Mechatronic systems simulation (146029) - MOD2

A Practical Approach

Enrico Bertolazzi

Copyright © 2025 Enrico Bertolazzi

PUBLISHED BY PUBLISHER

BOOK-WEBSITE.COM

Licensed under the Creative Commons Attribution-NonCommercial 4.0 License (the "License"). You may not use this file except in compliance with the License. You may obtain a copy of the License at https://creativecommons.org/licenses/by-nc-sa/4.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

First printing, May 2025



Contents

11

I Introduction

1 Ordinary Differential Equations (ODEs) 17

1.1 Application of ODEs 17
1.1.1 Physics 17
1.1.2 Charged Particle in an Electromagnetic Field 17
1.1.3 Chemistry 18
1.1.4 Engineering 18
1.1.5 Aerospace Engineering 19
1.1.6 Biology 19
1.1.7 Epidemiology 19
1.1.8 Economics 19
1.1.9 Electronics 20
1.1.10 Medicine 20

1.2 Classification of ODEs 21
1.2.1 Order of an ODE 21
1.2.2 Linearity of an ODE 21
1.2.3 Autonomous vs. Non-autonomous ODEs 21

1.3 Solution Techniques 21

2 Some properties of ODEs 23

2.1 Examples of single ODE 23

2.2 Examples of Systems of ODEs 24

2.3 Vector Form of ODEs 24
2.3.1 System of ODEs in Scalar Form 25
2.3.2 Compact Vector Form 25
2.3.3 Matrix Form of a Linear System 26

2.4 General ODE and properties . . . 26
2.5 Well-Posedness . . . 28
2.5.1 Definition of a Well-Posed Initial Value Problem . . . 28
2.5.2 Existence of Solutions . . . 28
2.5.3 Peano's Continuation Theorem . . . 28
2.6 Uniqueness of Solutions . . . 28
2.6.1 Lipschitz Condition . . . 29
2.6.2 Grönwall's Lemma . . . 29
2.7 Existence and Uniqueness . . . 29
2.7.1 Picard-Picard Theorem . . . 29
2.8 Example of ODE with Existence but No Uniqueness . . . 30
2.8.1 Existence of the Solution . . . 30
2.8.2 No Uniqueness of the Solution . . . 30
2.8.3 Multiple Solutions . . . 30
2.8.4 Conclusion . . . 31

3 Numerical Solutions for Initial Value Problems . . . 33
3.1 Euler Method . . . 34
3.1.1 General Form of the Euler Method . . . 34
3.2 Heun and Collatz Methods . . . 36
3.2.1 Heun's Method . . . 36
3.2.2 Collatz Method . . . 36
3.2.3 Comparison of Methods . . . 37

4 Runge-Kutta Methods . . . 39
4.1 Explicit Runge-Kutta Methods . . . 39
4.1.1 Butcher Tableau . . . 39
4.2 Implicit Runge-Kutta Methods . . . 43

II Differential-Algebraic Equations (DAEs) and Differential Index

5 DAEs Forms . . . 51
5.1 Semi-Explicit Form of DAEs . . . 51
5.2 Index-0 DAEs are ODEs . . . 52
5.3 Index-1 DAEs in Semi-Explicit Form . . . 52
5.4 The Implicit Function Theorem . . . 53
5.4.1 Illustrative Example: A Simple Index-1 DAE . . . 53
DAE of index 2 . . . 53
5.5 DAE of index 3 . . . 54
5.6 DAE in semi-esplicit form . . . 55

6 The Index of a DAE . . . 57
6.1 Differential Index Definition . . . 57
6.2 Example 1 . . . 58
6.2.1 Index-2 DAE . . . 58

6.3 Example 2 . . . 58
6.3.1 Time-Varying Coefficient DAE . . . 58
6.4 Example 3 . . . 59
6.4.1 Index-1 DAE . . . 59
Summary . . . 59

III **Linear DAE**

7 Linear DAE examples and properties . . . 65
7.1 Separation of algebraic and differential part . . . 69
7.1.1 Solution of $z_2'(t) = Jz_2(t) + g_2(t)$ . . . 70
7.1.2 Solution of $Nz_1'(t) = z_1(t) + g_1(t)$ . . . 70

8 Gaussian Elimination and Echelon form . . . 73
8.1 Gaussian Elimination . . . 73
8.2 Echelon Form . . . 78
8.2.1 Elementary Operations and Matrix Representation . . . 78
Step-by-Step Process . . . 78
8.2.2 Example of reduction to echelon form . . . 79
8.2.3 Example of reduction to echelen form for a singular matrix . . . 79
8.2.4 Summary of Echelon Form Reduction Algorithm . . . 80

9 Gauss-based Method for separating the algebraic part from the differential part . . . 83
9.1 Formulation of the DAE . . . 83
9.2 Change of Coordinates . . . 83
9.2.1 Special Structure of the Matrix . . . 84
9.2.2 Decoupling of Components . . . 84
9.2.3 Differentiation of Algebraic Constraints . . . 84
9.3 Rank Increase Observation . . . 84

IV **Nonlinear DAE**

10 DAE from Lagrangian mechanics . . . 93
10.1 Overview of Lagrangian Mechanics . . . 93
10.1.1 Generalized Coordinates and Energies . . . 93
10.1.2 The Lagrangian Function . . . 93
10.1.3 Euler-Lagrange Equations . . . 94
10.2 The Minimal Action Principle in Classical Mechanics . . . 95
10.2.1 Principle of Stationary Action . . . 95
10.2.2 Interpretation of the Action Functional . . . 96
10.2.3 Physical Implications and Broader Context . . . 96
10.3 The Calculus of Variations and the Stationary Action Principle . . . 97
10.3.1 The Variational Operator: A Simple Example . . . 98

10.4 Consequences of the Stationary Action Principle . . . 98
10.4.1 Integration by Parts and the Euler-Lagrange Equations . . . 99
10.4.2 Derivation of the Euler-Lagrange Equations . . . 100
10.5 Principle of Minimal Action and Coordinate Independence . . . 101
10.5.1 Coordinate Invariance of the Equations of Motion . . . 102
10.5.2 Beyond Conservative Systems: Generalized Forces . . . 102
10.6 Mechanical Equations of Motion . . . 103
10.6.1 Kinetic Energy Representation . . . 103
10.6.2 Deriving the Equations of Motion . . . 103
10.7 Constrained Systems and Lagrange Multipliers . . . 105

11 Numerical Integration of DAE . . . 109
11.1 Reduction to ODE . . . 110
11.2 The drift effect . . . 112

V Index Reduction

12 Index reduction of a Linear DAE . . . 119
12.1 Steps to Compute the Index . . . 119
12.2 Practical construction of Q_a and Q_d . . . 121
12.2.1 Simplification . . . 121
12.4 Example 1 . . . 122
12.4.1 Index Reduction of a Linear DAE . . . 122
12.4.2 First reduction . . . 122
12.3 Index Reduction Algorithm for linear DAEs . . . 123
12.4.3 Second Reduction . . . 124
12.5 Example 2 . . . 125
12.6 Example 3 . . . 128

13 Index reduction of a NonLinear DAE . . . 131
13.1 General DAE . . . 131
13.2 Semi-Explicit Form of DAEs . . . 131
13.3 Matrix Representation . . . 132
13.4 Steps to Compute the Index of a nonlinear DAE . . . 132

14 Index Reduction of DAEs Using Dummy Variables . . . 135
14.1 Introduction . . . 135
14.2 System Requirements . . . 135
14.3 Preprocessing . . . 136
14.4 Index reduction algorithm . . . 137

15 Linear algebra tools used in the index reduction process . . . 141
15.1 Index Reduction through Kernel Computation . . . 141
15.1.1 Kernel computation methods . . . 143

16 Examples of index reduction . . . 145
16.1 Example 1 . . . 145
16.1.1 First reduction . . . 145
16.1.2 Second Reduction . . . 146
16.2 Example 2 . . . 147
16.3 Example 3 . . . 150

VI Numerical Method for DAE

17 Numerical methods for linear DAE . . . 157
17.1 Consistent initial values . . . 157
17.2 Explicit Euler . . . 158
17.3 Implicit Euler . . . 158
17.4 Numerical result . . . 158
17.4.1 Crank-Nicolson . . . 158
17.5 Damped stabilization (Baumgarte) . . . 160
17.6 Numerical result . . . 161

18 Baumgarte's stabilization . . . 163
18.1 Differential-Algebraic Equations (DAEs) in Mechanics . . . 163
18.1.1 Notation and Setup . . . 163
18.1.2 Reformulating as a DAE . . . 163
18.2 Numerical Solution of DAEs . . . 164
18.2.1 Time Discretization . . . 164
18.2.2 Computing p'' and λ . . . 164
18.2.3 Numerical Integration Algorithm . . . 165
18.2.4 Handling Constraint Drift . . . 165
18.3 Example: Pendulum in Redundant Coordinates . . . 165
18.3.1 System Description . . . 165
18.3.2 Equations of Motion . . . 166
18.4 Numerical Solution . . . 166
18.4.1 Time Discretization . . . 167

19 Baumgarte's stabilization . . . 169
19.0.1 Baumgarte details . . . 171
19.0.2 Modified Linear System . . . 174
19.1 Pendulum DAE reduced to first order . . . 174
19.2 Integration of the DAE without stabilization . . . 176
19.2.1 Numerical Results . . . 181
19.3 Baumgarte stabilization . . . 181
19.3.1 Results with Crank-Nicolson . . . 181

20 Projection method . . . 183
20.1 Projection Methods in general . . . 184
20.1.1 Key Principles of the Projection Method . . . 184

20.1.2 Requirements for the Projection Method . . . 185
20.2 Mathematical Formulation . . . 185
20.3 Numerical Integration with Projection . . . 186
Step 1: ODE Integration . . . 186
Step 2: Projection onto the Constraint Manifold . . . 186
20.4 Generic Minimization Problem with Constraints . . . 186
Solution Using Lagrange Multipliers . . . 187
Linearization of the Minimization Problem . . . 187

21 Projection Algorithm for index 1,2 and 3 . . . 189
21.1 Index-1 DAE Projection Algorithm . . . 189
21.1.1 Projection Phase . . . 190
21.2 Index 2 DAE projection algorithm . . . 191
21.2.1 Constraint Differentiation . . . 192
21.2.2 Numerical Solution . . . 192
21.3 Index 3 DAE projection algorithm . . . 195

22 Projection with implicit numerical scheme . . . 199
22.1 Index-1 DAE . . . 199
22.1.1 Advancing the ODE: Explicit Methods . . . 199
22.1.2 Advancing the ODE: Implicit Methods . . . 199
22.1.3 RHS Jacobian . . . 200
22.1.4 Practical Computation of the Jacobian . . . 200
22.2 Index-3 DAE Special Case . . . 200

23 Coordinate Partitioning for Pendulum in cartesian coordinates . 203
23.1 Step 1 choose independent coordinates . . . 203
23.2 Step 2a propagate constraints . . . 203
23.3 Step 2b propagate constraints . . . 204
23.4 Step 2c propagate constraints . . . 204
23.5 Step 2d propagate constraints . . . 204
23.6 Collatz . . . 204
23.7 Runge-Kutta 4th order . . . 205

VII Examples

24 Example 1 . . . 211
24.1 Index reduction . . . 211
24.1.1 First reduction of the index by one . . . 211
24.1.2 Second reduction of the index by one . . . 213
24.1.3 Third reduction of the index by one . . . 213
24.1.4 DAE reduced to an ODE . . . 215

24.2 Index reduction using dummy variables . . . 215
24.2.1 Variable classification . . . 216
24.2.2 Index Reduction Procedure . . . 216
24.2.3 DAE reduced to ODE . . . 218
24.2.4 Baumgarte stabilization . . . 218
24.2.5 Jacobians . . . 218

25 Example 2 . . . 221
25.1 3D Pendulum . . . 221
25.2 Index reduction . . . 222
25.3 Index reduction using dummy variable . . . 224
25.4 Stabilization of the DAE System via Hidden Constraints . . . 225

26 Example 3 . . . 229
26.1 Index reduction . . . 229
26.1.1 First reduction of the index by one . . . 230
26.1.2 Second reduction of the index by one . . . 230
26.1.3 Third reduction of the index by one . . . 231
26.1.4 Fourth reduction of the index by one . . . 231
26.1.5 Fifth reduction of the index by one . . . 232
26.1.6 Sixth reduction of the index by one . . . 233
26.1.7 DAE reduced to an ODE . . . 233
26.2 Index reduction using dummy variable . . . 234

27 Example 4 . . . 237
27.1 Double Pendulum . . . 237
27.2 Index reduction . . . 239
27.2.1 First reduction of the index by one . . . 240
27.2.2 Second reduction of the index by one . . . 240
27.3 Index reduction using dummy variable . . . 244
27.3.1 The DAE of index 1 . . . 248
27.3.2 Baumgarte Stabilization . . . 248

28 Example 5 . . . 251
28.1 A Constrained Van der Pol System and a Dummy-Variable Reformulation 251
28.2 Formulation as a Stiff DAE . . . 251
28.2.1 Dummy-Variable Reformulation . . . 251
28.2.2 Index Reduction Procedure . . . 252
28.2.3 The ODE . . . 252
28.3 Baumgarte Stabilization . . . 253
28.4 Numerical Examples . . . 254

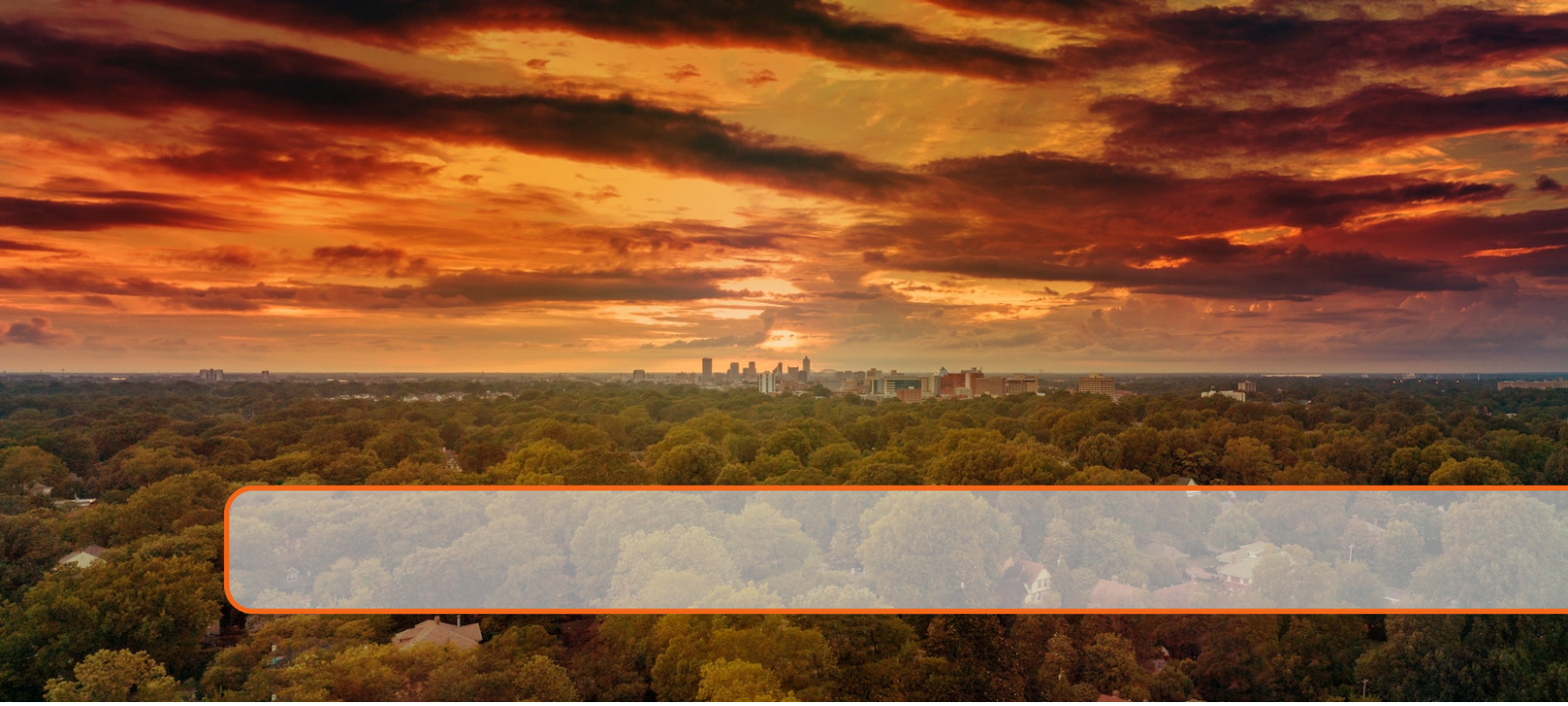
29 Example 6 . . . 255
29.1 Introduction . . . 255
29.1.1 Semi-Explicit Formulation . . . 255
29.1.2 Index Reduction Procedure . . . 255
29.2 Baumgarte Stabilization . . . 256

30 Example 7 . . . 259
30.1 Circuit Description . . . 259
30.1.1 Semi-Explicit Formulation . . . 259
30.1.2 Index Reduction Procedure . . . 259
30.2 Baumgarte Stabilization . . . 261
30.3 Simulation Results . . . 262

31 Example 8 . . . 263
31.1 The DAE . . . 263
31.1.1 Semi-Explicit Formulation . . . 264
31.1.2 Index Reduction Procedure . . . 265

32 Example 9 . . . 273
32.1 Double Pendulum System with Connecting Bar . . . 274
32.1.1 Complete DAE Structure . . . 275
32.1.2 Semi-Explicit Formulation . . . 275
32.1.3 Index Reduction Procedure . . . 276

References . . . 281



Ordinary differential equations are a powerful and essential tool in both theoretical and applied mathematics. Their ability to model dynamic systems, whether simple or complex, makes them indispensable in understanding the behavior of many real-world systems.

With advanced techniques and numerical methods, ODEs continue to play a crucial role in scientific research, engineering design, and beyond. Understanding ODEs is therefore central to many fields of study, from physics and engineering to biology and economics.

Acknowledgement



Figure 1: License: CC BY 4.0

This project is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0). Some pages may contain materials that are subject to copyright, in which case you will see the copyright notice. make

Introduction

1 Ordinary Differential Equations (ODEs) 17
1.1 Application of ODEs 17
1.2 Classification of ODEs 21
1.3 Solution Techniques 21

2 Some properties of ODEs 23
2.1 Examples of single ODE 23
2.2 Examples of Systems of ODEs 24
2.3 Vector Form of ODEs 24
2.4 General ODE and properties 26
2.5 Well-Posedness 28
2.6 Uniqueness of Solutions 28
2.7 Existence and Uniqueness 29
2.8 Example of ODE with Existence but No Uniqueness 30

3 Numerical Solutions for Initial Value Problems 33
3.1 Euler Method 34
3.2 Heun and Collatz Methods 36

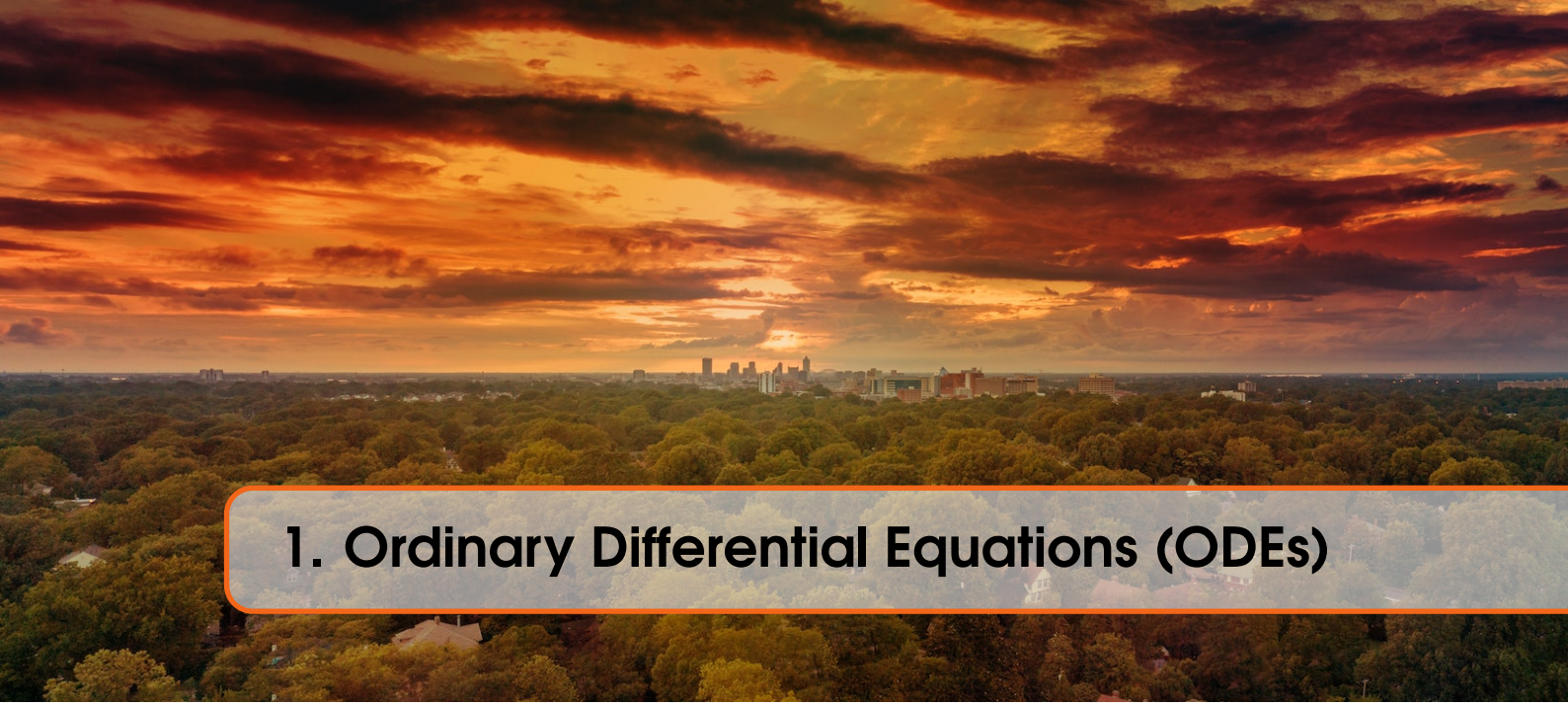
4 Runge-Kutta Methods 39
4.1 Explicit Runge-Kutta Methods 39
4.2 Implicit Runge-Kutta Methods 43

Ordinary differential equations are a powerful and essential tool in both theoretical and applied mathematics. Their ability to model dynamic systems, whether simple or complex, makes them indispensable in understanding the behavior of many real-world systems.

