaking…

Pantelides' algorithm [Pantelides '88, Pryce '01] uses structural information of DAEs and ignores numerical information.

→ Hidden numerical cancellations cause the failure.

When the MS-algorithm succeeds?

Σ-Jacobian

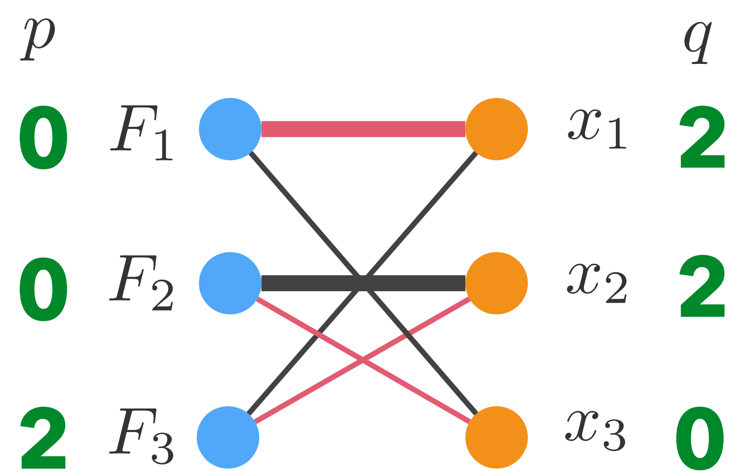
$$D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j}$$

(p, q) : dual opt. sol.

The MS-alg. succeeds if the **Σ-Jacobian** is nonsingular.

Example

$$\begin{cases} F_1 : \ddot{x}_1 + x_1 x_3 & = 0 \\ F_2 : \ddot{x}_2 + x_2 x_3 - g & = 0 \\ F_3 : x_1^2 + x_2^2 - L^2 & = 0 \end{cases}$$



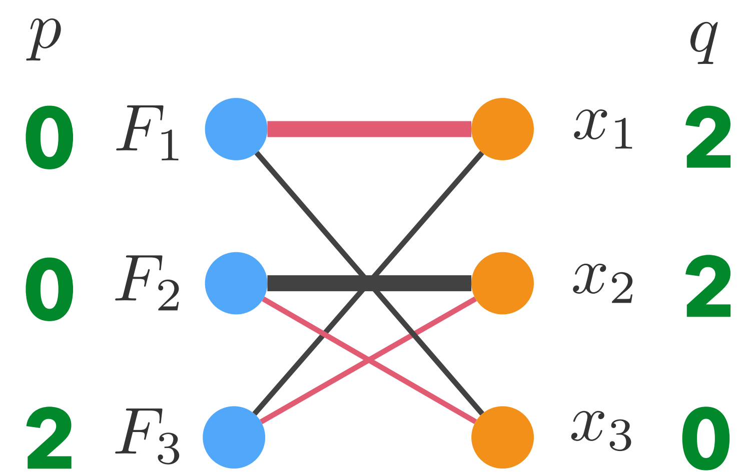
Σ-Jacobian

$$D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \frac{\partial F_1}{\partial \ddot{x}_1} & \frac{\partial F_1}{\partial \ddot{x}_2} & \frac{\partial F_1}{\partial x_3} \\ \frac{\partial \ddot{F}_2}{\partial \ddot{x}_1} & \frac{\partial F_2}{\partial \ddot{x}_2} & \frac{\partial F_2}{\partial x_3} \\ \frac{\partial \ddot{F}_3}{\partial \ddot{x}_1} & \frac{\partial \ddot{F}_3}{\partial \ddot{x}_2} & \frac{\partial \ddot{F}_3}{\partial x_3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_1 \\ 0 & 1 & x_2 \\ 2x_1 & 2x_2 & 0 \end{pmatrix}$$

$$\dot{F}_3 : 2\dot{x}_1 x_1 + 2\dot{x}_2 x_2$$
$$\ddot{F}_3 : 2\ddot{x}_1 x_1 + 2\dot{x}_1^2 + 2\ddot{x}_2 x_2 + 2\dot{x}_2^2$$

Example

$$\begin{cases} F_1 : \ddot{x}_1 + x_1 x_3 = 0 \\ F_2 : \ddot{x}_2 + x_2 x_3 - g = 0 \\ F_3 : x_1^2 + x_2^2 - L^2 = 0 \end{cases}$$



Σ-Jacobian

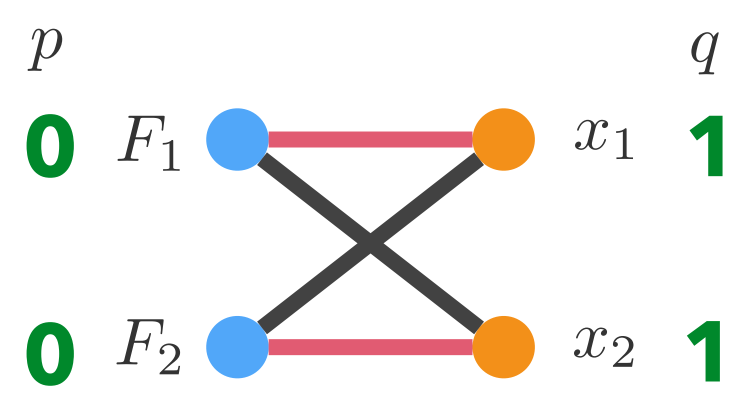
$$D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \frac{\partial F_1}{\partial \ddot{x}_1} & \frac{\partial F_1}{\partial \ddot{x}_2} & \frac{\partial F_1}{\partial x_3} \\ \frac{\partial \ddot{F}_2}{\partial \ddot{x}_1} & \frac{\partial \ddot{F}_2}{\partial \ddot{x}_2} & \frac{\partial F_2}{\partial x_3} \\ \frac{\partial \ddot{F}_3}{\partial \ddot{x}_1} & \frac{\partial \ddot{F}_3}{\partial \ddot{x}_2} & \frac{\partial \ddot{F}_3}{\partial x_3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_1 \\ 0 & 1 & x_2 \\ 2x_1 & 2x_2 & 0 \end{pmatrix}$$

$$\det D = -2x_1^2 - 2x_2^2 = -L^2 \neq 0$$

The MS-algorithm works!