Learning objectives

After this part, you should be able to formulate an initial value problem, identify the assumptions that ensure a meaningful solution, and select and analyse a one-step numerical method in terms of order and stability.

With advanced techniques and numerical methods, ODEs continue to play a crucial role in scientific research, engineering design, and beyond. Understanding ODEs is therefore central to many fields of study, from physics and engineering to biology and economics.



1. Ordinary Differential Equations (ODEs)

In mathematics, an *ordinary differential equation* (ODE) is a type of *differential equation* that involves one or more unknown functions of a single independent variable, along with their derivatives. The term *ordinary* is used to distinguish these equations from *partial differential equations* (PDEs), which involve multiple independent variables. ODEs describe the relationship between a function and its rates of change with respect to one independent variable, providing a framework to model dynamic systems in various fields.

1.1 Application of ODEs

The importance of ordinary differential equations (ODEs) lies in their ability to model a vast range of phenomena across numerous scientific and engineering disciplines. ODEs are used to describe processes that evolve over time or space, making them powerful tools in understanding dynamic systems. These processes could include the motion of objects, population growth, the spread of diseases, fluid flow, and other natural or man-made systems that change continuously. The solutions to ODEs describe how a system evolves from one state to another, which allows us to predict, optimize, and gain deeper insights into the behavior of such systems.

1.1.1 Physics

In **physics**, ODEs play a critical role in describing the motion of particles and objects under the influence of forces. One of the simplest yet most foundational examples is **Newton's second law of motion**, which states that the force acting on an object is proportional to its acceleration. This leads to the second-order differential equation:

$$\mathscr{F}(z(t)) = m\frac{\mathrm{d}^2}{\mathrm{d}t^2}z(t)$$

where $\mathscr{F}$ is the force, $z(t)$ is the position of the object at time t, and m is its mass. Solving this equation allows us to determine the trajectory of an object when a known force is applied. For example, this law is used to predict the movement of planets in space or the motion of a pendulum.

1.1.2 Charged Particle in an Electromagnetic Field

The motion of a charged particle in the presence of electric ($\vec{E}$) and magnetic ($\vec{B}$) fields is described by the **Lorentz force law**. The Lorentz force law states that the force acting on a charged particle is the sum of the electric and magnetic forces:

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$$

where: - $\vec{F}$ is the force acting on the particle, - q is the charge of the particle, - $\vec{E}$ is the electric field, - $\vec{B}$ is the magnetic field, - $\vec{v}$ is the velocity of the particle.

The Lorentz force law can be expressed as a second-order ordinary differential equation that governs the motion of the particle. Using Newton's second law ($\vec{F} = m\vec{a}$), where m is the mass of the particle and $\vec{a}$ is its acceleration, we get the following system of ODEs for the particle's motion:

$$m\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} = q(\vec{E} + \vec{v} \times \vec{B})$$

where $\vec{r}(t)$ is the position vector of the particle at time (t), and the velocity

$$\vec{v}(t) = \frac{\mathrm{d}\vec{r}}{\mathrm{d}t}$$

This equation describes how the position and velocity of the particle change over time under the influence of electric and magnetic fields.

1.1.3 Chemistry

In **chemistry**, ODEs are crucial in modeling reaction rates, chemical kinetics, and the processes of diffusion and mixing. A well-known example is the rate law in chemical reactions. For a simple first-order reaction, the rate at which a reactant is consumed is proportional to its concentration, which can be modeled with the following first-order ODE:

$$\frac{\mathrm{d}[A]}{\mathrm{d}t} = -k[A]$$

where $[A]$ is the concentration of the reactant at time t, and k is the rate constant. The solution to this equation describes how the concentration of the reactant decreases over time as the reaction progresses. For more complex reactions, such as second-order reactions or systems involving multiple reactants, systems of coupled ODEs are often used to model the interdependence between concentrations.

In **chemical engineering**, ODEs are used to describe the rates of flow in reactors, heat exchangers, and distillation columns, as well as in the modeling of fluid dynamics and transport processes. These equations are crucial for designing processes in which materials and energy are exchanged between different systems.

1.1.4 Engineering

In **mechanical engineering**, ODEs are fundamental in the study of vibrations and structural dynamics. For instance, the motion of a damped harmonic oscillator, which can be modeled as:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + c\frac{\mathrm{d}x}{\mathrm{d}t} + kx = 0$$

where m is the mass, c is the damping coefficient, and k is the spring constant. This second-order ODE describes how an object behaves when subject to both restoring forces (from a spring) and damping forces (such as friction). These equations are used in the design of suspension systems, buildings, and machinery that need to withstand vibrations.

In **civil engineering**, ODEs are applied to model the deflection and stress in beams and bridges, particularly in response to forces such as wind, gravity, or earthquakes. Structural engineers use these models to ensure safety and optimal design. A famous example is the beam bending equation, which can be modeled by the following ODE:

$$EI\frac{\mathrm{d}^4y}{\mathrm{d}x^4} = q(x)$$

where E is the Young's modulus of the material, I is the moment of inertia, $y(x)$ is the deflection of the beam, and $q(x)$ is the load applied at position x. The solution of this equation helps engineers determine the bending of beams under various loading conditions.

1.1.5 Aerospace Engineering

In **aerospace engineering**, ODEs are used extensively in the modeling and simulation of flight dynamics, orbital mechanics, and spacecraft trajectory planning. For example, in the study of **orbital mechanics**, the motion of a satellite under the influence of gravity is governed by Newton's law of gravitation, leading to an ODE that describes the satellite's trajectory:

$$\frac{\mathrm{d}^2r}{\mathrm{d}t^2} = -\frac{GM}{r^2}$$

where r is the distance between the satellite and the center of the Earth, G is the gravitational constant, and M is the mass of the Earth. Solving this ODE allows engineers to predict satellite orbits, plan space missions, and ensure that spacecraft stay on the correct trajectory.

In **aircraft dynamics**, ODEs are used to describe the motion of an aircraft in the atmosphere. The aircraft's acceleration, velocity, and position are governed by forces such as lift, drag, thrust, and weight, and these forces are modeled by a system of ODEs. By solving these equations, aerospace engineers can optimize the performance of the aircraft, design control systems, and simulate flight behavior under different conditions, including turbulence or extreme weather.

1.1.6 Biology

In **biology**, ODEs are used to model the growth of populations, the spread of diseases, and the dynamics of ecosystems. For example, the **logistic growth model**, a first-order ODE, describes how a population grows over time given a limited carrying capacity of the environment:

$$\frac{\mathrm{d}P}{\mathrm{d}t} = rP\left(1 - \frac{P}{K}\right)$$

where P is the population size, r is the growth rate, and K is the carrying capacity of the environment. This model is widely used in ecology to predict population dynamics, taking into account both growth and environmental limitations.

1.1.7 Epidemiology

In **epidemiology**, ODEs are fundamental in modeling the spread of diseases. The classic **SIR model** (Susceptible-Infected-Recovered) is a system of ODEs used to study infectious diseases. It describes the rates at which individuals move from one compartment (e.g., susceptible, infected, recovered) to another:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta SI, \quad \frac{\mathrm{d}I}{\mathrm{d}t} = \beta SI - \gamma I, \quad \frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I$$

where S is the number of susceptible individuals, I is the number of infected individuals, and R is the number of recovered individuals. These models help public health officials predict the spread of diseases and determine strategies for controlling outbreaks.

1.1.8 Economics

In **economics**, ODEs are used to model the growth of markets, the value of assets, and various macroeconomic factors over time. For example, the **Solow growth model**, which is used to

understand long-term economic growth, involves an ODE that describes the change in capital per worker over time:

$$\frac{dk}{dt} = sf(k) - \delta k$$

where k is the capital per worker, s is the savings rate, $f(k)$ is the output per worker, and δ is the depreciation rate. This equation helps economists understand how economies evolve, accounting for factors like savings, investment, and technological progress.

In **financial markets**, ODEs are also used to model the price of assets, such as stocks and bonds, using stochastic differential equations. For instance, the famous **Black-Scholes model** for pricing options involves solving a partial differential equation, which, under certain assumptions, can be reduced to an ODE. This model is essential in determining the fair price of options in markets.

1.1.9 Electronics

In **electronics**, ODEs are crucial for modeling circuits, especially in the context of RC (resistor-capacitor) and RLC (resistor-inductor-capacitor) circuits, which are used in signal processing, filtering, and oscillation. For example, in an **RC circuit**, the voltage across the capacitor $V_C(t)$ changes over time according to the following first-order ODE:

$$\frac{dV_C}{dt} = -\frac{1}{RC}V_C(t)$$

where R is the resistance, C is the capacitance, and $V_C(t)$ is the voltage across the capacitor. This ODE describes how the capacitor charges and discharges over time, and solving it allows engineers to design circuits with specific time constants for various applications, such as smoothing signals or timing applications.

In an **RLC circuit**, which includes a resistor, an inductor, and a capacitor, the second-order ODE governing the current $I(t)$ is given by:

$$L\frac{d^2I}{dt^2} + R\frac{dI}{dt} + \frac{1}{C}I = 0$$

where L is the inductance, R is the resistance, and C is the capacitance. This ODE is essential for analyzing the behavior of oscillating circuits, resonance, and the decay of current in various electrical systems.

1.1.10 Medicine

In **medicine**, ODEs are widely used to model physiological processes, drug interactions, and the spread of diseases. A classic example is **pharmacokinetics**, which describes the absorption, distribution, metabolism, and excretion (ADME) of drugs in the human body. For instance, the **one-compartment model** for the elimination of a drug can be represented by the following first-order ODE:

$$\frac{dC}{dt} = -kC$$

where $C(t)$ is the concentration of the drug in the bloodstream at time t, and k is the elimination rate constant. Solving this ODE helps determine how the drug concentration decreases over time, which is crucial for dosing and treatment scheduling.

In **cardiovascular modeling**, ODEs are used to describe the heart's pumping action and blood flow. The dynamics of the heart's contraction can be modeled by a system of ODEs that account for factors such as heart rate, stroke volume, and the resistance in the circulatory system. One example is the **Windkessel model**, which represents the pressure and volume in the arterial system as a set of ODEs:

$$\frac{dp}{dt} = \frac{1}{R}(Q - p)$$

where $p(t)$ is the pressure in the arteries, Q is the flow rate, and R is the resistance. This model helps in understanding and treating cardiovascular diseases such as hypertension.

1.2 Classification of ODEs

ODEs can be classified in several ways, including by the order of the equation, its linearity, and whether the equation is autonomous or non-autonomous.

1.2.1 Order of an ODE

The order of an ODE refers to the highest derivative that appears in the equation. For example, an ODE involving the first derivative of a function is called a *first-order ODE*, while one that involves the second derivative is called a *second-order ODE*. The order of an ODE determines the number of independent conditions (such as initial conditions) required to uniquely determine a solution. For instance, the general solution of a second-order ODE requires two initial conditions.

1.2.2 Linearity of an ODE

An ODE is said to be *linear* if the unknown function and its derivatives appear only to the first power and are not multiplied together. For example, a first-order linear ODE can be written as:

$$\frac{d}{dt}y(t) + p(t)y(t) = q(t)$$

where $p(t)$ and $q(t)$ are known functions of the independent variable t, and $y(t)$ is the unknown function. Linear ODEs are well understood and are often solvable using analytical methods.

In contrast, a *nonlinear ODE* is one where the unknown function and its derivatives appear in nonlinear terms, such as products or powers. Nonlinear ODEs are more difficult to solve and often require numerical methods for finding approximate solutions. They are crucial in modeling complex real-world phenomena like turbulence in fluids or nonlinear population dynamics.

1.2.3 Autonomous vs. Non-autonomous ODEs

An ODE is called *autonomous* if the independent variable does not explicitly appear in the equation. In other words, the equation depends only on the dependent variable and its derivatives. For example, the ODE:

$$\frac{d}{dt}y(t) = f(y(t))$$

is autonomous. If the independent variable appears explicitly in the equation, the ODE is *non-autonomous*.

1.3 Solution Techniques

While simple ODEs can often be solved analytically, many real-world problems lead to more complex equations that cannot be solved explicitly. In such cases, several advanced techniques come into play:

1. **Numerical Methods** For many ODEs, particularly nonlinear ones, explicit analytical solutions are not possible. In these cases, numerical methods such as Euler's method, Runge-Kutta methods, or more advanced techniques like the finite difference method are used to approximate solutions. These methods involve discretizing the independent variable and solving for the function at each step.

2. **Qualitative Analysis** In addition to finding explicit solutions, it is often useful to study the *qualitative behavior* of ODE solutions. This involves analyzing properties such as stability, equilibrium points, and periodic solutions. The study of dynamical systems and stability theory allows for a deeper understanding of the behavior of systems described by ODEs, even without finding exact solutions. Techniques like phase plane analysis and Lyapunov's stability criteria are often employed to investigate the behavior of solutions in the long term.
3. **Series Solutions** For some ODEs, particularly those with variable coefficients or singularities, solutions may be found using power series expansions. Methods such as the Frobenius method are used to find series solutions near singular points, allowing for approximate solutions to be obtained.
4. **Transform Methods** In some cases, tools like the Laplace transform or Fourier transform are used to convert an ODE into an algebraic equation, which is easier to solve. Once the equation is solved in the transformed domain, the inverse transform is applied to obtain the solution to the original problem.



2. Some properties of ODEs

A system of Ordinary Differential Equations (ODEs) consists of one or more equations that describe the relationship between one or more unknown functions and their derivatives (see [6],[11])

2.1 Examples of single ODE

In this section, we present several examples of **first-order** ordinary differential equations, highlighting their structure and how they are defined in terms of a function $f(x,y)$.

■ **Example 2.1 — Nonlinear First-Order ODE**.**

$$\begin{cases} y' = y^2 + x \\ y(0) = 1 \end{cases} \Longrightarrow \begin{array}{l} y'(x) = y(x)^2 + x = f(x, y(x)), \\ f(x,y) = y^2 + x \end{array}$$

■

■ **Example 2.2 — ODE with Mixed Linear and Nonlinear Terms**.**

$$\begin{cases} y' = xy + y^2 \\ y(0) = 2 \end{cases} \Longrightarrow \begin{array}{l} y'(x) = xy(x) + y(x)^2 = f(x, y(x)), \\ f(x,y) = xy + y^2 \end{array}$$

■

■ **Example 2.3 — ODE with a Trigonometric Nonlinearity**.**

$$\begin{cases} z' = x^2 + \sin z \\ z(-1) = 0 \end{cases} \Longrightarrow \begin{array}{l} z'(x) = x^2 + \sin z(x) = f(x, z(x)), \\ f(x,z) = x^2 + \sin z \end{array}$$

■

■ **Example 2.4 — Logistic Growth Model (Common in Biology)**.**

$$\begin{cases} p' = rp\left(1 - \dfrac{p}{K}\right) \\ p(0) = p_0 \end{cases} \Longrightarrow p'(t) = rp(t)\left(1 - \frac{p(t)}{K}\right)$$

- Models **population growth**, where p represents the population size.
- **Parameters:**

  - r = growth rate.
  - K = carrying capacity of the environment.
- **Applications:** Used in **ecology** and **epidemiology**.

■

■ **Example 2.5 — Newton's Law of Cooling (Physics Application)**.**

$$\begin{cases} T' = -k(T - T_{\text{env}}) \\ T(0) = T_0 \end{cases} \implies T'(t) = -k(T - T_{\text{env}})$$

- Describes how an object cools down over time.
- **Parameters:**
  - k = heat transfer coefficient.
  - T_{env} = ambient temperature.

■

2.2 Examples of Systems of ODEs

■ **Example 2.6 — A Simple Two-Variable System.** The following system describes the evolution of two state variables $x(t)$ and $y(t)$:

$$\left\{ \begin{array}{l} x' = x + y \\ y' = \mathrm{e}^x - y \end{array} \right\} \text{ ODE equations}$$
$$\left\{ \begin{array}{l} x(0) = 1 \\ y(0) = 1 \end{array} \right\} \text{ Initial conditions}$$

- The first equation describes the rate of change of x, which depends on both x and y.
- The second equation models the evolution of y, which depends on an exponential function of x and y itself.
- The system starts from the initial state $(x(0), y(0)) = (1, 1)$.

■

■ **Example 2.7 — A Three-Variable Coupled System**.** This example involves three interdependent variables x, w, and z, with an explicit time dependence in the third equation.

$$\left\{ \begin{array}{l} z' = x + z \\ w' = z \\ x' = wz + t \end{array} \right\} \text{ ODE equations}$$
$$\left\{ \begin{array}{l} x(0) = 1 \\ w(0) = 1 \\ z(0) = 1 \end{array} \right\} \text{ Initial conditions}$$

■

2.3 Vector Form of ODEs

A system of first-order ODEs can be expressed in vector notation, which provides a more compact and structured representation of the problem. This notation is particularly useful in numerical methods and theoretical analysis.

2.3.1 System of ODEs in Scalar Form

Consider a system of n first-order ordinary differential equations:

$$y_1(t), y_2(t), \ldots, y_n(t), \qquad \text{functions of } t$$

The system is given by:

$$\begin{cases} \left.\begin{aligned} y_1' &= f_1(t, y_1, y_2, \ldots, y_n) \\ y_2' &= f_2(t, y_1, y_2, \ldots, y_n) \\ &\vdots \\ y_n' &= f_n(t, y_1, y_2, \ldots, y_n) \end{aligned}\right\} \text{ODE system} \\ \left.\begin{aligned} y_1(0) &= y_1^0, \\ y_2(0) &= y_2^0, \\ &\vdots \\ y_n(0) &= y_n^0, \end{aligned}\right\} \text{Initial conditions} \end{cases}$$

2.3.2 Compact Vector Form

To write this system in vector notation, define:

- The state vector:

$$\boldsymbol{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{pmatrix}$$

- The function that describes the system dynamics:

$$\boldsymbol{f}(t, \boldsymbol{y}) = \begin{pmatrix} f_1(t, y_1, y_2, \ldots, y_n) \\ f_2(t, y_1, y_2, \ldots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \ldots, y_n) \end{pmatrix}$$

- The initial condition vector:

$$\boldsymbol{y}_0 = \begin{pmatrix} y_1^0 \\ y_2^0 \\ \vdots \\ y_n^0 \end{pmatrix}$$

Thus, the system can be rewritten in **vector form** as:

$$\begin{cases} \boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases} \qquad \Longrightarrow \qquad \boldsymbol{y}(t) = \boldsymbol{f}(t, \boldsymbol{y}(t))$$

This notation simplifies analysis and computation, particularly in numerical solvers.

2.3.3 Matrix Form of a Linear System

When the system is **linear**, meaning the functions $f_i(t, \boldsymbol{y})$ depend linearly on $\boldsymbol{y}$, the system can be written as:

$$\boldsymbol{y}' = \boldsymbol{A}(t)\boldsymbol{y} + \boldsymbol{b}(t)$$

where:

- $\boldsymbol{A}(t)$ is an $n \times n$ matrix of coefficients.
- $\boldsymbol{b}(t)$ is a vector of external inputs (or forcing terms).
- If $\boldsymbol{b}(t) = \boldsymbol{0}$, the system is **homogeneous**; otherwise, it is **inhomogeneous**.

A simple example of a **homogeneous linear system** is:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

This formulation allows for powerful solution techniques, such as matrix exponentiation and eigenvalue analysis.

2.4 General ODE and properties

Before exploring numerical solution methods, it is useful to briefly review some key definitions and properties of ordinary differential equations (ODEs) and initial value problems (IVPs).

Definition 2.1 — Ordinary Differential Equation. An *ordinary differential equation* (ODE) is an equation involving a function $z(t)$, defined on an interval $I \subset \mathbb{R}$, with values in either real or complex numbers, or in the vector spaces $\mathbb{R}^d$ or $\mathbb{C}^d$. The general form of an ODE is:

$$\boldsymbol{f}\left(t, \boldsymbol{z}(t), \boldsymbol{z}'(t), \boldsymbol{z}''(t), \ldots, \boldsymbol{z}^{(n)}(t)\right) = 0$$

Here, $\boldsymbol{f}$ is a function that depends on time t, the unknown function $\boldsymbol{z}(t)$, and its derivatives up to order n. The *order* of the differential equation is determined by the highest derivative $\boldsymbol{z}^{(n)}(t)$ that appears.

If $\boldsymbol{z}(t)$ takes values in a space of dimension $d > 1$, the equation is called a *system of differential equations*.

This form is generally too much general, the explicit form is often available in practical problems:

Definition 2.2 — Explicit ODE. An *explicit* first-order differential equation has the form:

$$\boldsymbol{z}'(t) = \boldsymbol{f}(t, \boldsymbol{z}(t)) \quad \text{or briefly:} \quad \boldsymbol{z}' = \boldsymbol{f}(t, \boldsymbol{z}) \tag{2.1}$$

For a higher-order equation (order n), the equation is called explicit if it can be written as:

$$\boldsymbol{z}^{(n)}(t) = \boldsymbol{f}\left(t, \boldsymbol{z}(t), \boldsymbol{z}'(t), \boldsymbol{z}''(t), \ldots, \boldsymbol{z}^{(n-1)}(t)\right)$$

Definition 2.3 — Autonomous ODE. A differential equation of the form Equation 2.1 is called *autonomous* if the right-hand side f does not explicitly depend on t, meaning:

$$\boldsymbol{z}'(t) = \boldsymbol{f}(\boldsymbol{z}(t))$$

Numerical methods are often designed specifically for first-order ODEs. Therefore, higher-order ODEs must be transformed into an equivalent system of first-order equations before applying these methods.

Lemma 2.1 — Reduction to First-Order Systems. Every differential equation of higher order can be rewritten as a system of first-order differential equations. If the original equation is explicit, the resulting system will also be explicit.

Proof. To transform a higher-order differential equation into a system of first-order equations, we introduce auxiliary variables for each derivative of $\boldsymbol{z}(t)$:

$$\begin{aligned} z_0(t) &= z(t), \\ z_1(t) &= z'(t), \\ &\vdots \\ z_{n-1}(t) &= z^{(n-1)}(t) \end{aligned}$$

This allows us to rewrite the original n-th order differential equation as a system of n first-order equations:

$$\begin{pmatrix} z_0'(t) - z_1(t) \\ z_1'(t) - z_2(t) \\ \vdots \\ z_{n-2}'(t) - z_{n-1}(t) \\ f\left(t, z_0(t), z_1(t), \ldots, z_{n-1}(t), z'n-1(t)\right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

In the case of an explicit equation, this system becomes:

$$\begin{pmatrix} z_0'(t) \\ z_1'(t) \\ \vdots \\ z_{n-2}'(t) \\ z_{n-1}'(t) \end{pmatrix} = \begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_{n-1}(t) \\ f\left(t, z_0(t), z_1(t), \ldots, z_{n-1}(t)\right) \end{pmatrix}$$

■

Most often, ODEs come with additional information in the form of an *initial condition*, leading to what is known as an *initial value problem* (IVP), also referred to as a **Cauchy problem**. The initial condition specifies the value of the function z at a given point t_0 within the domain, helping to define a unique solution to the differential equation.

Definition 2.4 — Initial Value Problem (IVP). Given an initial point

$$(t_0, \boldsymbol{z}_0) \in \mathbb{R} \times \mathbb{R}^d$$

let $f(t, \boldsymbol{z}(t))$ be a function with values in $\mathbb{R}^d$, defined in a neighborhood

$$I \times U \subset \mathbb{R} \times \mathbb{R}^d$$

around the initial point. The *initial value problem* (IVP) is then the task of finding a function $\boldsymbol{z}(t)$ such that:

$$\begin{cases} \boldsymbol{z}'(t) = f(t, \boldsymbol{z}(t)) \\ \boldsymbol{z}(t_0) = \boldsymbol{z}_0 \end{cases}$$

In the study of ordinary differential equations (ODEs), the concept of a local solution plays a fundamental role in understanding the behavior of dynamical systems. Given an initial value problem (IVP), a local solution refers to a function that satisfies the differential equation and initial condition within a limited region around the initial time.

Local solutions are particularly important because existence theorems, such as the Picard-Lindelöf Theorem 2.2, often guarantee solutions only within a small interval rather than over the entire domain. This is due to factors like nonlinearities in the system, singularities, or rapidly growing solutions that may prevent global existence.

Definition 2.5 — Local Solution. A *local solution* to the IVP is a continuously differentiable function $\boldsymbol{z}(t)$ that satisfies $\boldsymbol{z}(t_0) = \boldsymbol{z}_0$. This solution exists within some neighborhood J of t_0, where both $\boldsymbol{z}(t)$ and $\boldsymbol{f}(t, \boldsymbol{z}(t))$ are well-defined, and the equation holds for all $t \in J$.

2.5 Well-Posedness

2.5.1 Definition of a Well-Posed Initial Value Problem

A mathematical problem is said to be **well-posed** if it satisfies the following conditions (**Hadamard conditions**):

- **Existence**: a solution exists.
- **Uniqueness**: the solution is unique.
- **Continuous Dependence**: the solution continuously depends on the initial data.

If any of these conditions fail, the problem is considered **ill-posed**, meaning small changes in the initial conditions can lead to large deviations in the solution.

2.5.2 Existence of Solutions

The following result provides a **general existence theorem** for first-order differential equations.

Theorem 2.1 — Peano's Existence Theorem. Let $f(t, \boldsymbol{z})$ be a continuous function defined on the closed set:

$$\bar{D} = \left\{ (t, \boldsymbol{z}) \in \mathbb{R} \times \mathbb{R}^d \;\middle|\; |t - t_0| \leq \alpha, |\boldsymbol{z} - \boldsymbol{z}_0| \leq \beta \right\}$$

where $\alpha, \beta > 0$. Then, there exists a continuously differentiable solution $\boldsymbol{z}(t) \in C^1(I)$ on the interval $I = [t_0 - T, t_0 + T]$, where:

$$T = \min\left(\alpha, \frac{\beta}{M}\right), \quad M = \max_{(t,z)\in\bar{D}} |f(t, z)|$$

This theorem **guarantees the existence** of a solution but does **not** ensure uniqueness.

2.5.3 Peano's Continuation Theorem

Under the assumptions of Peano's Existence Theorem, the solution can be **extended** to a larger interval $I_m = [t_-, t_+]$, such that the boundary points $(t_-, z(t_-))$ and $(t_+, z(t_+))$ lie on the boundary of the domain D.

- The solution exists as long as $f(t, z)$ remains **bounded**.
- The interval I_m may be **finite or infinite**, depending on the properties of $f(t, z)$.

2.6 Uniqueness of Solutions

To ensure *uniqueness*, we require stronger conditions such as **Lipschitz continuity**. We also introduce **Grönwall's Lemma**, a crucial tool for proving uniqueness.

2.6.1 Lipschitz Condition

A function $f(t,z)$ satisfies a **Lipschitz condition** on its domain $D = I \times \Omega \subset \mathbb{R} \times \mathbb{R}^d$ if there exists a positive constant L such that:

$$\forall t \in I, \forall x, z \in \Omega : |f(t,x) - f(t,z)| \leq L|x - z|$$

- If L is a **global constant**, the function is said to satisfy a **global Lipschitz condition**.
- If the condition holds only for compact subsets of D, it is called a **local Lipschitz condition**.

The **Lipschitz condition ensures uniqueness** of solutions.

2.6.2 Grönwall's Lemma

Grönwall's Lemma is a fundamental result used to establish **bounds on solutions** and prove uniqueness.

Lemma 2.2 — Grönwall. Let

- $w(t)$, $a(t)$, and $b(t)$ be non-negative, integrable functions
- the product $a(t)w(t)$ is also integrable
- $b(t)$ is non-decreasing

moreover $w(t)$ satisfies the inequality:

$$w(t) \leq b(t) + \int_{t_0}^{t} a(s)w(s)\,\mathrm{d}s, \qquad t \geq t_0$$

Then for almost all $t \geq t_0$, we have:

$$w(t) \leq b(t) \exp\left(\int_{t_0}^{t} a(s)\,\mathrm{d}s\right)$$

This result is **key** in proving the uniqueness of solutions in the **Picard-Lindelöf Theorem**.

2.7 Existence and Uniqueness

2.7.1 Picard-Picard Theorem

The **Picard-Lindelöf theorem** (also called the **Cauchy-Lipschitz theorem**) strengthens Peano's theorem by guaranteeing **both existence and uniqueness** of solutions.

Theorem 2.2 — Picard-Lindelöf. Let $f(t,\boldsymbol{z})$ be continuous on the **cylinder**:

$$D = \{(t,\boldsymbol{z}) \in \mathbb{R} \times \mathbb{R}^d \mid |t - t_0| \leq a, |\boldsymbol{z} - \boldsymbol{z}_0| \leq b\}$$

Assume that:

1. $\boldsymbol{f}(t,\boldsymbol{z})$ is **bounded**, meaning there exists a constant M such that:

$$\|\boldsymbol{f}(t,\boldsymbol{z})\| \leq M, \quad \forall (t,\boldsymbol{z}) \in D$$

2. f satisfies a **Lipschitz condition** with constant L:

$$\|\boldsymbol{f}(t,\boldsymbol{x}) - \boldsymbol{f}(t,\boldsymbol{z})\| \leq L\|\boldsymbol{x} - \boldsymbol{z}\|, \quad \forall \boldsymbol{x}, \boldsymbol{z} \in D$$

Then, the initial value problem:

$$\begin{cases} \boldsymbol{z}'(t) = \boldsymbol{f}(t,\boldsymbol{z}(t)) \\ \boldsymbol{z}(t_0) = \boldsymbol{z}_0 \end{cases}$$

has a **unique solution** on the interval $I = [t_0 - T, t_0 + T]$, where:

$$T = \min\left(a, \frac{b}{M}\right)$$

2.8 Example of ODE with Existence but No Uniqueness

Consider the ordinary differential equation (ODE):

$$y' = \sqrt{|y|}$$

with the initial condition:

$$y(0) = 0.$$

2.8.1 Existence of the Solution

Peano's Theorem guarantees that if a function $f(t,y)$ is continuous, then there exists at least one solution within some interval around the initial point. In this case, the function $f(y) = \sqrt{|y|}$ is continuous for all values of y, so Peano's theorem tells us that there exists **at least one solution** for the given ODE, at least in a small interval around $t = 0$.

Thus, we can assert that there exists a function $y(t)$ that satisfies the differential equation and the initial condition $y(0) = 0$.

2.8.2 No Uniqueness of the Solution

Picard-Lindelöf Theorem 2.2 states that, in addition to existence, a solution is also **unique** if the function $f(t,y)$ is **Lipschitz continuous** with respect to y in an interval around the initial condition y_0. However, in our case, the function $f(y) = \sqrt{|y|}$ is not Lipschitz continuous at $y = 0$.

To verify this, let's compute the derivative of $f(y) = \sqrt{|y|}$:

$$f'(y) = \begin{cases} \dfrac{1}{2\sqrt{y}} y \geq 0 \\ -\dfrac{1}{2\sqrt{-y}} y \geq 0 \end{cases} = \frac{\text{sign}(y)}{2\sqrt{|y|}}$$

Notice that this derivative becomes infinite as $y \to 0$, so the function is not Lipschitz continuous at $y = 0$. The Lipschitz condition requires the derivative to be bounded, but in this case, it is not.

Since $f(y) = \sqrt{|y|}$ does not satisfy the Lipschitz condition at $y = 0$, **Picard-Lindelöf** Theorem 2.2 does not guarantee the uniqueness of the solution. This means there may be more than one solution that satisfies the same initial condition $y(0) = 0$.

2.8.3 Multiple Solutions

Even though a solution exists, it is not unique. In fact, there are **many solutions** that satisfy the ODE and the initial condition $y(0) = 0$. Here are a few examples:

1. **Trivial (Zero) Solution**
   One solution is the **zero solution**, where $y(x) = 0$ for all x. This is a solution that satisfies both the ODE and the initial condition $y(0) = 0$, and it holds for every value of x.

   $$y(x) = 0 \quad \text{for all} \quad x.$$

2. **Explicit (Non-Zero) Solution**
   Another solution can be found by solving the ODE by separating the variables. We start by writing the ODE as:

$$\frac{dy}{\sqrt{|y|}} = dx.$$

Integrating both sides:

$$2\sqrt{|y|} = x + C.$$

Setting the initial condition $y(0) = 0$, we get $C = 0$. Therefore, the solution becomes:

$$y(x) = \frac{x^2}{4}, \quad \text{for} \quad x \geq 0.$$

This is an explicit solution for $x \geq 0$ and satisfies the ODE and the initial condition.
For $x < 0$, the solution must be:

$$y(x) = -\frac{x^2}{4}, \quad \text{for} \quad x < 0,$$

to match the sign of the right-hand side. Thus, the complete solution is:

$$y(x) = \frac{x^2}{4} \begin{cases} 1 & x \geq 0, \\ -1 & x < 0. \end{cases}$$

This can also be written more compactly as:

$$y(x) = \frac{x|x|}{4}.$$

The derivative of this function is:

$$y'(x) = \frac{x}{2} \begin{cases} 1 & x \geq 0, \\ -1 & x < 0. \end{cases}$$

Which simplifies to:

$$y'(x) = \frac{|x|}{2}.$$

Notice that:

$$\sqrt{|y(x)|} = \sqrt{\left|\frac{x|x|}{4}\right|} = \sqrt{\frac{|x|^2}{4}} = \frac{|x|}{2} = y'(x).$$

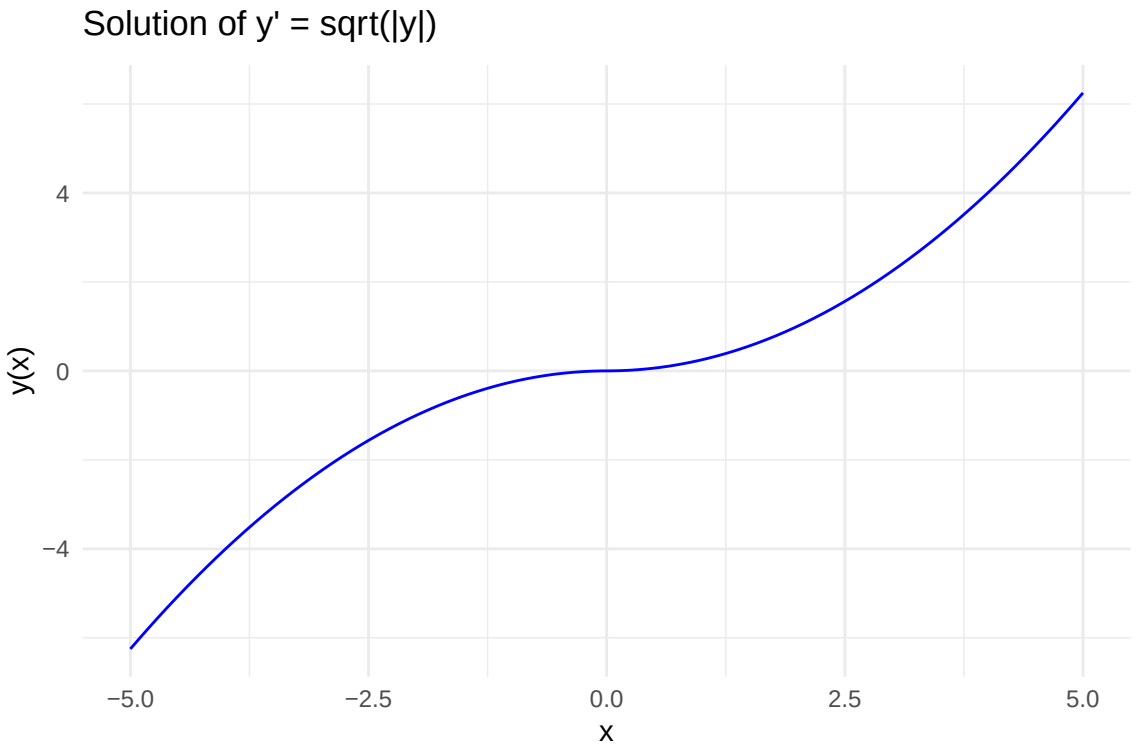
Thus, the function $y(x) = \frac{x|x|}{4}$ satisfies the ODE, and it is indeed a valid solution.

2.8.4 Conclusion

The ODE $y' = \sqrt{|y|}$ with the initial condition $y(0) = 0$ has multiple solutions. In addition to the trivial solution $y(x) = 0$, we also have non-zero solutions of the form:

$$y(x) = \frac{x|x|}{4}.$$

These multiple solutions are possible due to the lack of uniqueness in this case, as the function $f(y) = \sqrt{|y|}$ is not Lipschitz continuous at $y = 0$.


Solution of y' = sqrt(|y|)
y(x)
4
0
−4
−5.0
−2.5
0.0
2.5
5.0
x



3. Numerical Solutions for Initial Value Problems

Learning objectives

After this chapter, you should be able to:

- distinguish local truncation error from global error;
- derive the explicit and implicit Euler updates for an initial value problem;
- explain why implicit methods require the solution of an algebraic or nonlinear system at every step; and
- relate convergence and absolute stability to the choice of a time step.

In many cases, finding an analytical solution to initial value problems (IVPs) is either impossible or impractical due to the complexity of the integrals involved. Therefore, numerical methods are commonly used to approximate solutions. These methods can be applied not only to single IVPs but also extend naturally to systems of differential equations. Additionally, since higher-order differential equations can be reformulated as systems of first-order equations, we will focus on numerical methods for first-order systems.

Definition 3.1 — Time Step. On a time interval $I = [t_0, t_0 + T]$, we partition the interval into n subintervals, known as *time steps*. Each subinterval

$$I_k = [t_{k-1}, t_k]$$

has a step size $\Delta t_k = t_k - t_{k-1}$. The total partitioning into n steps ensures that $t_n = t_0 + T$. The term *k-th time step* may refer either to the subinterval I_k or the specific time point t_k, depending on the context. For simplicity, we often assume evenly spaced time steps, where $\Delta t_k = h$ for all k.

Numerical methods for solving IVPs are divided into two primary categories: *one-step methods* and *multi-step methods*. In one-step methods, the next approximation z_{k+1} depends only on the current step z_k. In contrast, multi-step methods calculate z_{k+1} using information from several previous steps, including $z_{k-1}, z_{k-2}, \ldots, z_{k-n}$.

Definition 3.2 — Explicit Methods. An *explicit method* (also called forward integration) calculates approximations sequentially. Starting with an initial condition z_0 at time t_0, it computes the sequence $z_1, z_2, \ldots, z_n$ using the following update formula:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta t_k \boldsymbol{F}(t_k, \boldsymbol{z}_k, \ldots, \boldsymbol{z}_{k-p}; \Delta t_k)$$

Here, $\boldsymbol{F}$ is called the *increment function*, which determines the change from one time step to the next. The parameter p refers to the order of the numerical multi-step method. If $p = 1$, the method is a *one-step method*.

Definition 3.3 — Implicit Methods. An *implicit method* (also called backward integration) is similar in structure to explicit methods but requires solving for z_k implicitly. The update formula is given by:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta t_k \boldsymbol{F}(t_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{z}_k, \ldots, \boldsymbol{z}_{k-p}; \Delta t_h)$$

In this case, the increment function $\boldsymbol{F}$ depends on $\boldsymbol{z}_{k+1}$, meaning the equation must be solved iteratively or using numerical techniques to find $\boldsymbol{z}_{k+1}$. Implicit methods often provide better stability, especially for stiff differential equations, but are computationally more intensive because of the need to solve for $\boldsymbol{z}_{k+1}$ at each step.

In summary, the distinction between explicit and implicit methods is primarily in how the next step is computed. Explicit methods are generally simpler but may require smaller time steps for stability, whereas implicit methods can handle larger time steps but at the cost of increased computational effort.

3.1 Euler Method

The Euler method is the simplest one-step numerical method for solving initial value problems (IVPs). One intuitive way to derive this method is through the truncated Taylor series expansion.

Consider the following IVP:

$$\begin{cases} \boldsymbol{z}'(t) = \boldsymbol{f}(t, \boldsymbol{z}), \\ \boldsymbol{z}(t_0) = \boldsymbol{z}_0, \end{cases}$$

The Taylor series expansion of the solution $\boldsymbol{z}(t)$ around t_0 gives us:

$$\boldsymbol{z}(t_0 + h) = \boldsymbol{z}(t_0) + \Delta t \boldsymbol{z}'(t_0) + \mathcal{O}(\Delta t^2),$$

where $\boldsymbol{z}'(t_0) = \boldsymbol{f}(t_0, \boldsymbol{z}_0)$. Neglecting higher-order terms, we approximate $\boldsymbol{z}(t_0 + \Delta t)$ as:

$$\boldsymbol{z}(t_0 + \Delta t) \approx \boldsymbol{z}(t_0) + \Delta t \boldsymbol{z}'(t_0),$$

The difference between this approximation and the exact solution $\boldsymbol{z}(t_0 + \Delta t)$ is proportional to Δt^2, meaning the error decreases as Δt becomes smaller.

3.1.1 General Form of the Euler Method

To generalize this for any time interval $I = [t_0, t_0 + T]$, we divide the interval into subintervals and apply the approximation at each step. For the k-th subinterval

$$I_k = [t_{k-1}, t_k]$$

the *explicit Euler method* for the k-th step is given by:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta t \boldsymbol{f}(t_k, \boldsymbol{z}_k), \quad k = 0, 1, 2, \ldots, n,$$

where Δt is the step size.

On the other hand, the *implicit Euler method* for the k-th step is defined as:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta t \boldsymbol{f}(t_{k+1}, \boldsymbol{z}_{k+1}), \quad k = 0, 1, 2, \ldots, n-1.$$

In this case, the update depends on the unknown value $\boldsymbol{z}_{k+1}$, so an algebraic equation (nonlinear when $\boldsymbol{f}$ is nonlinear) must be solved at every step.

Local and global error

Two different errors must not be confused. The **local truncation error** is the defect produced by one numerical step when its input is the exact solution. For a method of global order p, it is typically $\mathcal{O}(\Delta t^{p+1})$. The **global error** is the difference between the numerical and exact solutions after many steps; on a fixed time interval it is typically $\mathcal{O}(\Delta t^{p})$, provided the method propagates perturbations in a stable way. The loss of one power of Δt reflects the approximately $T/\Delta t$ steps taken over an interval of length T.

Convergence and Stability

Both the explicit and implicit Euler methods are first-order accurate, meaning that their local truncation error is $\mathcal{O}(\Delta t^2)$, leading to a global error that scales as $\mathcal{O}(\Delta t)$. As the step size $\Delta t \to 0$, both methods converge to the exact solution, with the error at any time

$$t \in I = [t_0, t_0 + T],$$

bounded by $C\Delta t$, where C is a constant. However, in practice, this constant can grow exponentially with T, following

$$C \propto e^{LT},$$

where L is the **Lipschitz constant** of the function $\boldsymbol{f}(t, \boldsymbol{z})$. This growth in error amplification is particularly problematic for large-time integration.

Stability Considerations

One of the key differences between the explicit and implicit Euler methods lies in their stability properties. The explicit Euler method has a **stability constraint** dictated by the step size Δt, requiring Δt to be sufficiently small to ensure numerical stability. Specifically, for a linear test equation of the form

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \lambda z, \quad \lambda \in \mathbb{C}$$

the explicit Euler method is stable only if h satisfies the **stability condition**:

$$|1 + \Delta t \lambda| \leq 1.$$

For *stiff problems*, where solutions evolve on vastly different time scales and exhibit rapid transient dynamics, this condition can impose **severely restrictive time-step limitations**, making explicit Euler impractical.

In contrast, the **implicit Euler method** is *unconditionally stable* for all values of h when applied to the same test equation. This property makes it particularly advantageous for stiff problems, where it allows for significantly larger time steps without numerical instability. However, this comes at the cost of solving a nonlinear equation at each step, increasing computational expense.

Thus, while both methods share the same convergence rate of $\mathcal{O}(\Delta t)$, their stability properties determine their applicability:

- **Explicit Euler**: Suitable for *non-stiff problems* but requires small Δt for stability.
- **Implicit Euler**: More robust for *stiff problems*, allowing larger Δt, but requires solving implicit equations at each step.

3.2 Heun and Collatz Methods

3.2.1 Heun's Method

Heun's method, also known as the improved Euler method, is a second-order numerical method for solving initial value problems (IVPs). It refines the accuracy of the explicit Euler method by incorporating an additional correction step.