Example

$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 \quad = 0 \end{cases}$$



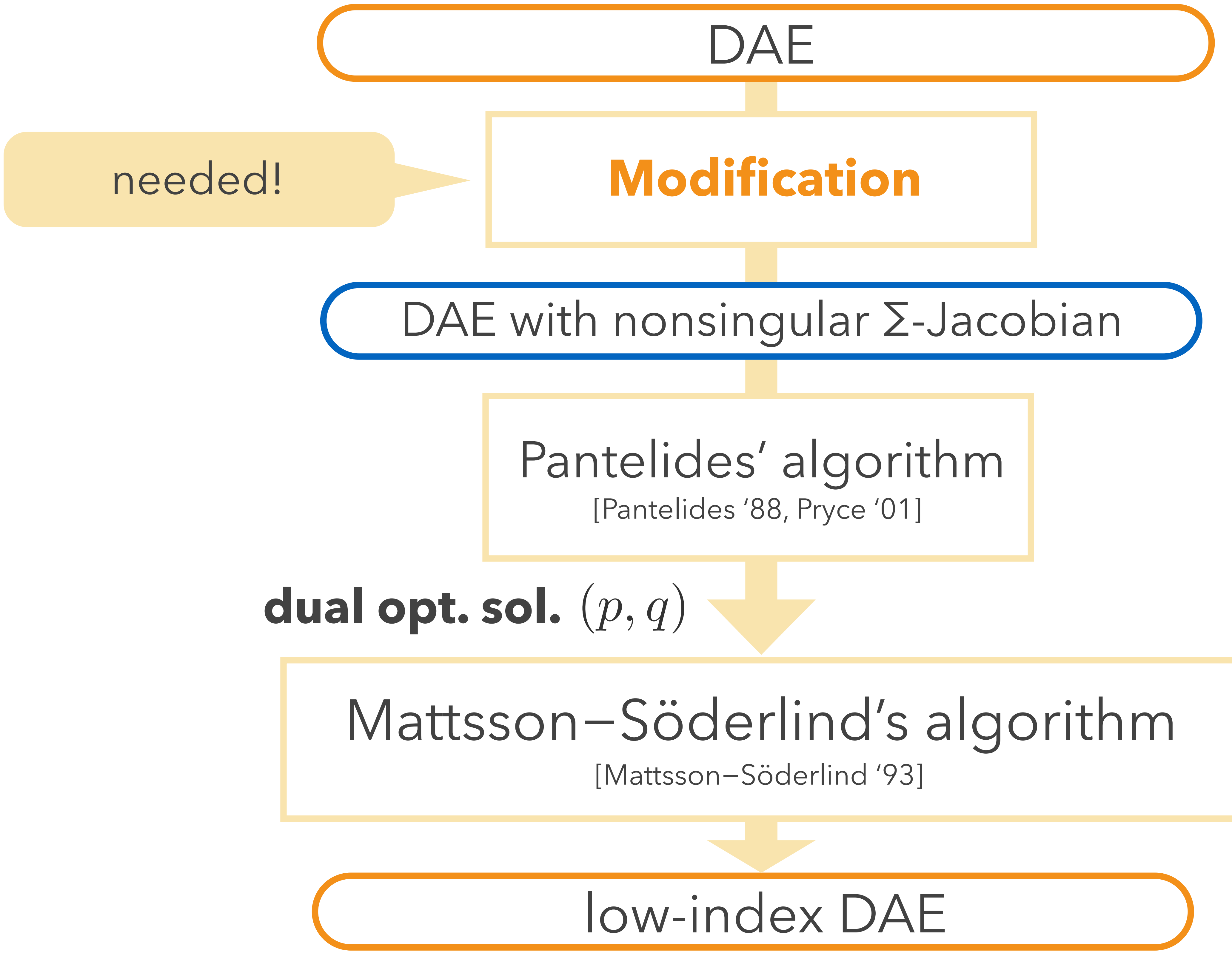
Σ-Jacobian

$$D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \frac{\partial F_1}{\partial \dot{x}_1} & \frac{\partial F_1}{\partial \dot{x}_2} \\ \frac{\partial F_2}{\partial \dot{x}_1} & \frac{\partial F_2}{\partial \dot{x}_2} \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$$

$$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$$
$$\beta = 3(\dot{x}_1 + \dot{x}_2)^2$$

The MS-algorithm does not work…

Preprocess for the MS-algorithm



Combinatorial Relaxation

[Murota '90 '95, Wu+ '13]

- The **combinatorial relaxation** was originally invented by [Murota '90] for the computation of the degree of the determinant of a polynomial matrix.