Consider the IVP:

$$\begin{cases} z'(t) = f(t,z), \\ z(t_0) = z_0, \end{cases}$$

Similar to the Euler method, Heun's method begins by estimating the next step using the explicit Euler approximation:

$$\tilde{z}_{k+1} = z_k + \Delta t f(t_k, z_k).$$

This intermediate value $\tilde{z}_{k+1}$ is then used to refine the estimate by averaging the slopes at t_k and t_{k+1}:

$$z_{k+1} = z_k + \frac{\Delta t}{2}\left[f(t_k, z_k) + f(t_{k+1}, \tilde{z}_{k+1})\right].$$

This two-stage approach makes Heun's method **more accurate than explicit Euler**, achieving a global error of $\mathcal{O}(\Delta t^2)$ instead of $\mathcal{O}(\Delta t)$.

Convergence and Stability

Heun's method is second-order accurate, meaning that its local truncation error is $\mathcal{O}(\Delta t^3)$, leading to a global error of $\mathcal{O}(\Delta t^2)$. Compared to the explicit Euler method, it provides improved accuracy for the same step size.

In terms of stability, Heun's method is **conditionally stable** and has a larger stability region than explicit Euler. However, it still suffers from stability constraints in stiff problems, requiring smaller time steps to maintain numerical stability.

3.2.2 Collatz Method

The Collatz method, is another improvement over the Euler method. It introduces multiple stages to achieve better accuracy by approximating the solution using additional intermediate steps.

Formulation of the Collatz Method

For a given step size Δt, the Collatz method approximates the solution as follows:

1. Compute an initial (half) Euler step:

$$\tilde{z} = z_k + \frac{\Delta t}{2} f\left(t_k, z_k\right).$$

2. Compute a midpoint correction:

$$z_{k+1} = z_k + \Delta t f\left(t_k + \frac{h}{2}, \tilde{z}\right).$$

This results in an overall second-order accuracy, improving both the accuracy and stability of the method over the simple Euler approach.

Convergence and Stability

Like Heun's method, the Collatz method achieves a global error of $\mathcal{O}(\Delta t^2)$, making it more accurate than the first-order Euler method. Its stability region is also significantly larger, making it more suitable for problems where Euler's method struggles.

However, similar to Heun's method, the Collatz method is still **not unconditionally stable** and may require small step sizes for stiff problems.

3.2.3 Comparison of Methods

Method	Order of Accuracy	Stability	Advantages	Disadvantages
Explicit Euler	$\mathcal{O}(\Delta t)$	Conditionally stable	Simple and fast	Requires small Δt for stability
Heun's Method	$\mathcal{O}(\Delta t^2)$	Improved but not unconditionally stable	More accurate than Euler	Still requires small h for stiff problems
Collatz Method	$\mathcal{O}(\Delta t^2)$	Improved but not unconditionally stable	Better than Heun for some problems	Still requires small h for stiff problems

Both Heun's and Collatz methods improve upon the Euler method by incorporating corrections that enhance accuracy and stability, making them more suitable for practical numerical integration.



4. Runge-Kutta Methods

Although the Euler method works adequately in many applications, its accuracy and efficiency diminish for large problems, high dimensions, or stiff equations. For more accurate results, especially when smaller error tolerances are required, we turn to *Runge-Kutta methods* (see [5],[9])

4.1 Explicit Runge-Kutta Methods

The general form of an explicit Runge-Kutta method is a one-step method that approximates the solution by using multiple intermediate stages within each time step. The method is represented as follows:

$$
\begin{aligned}
\boldsymbol{z}_{k+1} &= \boldsymbol{z}_k + \sum_{i=1}^{s} b_i \boldsymbol{K}_i, \\
&\begin{cases}
\boldsymbol{K}_1 = h\boldsymbol{f}\left(t_k, \boldsymbol{z}_k\right), \\
\boldsymbol{K}_2 = h\boldsymbol{f}\left(t_k + hc_2, \boldsymbol{z}_k + a_{21}\boldsymbol{K}_1\right), \\
\boldsymbol{K}_3 = h\boldsymbol{f}\left(t_k + hc_3, \boldsymbol{z}_k + a_{31}\boldsymbol{K}_1 + a_{32}\boldsymbol{K}_2\right), \\
\quad \vdots \\
\boldsymbol{K}_s = h\boldsymbol{f}\left(t_k + hc_s, \boldsymbol{z}_k + \sum_{j=1}^{s-1} a_{sj}\boldsymbol{K}_j\right),
\end{cases}
\end{aligned}
$$

where $\boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_s$ are intermediate values called **stages**, and s is the number of stages. The constants a_{ij}, b_i, and c_i define the specific Runge-Kutta method and are chosen to achieve higher-order accuracy. The method computes multiple weighted sums of function evaluations (the $\boldsymbol{K}_i$ values) to achieve more accurate approximations of the solution.

The values $t_k + hc_i$ represent quadrature points in the time interval $[t_k, t_{k+1}]$, and the quantities K_i/h approximate the function values at these points.

4.1.1 Butcher Tableau

To simplify the notation for Runge-Kutta methods, the coefficients a_{ij}, b_i, and c_i are often summarized in a compact form called the *Butcher tableau*. The tableau for the explicit Runge-Kutta method is given by:

$$\begin{array}{c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 & 0 & 0 \\ c_3 & a_{31} & a_{32} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

Explicit Euler

The Explicit Euler Method Represented Using a Butcher Tableau:

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \tag{4.1}$$

Midpoint method

The Midpoint Method, also known as the Collatz Method, is a variant of the explicit Euler Method. This approach is classified as a second-order method and consists of two stages, allowing for improved accuracy in numerical integration.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \tag{4.2}$$

Heun's method

Heun's Method, often referred to as the explicit trapezoid rule, enhances the standard explicit Euler Method. This technique is a second-order method that employs two stages to achieve greater accuracy in numerical integration.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \tag{4.3}$$

Ralston's method

Ralston's method is a second-order technique with two stages, specifically designed to achieve a minimal local error bound.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 2/3 & 2/3 & 0 \\ \hline & 1/4 & 3/4 \end{array} \tag{4.4}$$

Generic second-order method

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \alpha & \alpha & 0 \\ \hline & 1-\dfrac{1}{2\alpha} & \dfrac{1}{2\alpha} \end{array} \qquad \alpha \in (0,1) \tag{4.5}$$

Runge-Kutta's third-order method (RK3)

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{4.6}$$

Heun's third-order method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 \\ 1/3 & 0 & 2/3 & 0 \\ \hline & 1/4 & 0 & 3/4 \end{array} \tag{4.7}$$

Ralston's third-order method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 3/4 & 0 & 3/4 & 0 \\ \hline & 2/9 & 1/3 & 4/9 \end{array} \tag{4.8}$$

Strong Stability Preserving Runge-Kutta (SSPRK3)

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1/2 & 1/4 & 1/4 & 0 \\ \hline & 1/6 & 1/6 & 2/3 \end{array} \tag{4.9}$$

Generic third-order method

With $\alpha \neq 0, 2/3, 1$:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \alpha & \alpha & 0 & 0 \\ 1 & 1+\dfrac{1-\alpha}{\alpha(3\alpha-2)} & -\dfrac{1-\alpha}{\alpha(3\alpha-2)} & 0 \\ \hline & \dfrac{1}{2}-\dfrac{1}{6\alpha} & \dfrac{1}{6\alpha(1-\alpha)} & \dfrac{2-3\alpha}{6\alpha(1-\alpha)} \end{array} \tag{4.10}$$

Classical Runge-Kutta's method (RK4)

The classical fourth-order Runge-Kutta:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array} \tag{4.11}$$

Ralston's fourth-order method

It is a second-order method with two stages, designed to minimize truncation error.

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 2/5 & 2/5 & 0 & 0 & 0 \\ c_3 & a_{3,1} & a_{3,2} & 0 & 0 \\ 1 & a_{4,1} & a_{4,2} & a_{4,3} & 0 \\ \hline & & & & \\ & b_1 & b_2 & b_3 & b_4 \end{array} \tag{4.12}$$

where

$$\begin{aligned} b_1 &= \frac{263}{1812} + \frac{2}{151}\sqrt{5} \\ b_2 &= \frac{125}{3828} - \frac{250}{957}\sqrt{5} \\ b_3 &= \frac{3426304}{5924787} + \frac{553984}{1974929}\sqrt{5} \\ b_4 &= \frac{10}{41} - \frac{4}{123}\sqrt{5} \\ c_3 &= \frac{7}{8} - \frac{3}{16}\sqrt{5} \\ a_{3,1} &= \frac{357}{256}\sqrt{5} - \frac{2889}{1024} \\ a_{4,1} &= \frac{1047}{3020}\sqrt{5} - \frac{673}{1208} \\ a_{3,2} &= \frac{3785}{1024} - \frac{405}{256}\sqrt{5} \\ a_{4,2} &= -\frac{975}{2552} - \frac{1523}{1276}\sqrt{5} \\ a_{4,3} &= \frac{93408}{48169} + \frac{203968}{240845}\sqrt{5} \end{aligned}$$

3/8-rule fourth-order method

It is a second-order method with two stages that minimizes truncation error.

0	0	0	0	0
1/3	1/3	0	0	0
2/3	−1/3	1	0	0
1	1	−1	1	0
	1/8	3/8	3/8	1/8

4.2 Implicit Runge-Kutta Methods

The methods discussed so far are all *explicit Runge-Kutta methods*. However, when dealing with *stiff* equations, it is advisable to use *implicit Runge-Kutta methods*.

The generic *implicit Runge-Kutta method* is a one-step method characterized by the following representation:

$$
\begin{aligned}
\boldsymbol{z}_{k+1} &= \boldsymbol{z}_k + \sum_{i=1}^{s} b_i \boldsymbol{K}_i \\
&\begin{cases}
\boldsymbol{K}_1 = h\boldsymbol{f}\left(t_k + hc_1, \boldsymbol{z}_k + \sum_{j=1}^{s} a_{1j}\boldsymbol{K}_j\right) \\
\boldsymbol{K}_2 = h\boldsymbol{f}\left(t_k + hc_2, \boldsymbol{z}_k + \sum_{j=1}^{s} a_{2j}\boldsymbol{K}_j\right) \\
\quad \vdots \\
\boldsymbol{K}_s = h\boldsymbol{f}\left(t_k + hc_s, \boldsymbol{z}_k + \sum_{j=1}^{s} a_{sj}\boldsymbol{K}_j\right)
\end{cases}
\end{aligned}
\tag{4.13}
$$

The butcher tableau for Equation 4.13 will be then of the form:

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	$b_s s$

Implicit Euler method

The Implicit Euler Method Represented Using a Butcher Tableau:

1	1
	1

(4.14)

Implicit Midpoint Method

The implicit midpoint method is an adaptation of the explicit Euler method, often referred to as the Collatz method. It is a second-order method characterized by two stages, providing enhanced accuracy for solving ordinary differential equations.

$$\begin{array}{c|cc} 1/2 & 1/2 \\ \hline & 1 \end{array} \tag{4.15}$$

Crank-Nicolson method

The Crank-Nicolson method corresponds to the implicit trapezoidal rule.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \tag{4.16}$$

Gauss-Legendre methods

These methods are based on the points of Gauss-Legendre quadrature.

Gauss-Legendre second-order method

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2}+\frac{\sqrt{3}}{2} & \frac{1}{2}-\frac{\sqrt{3}}{2} \end{array} \tag{4.17}$$

Gauss-Legendre sixth-order method

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \\ & -\frac{5}{6} & \frac{8}{3} & -\frac{5}{6} \end{array} \tag{4.18}$$

Lobatto methods

Lobatto IIIA fourth-order method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 5/24 & 1/3 & -1/24 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{4.19}$$

Lobatto IIIB fourth-order method

$$\begin{array}{c|ccc} 0 & 1/6 & -1/6 & 0 \\ 1/2 & 1/6 & 1/3 & 0 \\ 1 & 1/6 & 5/6 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{4.20}$$

Lobatto IIIC fourth-order method

$$\begin{array}{c|ccc} 0 & 1/6 & -1/3 & 1/6 \\ 1/2 & 1/6 & 5/12 & -1/12 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{4.21}$$

Lobatto IIIC* fourth-order method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/4 & 1/4 & 0 \\ 1 & 0 & 1 & 0 \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{4.22}$$

Radau methods

Radau IA fifth-order method

$$\begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{3}{5}-\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & \frac{11}{45}-\frac{43\sqrt{6}}{360} \\ \frac{3}{5}+\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{43\sqrt{6}}{360} & \frac{11}{45}-\frac{7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{4}{9}-\frac{\sqrt{6}}{36} \end{array} \tag{4.23}$$

Radau II fifth-order method

$$
\begin{array}{c|cccc}
\frac{2}{5}-\frac{\sqrt{6}}{10} & \frac{11}{45}-\frac{7\sqrt{6}}{360} & \frac{37}{225}-\frac{169\sqrt{6}}{1800} & -\frac{2}{225}+\frac{\sqrt{6}}{75} \\
\frac{2}{5}+\frac{\sqrt{6}}{10} & \frac{37}{225}+\frac{169\sqrt{6}}{1800} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & -\frac{2}{225}-\frac{\sqrt{6}}{75} \\
1 & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \\
\hline
 & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9}
\end{array}
\tag{4.24}
$$

uations (DAEs) and Differential Index

II

5 DAEs Forms 51
5.1 Semi-Explicit Form of DAEs 51
5.2 Index-0 DAEs are ODEs 52
5.3 Index-1 DAEs in Semi-Explicit Form 52
5.4 The Implicit Function Theorem 53
DAE of index 2 53
5.5 DAE of index 3 54
5.6 DAE in semi-esplicit form 55

6 The Index of a DAE 57
6.1 Differential Index Definition 57
6.2 Example 1 58
6.3 Example 2 58
6.4 Example 3 59
Summary 59

Differential-Algebraic Equations (DAEs) arise naturally in many applications, including constrained mechanical systems, electrical circuits, chemical kinetics, and optimal control. Unlike ordinary differential equations (ODEs), DAEs include algebraic constraints that must be satisfied alongside the differential dynamics This added structure introduces fundamental differences in both the theoretical analysis and numerical treatment of such systems.

One of the most important characteristics of a DAE is its **index**. Informally, the index measures how many times the algebraic constraints must be differentiated with respect to time in order to rewrite the DAE as an explicit ODE. The index provides critical insight into the inherent difficulty of the problem:

- low-index DAEs are often tractable with standard numerical solvers
- while high-index systems may be ill-posed or require specialized techniques.

In this chapter, we introduce the concept of the **differential index** and explain how it arises from the structure of the DAE. We discuss its formal definition, illustrate it with concrete examples, and explore its implications for both analytical understanding and numerical solution.

Learning objectives

After this part, you should be able to state the local nature of an index, check the regularity assumptions behind a semi-explicit DAE, and determine the differential index in elementary examples.



5. DAEs Forms

Many physical systems are governed not by ordinary differential equations (ODEs) but by **differential-algebraic equations (DAEs)**, which combine differential dynamics with algebraic constraints. A general DAE can be written as:

$$\boldsymbol{f}(\boldsymbol{x}'(t),\boldsymbol{x}(t),t) = \boldsymbol{0},$$

where $bmx(t) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{x}'(t)$ is its time derivative, and $\boldsymbol{f}$ represents a (possibly nonlinear) vector function.

Unlike ODEs, DAEs may not permit direct solution for $\boldsymbol{x}'(t)$ due to the potential **singularity** of the Jacobian:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}'} = \frac{\partial \boldsymbol{f}(\boldsymbol{x}',\boldsymbol{x},t)}{\partial \boldsymbol{x}'}.$$

When this Jacobian is singular (i.e., not invertible), implicit or indirect techniques are needed. The **differential index** of a DAE quantifies how many times we must differentiate the equation to isolate $\boldsymbol{x}'(t)$.

5.1 Semi-Explicit Form of DAEs

A more structured (and often more tractable) form of DAEs is:

$$\boldsymbol{A}(\boldsymbol{x},t)\boldsymbol{x}' = \boldsymbol{g}(\boldsymbol{x},t),$$

where $\boldsymbol{A}(\boldsymbol{x},t)$ is singular. When $\boldsymbol{A}(\boldsymbol{x},t)$ has a **block-diagonal structure**:

$$\boldsymbol{A}(\boldsymbol{x},t) = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix},$$

the DAE is said to be in **semi-explicit form**.

Why is this useful?

- The semi-explicit form naturally separates the system into **differential** and **algebraic** components:
  - **Differential part**: Explicit ODEs (e.g., $\boldsymbol{y}' = \boldsymbol{h}(\boldsymbol{y},\boldsymbol{z},t)$).
  - **Algebraic part**: Constraints (e.g., $\boldsymbol{0} = \boldsymbol{k}(\boldsymbol{y},\boldsymbol{z},t)$).

- This separation simplifies **index analysis** (discussed later).

> **General DAE to semi-explicit form**
>
> A generic DAE:
>
> $$\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{x}, t) = \boldsymbol{0}$$
>
> can **always** be rewritten in semi-explicit form by introducing the auxiliary variable $\boldsymbol{x}_{\text{dot}}(t) = \boldsymbol{x}'(t)$:
>
> $$\boldsymbol{x}' = \boldsymbol{x}_{\text{dot}}$$
>
> $$\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{x}_{\text{dot}}, \boldsymbol{x}, t)$$
>
> **Important:**
> - This transformation **increases the differential index by 1**.
> - For example, an **index-1 DAE** becomes **index-2** after conversion.

5.2 Index-0 DAEs are ODEs

By definition, **index-0 DAEs** are simply ODEs of the form:

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

where the system is fully explicit with no algebraic constraints.

5.3 Index-1 DAEs in Semi-Explicit Form

An **index-1 DAE** in **semi-explicit form** is:

$$\begin{cases} \boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, t) & \text{(Differential part)}, \\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t) & \text{(Algebraic part)}, \end{cases}$$

where:
- $\boldsymbol{x} \in \mathbb{R}^n$ are the **differential variables**,
- $\boldsymbol{y} \in \mathbb{R}^m$ are the **algebraic variables**,
- $\boldsymbol{f} \in C^1(\mathbb{R}^{n+m+1}, \mathbb{R}^n)$,
- $\boldsymbol{g} \in C^1(\mathbb{R}^{n+m+1}, \mathbb{R}^m)$.

The Jacobian of $\boldsymbol{g}$ w.r.t. $\boldsymbol{y}$,

$$\boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{y}, t) = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}},$$

must be **non-singular**. This ensures that the **Implicit Function Theorem** applies, allowing us to express $\boldsymbol{y}$ as a function of $\boldsymbol{x}$ and t:

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}, t).$$

Substituting back into the differential part reduces the DAE to an ODE:

$$\boldsymbol{x}'(t) = \boldsymbol{g}(\boldsymbol{x}(t),t), \qquad \text{where} \qquad \boldsymbol{g}(\boldsymbol{x},t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t)$$

5.4 The Implicit Function Theorem

The **Implicit Function Theorem** ensures that the algebraic constraints can be locally solved for $\boldsymbol{y}$.

Theorem 5.1 Let $A \subset \mathbb{R}^{n+m}$ an open set and $\boldsymbol{f} \in C^1(A, \mathbb{R}^m)$, moreover

1. $(\boldsymbol{x}_0, \boldsymbol{y}_0) \in A$;
2. $\boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{y}_0) = \boldsymbol{0}$;
3. $\dfrac{\partial \boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{y}_0)}{\partial \boldsymbol{y}}$ non-singular.

Then, there exists two open set

$$U \subset \mathbb{R}^n \quad \text{and} \quad V \subset \mathbb{R}^m$$

and a function $\boldsymbol{\phi} : U \mapsto V$ such that

- $\boldsymbol{y}_0 = \boldsymbol{\phi}(\boldsymbol{x}_0)$;
- $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{0}$ iff $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ (for $\boldsymbol{x} \in U$ and $\boldsymbol{y} \in V$);
- $\boldsymbol{\phi} \in C^1(U,V)$

and

$$\frac{\partial \boldsymbol{\phi}(\boldsymbol{x})}{\partial \boldsymbol{x}} = -\left(\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\phi}(\boldsymbol{x}))}{\partial \boldsymbol{y}}\right)^{-1} \frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\phi}(\boldsymbol{x}))}{\partial \boldsymbol{x}}$$

5.4.1 Illustrative Example: A Simple Index-1 DAE

Consider the semi-explicit DAE:

$$\begin{aligned} x' &= -y, \\ 0 &= x + y - \sin(t). \end{aligned}$$

Step 1: Apply the Implicit Function Theorem

The algebraic equation is:

$$y = \sin(t) - x.$$

Step 2: Substitute Back into the ODE

The system reduces to:

$$x' = -\sin(t) + x,$$

which is an ODE solvable with standard methods.

DAE of index 2

DAE of index 1 are in practice ODE with implicit parameters. Index 2 DAE are more complex

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)$$

$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},t)$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{y}(t) \in \mathbb{R}^m$, $\boldsymbol{f} \in C^1(\mathbb{R}^{n+m+1}, \mathbb{R}^n)$ and $\boldsymbol{g} \in C^1(\mathbb{R}^{n+1}, \mathbb{R}^m)$. Moreover

$$\frac{\partial \boldsymbol{g}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \times \frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}}$$

ia a **non singular** matrix. In this case **implicit function Theorem** cannot be used to determine $\boldsymbol{y}$ as a function of $\boldsymbol{x}$.

To check the index perform one derivation of the constraints

$$\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}(t),t)}{\mathrm{d}t} = \frac{\partial \boldsymbol{g}(\boldsymbol{x}(t),t)}{\partial \boldsymbol{x}}\boldsymbol{x}'(t) + \frac{\partial \boldsymbol{g}(\boldsymbol{x}(t),t)}{\partial t} = \boldsymbol{0}$$

obtaining the new (hidden) constraints

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) = \boldsymbol{0}$$

where

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) = \frac{\partial \boldsymbol{g}(\boldsymbol{x},t)}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) + \frac{\partial \boldsymbol{g}(\boldsymbol{x},t)}{\partial t}$$

Thus, a new DAE is obtained

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)$$

$$\boldsymbol{0} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t)$$

The derived DAE is now of index 1:

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)$$

$$\boldsymbol{0} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t)$$

to check it must be

$$\frac{\partial \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}} = \frac{\partial \boldsymbol{g}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \times \frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}}$$

non singular.