- [Wu+ '13] pointed out that the combinatorial relaxation can be used for the preprocessing of the MS-algorithm for linear DAEs with constant coefficients

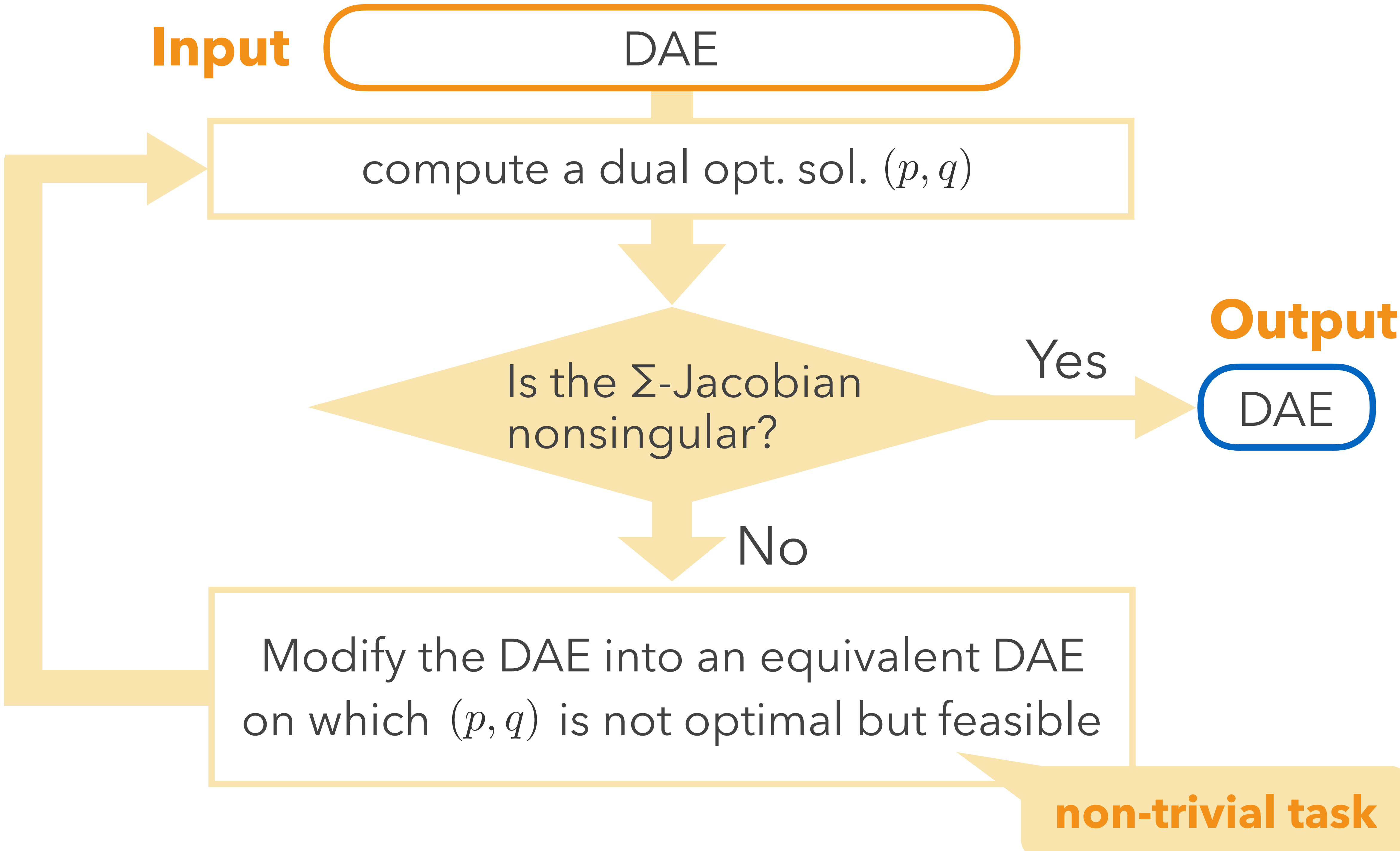
$$\sum_{k=0}^{l} A_k x^{(k)} = f(t)\,.$$

$A_0, \ldots, A_l \in \mathbb{R}^{n \times n}$

$f\colon \mathbb{R} \to \mathbb{R}^n$

Flowchart of the Combinatorial Relaxation

[Murota '90 '95, Wu+ '13]



Equivalent Condition

Thm (complementarity theorem) [Murota '90]

(p, q) : feasible dual sol. on a DAE

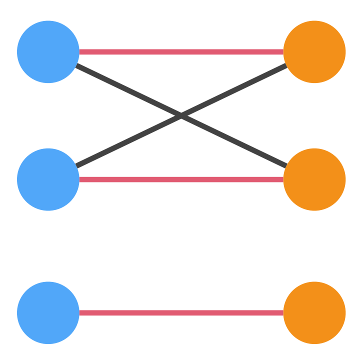
D : Σ-Jacobian w.r.t. (p, q)

$$(p, q) \text{ is optimal} \Leftrightarrow \text{t-rank}\, D = n$$

What is the term-rank of a matrix?

the max. size of a matching in the associated bipartite graph

$$D = \begin{pmatrix} \alpha & \beta & 0 \\ \alpha & \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



$$\text{t-rank}\, D = 3$$

Problem Formulation of Modification

Modify the DAE into an equivalent DAE on which (p, q) is not optimal but feasible

non-trivial task

Input

DAE $F(t, x, \dot{x}, \dots .x^{(l)}) = 0$

dual opt. sol. (p, q)

s.t. the Σ-Jacobian $D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j}$ is singular

Output

equivalent DAE $\bar{F}(t, x, \dot{x}, \dots, x^{(l)}) = 0$

s.t. the Σ-Jacobian $\bar{D}$ w.r.t. (p, q) has the t-rank less than n

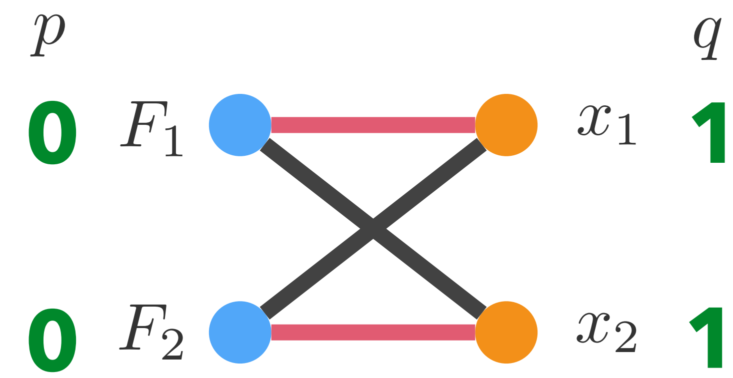
DAE Modification Algorithms

Algorithm	Target Class of DAEs	Modification	How many DAEs can be handled?
[Wu–Zeng–Cao '13]	Linear DAEs with constant coefficients	unimodular transformation by [Murota '90]	**all instances**
[Iwata–O.–Takamatsu '17]	Linear DAEs with mixed poly. matrices	unimodular transformation	**all instances**
LC-method [Tan–Nedialkov–Pryce '16]	Nonlinear DAEs	row operation	not so many
ES-method [Tan–Nedialkov–Pryce '16]	Nonlinear DAEs	new variable substitution	not so many
Proposed	Nonlinear DAEs	implicit function theorem	**many**

Modification for Linear DAEs

DAE
$$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : \dot{x}_1 + \dot{x}_2 \qquad = 0 \end{cases}$$

1

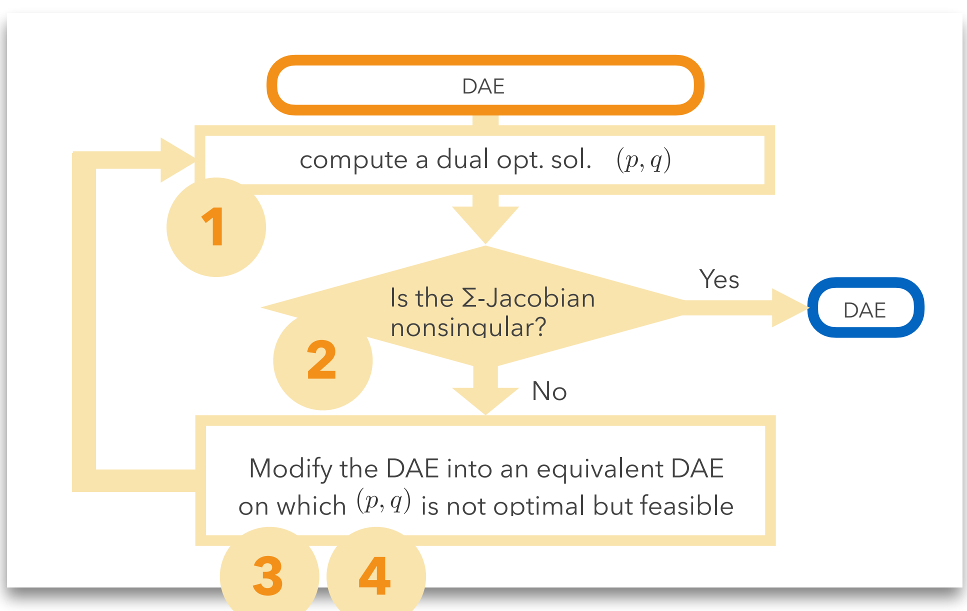


2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$: singular

3 Transform $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ into $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ by a row operation

t-rank = 2

t-rank = 1

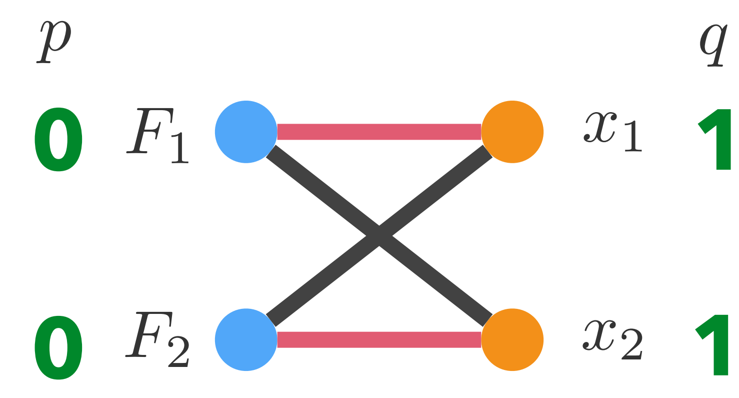


Modification for Linear DAEs

[Murota '90, Wu+ '13]

DAE $\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : \dot{x}_1 + \dot{x}_2 \qquad\;\; = 0 \end{cases}$

1



2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$: singular

3 Transform $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ into $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ by a row operation

4 Perform the "corresponding" transformation on the DAE

$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : \dot{x}_1 + \dot{x}_2 \qquad\;\; = 0 \end{cases}$ subtract $\Longrightarrow$ $\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ \bar{F}_2 : \qquad\qquad\; - x_2 = 0 \end{cases}$

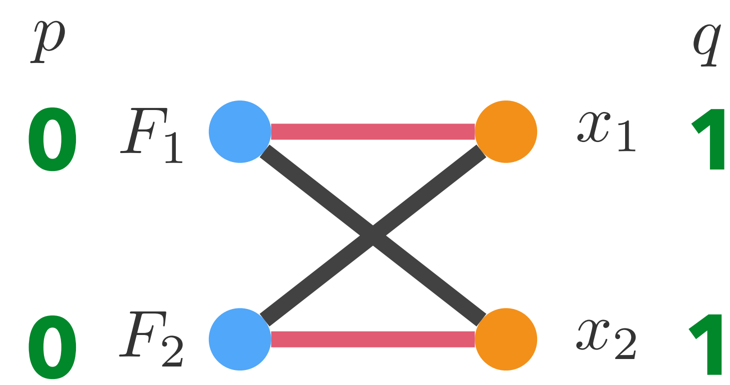
Modification for Nonlinear DAEs by LC-method