5.5 DAE of index 3

Index 3 DAE semi-esplicit take the form

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)$$

$$\boldsymbol{y}' = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)$$

$$\boldsymbol{0} = \boldsymbol{h}(\boldsymbol{y},t)$$

where

$$\boldsymbol{x}(t) \in \mathbb{R}^n \qquad \boldsymbol{y}(t) \in \mathbb{R}^m \qquad \boldsymbol{z}(t) \in \mathbb{R}^p \, \boldsymbol{f} \in C^1(\mathbb{R}^{n+m+p+1}, \mathbb{R}^n) \quad \boldsymbol{g} \in C^1(\mathbb{R}^{n+m+1}, \mathbb{R}^m) \quad \boldsymbol{h} \in C^1(\mathbb{R}^{m+1}, \mathbb{R}^n)$$

To check the index derive the constraints

$$\frac{\mathrm{d}\boldsymbol{h}(\boldsymbol{y}(t),t)}{\mathrm{d}t} = \frac{\partial \boldsymbol{h}(\boldsymbol{y}(t),t)}{\partial \boldsymbol{y}}\boldsymbol{y}'(t) + \frac{\partial \boldsymbol{h}(\boldsymbol{y}(t),t)}{\partial t} = \boldsymbol{0}$$

obtaining the new (hidden) constraints

$$\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)=\boldsymbol{0}$$

where

$$\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)=\frac{\partial \boldsymbol{h}(\boldsymbol{y},t)}{\partial \boldsymbol{y}}\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)+\frac{\partial \boldsymbol{h}(\boldsymbol{y},t)}{\partial t}$$

Thus, a new DAE is obtained

$$\boldsymbol{x}'=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)$$

$$\boldsymbol{y}'=\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)$$

$$\boldsymbol{0}=\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)$$

The DAE

$$\begin{pmatrix}\boldsymbol{x}'\\ \boldsymbol{y}'\end{pmatrix}=\begin{pmatrix}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)\\ \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)\end{pmatrix}$$

$$\boldsymbol{0}=\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)$$

is index 2 iff

$$\frac{\partial \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{x}}\times\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)}{\partial \boldsymbol{z}}+\frac{\partial \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}}\times\frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{z}}$$

is non singular, i.e.

$$\frac{\partial \boldsymbol{h}(\boldsymbol{y},t)}{\partial \boldsymbol{y}}\times\frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{x}}\times\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)}{\partial \boldsymbol{z}}$$

is non singular.

5.6 DAE in semi-esplicit form

DAE of index 1

$$\begin{cases}\boldsymbol{x}'=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)\\ \boldsymbol{0}=\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)\end{cases}\qquad \frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}}\quad \textbf{non singular}$$

DAE of index 2

$$\begin{cases}\boldsymbol{x}'=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)\\ \boldsymbol{0}=\boldsymbol{g}(\boldsymbol{x},t)\end{cases}\qquad \underbrace{\frac{\partial \boldsymbol{g}(\boldsymbol{x},t)}{\partial \boldsymbol{x}}\times\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}}}_{\textbf{non singular}}$$

DAE of index 3

$$\begin{cases}\boldsymbol{x}'=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)\\ \boldsymbol{y}'=\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)\\ \boldsymbol{0}=\boldsymbol{h}(\boldsymbol{y},t)\end{cases}\qquad \underbrace{\frac{\partial \boldsymbol{h}(\boldsymbol{y},t)}{\partial \boldsymbol{y}}\times\frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{x}}\times\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},t)}{\partial \boldsymbol{z}}}_{\textbf{non singular}}$$

DAE of index μ (in the Hessemberg form)

The DAE

$$
\begin{aligned}
x_1' &= f_1(x_1, x_2, \ldots, x_\mu, t) \\
x_2' &= f_2(x_1, x_2, \ldots, x_{\mu-1}, t) \\
&\vdots \\
x_k' &= f_k(x_{k-1}, x_k, \ldots, x_{\mu-1}, t) \qquad k = 3, \ldots, \mu - 1 \\
&\vdots \\
0 &= f_\mu(x_{\mu-1}, t)
\end{aligned}
$$

with

$$
\frac{\partial f_\mu}{\partial x_{\mu-1}} \times \frac{\partial f_{\mu-1}}{\partial x_{\mu-2}} \times \cdots \times \frac{\partial f_2}{\partial x_1} \times \frac{\partial f_1}{\partial x_\mu}
$$

non singular is a DAE of index μ.



6. The Index of a DAE

One of the most important characteristics of a DAE is its **index**. Informally, the index measures how many times the algebraic constraints must be differentiated with respect to time in order to rewrite the DAE as an explicit ODE. Several index notions exist in the literature—such as the tractability index and perturbation index—but we focus primarily on the differential index, which is most directly connected to the process of index reduction and integration.

6.1 Differential Index Definition

Throughout this chapter, an index statement is local: it is valid only where the Jacobians required to solve the differentiated equations have full rank. This regularity condition is essential both for the definition and for the examples below.

A DAE has **differential index** q if q is the smallest integer such that repeated differentiation yields a system where $\boldsymbol{x}'(t)$ can be expressed explicitly in terms of $\boldsymbol{x}(t)$ and t:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}(\boldsymbol{x}',\boldsymbol{x},t) &= \boldsymbol{0},\\
\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{f}(\boldsymbol{x}',\boldsymbol{x},t) &= \boldsymbol{0},\\
&\vdots\\
\frac{\mathrm{d}^q}{\mathrm{d}t^q}\boldsymbol{f}(\boldsymbol{x}',\boldsymbol{x},t) &= \boldsymbol{0}.
\end{aligned}$$

Definition 6.1 — Differential Index. A DAE system

$$\boldsymbol{f}(\boldsymbol{x}'(t),\boldsymbol{x}(t),t) = \boldsymbol{0}$$

has **differential index** q if q is the minimum number of time differentiations needed to obtain an explicit expression

$$\boldsymbol{x}'(t) = \boldsymbol{g}(\boldsymbol{x}(t),t)$$

for $\boldsymbol{x}'(t)$.

6.2 Example 1

6.2.1 Index-2 DAE

Consider the system:

$$x(t) = \sin(t), \qquad x'(t) + y(t) = 0. \tag{6.1}$$

The first equation fixes $x(t)$ and the second fixes $y(t)$, but neither equation directly provides an ODE for the complete state (x, y). First, differentiate the constraint $x(t) - \sin(t) = 0$ once:

1. $$x'(t) = \cos(t).$$

   Together with $x'(t) + y(t) = 0$, this gives $y(t) = -\cos(t)$, but it does not yet provide an equation for $y'(t)$.
2. Differentiate the constraint a second time and the differential equation once:

$$x''(t) = -\sin(t), \qquad x''(t) + y'(t) = 0 \quad \Rightarrow \quad y'(t) = -x''(t) = \sin(t).$$

Since two differentiations were needed to convert to an explicit ODE system, the **differential index** is $q = 2$.

6.3 Example 2

6.3.1 Time-Varying Coefficient DAE

Consider the time-dependent system from [4]:

$$\boldsymbol{E}(t) \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} + \boldsymbol{A}(t) \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \boldsymbol{c}(t),$$

where:

$$\boldsymbol{E}(t) = \begin{pmatrix} 1 & t \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}(t) = \begin{pmatrix} 0 & 0 \\ 1 & t \end{pmatrix}, \quad \boldsymbol{c}(t) = \begin{pmatrix} c_1(t) \\ c_2(t) \end{pmatrix}.$$

Step 1: System Equations

From the matrix form:

$$\begin{cases} x'(t) + t y'(t) = c_1(t), \\ x(t) + t y(t) = c_2(t). \end{cases}$$

Differentiate the second equation:

$$x'(t) + t y'(t) + y(t) = c_2'(t).$$

Substitute from the first equation:

$$c_1(t) + y(t) = c_2'(t) \Rightarrow y(t) = c_2'(t) - c_1(t).$$

Step 2: Derive ODEs

Differentiate $y(t)$:

$$y'(t) = c_2''(t) - c_1'(t).$$

Substitute back into the first equation:

$$x'(t) = c_1(t) + t(c_1'(t) - c_2''(t)).$$

Thus, the system becomes:

$$\begin{cases} x'(t) = c_1(t) + t(c_1'(t) - c_2''(t)), \\ y'(t) = c_2''(t) - c_1'(t). \end{cases}$$

Two differentiations were needed: hence, the **differential index is 2**.

6.4 Example 3

6.4.1 Index-1 DAE

Consider the system:

$$\begin{aligned} x'(t) &= y(t), \\ 0 &= x(t)^2 + y(t)^2 - 1. \end{aligned}$$

Differentiate the algebraic constraint:

$$2x(t)x'(t) + 2y(t)y'(t) = 0.$$

Substitute $x'(t) = y(t)$:

$$2x(t)y(t) + 2y(t)y'(t) = 0 \Rightarrow y'(t) = -x(t),$$

provided $y(t) \neq 0$. Thus the DAE is index 1 locally on the regular part of the constraint manifold. At points where $y = 0$, the Jacobian of the algebraic constraint with respect to the algebraic variable is singular, so this example does not satisfy the regularity hypothesis there.

Now we have an explicit ODE system:

$$x'(t) = y(t), \qquad y'(t) = -x(t).$$

Only one differentiation was needed: the system has **index 1**.

Summary

- The **differential index** measures how far a DAE is from being an ODE.
- Higher-index systems are more challenging to analyze and simulate.
- Index-1 DAEs are typically well-posed and numerically tractable.
- Index-2 or higher systems often require **index reduction** techniques before numerical integration.

III Linear DAE

7 Linear DAE examples and properties . 65
7.1 Separation of algebraic and differential part . 69

8 Gaussian Elimination and Echelon form 73
8.1 Gaussian Elimination 73
8.2 Echelon Form 78

9 Gauss-based Method for separating the algebraic part from the differential part 83
9.1 Formulation of the DAE 83
9.2 Change of Coordinates 83
9.3 Rank Increase Observation 84

In this chapter, we will explore linear implicit differential equations, also known as Differential Algebraic Equations (DAEs), characterized by the following form:

$$\boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{f}(t) \tag{6.2}$$

In the simplest scenario, the matrices $\boldsymbol{E}$ and $\boldsymbol{A}$ are square constant matrices belonging to $\mathbb{R}^{n\times n}$, and $\boldsymbol{f}(t) : \mathbb{R} \to \mathbb{R}^n$ represents an external perturbation that is independent of $\boldsymbol{x}(t)$. Such equations frequently arise when linearizing autonomous nonlinear problems around constant (or critical) solutions, where $\boldsymbol{f}(t)$ acts as a perturbation that displaces the solution from the nominal trajectory. Other applications include modeling linear electrical circuits, simple mechanical systems, and, more generally, linear systems with additional algebraic constraints.

Learning objectives

After this part, you should be able to distinguish an ODE from a singular linear DAE, reduce simple linear DAEs by row operations, and identify the consistency conditions imposed on the initial state.



7. Linear DAE examples and properties

A primary consideration is the existence and uniqueness of a solution

$$\boldsymbol{x}(t) : \mathbb{R} \to \mathbb{R}^n$$

The intriguing case arises when $\boldsymbol{E}$ is singular (i.e., not invertible). In contrast, if $\boldsymbol{E}$ is non-singular, we can simply invert it, allowing us to express the DAEs as ordinary differential equations (ODEs) as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{E}^{-1}\boldsymbol{f}(t). \tag{7.1}$$

For the equation in Equation 7.1, it suffices to assume that $\boldsymbol{f}(t)$ is piecewise continuous[1], which guarantees the existence and uniqueness of solutions. For a detailed discussion, see, for instance, [6].

Conversely, when the matrix $\boldsymbol{E}$ is singular, the existence of a solution becomes uncertain. A singular matrix $\boldsymbol{E}$ indicates that ODEs are coupled with linear algebraic constraints. Therefore, it is essential to consider certain regularity conditions for the pair $(\boldsymbol{E},\boldsymbol{A})$, as illustrated in the following example.

Example

Consider the differential algebraic equations (DAEs) with the state vect

$$\boldsymbol{x}(t) = (x_1(t); x_2(t)) \in \mathbb{R}^2$$

and the external perturbation $\boldsymbol{f}(t) = 0$ for all $t \in [0,\infty)$. The matrices $\boldsymbol{E}$ and $\boldsymbol{A}$ are defined as follows:

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

This DAE corresponds to the scalar differential equation:

$$\dot{x}_1(t) = x_2(t).$$

Assuming an initial condition $\boldsymbol{x}_0 = (0;0) \in \mathbb{R}^2$, a general solution can be expressed as:

[1] Weaker conditions can be applied to $\boldsymbol{f}(t)$; for instance, it is sufficient for $\boldsymbol{f}(t)$ to be Lebesgue measurable.

$$\boldsymbol{x}(t) = \int_0^t \begin{bmatrix} x_2(\tau) \\ \alpha(\tau) \end{bmatrix} \mathrm{d}\tau,$$

where $\alpha(\tau)$ is an arbitrary function that satisfies the initial condition.
For example, if we choose $\alpha(t) = \sin(t)$, which is compatible with the initial condition ($\sin(0) = 0$), the solution becomes:

$$\boldsymbol{x}(t) = \begin{bmatrix} -\sin(t) \\ -\cos(t) \end{bmatrix}.$$

Alternatively, if we select $\alpha(t) = t$, the solution is given by:

$$\boldsymbol{x}(t) = \frac{1}{6} \begin{bmatrix} t^3 \\ 3t^2 \end{bmatrix}.$$

This illustrates that the solution is not unique, highlighting the need for certain regularity assumptions regarding the pair $(\boldsymbol{E}, \boldsymbol{A})$.

To establish the existence and uniqueness of a solution for equation Equation 6.2, we need to introduce the following definition.

Definition 7.1 — Regular Pencil. A matrix pair $(\boldsymbol{E}, \boldsymbol{A})$ is defined as a *regular pencil* if there exists a scalar $\lambda_0 \in \mathbb{R}$ such that:

$$\left|\boldsymbol{A} - \lambda_0 \boldsymbol{E}\right| \neq 0.$$

Example

Consider the following pair of matrices $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{3\times 3}$, which represents a non-regular (singular) pencil:

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 0 & 0 \\ 0 & -2 & 0 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

For this pair, the determinant of the pencil evaluates to zero for every value of $\lambda_0 \in \mathbb{R}$:

$$\left|\boldsymbol{A} - \lambda_0 \boldsymbol{E}\right| = \begin{vmatrix} 1-\lambda_0 & 0 & 0 \\ 2\lambda_0 & 1 & 0 \\ 0 & 2\lambda_0 & 0 \end{vmatrix} = 0.$$

The central idea is that if the pair $(\boldsymbol{E}, \boldsymbol{A})$ is a regular pencil, then we can simplify the problem through an appropriate change of coordinates. This concept is further clarified by the following theorem.

Theorem 7.1 — Weierstrass Canonical Form. A matrix pair $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{n\times n}$ is classified as a *regular pencil* if, and only if, there exist invertible matrices

$$\boldsymbol{P} \in \mathbb{R}^{n\times n} \quad \text{and} \quad \boldsymbol{Q} \in \mathbb{R}^{n\times n}$$

such that:

$$(\boldsymbol{PEQ}, \boldsymbol{PAQ}) = \left(\begin{bmatrix} \boldsymbol{N} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}, \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{J} \end{bmatrix} \right), \tag{7.2}$$

where $\boldsymbol{N}$ is an upper triangular nilpotent matrix, $\boldsymbol{J}$ is a matrix in Jordan canonical form, and $\boldsymbol{I}$ is the identity matrix of appropriate dimensions.

To enhance understanding, we will briefly recall the definition of a nilpotent matrix.

Definition 7.2 — Nilpotent Matrix. A square matrix $\boldsymbol{N}$ is termed *nilpotent* if there exists an integer $v \in \mathbb{N}$ such that:

$$\boldsymbol{N}^v = 0.$$

Remark 7.1 It is important to note that the smallest integer v for which $\boldsymbol{N}^v = 0$ is referred to as the *degree* of the nilpotent matrix $\boldsymbol{N}$. Furthermore, any upper triangular matrix that has zeros along its main diagonal qualifies as a nilpotent matrix.

Example

Consider the matrix $\boldsymbol{N} \in \mathbb{R}^{3\times 3}$ defined as follows:

$$\boldsymbol{N} = \begin{bmatrix} 0 & 1 & 3 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

This matrix is nilpotent with a **degree** $v = 3$. To illustrate this, we compute the following powers of $\boldsymbol{N}$:

$$\boldsymbol{N}^2 = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{N}^3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The decoupling form presented in Equation 7.2 is known as the *Weierstrass canonical form*. For further details, refer to [7]. The matrices $\boldsymbol{N}$ and $\boldsymbol{J}$ in this form possess the following structures:

$$\boldsymbol{N} = \begin{bmatrix} \boldsymbol{N}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{N}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{N}_{n_1} \end{bmatrix},$$

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{J}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{J}_{n_2}. \end{bmatrix}$$

Here, $\boldsymbol{N}_i$ for $i = 1, \dots, n_1$ and $\boldsymbol{J}_i$ for $i = 1, \dots, n_2$ are Jordan blocks. The blocks $\boldsymbol{N}_i$ are associated with the zero eigenvalues of the matrix $\boldsymbol{E}$, and are structured as follows:

$$\boldsymbol{N}_i = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & 0 \end{bmatrix}, \quad i = 1, \dots, n_1.$$

The dimension of each block $\boldsymbol{N}_i$ corresponds to the difference between the algebraic and geometric multiplicities of the respective eigenvalue. Conversely, the blocks $\boldsymbol{J}_i$ for $i = 1, \dots, n_2$ are classical Jordan blocks, structured as follows:

$$\boldsymbol{J}_k = \begin{bmatrix} \lambda_k & 1 & & & \\ & \lambda_k & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda_k \end{bmatrix}, \quad k = 1, \ldots, n_2.$$

Next, we provide a constructive proof of the Weierstrass canonical form. This proof is based on a sequential multiplication of non-singular matrices. We will demonstrate only the sufficiency part; for the necessity, please refer to [7].

Proof. Let us consider the regular pencil pair $(\boldsymbol{E}, \boldsymbol{A})$. We can express this as:

$$(\boldsymbol{E}, \boldsymbol{A} + \lambda_0 \boldsymbol{E} - \lambda_0 \boldsymbol{E}).$$

Now, define the matrix $\boldsymbol{T}_0 = (\boldsymbol{A} - \lambda_0 \boldsymbol{E})$, which is non-singular due to the regular pencil assumption. This leads us to consider the expression:

$$\boldsymbol{T}_0^{-1}(\boldsymbol{E}, \boldsymbol{A} + \lambda_0 \boldsymbol{E} - \lambda_0 \boldsymbol{E}) = \left(\boldsymbol{T}_0^{-1}\boldsymbol{E}, \boldsymbol{I} + \lambda_0 \boldsymbol{T}_0^{-1}\boldsymbol{E}\right).$$

Next, let $\boldsymbol{T}_1$ be the non-singular transformation matrix such that $\boldsymbol{T}_1^{-1}\boldsymbol{T}_0^{-1}\boldsymbol{E}\boldsymbol{T}_1$ is in Jordan canonical form. Thus, the pair

$$\boldsymbol{T}_1^{-1}\left(\boldsymbol{T}_0^{-1}\boldsymbol{E}, \boldsymbol{I} + \lambda_0 \boldsymbol{T}_0^{-1}\boldsymbol{E}\right)\boldsymbol{T}_1$$

takes the following structure:

$$\left(\begin{bmatrix} \tilde{\boldsymbol{N}} & 0 \\ 0 & \tilde{\boldsymbol{J}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{N}} & 0 \\ 0 & \boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{J}} \end{bmatrix}\right). \tag{7.3}$$

Here, the block $\tilde{\boldsymbol{N}}$ corresponds to the zero eigenvalues of the singular matrix $\boldsymbol{T}_0^{-1}\boldsymbol{E}$. Therefore, $\tilde{\boldsymbol{N}}$ is upper triangular, with zeros on the main diagonal. As a result, the matrix $(\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{N}})$ is invertible.

Next, we define the non-singular transformation matrix $\boldsymbol{T}_2$ as follows:

$$\boldsymbol{T}_2 = \begin{bmatrix} (\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{N}}) & 0 \\ 0 & \tilde{\boldsymbol{J}} \end{bmatrix}.$$

Multiplying the expression in Equation 7.3 by $\boldsymbol{T}_2^{-1}$ from the left, we obtain the following structure:

$$\mathscr{S} = \left(\begin{bmatrix} (\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{N}})^{-1}\tilde{\boldsymbol{N}} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}, \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \tilde{\boldsymbol{J}}^{-1}(\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{J}}) \end{bmatrix}\right). \tag{7.4}$$

Finally, we need to consider matrices $\boldsymbol{T}_{31}$ and $\boldsymbol{T}_{32}$ such that:

- $\boldsymbol{T}_{31}^{-1}(\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{N}})^{-1}\tilde{\boldsymbol{N}}\boldsymbol{T}_{31}$ is in Jordan canonical form.
- $\boldsymbol{T}_{32}^{-1}\tilde{\boldsymbol{J}}^{-1}(\boldsymbol{I} + \lambda_0 \tilde{\boldsymbol{J}})\boldsymbol{T}_{32}$ is also in Jordan canonical form.

We construct the transformation matrix $\boldsymbol{T}_3$ as follows:

$$\boldsymbol{T}_3 = \begin{bmatrix} \boldsymbol{T}_{31} & 0 \\ 0 & \boldsymbol{T}_{32} \end{bmatrix}.$$

Thus, from Equation 7.4 the final change of coordinates yields:

$$\boldsymbol{T}_3^{-1}\mathscr{S}\boldsymbol{T}_3 = \left(\begin{bmatrix} \boldsymbol{N} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}, \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{J} \end{bmatrix}\right).$$

It remains to show that the matrix $\boldsymbol{N}$ is nilpotent. To do this, we need to demonstrate that the matrix

$$(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1}\tilde{\boldsymbol{N}}$$

is nilpotent. First, observe that the matrices $(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})$ and $\tilde{\boldsymbol{N}}$ commute since:

$$(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})\tilde{\boldsymbol{N}} = \tilde{\boldsymbol{N}}+\lambda_0\tilde{\boldsymbol{N}}^2 = \tilde{\boldsymbol{N}}(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}}).$$

Furthermore, $(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1}$ also commutes with $\tilde{\boldsymbol{N}}$:

$$\begin{aligned}(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1}\tilde{\boldsymbol{N}} &= (\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1}\tilde{\boldsymbol{N}}(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1}\\ &= \tilde{\boldsymbol{N}}(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{-1},\end{aligned}$$

where we have used the fact that

$$\tilde{\boldsymbol{N}}(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}}) = (\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})\tilde{\boldsymbol{N}}$$

Given that these matrices commute, let ν be the index such that $\tilde{\boldsymbol{N}}^{\nu}=0$. The following holds:

$$\left[(\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})\tilde{\boldsymbol{N}}\right]^{\nu} = (\boldsymbol{I}+\lambda_0\tilde{\boldsymbol{N}})^{\nu}\tilde{\boldsymbol{N}}^{\nu} = 0.$$

This completes the proof.