[Tan+ '16]

DAE

$$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : x_1\dot{x}_1 + x_1\dot{x}_2 = 0 \end{cases}$$

1



2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} 1 & 1 \\ x_1 & x_1 \end{pmatrix}$: singular

3 Transform $\begin{pmatrix} 1 & 1 \\ x_1 & x_1 \end{pmatrix}$ into $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ by a row operation

4 Perform the "corresponding" transformation on the DAE

$$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : x_1\dot{x}_1 + x_1\dot{x}_2 = 0 \end{cases}$$

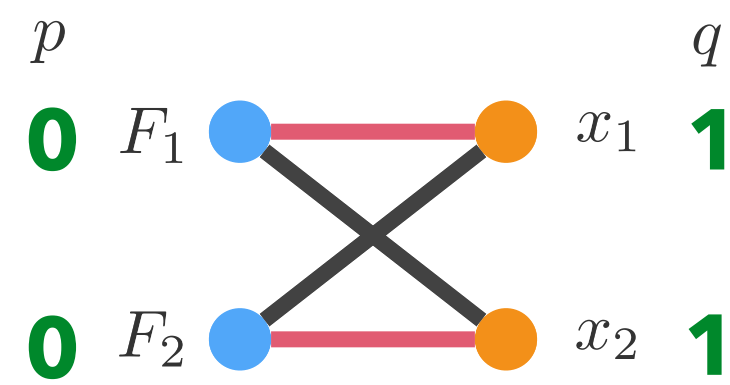
subtract
1st row × x_1

$$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ \bar{F}_2 : \ -x_1 x_2 = 0 \end{cases}$$

Modification for Nonlinear DAEs by LC-method

DAE $\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : x_1\dot{x}_1 + x_1\dot{x}_2 = 0 \end{cases}$

1



2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right) = \begin{pmatrix} 1 & 1 \\ x_1 & x_1 \end{pmatrix}$: singular

3 Transform $\begin{pmatrix} 1 & 1 \\ x_1 & x_1 \end{pmatrix}$ into

> replace an equation by
> a linear combination of equations
> LC = "Linear Combination"

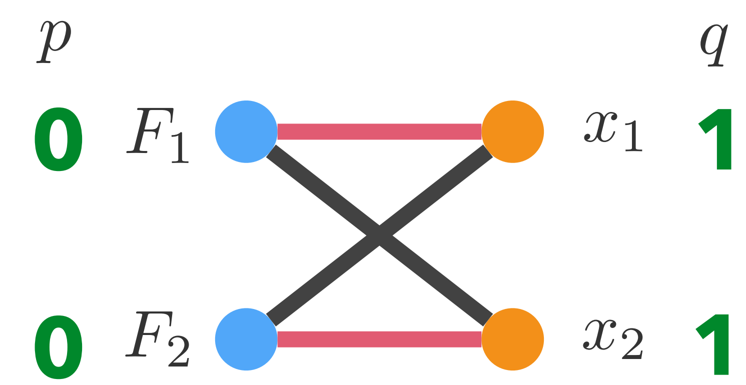
4 Perform the "corresponding" transformation on the DAE

$\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ F_2 : x_1\dot{x}_1 + x_1\dot{x}_2 = 0 \end{cases}$ subtract 1st row × x_1 → $\begin{cases} F_1 : \dot{x}_1 + \dot{x}_2 + x_2 = 0 \\ \bar{F}_2 : \ - x_1 x_2 = 0 \end{cases}$

Failure Case of LC-method

DAE
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

1



2 Σ-Jacobian: $D = \left(\frac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$: singular

$$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$$
$$\beta = 3(\dot{x}_1 + \dot{x}_2)^2$$

3 Transform $\begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$ into $\begin{pmatrix} \alpha & \alpha \\ 0 & 0 \end{pmatrix}$ by a row operation

$\begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$ subtract 1st row × β/α → $\begin{pmatrix} \alpha & \alpha \\ 0 & 0 \end{pmatrix}$

Failure Case of LC-method

DAE
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

4 Perform the "corresponding" transformation on the DAE

$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

subtract
1st row × β/α

$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$
$\beta = 3(\dot{x}_1 + \dot{x}_2)^2$

$$\rightarrow \begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ \bar{F}_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 - \frac{\beta}{\alpha}\{\exp(\dot{x}_1 + \dot{x}_2) + x_1\} = 0 \end{cases}$$

$\dot{x}_1$ and $\dot{x}_2$ do not disappear…

Σ-Jacobian $D = \begin{pmatrix} \alpha & \alpha \\ \gamma & \gamma \end{pmatrix}$ still has t-rank = 2. **FAILED!**

Conversion to DAE for which Existing Methods Fails

$F(t, x, \dot{x}) = 0$: DAE to which the MS-algorithm is not applicable

$\varphi\colon\ \mathbb{R}^n \to \mathbb{R}^n$: nonlinear diffeomorphism
$\psi\colon\ \mathbb{R}^n \to \mathbb{R}^n$: nonlinear diffeomorphism with $\psi(v) = 0 \iff v = 0$

Coordinate change

- variable: $x = \varphi(u)$, $\dot{x} = \varphi'(u)\dot{u}$
- equation: $F \mapsto \psi(F)$

$$F(t, x, \dot{x}) = 0 \iff \psi(F(t, \varphi(u), \varphi'(u)\dot{u})) = 0$$

$$=: G(t, u, \dot{u})$$

Then, $G(t, u, \dot{u}) = 0$ cannot be dealt with by existing methods.