In summary, the overall transformation matrices $(\boldsymbol{P},\boldsymbol{Q})$ are given by:

$$\boldsymbol{P} = (\boldsymbol{T}_0\boldsymbol{T}_1\boldsymbol{T}_2\boldsymbol{T}_3)^{-1}, \quad \boldsymbol{Q} = \boldsymbol{T}_1\boldsymbol{T}_3.$$

■

7.1 Separation of algebraic and differential part

The Weierstrass canonical form is particularly valuable because it allows us to rewrite the system described by Equation 6.2 using the transformation matrices $(\boldsymbol{P},\boldsymbol{Q})$ defined earlier. This results in the following expression:

$$\boldsymbol{P}\boldsymbol{E}\boldsymbol{Q}\boldsymbol{Q}^{-1}\dot{\boldsymbol{x}}(t) = \boldsymbol{P}\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^{-1}\boldsymbol{x}(t)+\boldsymbol{P}\boldsymbol{f}(t).$$

Next, we apply the following change of coordinates:

$$\begin{aligned}\boldsymbol{z}(t) &= \boldsymbol{Q}^{-1}\boldsymbol{x}(t),\\ \boldsymbol{g}(t) &= \boldsymbol{P}\boldsymbol{f}(t),\end{aligned} \tag{7.5}$$

leading to the equivalent differential-algebraic equations (DAEs):

$$\begin{bmatrix}\boldsymbol{N} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{z}}_1(t)\\ \dot{\boldsymbol{z}}_2(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{I} & 0\\ 0 & \boldsymbol{J}\end{bmatrix}\begin{bmatrix}\boldsymbol{z}_1(t)\\ \boldsymbol{z}_2(t)\end{bmatrix}+\begin{bmatrix}\boldsymbol{g}_1(t)\\ \boldsymbol{g}_2(t)\end{bmatrix}.$$

In this formulation, the vectors $\boldsymbol{z}(t)$ and $\boldsymbol{g}(t)$ are partitioned according to the dimensions of matrices $\boldsymbol{J}$ and $\boldsymbol{N}$. Specifically, we have:

$$\boldsymbol{z}(t) = \begin{pmatrix}\boldsymbol{z}_1(t)\\ \boldsymbol{z}_2(t)\end{pmatrix}, \quad \boldsymbol{g}(t) = \begin{pmatrix}\boldsymbol{g}_1(t)\\ \boldsymbol{g}_2(t)\end{pmatrix}.$$

Notably, the two sets of equations corresponding to $\boldsymbol{z}_1(t)$ and $\boldsymbol{z}_2(t)$ are decoupled, yielding:

$$\begin{aligned}\boldsymbol{N}\dot{\boldsymbol{z}}_1 &= \boldsymbol{z}_1+\boldsymbol{g}_1(t),\\ \dot{\boldsymbol{z}}_2 &= \boldsymbol{J}\boldsymbol{z}_2+\boldsymbol{g}_2(t).\end{aligned} \tag{7.6}$$

7.1.1 Solution of $z_2'(t) = Jz_2(t) + g_2(t)$

The equation associated with $z_2(t)$ is straightforward; it represents an ordinary differential equation (ODE). Therefore, the unique solution given the initial condition $z_2(0)$ is:

$$z_2(t) = \mathrm{e}^{Jt} z_2(0) + \int_0^t \mathrm{e}^{J(t-\tau)} g_2(\tau)\,\mathrm{d}\tau, \quad \forall t \geq 0.$$

where matrix exponential is

$$\exp A = I + A + \frac{1}{2}A^2 + \frac{1}{6}A^3 + \cdots + \frac{1}{k!}A^k + \cdots$$

For further details, refer to [10].

7.1.2 Solution of $Nz_1'(t) = z_1(t) + g_1(t)$

On the other hand, Equation 7.6 remains a DAE. However, since the matrix N is nilpotent, we can manipulate it effectively. To facilitate this, we can rewrite Equation 7.6 as:

$$z_1(t) = N\dot{z}_1(t) - g_1(t). \tag{7.7}$$

Substituting Equation 7.7 into itself and differentiating gives us:

$$\begin{aligned}
z_1(t) &= N\frac{\mathrm{d}}{\mathrm{d}t}\left[N\dot{z}_1(t) - g_1(t)\right] - g_1(t) \\
&= N^2\frac{\mathrm{d}^2 z_1(t)}{\mathrm{d}t^2} - N\frac{\mathrm{d}g_1(t)}{\mathrm{d}t} - g_1(t) \\
&= N^3\frac{\mathrm{d}^3 z_1(t)}{\mathrm{d}t^3} - N^2\frac{\mathrm{d}^2 g_1(t)}{\mathrm{d}t^2} - \cdots \\
&\vdots \\
&= N^\nu\frac{\mathrm{d}^\nu z_1(t)}{\mathrm{d}t^\nu} - N^{\nu-1}\frac{\mathrm{d}^{\nu-1} g_1(t)}{\mathrm{d}t^{\nu-1}} - \cdots - N\frac{\mathrm{d}g_1(t)}{\mathrm{d}t} - g_1(t).
\end{aligned}$$

This process establishes differential properties of the solutions. Note that the procedure terminates when we reach the index ν of matrix N, since $N^\nu = 0$. Consequently, we can express a unique solution as a finite summation of increasing derivatives of $g(t)$:

$$z_1(t) = -\sum_{k=0}^{\nu-1} N^k \frac{\mathrm{d}^k g_1(t)}{\mathrm{d}t^k}.$$

Remark 7.2 The index ν is sometimes referred to as the *Kronecker index*. It is important to note that ν represents the number of times the function $g_1(t)$ is differentiated in order to obtain the solution $z_1(t)$.

With this understanding, we can now express the unique solution to the original problem stated in Equation 6.2. To do this, we simply invert the change of coordinates described in Equation 7.5. The solution is given by:

$$x(t) = Qz(t) = Q\begin{bmatrix} -\sum_{k=0}^{\nu-1} N^k \frac{\mathrm{d}^k g_1(t)}{\mathrm{d}t^k} \\ \mathrm{e}^{Jt} z_2(0) + \int_0^t \mathrm{e}^{J(t-\tau)} g_2(\tau)\,\mathrm{d}\tau \end{bmatrix} \tag{7.8}$$

Remark 7.3 It is crucial to observe that the initial condition $\boldsymbol{z}_1(0)$ cannot be chosen arbitrarily; it must satisfy the following relationship:

$$\boldsymbol{z}_1(0) = -\sum_{k=0}^{\nu-1} \boldsymbol{N}^k \frac{\mathrm{d}^k \boldsymbol{g}_1(0)}{\mathrm{d}t^k}.$$

As a result, the initial condition $\boldsymbol{x}_0 = \boldsymbol{x}(t=0)$ for Equation Equation 6.2 is also not freely chosen. This reflects the requirement that the initial condition must be consistent with the constraints imposed by the matrix $\boldsymbol{E}$.

To summarize our discussion, we can outline the conditions for the existence of a solution to the problem stated in Equation 6.2:

Theorem 7.2 The problem in Equation 6.2 with initial condition $\boldsymbol{x}_0 = \boldsymbol{x}(t=0)$ has a solution if the following conditions are satisfied:

- The function $\boldsymbol{f}(t)$ is piecewise continuous.
- The pair $(\boldsymbol{E}, \boldsymbol{A})$ forms a regular pencil.
- The initial condition $\boldsymbol{x}_0$ meets the requirement:

$$\boldsymbol{x}_0 = \boldsymbol{Q} \begin{bmatrix} -\sum_{k=0}^{\nu-1} \boldsymbol{N}^k \frac{\mathrm{d}^k \boldsymbol{g}_1(0)}{\mathrm{d}t^k} \\ \boldsymbol{z}_2(0) \end{bmatrix}$$

Furthermore, the corresponding solution is given by Equation 7.8.



8. Gaussian Elimination and Echelon form

The *Weierstrass canonical form* is quite valuable from a theoretical standpoint. However, it can be computationally intensive since it necessitates the use of several Jordan canonical forms. In the following section, we will present a more numerically efficient algorithm for reducing a Differential Algebraic Equation (DAE) to an Ordinary Differential Equation (ODE). This method leverages the well-known Gaussian elimination technique, enhanced with differentiation to achieve the desired reduction.

8.1 Gaussian Elimination

Before diving into DAE reduction to ODE, we will revisit some fundamental concepts and definitions that are pertinent to this process.

Definition 8.1 Given a matrix

$$\boldsymbol{M} \in \mathbb{R}^{n\times m}$$

we define elementary operations as follows:

- **Permutation of two rows** (or columns),
- **Linear combinations of rows** (or columns).

These operations can also be viewed as specific forms of matrix multiplication.

Definition 8.2 A non-singular matrix

$$\boldsymbol{P}_{[i,j]} \in \mathbb{R}^{n\times n}$$

is referred to as a **permutation matrix** (**row** or **column** depend on the application) if it is derived from the identity matrix $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ by swapping column i with column j.

Permutation matrices exhibit several noteworthy properties:

- $\boldsymbol{P}_{[i,j]}^{-1} = \boldsymbol{P}_{[i,j]}^{\mathsf{T}}$: This shows that $\boldsymbol{P}_{[i,j]}$ is an orthogonal matrix.
- $\boldsymbol{P}_{[i,j]}^{2} = \boldsymbol{I}$: This indicates that $\boldsymbol{P}_{[i,j]}$ is an involution matrix.
- $\boldsymbol{P}_{[i,j]}\boldsymbol{M}$: This operation exchanges row i with row j.
- $\boldsymbol{M}\boldsymbol{P}_{[i,j]}$: This operation exchanges column i with column j.

Example

Consider the permutation matrix

$$\boldsymbol{P}_{[1,2]} \in \mathbb{R}^{3\times 3} \quad \text{and the matrix} \quad \boldsymbol{M} \in \mathbb{R}^{3\times 3}$$

The product $\boldsymbol{P}_{[1,2]}M$ exchanges rows 1 and 2 of matrix $\boldsymbol{M}$:

$$\boldsymbol{P}_{[1,2]}\boldsymbol{M} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

Similarly, the product $\boldsymbol{M}\boldsymbol{P}_{[1,2]}$ exchanges columns 1 and 2 of matrix $\boldsymbol{M}$:

$$\boldsymbol{M}\boldsymbol{P}_{[1,2]} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \\ 8 & 7 & 9 \end{bmatrix}$$

In a similar manner, we can define matrices that perform linear combinations of rows (or columns). These matrices are classified as **elementary matrices**.

Definition 8.3 A matrix

$$\boldsymbol{F}_{[i,j]}(\alpha) \in \mathbb{R}^{n\times n}$$

is referred to as a **row (or column) elementary matrix** if it has the following structure:

$$\boldsymbol{F}_{[i,j]}(\alpha) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & 1 & & 0 & & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & \alpha & \dots & 1 & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix}$$

where the (i, j) entry equals α.

Remark 8.1 An elementary matrix $\boldsymbol{F}_{[i,j]}(\alpha) \in \mathbb{R}^{n\times n}$ is identical to the identity matrix of dimension n in all entries except for the (i, j) position, which is set to α.

Elementary matrices have several key properties:

- The operation $\boldsymbol{F}_{[i,j]}(\alpha)\boldsymbol{M}$ adds row j multiplied by α to row i.
- The operation $\boldsymbol{M}\boldsymbol{F}_{[i,j]}(\alpha)$ adds column i multiplied by α to column j.

Theorem 8.1 Given a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times m}$ with $n \leq m$ and rank $\boldsymbol{M} = r \leq n$, there exist a pair of nonsingular matrices

- $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and
- $\boldsymbol{R} \in \mathbb{R}^{m\times m}$

such that

$$\boldsymbol{L}\boldsymbol{M}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\Lambda} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r}$ is a nonsingular diagonal matrix. The blocks of zeros have dimensions compatible with the partitioning and may be empty.

Proof. The proof follows an algorithmic approach based on sequential applications of Gaussian elimination. We define

$$\boldsymbol{M}^{(k)}$$

as the matrix $\boldsymbol{M}$ at the k-th iteration.

Note that parentheses indicate that $\boldsymbol{M}^{(k)}$ is not the matrix $\boldsymbol{M}$ raised to the power k. We initialize the algorithm with $\boldsymbol{M}^{(1)} = \boldsymbol{M}$ and set $k = 1$.

The first step is to locate the first non-zero element within the minor

$$\boldsymbol{M}_{k\ldots n,k\ldots m} = \{M_{ij}\}_{i,j=1}^{i=n,j=m}$$

At the first iteration, we explore all entries. We move first along the rows and then along the columns. Unless the matrix is entirely composed of zeros, we will identify a pair of indices such that $M_{pq}^{(k)} \neq 0$. The goal is to move this non-zero element to the (k,k) position using row and column permutations as follows:

$$\overline{\boldsymbol{M}}^{(k)} = \boldsymbol{P}_{[k,p]} \boldsymbol{M}^{(k)} \boldsymbol{P}_{[q,k]}.$$

Notice that after permutation

$$\overline{M}_{k,k}^{(k)} = M_{p,q}^{(k)}$$

thus, the non-zero element in position (p,q) of matrix $\boldsymbol{M}^{(k)}$ is relocated to the (k,k) position of matrix $\overline{\boldsymbol{M}}^{(k)}$.

Next, we use this non-zero element to zero out all other entries in the k-th column and the k-th row. This can be achieved using a composition of linear combination matrices. Specifically, the matrix

$$\overline{\boldsymbol{L}}^{(k)} \in \mathbb{R}^{n \times n}$$

is designed to set all elements in the range $(k+1\ldots n,k)$ to zero:

$$\begin{aligned}
\overline{\boldsymbol{L}}^{(k)} &= \boldsymbol{F}_{[k+1,k]}\left(-\frac{\overline{M}_{k+1,k}}{\overline{M}_{k,k}}\right)\cdots\boldsymbol{F}_{[n-1,k]}\left(-\frac{\overline{M}_{n-1,k}}{\overline{M}_{k,k}}\right)\boldsymbol{F}_{[n,k]}\left(-\frac{\overline{M}_{n,k}}{\overline{M}_{k,k}}\right) \\
&= \prod_{j=k+1}^{n} \boldsymbol{F}_{[j,k]}\left(-\frac{\overline{M}_{j,k}}{\overline{M}_{k,k}}\right).
\end{aligned}$$

Similarly, we can perform the same operation along the columns. In this case, the matrix $\overline{\boldsymbol{R}}^{(k)}$ will set all elements in the range $(k,k+1\ldots m)$ to zero:

$$\begin{aligned}
\overline{\boldsymbol{R}}^{(k)} &= \boldsymbol{F}_{[k,k+1]}\left(-\frac{\overline{M}_{k,k+1}}{\overline{M}_{k,k}}\right)\cdots\boldsymbol{F}_{[k,n-1]}\left(-\frac{\overline{M}_{k,n-1}}{\overline{M}_{k,k}}\right)\boldsymbol{F}_{[k,n]}\left(-\frac{\overline{M}_{k,n}}{\overline{M}_{k,k}}\right) \\
&= \prod_{j=k+1}^{m} \boldsymbol{F}_{[k,j]}\left(-\frac{\overline{M}_{k,h}}{\overline{M}_{k,k}}\right).
\end{aligned}$$

For notational convenience, we define the following quantities:

$$\begin{aligned}
\boldsymbol{L}^{(k)} &= \overline{\boldsymbol{L}}^{(k)} \boldsymbol{P}_{[k,p]}, \\
\boldsymbol{R}^{(k)} &= \boldsymbol{P}_{[q,k]} \overline{\boldsymbol{R}}^{(k)}.
\end{aligned}$$

Using both matrices $\boldsymbol{L}^{(k)} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}^{(k)} \in \mathbb{R}^{m\times m}$ simultaneously, we can zero out all elements along the k-th row and the k-th column, except for the (k,k) position. The matrix $\boldsymbol{M}$ at the next iteration is given by:

$$\boldsymbol{M}^{(k+1)} = \boldsymbol{L}^{(k)}\boldsymbol{M}^{(k)}\boldsymbol{R}^{(k)}.$$

It is now evident that by iterating this process, after exactly r iterations, the matrix $\boldsymbol{M}^{(r+1)} \in \mathbb{R}^{n\times m}$ will have the following structure:

$$\boldsymbol{M}^{(r+1)} = \begin{bmatrix} \boldsymbol{\Lambda} & 0 \\ 0 & 0 \end{bmatrix}.$$

Moreover, the matrices $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ that achieve this structure are given by:

$$\boldsymbol{L} = \boldsymbol{L}^{(r)}\boldsymbol{L}^{(r-1)}\cdots\boldsymbol{L}^{(1)},$$
$$\boldsymbol{R} = \boldsymbol{R}^{(1)}\boldsymbol{R}^{(2)}\cdots\boldsymbol{R}^{(r)}.$$

It is important to note that both $\boldsymbol{L}$ and $\boldsymbol{R}$ are nonsingular since they are obtained through the product of nonsingular matrices. ■

To illustrate the algorithm clearly, we provide the following example.

Example

Consider the matrix $\boldsymbol{M} \in \mathbb{R}^{3\times 4}$ with rank rank $\boldsymbol{M} = r = 2$, defined as follows:

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 4 & 2 & 0 \end{bmatrix}.$$

Step 1: Initialization We begin by setting

$$\boldsymbol{M}^{(1)} = \boldsymbol{M} \quad \text{and} \quad k = 1.$$

We search for the first non-zero element, starting from the position $(k,k) = (1,1)$ and moving down the first column, then across the rows.
At the first iteration, we find the first non-zero element at position $(p,q) = (2,2)$. To move this element to the $(1,1)$ position, we use the following permutation matrices:

- $\boldsymbol{P}_{[1,2]} \in \mathbb{R}^{3\times 3}$
- $\boldsymbol{P}_{[2,1]} \in \mathbb{R}^{4\times 4}$.

Applying these matrices yields:

$$\begin{aligned} \overline{\boldsymbol{M}}^{(1)} &= \boldsymbol{P}_{[1,2]}\boldsymbol{M}^{(1)}\boldsymbol{P}_{[2,1]} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{M}^{(1)} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 4 & 0 & 2 & 0 \end{bmatrix}. \end{aligned}$$

Step 2: Zeroing Out Next, we need to set all non-zero elements in the first row and column to zero. The only non-zero element in the first column is $\overline{M}^{(1)}_{3,1} = 4$. We use the row combination matrix:

$$\boldsymbol{C}_{[3,1]}(-2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

to achieve the goal:

$$\begin{aligned} \boldsymbol{M}^{(2)} &= \boldsymbol{C}_{[3,1]}(-2)\overline{\boldsymbol{M}}^{(1)} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 4 & 0 & 2 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 \end{bmatrix}. \end{aligned}$$

Step 3: Next Iteration Now, we set $k = 2$ and search for the first non-zero element in $\boldsymbol{M}^{(2)}$. The first non-zero element is at position $(p,q) = (2,3)$. Thus, we can use the permutation matrices:

- $\boldsymbol{P}_{[2,2]} = \boldsymbol{I} \in \mathbb{R}^{3\times 3}$
- $\boldsymbol{P}_{[3,1]} \in \mathbb{R}^{4\times 4}$.

We apply these matrices:

$$\begin{aligned} \overline{\boldsymbol{M}}^{(2)} &= \boldsymbol{M}^{(2)}\boldsymbol{P}_{[3,1]} \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Step 4: Final Zeroing Finally, we apply the row combination matrix:

$$\boldsymbol{C}_{[3,2]}(-2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

to complete the process:

$$\begin{aligned} \boldsymbol{M}^{(3)} &= \boldsymbol{C}_{[3,2]}(-2)\overline{\boldsymbol{M}}^{(2)} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

The diagonal block is given by:

$$\boldsymbol{\Lambda} \in \mathbb{R}^{2\times 2} = \operatorname{diag}(2,1).$$

This clearly shows that the rank of matrix $\boldsymbol{M}$ is equal to $r = 2$.

8.2 Echelon Form

A matrix is in **row echelon form** if:

- All nonzero rows are above rows of all zeros.
- Each leading entry (first nonzero number from the left) of a row is strictly to the right of the leading entry of the row above it.
- All entries in a column below a leading entry are zeros.

Example of a matrix in row echelon form:

$$\begin{bmatrix} 1 & 2 & 0 & 3 \\ 0 & 4 & 1 & 5 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Another example of a matrix in row echelon form:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

8.2.1 Elementary Operations and Matrix Representation

To transform a matrix into echelon form, we use three elementary row operations, each of which corresponds to multiplying on the left by an **elementary matrix**:

1. **Swapping two rows** $\Rightarrow$ Multiply by a **permutation matrix**.
2. **Multiplying a row by a nonzero scalar** $\Rightarrow$ Multiply by a **diagonal matrix**.
3. **Adding or subtracting a multiple of one row to another row** $\Rightarrow$ Multiply by a **lower triangular matrix**.

Each elementary matrix is obtained by applying the corresponding row operation to the identity matrix.

Step-by-Step Process

Step 1: Identify the Leftmost Nonzero Column (Pivot Column)

- Scan the columns from left to right to find the first column that contains a nonzero entry.
- The first nonzero number in that column will be our **pivot**.

Step 2: Ensure the Pivot is in the First Row of the Submatrix

- If necessary, swap rows to move a row with a nonzero pivot to the top.
- This corresponds to multiplying by a **permutation matrix** P.

Step 3: Zero Out All Entries Below the Pivot

- For each row below the pivot row, subtract an appropriate multiple of the pivot row to make the entries below the pivot **zero**.
- This operation is equivalent to multiplying by a **lower triangular elementary matrix** L.

Step 4: Normalize the Pivot (Optional but Useful)

- The pivot can be scaled to 1 by dividing the row by the pivot value.
- This step is equivalent to multiplying by a **diagonal matrix** $\boldsymbol{D}$.

Step 5: Move to the Next Column and Repeat

- Find the next pivot column (to the right of the current pivot).
- Repeat Steps 2-4 for the remaining sub-matrix.
- Continue this process until all pivot positions are correctly set.

At each step, the transformation can be written as:

$$\boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{P}_k\cdots\boldsymbol{L}_2\boldsymbol{D}_2\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{D}_1\boldsymbol{P}_1\boldsymbol{A}=\boldsymbol{U}$$

where $\boldsymbol{U}$ is the echelon form of $\boldsymbol{A}$, and the matrices $\boldsymbol{P}$, $\boldsymbol{D}$, $\boldsymbol{L}$ represent elementary row operations.

8.2.2 Example of reduction to echelon form

Given the matrix:

$$\boldsymbol{A}=\begin{bmatrix}2&4&6\\1&3&5\\3&7&8\end{bmatrix}$$

Step 1: Make the First Pivot 1

Swap row 1 and row 2 using permutation matrix $\boldsymbol{P}$:

$$\boldsymbol{PA}=\begin{bmatrix}1&3&5\\2&4&6\\3&7&8\end{bmatrix}$$

Step 2: Zero Out Below the Pivot

Subtract 2 times row 1 from row 2 and 3 times row 1 from row 3 using lower triangular matrix $\boldsymbol{L}_1$:

$$\boldsymbol{L}_1\boldsymbol{PA}=\begin{bmatrix}1&3&5\\0&-2&-4\\0&-2&-7\end{bmatrix}$$

Step 3: Normalize the Second Pivot

Multiply row 2 by $-\frac{1}{2}$ using diagonal matrix $\boldsymbol{D}_1$:

$$\boldsymbol{D}_1\boldsymbol{L}_1\boldsymbol{PA}=\begin{bmatrix}1&3&5\\0&1&2\\0&-2&-7\end{bmatrix}$$

Step 4: Zero Out Below the New Pivot

Add 2 times row 2 to row 3 using lower triangular matrix $\boldsymbol{L}_2$:

$$\boldsymbol{L}_2\boldsymbol{D}_1\boldsymbol{L}_1\boldsymbol{PA}=\begin{bmatrix}1&3&5\\0&1&2\\0&0&-3\end{bmatrix}$$

This matrix is now in **row echelon form**.

8.2.3 Example of reduction to echelen form for a singular matrix

Consider the singular matrix:

$$\boldsymbol{A}=\begin{bmatrix}2&4&6\\1&2&3\\3&6&9\end{bmatrix}$$

Step 1: Normalize the First Pivot

Swap row 1 and row 2:

$$\boldsymbol{PA} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

Step 2: Zero Out Below the Pivot

Subtract 2 times row 1 from row 2 and 3 times row 1 from row 3:

$$\boldsymbol{L}_1\boldsymbol{PA} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Since the last two rows are all zeros, the **rank** of the matrix is **1**, meaning:

- The matrix has **linearly dependent** rows.
- The determinant is **zero**, confirming singularity.
- The system $\boldsymbol{Ax} = \boldsymbol{b}$ may have either infinitely many solutions or no solution, depending on b.

Thus, the matrix does not have an inverse.

8.2.4 Summary of Echelon Form Reduction Algorithm

To transform a matrix into row echelon form:

1. Identify the leftmost **pivot** column.
2. Swap rows if necessary to bring a nonzero pivot to the top $\boldsymbol{PA}$.
3. Zero out all entries below the pivot $\boldsymbol{L}_1, \boldsymbol{L}_2, \ldots$.
4. Normalize pivots if needed $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots$.
5. Move to the next pivot column and repeat.
6. Continue until all pivots are set correctly.

Each row operation corresponds to multiplication by an **elementary matrix**, leading to the factorization:

$$\boldsymbol{E}_k \cdots \boldsymbol{E}_2 \boldsymbol{E}_1 \boldsymbol{A} = \boldsymbol{U}$$

where $\boldsymbol{U}$ is in echelon form, and $\boldsymbol{E}_i$ are elementary matrices.

This method is a **systematic** approach to solving linear systems, computing ranks, and checking singularity conditions.

Algorithm. Row Echelon Form Reduction

Input: A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
Output: Matrix in row echelon form
$k \leftarrow 1$, $j \leftarrow 1$ `// Initialize`

while $k \leq m$ ***and*** $j \leq n$ **do**
  $p \leftarrow k$
  while $p \leq m$ ***and*** $A_{pj} = 0$ **do**
    $p \leftarrow p + 1$
  end
  if $p > m$ **then**
    $j \leftarrow j + 1$ `// No pivot in this column, move right`
  else
    if $p \neq k$ **then**
      Swap rows k and p `// Bring pivot to current row`
    end
    for $i \leftarrow k + 1$ **to** m **do**
      $t \leftarrow A_{ij}/A_{kj}$
      $A_{i,\bullet} \leftarrow A_{i,\bullet} - t \cdot A_{k,\bullet}$ `// Eliminate below pivot`
    end
    $k \leftarrow k + 1$, $j \leftarrow j + 1$ `// Move to next pivot position`
  end
end
return $\boldsymbol{A}$

Figure 8.1: **Algorithm:** Row Echelon Form Reduction (with row swaps only). Transform a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ into row echelon form.

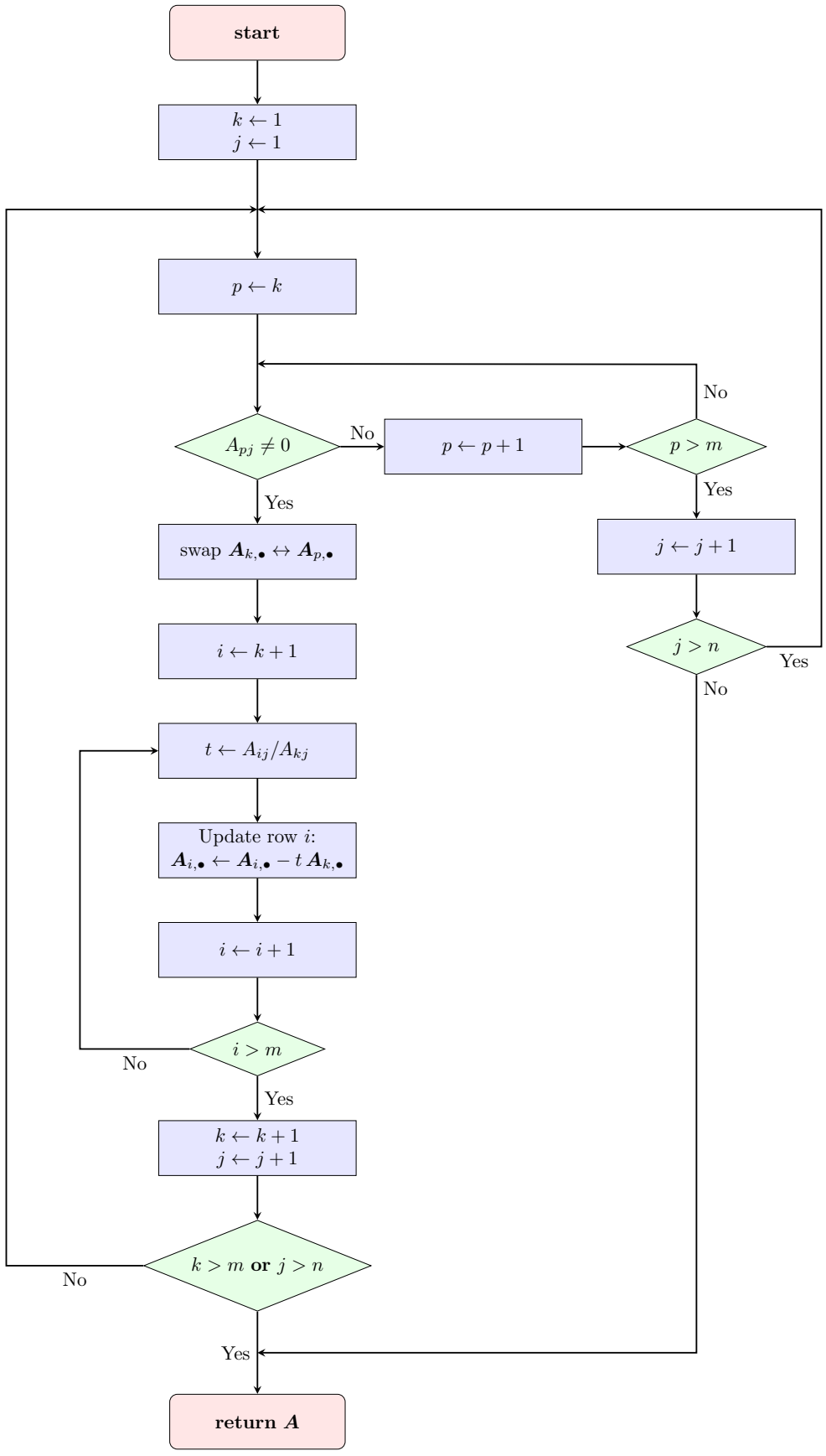
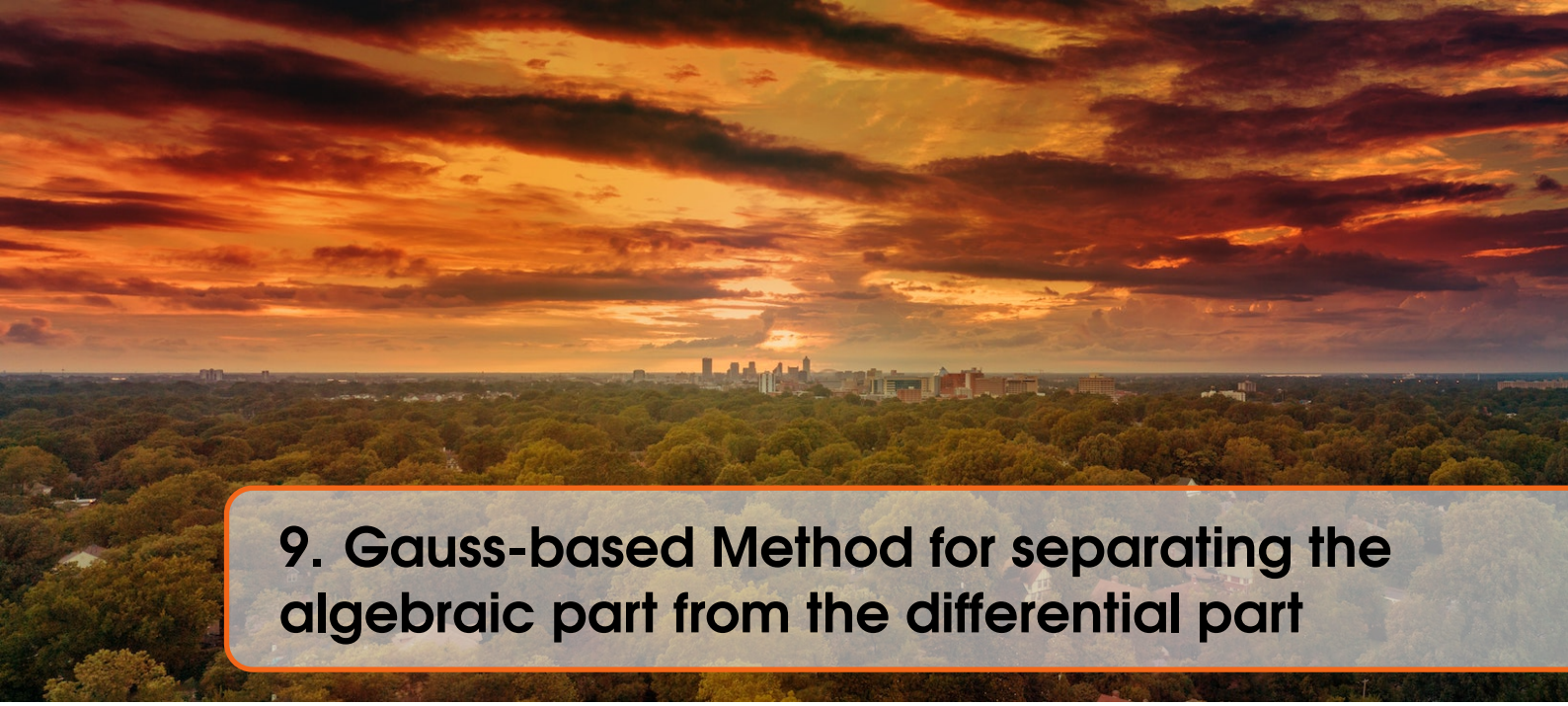


Figure 8.2: Algorithm: Reduction to Row Echelon Form (flow chart)



9. Gauss-based Method for separating the algebraic part from the differential part

We are now prepared to explore the utility of the Gauss-like method Theorem 8.1 in determining the index of a Differential-Algebraic Equation (DAE) and transforming a generic linear DAE into an Ordinary Differential Equation (ODE).

9.1 Formulation of the DAE

Consider a generic linear DAE expressed in the following form:

$$\boldsymbol{E}\boldsymbol{x}' + \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}(t). \tag{9.1}$$

The matrix $\boldsymbol{E}$ and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Define the rank of $\boldsymbol{E}$ as:

$$\text{rank } \boldsymbol{E} = r \leq n.$$

9.2 Change of Coordinates

Now, consider a pair of matrices $\boldsymbol{L}^{(0)} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}^{(0)} \in \mathbb{R}^{n\times n}$ such that:

$$\boldsymbol{L}^{(0)}\boldsymbol{E}\boldsymbol{R}^{(0)} = \boldsymbol{E}^{(0)} = \begin{bmatrix} \boldsymbol{\Lambda}^{(0)} & 0 \\ 0 & 0 \end{bmatrix}.$$

As in the previous section, the superscript $^{(0)}$ indicates the iteration number, which corresponds to the Kronecker index of the DAE. At this first iteration, $\boldsymbol{\Lambda}^{(0)} \in \mathbb{R}^{r\times r}$.

If $r = n$, we are done since the DAE has index $\nu = 0$, and the matrix $\boldsymbol{E}$ is full rank and invertible. If $r < n$, we introduce the following change of coordinates:

$$\boldsymbol{x}^{(0)} = \boldsymbol{R}^{(0)}\boldsymbol{x}, \qquad (\boldsymbol{x}^{(0)})' = \boldsymbol{R}^{(0)}\boldsymbol{x}',$$

and define the matrix $\boldsymbol{A}^{(0)}$ and vector $\boldsymbol{f}^{(0)}$ as follows:

$$\begin{aligned} \boldsymbol{A}^{(0)} &:= \boldsymbol{L}^{(0)}\boldsymbol{A}\boldsymbol{R}^{(0)}, \\ \boldsymbol{f}^{(0)} &:= \boldsymbol{L}^{(0)}\boldsymbol{f}. \end{aligned}$$

so that

$$\boldsymbol{E}^{(0)}(\boldsymbol{x}^{(0)})' + \boldsymbol{A}^{(0)}\boldsymbol{x}^{(0)} = \boldsymbol{f}^{(0)}(t). \tag{9.2}$$

obtained by multiply the left-hand side of Equation 9.1 by $\boldsymbol{L}^{(0)}$.

9.2.1 Special Structure of the Matrix

Notice that by partitioning

$$(\boldsymbol{x}')_1^{(0)}, \boldsymbol{x}_1^{(0)} \in \mathbb{R}^r, \quad (\boldsymbol{x}')_2^{(0)}, \boldsymbol{x}_2^{(0)} \in \mathbb{R}^{n-r}, \quad \text{and} \quad \boldsymbol{x}^{(0)} = \begin{bmatrix} \boldsymbol{x}_1^{(0)} \\ \boldsymbol{x}_2^{(0)} \end{bmatrix}.$$

then Equation 9.2 has the following special structure:

$$\begin{bmatrix} \boldsymbol{\Lambda}^{(0)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} (\boldsymbol{x}')_1^{(0)}(t) \\ (\boldsymbol{x}')_2^{(0)}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{11}^{(0)} & \boldsymbol{A}_{12}^{(0)} \\ \boldsymbol{A}_{21}^{(0)} & \boldsymbol{A}_{22}^{(0)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^{(0)} \\ \boldsymbol{x}_2^{(0)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1^{(0)}(t) \\ \boldsymbol{f}_2^{(0)}(t) \end{bmatrix}. \tag{9.3}$$

The equation Equation 9.3 can also be expressed as a pair of matrix equations:

$$\begin{aligned} \boldsymbol{\Lambda}^{(0)}(\boldsymbol{x}')_1^{(0)} + \boldsymbol{A}_{11}^{(0)}\boldsymbol{x}_1^{(0)} + \boldsymbol{A}_{12}^{(0)}\boldsymbol{x}_2^{(0)} &= \boldsymbol{f}_1^{(0)}(t), \qquad (a) \\ \boldsymbol{A}_{21}^{(0)}\boldsymbol{x}_1^{(0)} + \boldsymbol{A}_{22}^{(0)}\boldsymbol{x}_2^{(0)} &= \boldsymbol{f}_2^{(0)}(t), \qquad (b) \end{aligned} \tag{9.4}$$

9.2.2 Decoupling of Components

Notice that

- Equation 9.4(a) is an ODE since $\boldsymbol{\Lambda}^{(0)}$ is invertible
- Equation 9.4(b) is a pure algebraic equation.

Hence, in these coordinates, the differential and algebraic components of the DAE are decoupled.

9.2.3 Differentiation of Algebraic Constraints

Now we differentiate Equation 9.4(b) with respect to time:

$$\boldsymbol{A}_{21}^{(0)}(\boldsymbol{x}')_1^{(0)} + \boldsymbol{A}_{22}^{(0)}(\boldsymbol{x}')_2^{(0)} = (\boldsymbol{f}_2'(t))^{(0)}. \tag{9.5}$$

Writing Equation 9.5 with Equation 9.4(a) in matrix form yields:

$$\begin{bmatrix} \boldsymbol{\Lambda}^{(0)} & \boldsymbol{0} \\ \boldsymbol{A}_{21}^{(0)} & \boldsymbol{A}_{22}^{(0)} \end{bmatrix} \begin{bmatrix} (\boldsymbol{x}')_1^{(0)} \\ (\boldsymbol{x}')_2^{(0)} \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{11}^{(0)} & \boldsymbol{A}_{12}^{(0)} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^{(0)} \\ \boldsymbol{x}_2^{(0)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_1^{(0)} \\ (\boldsymbol{f}_2'(t))^{(0)} \end{bmatrix}. \tag{9.6}$$

9.3 Rank Increase Observation

As observed, the differentiation with respect to time of the algebraic constraints in the matrix representation produces merely a shift to the left of the two blocks $\boldsymbol{A}_{21}^{(0)}$ and $\boldsymbol{A}_{22}^{(0)}$ while preserving the original structure.

Thus, in the transition from the DAE Equation 9.4 to the differentiated version Equation 9.6, we observe an increase in rank at each iteration while maintaining the structure of the equations. The final rank gives the Kronecker index of the original DAE.

Definition 9.1 The minimum integer such that $\dim \boldsymbol{\Lambda}^{(\nu)} = n$ is the Kronecker index (or differential index) of the DAE.

In this way, we are able to compute the index of a DAE systematically through a sequence of transformations, eventually leading to a formulation where the components are decoupled, thus facilitating the analysis and solution of the DAE.

Example

Consider the following set of linear differential equation,

$$\begin{aligned} x_1'(t)+x_2'(t)+x_3'(t) &= 2x_1(t)+x_3(t)+t,\\ x_1'(t)+x_2'(t)+x_3'(t) &= x_1(t)+x_2(t)+t^3,\\ 0 &= x_1(t)+x_2(t)+x_3(t)+\sin(t), \end{aligned}$$

that we can write in a more compact matrix form as follows,

$$\begin{bmatrix} 1 & 1 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1'\\ x_2'\\ x_3' \end{bmatrix} - \begin{bmatrix} 2 & 0 & 1\\ 1 & 1 & 0\\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1\\ x_2\\ x_3 \end{bmatrix} = \begin{bmatrix} t\\ t^3\\ \sin(t) \end{bmatrix}.$$

Which corresponds to the usual form $\boldsymbol{E}\boldsymbol{x}'(t)+\boldsymbol{A}\boldsymbol{x}(t)=\boldsymbol{f}(t)$ obvious meaning for the symbols. Now consider to form the table

$$\begin{bmatrix} \boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t) \end{bmatrix} \in \mathbb{R}^{n\times 2n+1}$$

in such a way that,

$$\begin{bmatrix} \boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}'\\ \boldsymbol{x}\\ -1 \end{bmatrix} = \boldsymbol{0}$$

First it is necessary to check if the pair $(\boldsymbol{E},\boldsymbol{A})$ is a regular pencil,

$$\left|\boldsymbol{A}-\lambda\boldsymbol{E}\right| = \begin{vmatrix} -2-\lambda & -\lambda & -1-\lambda\\ -1-\lambda & -1-\lambda & -\lambda\\ -1 & -1 & -1 \end{vmatrix} = -2$$

the determinant is different from zero, so the pair $(\boldsymbol{E},\boldsymbol{A})$ is a *regular pencil*.
Now it is easy to manipulate the following table

$$\begin{bmatrix} 1 & 1 & 1 & -2 & 0 & -1 & t\\ 1 & 1 & 1 & -1 & -1 & 0 & t^3\\ 0 & 0 & 0 & -1 & -1 & -1 & \sin(t) \end{bmatrix}$$

subtract first row to the second row

$$\begin{bmatrix} 1 & 1 & 1 & -2 & 0 & -1 & t\\ 0 & 0 & 0 & 1 & -1 & 1 & t^3-t\\ 0 & 0 & 0 & -1 & -1 & -1 & \sin(t) \end{bmatrix}$$

constraints:

$$\begin{aligned} x_1-x_2+x_3-t^3+t &= 0\\ x_1+x_2+x_3+\sin(t) &= 0 \end{aligned}$$

---

differentiate second and third row (reduce index by one)

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 1 & t \\ -1 & 1 & -1 & 0 & 0 & 0 & 3t^2-1 \\ -1 & -1 & -1 & 0 & 0 & 0 & \cos(t) \end{bmatrix}$$

add first row to the second and the third row

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 1 & t \\ 0 & 2 & 0 & 2 & 0 & 1 & 3t^2+t-1 \\ 0 & 0 & 0 & 2 & 0 & 1 & \cos(t)+t \end{bmatrix}$$

constraint:

$$2x_1 + x_3 - \cos(t) - t = 0$$

---

derive third row (reduce index by one)

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 1 & t \\ 0 & 2 & 0 & 2 & 0 & 1 & 3t^2+t-1 \\ 2 & 0 & 1 & 0 & 0 & 0 & -\sin(t)+1 \end{bmatrix}$$

subtract 2 times first row to third

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 1 & t \\ 0 & 2 & 0 & 2 & 0 & 1 & 3t^2+t-1 \\ 0 & -2 & -1 & -4 & 0 & -2 & -\sin(t)+1-2t \end{bmatrix}$$

add second row to third

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 1 & t \\ 0 & 2 & 0 & 2 & 0 & 1 & 3t^2+t-1 \\ 0 & 0 & -1 & -2 & 0 & -1 & 3t^2-t-\sin(t) \end{bmatrix}$$

add third row to first and then multiply third row by (-1) and second row by $(1/2)$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 3t^2-\sin(t) \\ 0 & 1 & 0 & 1 & 0 & 1 & \frac{1}{2}(3t^2+t-1) \\ 0 & 0 & 1 & 2 & 0 & 1 & t+\sin(t)-3t^2 \end{bmatrix}$$

subtract second row to first

$$\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 & \frac{3}{2}t^2+\frac{1}{2}(1-t)-\sin(t) \\ 0 & 1 & 0 & 1 & 0 & 1 & \frac{1}{2}(3t^2+t-1) \\ 0 & 0 & 1 & 2 & 0 & 1 & 3t^2-t-\sin(t) \end{bmatrix}$$

then we obtain the ODE

$$\begin{cases} x_1' = x_1 + x_3 + \dfrac{3}{2}t^2 + \dfrac{1}{2}(1-t) - \sin(t) \\ x_2' = -x_1 - x_3 + \dfrac{1}{2}(3t^2+t-1) \\ x_3' = -2x_1 - x_3 + 3t^2 - t - \sin(t) \end{cases}$$

with the hidden invariants (or constraints)

$$x_1 - x_2 + x_3 - t^3 + t = 0$$
$$x_1 + x_2 + x_3 + \sin(t) = 0$$
$$2x_1 + x_3 - \cos(t) - t = 0$$

IV Nonlinear DAE

10 DAE from Lagrangian mechanics 93
10.1 Overview of Lagrangian Mechanics 93
10.2 The Minimal Action Principle in Classical Mechanics 95
10.3 The Calculus of Variations and the Stationary Action Principle 97
10.4 Consequences of the Stationary Action Principle 98
10.5 Principle of Minimal Action and Coordinate Independence 101
10.6 Mechanical Equations of Motion 103
10.7 Constrained Systems and Lagrange Multipliers 105

11 Numerical Integration of DAE 109
11.1 Reduction to ODE 110
11.2 The drift effect 112

Nonlinear DAEs from the Principle of Stationary Action

Many physical systems are governed by constraints—relationships among variables that must be satisfied at all times. When such systems are described using variational principles, particularly the principle of stationary action, the resulting equations of motion often take the form of nonlinear differential-algebraic equations (DAEs).