Contents

1 Differential-Algebraic Equations

2 Combinatorial Relaxation

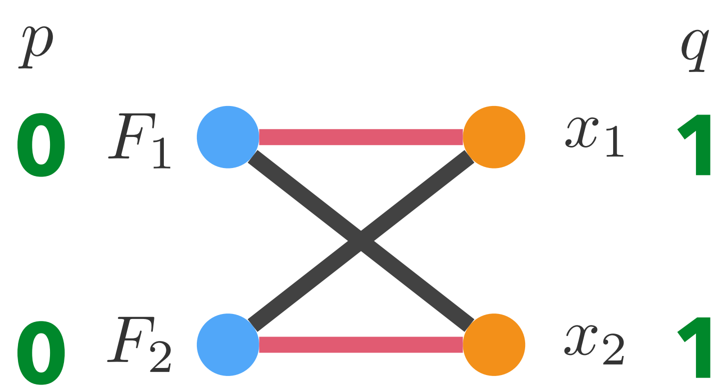
3 Proposed Algorithm

Our Algorithm

DAE

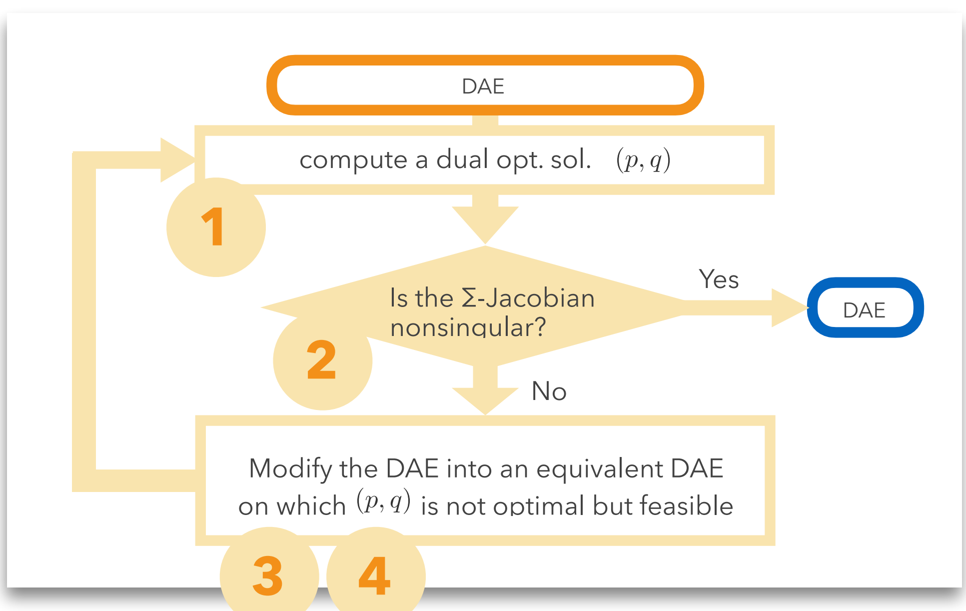
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

1



$$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$$
$$\beta = 3(\dot{x}_1 + \dot{x}_2)^2$$

2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$: singular



Our Algorithm

3 Find a row subset I, a column subset J and a row $r \notin I$

submatrix indexed by (I, J)

s.t.
- $|I| = |J|$
- $D[I, J]$ is nonsingular
- $D[I \cup \{r\}, C]$ is not full-row rank
- $p_r \leq p_i$ for all $i \in I$

$$\begin{array}{cc} & J \\ \begin{array}{c} I \\ r \end{array} & \begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix} \end{array}$$

: $D[I, J]$

Namely, the r-th row can be eliminated by $D[I, J]$.

Our Algorithm

(this is the simplified version where all $p_i = 0$)

4 Solve the eq. system $\{F_i = 0\}_{i \in I}$ for variables $\left\{x_j^{(q_j)}\right\}_{j \in J}$ and substitute it into $F_r = 0$.

this operation is guaranteed by the **Implicit Function Theorem**

DAE
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

Solve the 1st eq. $\exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0$ for $\dot{x}_1$

$$\Rightarrow \quad \dot{x}_1 = -\dot{x}_2 + \log(-x_1)$$

Substitute $\dot{x}_1 = \log(-x_1) - \dot{x}_2$ into the 2nd eq. $(\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0$

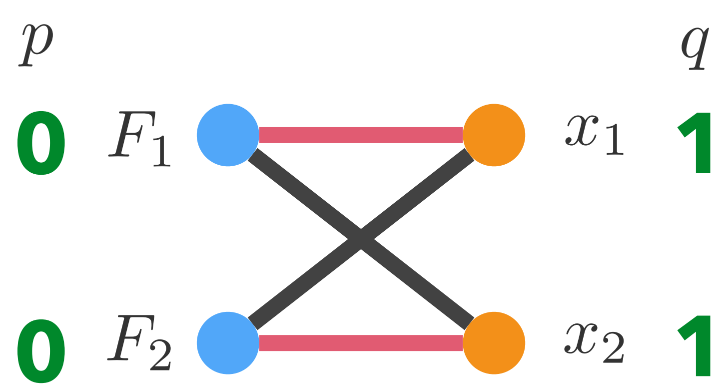
$$\Rightarrow \quad \begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ \bar{F}_2 : (\log(-x_1))^3 - x_2 = 0 \end{cases}$$

$\dot{x}_2$ is cancelled out!

Example

DAE
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ F_2 : (\dot{x}_1 + \dot{x}_2)^3 - x_2 = 0 \end{cases}$$

1



$$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$$
$$\beta = 3(\dot{x}_1 + \dot{x}_2)^2$$

2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \alpha & \alpha \\ \beta & \beta \end{pmatrix}$: singular

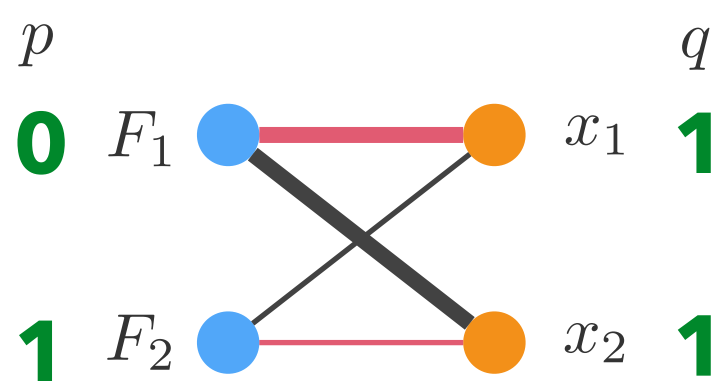
3 Modify the DAE into $\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ \bar{F}_2 : (\log(-x_1))^3 - x_2 = 0 \end{cases}$

Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \alpha & \alpha \\ 0 & 0 \end{pmatrix}$: t-rank < 2

Example

DAE
$$\begin{cases} F_1 : \exp(\dot{x}_1 + \dot{x}_2) + x_1 = 0 \\ \bar{F}_2 : (\log(-x_1))^3 - x_2 = 0 \end{cases}$$

1



$$\alpha = \exp(\dot{x}_1 + \dot{x}_2)$$

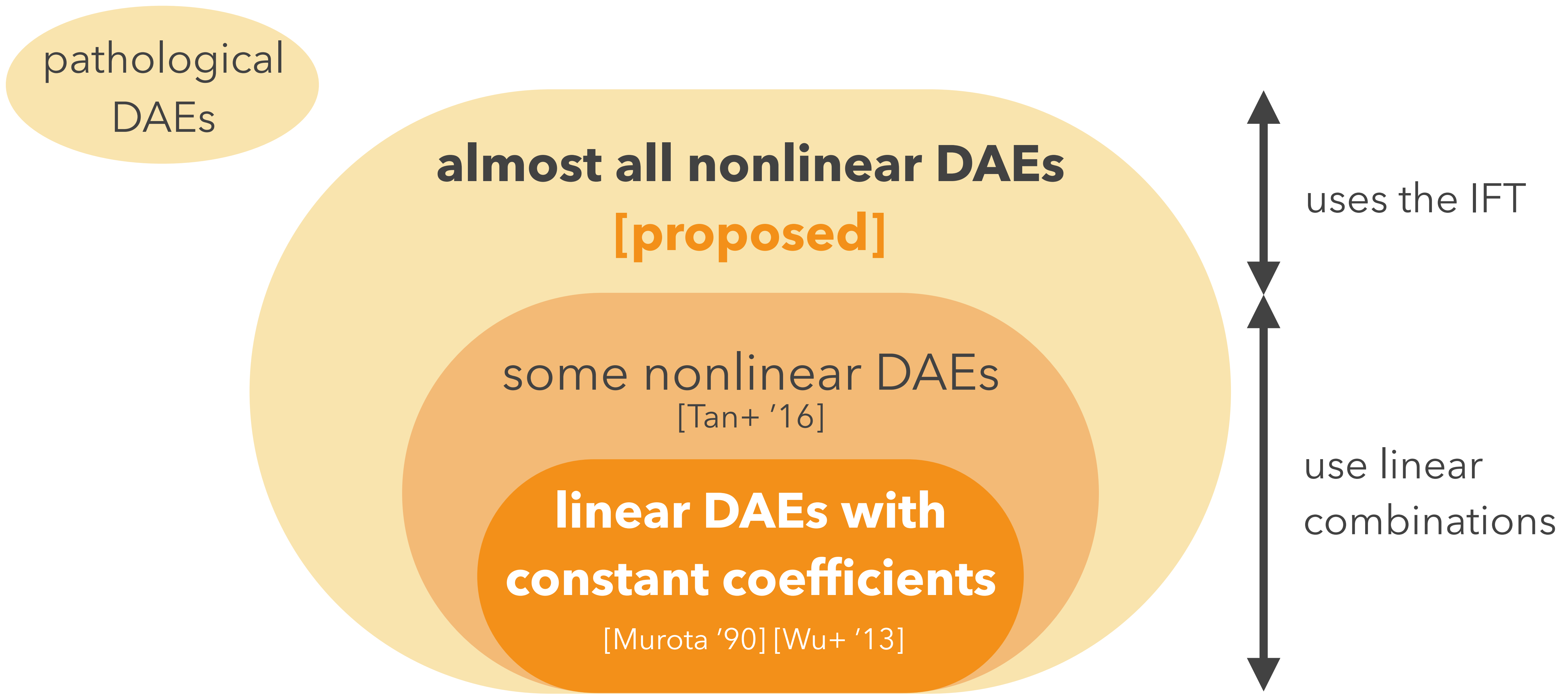
2 Σ-Jacobian: $D = \left(\dfrac{\partial F_i^{(p_i)}}{\partial x_j^{(q_j)}} \right)_{i,j} = \begin{pmatrix} \alpha & \alpha \\ -3x_1^{-1}(\log(-x_1))^2 & -1 \end{pmatrix}$

: generally nonsingular

→ We can reduce the index using the MS-algorithm!

Position of Our Algorithm

a first combinatorial relaxation algorithm
fully addresses nonlinear DAEs



Difficulties in Implementation

😩 The algorithm requires a feasible initial value…

😩 Symbolic operations are time consuming…

😩 Obtaining explicit functions is a difficult task…

Implementation Strategies

😩 The algorithm requires a feasible initial value…

➡ Execute the algorithm symbolically and obtain a feasible initial value afterward! ✔

😩 Symbolic operations are time consuming…

➡ Determine whether a mathematical formulation is identically zero or not by substituting random numbers! ✔

😩 Obtaining explicit functions is a difficult task…

➡ Represent a modified equation by a "system of equations" without using actual explicit functions! ✔

Implementation Strategy 3

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

Situation

Solve the 1st eq. for $\dot{x}_1$ → $\dot{x}_1 = \varphi(t, x_1, x_2, x_3, \dot{x}_2, \dot{x}_3)$

Substitute φ into $\dot{x}_1$ in F_2

→
$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, \varphi(t, x_1, x_2, x_3, \dot{x}_2, \dot{x}_3), \dot{x}_2, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

Then, $\dot{x}_2$s in the 2nd eq. are cancelled out.

Implementation Strategy 3

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

We memorize this modification as

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

Implicit Function

$$F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$$

y : intervening variable $(= \dot{x}_1)$

a : constant $(= \dot{x}_2)$

The value of y can be obtained by solving $F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$ numerically using the Newton–Raphson method.

Nested Case

DAE
$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ \bar{F}_2(t, x_1, x_2, x_3, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

$$\bar{F}_2 = F_2(t, x_1, x_2, x_3, \varphi(t, x_1, x_2, x_3, \dot{x}_2, \dot{x}_3), \dot{x}_2, \dot{x}_3)$$

Situation (cont'd)

Solve the 2nd eq. for $\dot{x}_3$ → $\dot{x}_3 = \psi(t, x_1, x_2, x_3)$

Substitute ψ into $\dot{x}_3$ in the 3rd eq.

$$\rightarrow \begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ \bar{F}_2(t, x_1, x_2, x_3, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \psi(t, x_1, x_2, x_3)) = 0 \end{cases}$$

Then, x_1s in the 3rd eq. are cancelled out.

Nested Case

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0 \\ F_3(t, x_1, x_2, x_3, \dot{x}_3) = 0 \end{cases}$$

Implicit Function

$$F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$$

modify

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0 \\ F_3(t, b, x_2, x_3, z) = 0 \end{cases}$$

Implicit Function

$$F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$$

$$F_1(t, b, x_2, x_3, w, a, z) = 0$$

$$F_2(t, b, x_2, x_3, w, a, z) = 0$$

y, w : intervening variables $(= \dot{x}_1)$

z : intervening variable $(= \dot{x}_3)$

a : constant $(= \dot{x}_2)$

b : constant $(= x_1)$

On Numerical Methods

On The implicit Euler method applied to $F(t, x, \dot{x}) = 0$

we solve $F\left(t_k, x^{(k)}, \frac{x^{(k)} - x^{(k-1)}}{h}\right) = 0$ for $x^{(k)}$ in every step

→ On the Newton–Raphson method

we need to evaluate F and its Jacobian in each iteration

→ On the evaluation of F =

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0 \\ F_3(t, b, x_2, x_3, z) = 0 \end{cases}$$

Implicit Function

$$F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$$
$$F_1(t, b, x_2, x_3, w, a, z) = 0$$
$$F_2(t, b, x_2, x_3, w, a, z) = 0$$

we need to run Newton–Raphson method

doubly nested Newon–Raphson method?

On Numerical Methods

DAE

$$\begin{cases} F_1(t, x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) = 0 \\ F_2(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0 \\ F_3(t, b, x_2, x_3, z) = 0 \end{cases}$$

Implicit Function

$$F_1(t, x_1, x_2, x_3, y, a, \dot{x}_3) = 0$$
$$F_1(t, b, x_2, x_3, w, a, z) = 0$$
$$F_2(t, b, x_2, x_3, w, a, z) = 0$$

$$F\left(t_k, x^{(k)}, \frac{x^{(k)} - x^{(k-1)}}{h}\right) = 0$$

We solve

$$\begin{cases} F_1(t_k, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, (x_1^{(k)} - x_1^{(k-1)})/h, (x_2^{(k)} - x_2^{(k-1)})/h, (x_3^{(k)} - x_3^{(k-1)})/h) = 0 \\ F_2(t_k, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, y, a, (x_3^{(k)} - x_3^{(k-1)})/h) = 0 \\ F_3(t_k, b, x_2^{(k)}, x_3^{(k)}, z) = 0 \\ F_1(t_k, x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, y, a, (x_3^{(k)} - x_3^{(k-1)})/h) = 0 \\ F_1(t_k, b, x_2^{(k)}, x_3^{(k)}, w, a, z) = 0 \\ F_2(t_k, b, x_2^{(k)}, x_3^{(k)}, w, a, z) = 0 \end{cases}$$

for $x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, y, z, w$

Only 1 call of the Newton–Raphson method!

Open Problems

What is a class of DAEs for which my algorithm works?

I want to mathematically clarify the “pathological” DAEs.

Is there a practical high-index DAE that can be dealt with by my algorithm for the first time?

While such DAEs can be constructed artificially,
I want to find a DAE naturally arising from dynamical systems.

Index reduction via algorithmic differentiation?

Can I exploit the computational graph representation of functions?

Acknowledgments

- Satoru Iwata, University of Tokyo
- Mizuyo Takamatsu, Chuo University

This work was supported by

- JST CREST Grant Number JPMJCR14D2
  "Developing Optimal Modeling Methods for Large-Scale Complex Systems" Team
- Grant-in-Aid for JSPS Research Fellow Grant Number JP18J22141