Unlike standard ordinary differential equations (ODEs), DAEs involve both differential terms and algebraic constraints. These constraints may arise naturally from the system's geometry, conservation laws, or imposed physical restrictions (such as rigid connections or incompressibility).

In this chapter, we explore how DAEs emerge from the variational formulation of mechanical systems with constraints. We begin by applying the stationary action principle to systems with holonomic and non-holonomic constraints, and show how Lagrange multipliers are introduced to enforce them. The resulting equations form a coupled system of differential and algebraic components—nonlinear in general—whose analysis and numerical solution require specialized techniques.

Our focus will be on understanding the structure of these equations, their geometric interpretation, and the challenges they pose for simulation and control.

Learning objectives

After this part, you should be able to derive constrained Euler–Lagrange equations, interpret the role of Lagrange multipliers, and explain why constraint differentiation can create drift in a numerical simulation.



10. DAE from Lagrangian mechanics

10.1 Overview of Lagrangian Mechanics

This section primarily draws from the foundational works of [1] and [13]. Mechanics explores the behavior of systems ranging from particles and rigid bodies to continuous media and field theories, such as electromagnetism and gravitation. Since Newton's formulation in the 17th century, mechanics has evolved into two dominant frameworks: *Lagrangian mechanics* and *Hamiltonian mechanics*. Here, we focus on the essentials of Lagrangian mechanics, with further details available in [13].

At the core of Lagrangian mechanics lies the principle of *stationary action* (or *least action*), a variational approach that generalizes Newton's second law, $\boldsymbol{F} = m\boldsymbol{a}$. This principle provides a powerful method for deriving equations of motion for a wide range of physical systems. While both Lagrangian and Hamiltonian mechanics share this foundation, we will restrict our discussion to systems with a finite number of *degrees of freedom*.

10.1.1 Generalized Coordinates and Energies

For such systems, the configuration is described by a set of *generalized coordinates*:

$$\boldsymbol{q} := (q_1, q_2, \ldots, q_n) \in \mathbb{R}^n,$$

where n is the minimum number of independent variables needed to specify the system's state. The choice of $\boldsymbol{q}$ is not unique; different sets of generalized coordinates may describe the same system.

Given a coordinate system $\boldsymbol{q}$, we express

- the system's *kinetic energy* $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$
- and *potential energy* $U(\boldsymbol{q}, t)$

as functions of these variables. The kinetic energy depends on both positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$, while the potential energy generally depends on positions and may also explicitly depend on time t:

$$T : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}, \quad U : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}.$$

Though $\boldsymbol{q}$ may parametrize more complex geometries (i.e., *manifolds*), in many practical cases, the configuration space can be locally approximated as Euclidean.

10.1.2 The Lagrangian Function

The *Lagrangian function*, $L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$, is defined as the difference between kinetic and potential energy:

$$L(\boldsymbol{q},\dot{\boldsymbol{q}},t) := T(\boldsymbol{q},\dot{\boldsymbol{q}}) - U(\boldsymbol{q},t). \tag{10.1}$$

This function encapsulates the entire dynamics of the system and serves as the foundation for the Lagrangian formulation. Its domain consists of positions, velocities, and time:

$$L : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}.$$

10.1.3 Euler-Lagrange Equations

The equations of motion follow from the *Euler-Lagrange equations*, derived by applying the principle of **stationary action**:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0, \quad i = 1,\ldots,n.$$

These equations provide a systematic method for obtaining the equations of motion for mechanical systems, from simple particles to more complex rigid-body dynamics.

The Lagrangian framework offers an elegant and unified approach to modeling physical phenomena, leveraging the principle of stationary action. By encoding dynamics in a single function $L(\boldsymbol{q},\dot{\boldsymbol{q}},t)$, it simplifies problem-solving and provides deep insights into the structure of classical mechanics.

▪ **Example 10.1 — Lagrangian Mechanics in Different Coordinate Systems.** Consider a free mass m moving in a plane under the influence of gravity, which acts downward along the y-axis (see Figure 10.1).

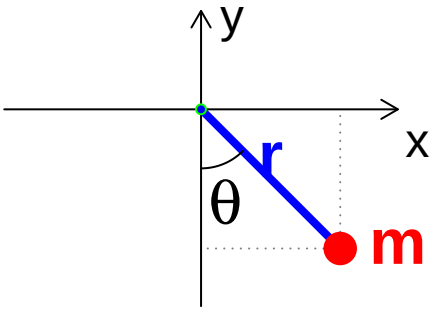


Figure 10.1: A free mass m in a plane with gravity acting downward.

Cartesian Coordinates

In Cartesian coordinates $\boldsymbol{q} = (x,y) \in \mathbb{R}^2$, the kinetic and potential energies of the system are given by:

$$T = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2), \quad U = mgy.$$

Thus, the Lagrangian is:

$$L = \frac{1}{2}m(\dot{\boldsymbol{q}}\cdot\dot{\boldsymbol{q}}) - mg(\boldsymbol{e}\cdot\boldsymbol{q}),$$

where $\boldsymbol{e}_2 = (0,1)$ is the unit vector in the y-direction.

Transformation to Polar Coordinates

Cartesian coordinates are not the only possible choice. Using *polar coordinates* (r,θ) [rotated by 90 degree], we have the transformations:

$$x = r\sin\theta, \quad y = -r\cos\theta.$$

Differentiating with respect to time:

$$\dot{x} = \dot{r}\sin\theta + r(\cos\theta)\dot{\theta}, \quad \dot{y} = -\dot{r}\cos\theta + r(\sin\theta)\dot{\theta}.$$

Substituting these into the expressions for kinetic and potential energy:

$$T = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\theta}^2), \quad U = -mgr\cos\theta.$$

Thus, in the generalized coordinates $\widetilde{\boldsymbol{q}} = (r,\theta)$, the Lagrangian takes the form:

$$L = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\theta}^2) + mgr\sin\theta.$$

This example demonstrates that the choice of generalized coordinates is not unique. Depending on the problem, selecting an appropriate coordinate system can simplify calculations. However, regardless of the chosen coordinates, the Lagrangian formulation remains consistent and yields the correct equations of motion.

■

10.2 The Minimal Action Principle in Classical Mechanics

A fundamental concept in classical mechanics is the *action* $S \in \mathbb{R}$. The action is defined as the integral of the Lagrangian function over time, evaluated between two instants t_1 and $t_2 \in \mathbb{R}$:

$$S := \int_{t_1}^{t_2} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\,\mathrm{d}t. \tag{10.2}$$

The action is a *functional*, meaning it assigns a scalar value to an entire trajectory rather than a single point. Specifically, it maps a curve $\boldsymbol{q}(\cdot)$ in the space of possible trajectories to a real number representing the system's total action over the time interval $[t_1, t_2]$.

10.2.1 Principle of Stationary Action

The *principle of stationary action* (or *least action principle*) states that the actual path taken by a system between two points in time is the one that makes the action stationary. This means that the physical trajectory $\boldsymbol{q}(t)$ satisfies:

$$\delta S = 0,$$

where δS represents the first-order variation of S due to small perturbations in the path. In most cases, this condition leads to minimizing the action, but there exist scenarios where the action is stationary *but not a minimum*.

This principle serves as the foundation for deriving the Euler-Lagrange equations, which govern the system's motion:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0, \quad i = 1,\ldots,n.$$

Thus, rather than directly applying Newton's second law, one can obtain the equations of motion by ensuring that small variations in the trajectory do not significantly change the action.

10.2.2 Interpretation of the Action Functional

Remark 10.1 — Action dependence. It's important to recognize that the action $S = S(\boldsymbol{q})$ depends on the entire trajectory $\boldsymbol{q}(t)$ over the time interval $[t_1, t_2]$, not just on individual points along the path.

To clarify notation:

- We use $\boldsymbol{q}(\cdot)$ to represent the function describing the full trajectory.
- $\boldsymbol{q}(t)$ refers to the specific position at a given time t.

Since S is a functional, different curves $\boldsymbol{q}(\cdot)$ generally yield different values of S. The principle of stationary action asserts that among all possible paths connecting the given endpoints, nature selects the one that makes S stationary.

10.2.3 Physical Implications and Broader Context

The action principle provides a unifying framework for classical mechanics and extends naturally into other areas of physics:

- In quantum mechanics, Feynman's *path integral formulation* interprets the action as a phase factor in a sum over all possible trajectories.
- In general relativity, Einstein's field equations emerge from an action principle applied to the curvature of spacetime.
- In field theory, the action integral is extended to functional over spacetime, leading to fundamental equations such as Maxwell's equations for electromagnetism.

By shifting the focus from forces to variational principles, the action principle offers a powerful and elegant approach to understanding physical laws.

▪ **Example 10.2 — Computing the Action for a Given Trajectory.** Consider a system with generalized coordinates

$$\boldsymbol{q} = (q_1, q_2) \in \mathbb{R}^2$$

evolving over the time interval $t \in [0, 1]$. Suppose the Lagrangian is given by:

$$L = q_1^2 + q_2^2.$$

The action functional, defined as the integral of the Lagrangian over time, takes the form:

$$S = \int_0^1 (q_1^2 + q_2^2)\,\mathrm{d}t.$$

To illustrate how the choice of trajectory affects the computed action, consider the specific path:

$$\boldsymbol{q}(t) = (2, t).$$

Substituting this trajectory into the action integral:

$$S = \int_0^1 (2^2 + t^2)\,\mathrm{d}t = \int_0^1 (4 + t^2)\,\mathrm{d}t.$$

Evaluating the integral:

$$S = \left[4t + \frac{t^3}{3}\right]_0^1 = 4(1) + \frac{1^3}{3} - \left(4(0) + \frac{0^3}{3}\right) = 4 + \frac{1}{3} = \frac{13}{3}.$$

This example demonstrates how different trajectories yield different action values, emphasizing the role of path selection in variational principles.

▪

10.3 The Calculus of Variations and the Stationary Action Principle

In classical analysis, we often look for extrema—points where a function reaches a minimum or maximum. A similar idea appears in mechanics: we may ask whether a particular trajectory of a system is, in some sense, optimal. The quantity to be optimized in this context is called the action functional.

This question leads to a fundamental area of mathematics known as the calculus of variations. Unlike traditional analysis, which focuses on functions of a few variables, the calculus of variations deals with functionals—that is, functions whose inputs are other functions, such as curves or trajectories.

A central concept is the principle of stationary action. It states that the actual path $\boldsymbol{q}(\cdot)$ followed by a mechanical system is one that makes the action functional stationary. This means that small variations in the path do not change the value of the action, at least to first order.

A useful analogy is that of light traveling through different media: according to Fermat's principle, light follows the path that makes the travel time minimal. Similarly, physical systems evolve along paths that extremize (not necessarily minimize) the action.

To formalize this principle, consider a system evolving between:

- An initial time t_1, where the system is at configuration $\boldsymbol{q}_1 = \boldsymbol{q}(t_1)$.
- A final time t_2, where the system reaches configuration $\boldsymbol{q}_2 = \boldsymbol{q}(t_2)$.

Among all possible paths connecting $\boldsymbol{q}_1$ to $\boldsymbol{q}_2$ within the time interval $[t_1, t_2]$, the physical trajectory satisfies:

$$\delta S = 0.$$

This variational condition (that will be discussed later in Section 10.4.2) leads directly to the *Euler-Lagrange equations*, which govern the motion of the system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0, \quad i = 1, \ldots, n.$$

Thus, the principle of stationary action provides a deep and elegant framework for deriving equations of motion across diverse physical systems, from simple particles to complex field theories.

Theorem 10.1 — Stationary Action Principle. The motion of a mechanical system evolving from configuration $\boldsymbol{q}_1$ to configuration $\boldsymbol{q}_2$ over a time interval $t \in [t_1, t_2]$ follows extremal curves of the action functional, as defined in Equation 10.2. Mathematically, this means:

$$\delta S = 0.$$

where δS represents an infinitesimal variation in the action due to a small perturbation in the trajectory $\boldsymbol{q}(\cdot)$. This condition implies that the physical path either minimizes, maximizes, or otherwise renders the action stationary.

A central idea in the calculus of variations is the *variational operator* δ, which quantifies how a **functional**—such as the action—responds to small perturbations in its input. In essence, δ serves a role analogous to that of the differential operator in classical calculus, but it acts on functionals rather than functions.

By systematically analyzing how the action varies under infinitesimal changes in the path, one obtains necessary conditions for a trajectory to make the action stationary. These conditions are precisely the **Euler–Lagrange equations**, which govern the motion of systems in this variational framework.

10.3.1 The Variational Operator: A Simple Example

■ **Example 10.3** The concept of variation δ is closely related to differentiation in standard calculus. To illustrate this, consider a function of two variables:

$$f(x,y) = \cos x + y^2.$$

A small perturbation $(\delta x, \delta y)$ in (x,y) induces a change in f, given by:

$$\delta f = \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y.$$

Explicitly computing the partial derivatives:

$$\delta f = (-\sin x)\delta x + (2y)\delta y.$$

In matrix form, this can be expressed as:

$$\delta f = \begin{pmatrix} -\sin x & 2y \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}.$$

A *stationary point* of f is characterized by $\delta f = 0$ for all small variations. Setting the partial derivatives to zero:

$$\sin x = 0, \quad 2y = 0.$$

Solving for x and y gives the stationary points:

$$x = k\pi, \quad y = 0, \quad k \in \mathbb{Z}.$$

This example mirrors the variational principle in mechanics: just as stationary points of a function correspond to extrema in calculus, stationary trajectories of the action functional correspond to the actual motion of a mechanical system.

■

Notice that

- The *calculus of variations* extends classical optimization to functionals over infinite-dimensional spaces.
- The *stationary action principle* states that physical trajectories extremize the action functional, leading to the Euler-Lagrange equations.
- The *variational operator* δ is a fundamental tool in analyzing changes in functionals, playing a role similar to differentiation in standard calculus.

By applying these principles, one can derive the laws of motion in a wide range of mechanical and physical systems, demonstrating the power and generality of Lagrangian mechanics.

10.4 Consequences of the Stationary Action Principle

The stationary action principle provides a powerful method for deriving the equations of motion of a system. If we assume that the Lagrangian L is sufficiently regular, we can exchange the order of variation and integration, allowing us to move the δ operator under the integral sign:

$$\delta S = \int_{t_1}^{t_2} \delta L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, \mathrm{d}t.$$

Expanding the variation of L in terms of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$:

$$\delta S = \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \boldsymbol{q}}\delta \boldsymbol{q} + \frac{\partial L}{\partial \dot{\boldsymbol{q}}}\delta \dot{\boldsymbol{q}} \right) \mathrm{d}t. \tag{10.3}$$

This equation expresses the change in action due to small variations $\delta \boldsymbol{q}(t)$ and their time derivatives $\delta \dot{\boldsymbol{q}}(t)$.

Remark 10.2 — Fixed-Time Variations. It is important to emphasize that variations are evaluated at *fixed times* t_1 and t_2. Consequently, the variation of L is given by:

$$\delta L(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), t) = \frac{\partial L}{\partial \boldsymbol{q}} \delta \boldsymbol{q} + \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \dot{\boldsymbol{q}},$$

and does *not* include a term involving δt:

$$\delta L(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), t) \neq \frac{\partial L}{\partial \boldsymbol{q}} \delta \boldsymbol{q} + \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \dot{\boldsymbol{q}} + \frac{\partial L}{\partial t} \delta t.$$

This distinction arises because time itself is not varied—only the trajectory $\boldsymbol{q}(\cdot)$ is perturbed. Additionally, we treat $\dot{\boldsymbol{q}}$ as an *independent* variable with respect to $\boldsymbol{q}$ during the variation process.

10.4.1 Integration by Parts and the Euler-Lagrange Equations

A crucial step in deriving the equations of motion from the action principle involves an integration by parts. Using the identity:

$$\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \dot{\boldsymbol{q}} = \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q} \right) - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q}, \tag{10.4}$$

we substitute this result into equation Equation 10.3:

$$\begin{aligned} \delta S &= \int_{t_1}^{t_2} \left(\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q} \right) - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q} + \frac{\partial L}{\partial \boldsymbol{q}} \delta \boldsymbol{q} \right) \mathrm{d}t \\ &= \left[\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q} \right]_{t_1}^{t_2} + \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \boldsymbol{q}} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \right) \delta \boldsymbol{q} \, \mathrm{d}t. \end{aligned} \tag{10.5}$$

The first term represents a *boundary contribution* that depends on the variations at the endpoints t_1 and t_2. The second integral must vanish for the action to be stationary.

Remark 10.3 — Vanishing of Boundary Terms. To ensure the physical trajectory $\boldsymbol{q}(t)$ is uniquely determined, we impose the boundary conditions:

$$\delta \boldsymbol{q}(t_1) = \delta \boldsymbol{q}(t_2) = 0.$$

This assumption reflects the fact that the initial and final positions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are fixed *a priori*. Under this condition, the boundary term:

$$\left[\frac{\partial L}{\partial \dot{\boldsymbol{q}}} \delta \boldsymbol{q} \right]_{t_1}^{t_2}$$

vanishes, leaving us with:

$$\int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \boldsymbol{q}} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \right) \delta \boldsymbol{q} \, \mathrm{d}t = 0.$$

Since this must hold for *any* infinitesimal variation $\delta \boldsymbol{q}$, the fundamental theorem of calculus of variations implies:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} = 0.$$

These are the *Euler-Lagrange equations*, which describe the equations of motion of the system.

Notice that

- By moving the variational operator inside the integral, we obtain an expression involving $\delta \boldsymbol{q}$ and $\delta \dot{\boldsymbol{q}}$.
- Applying integration by parts, we separate boundary terms from terms inside the integral.
- The assumption $\delta \boldsymbol{q}(t_1) = \delta \boldsymbol{q}(t_2) = 0$ removes the boundary term, leading directly to the Euler-Lagrange equations.

These equations form the cornerstone of Lagrangian mechanics, providing a general and elegant formulation of classical mechanics.

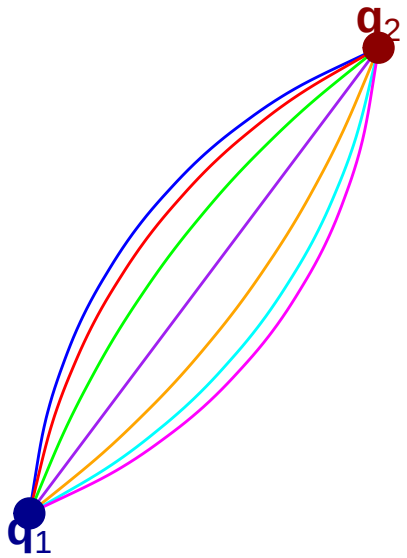


Figure 10.2: A graphical representation of the allowable variations with fixed endpoints $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$.

10.4.2 Derivation of the Euler-Lagrange Equations

Having established the variation of the action in equation Equation 10.5, we are now in a position to simplify it further using a fundamental result from the calculus of variations.

The Fundamental Lemma of the Calculus of Variations

A crucial step in our derivation is the *Fundamental Lemma of the Calculus of Variations*, which allows us to extract the equations of motion from the stationary condition of the action.

Lemma 10.1 — Fundamental Lemma of Calculus of Variations. Let S be a *differentiable* functional in the variational sense. Suppose that for every smooth variation $\delta \boldsymbol{q}(t)$ satisfying the boundary conditions

$$\delta \boldsymbol{q}(t_1) = \delta \boldsymbol{q}(t_2) = 0$$

we have:

$$\int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \boldsymbol{q}} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} \right) \delta \boldsymbol{q} \, \mathrm{d}t = 0.$$

Then, it necessarily follows that the integrand must vanish identically:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L^{\mathsf{T}}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L^{\mathsf{T}}}{\partial \boldsymbol{q}} = 0. \tag{10.6}$$

This result is fundamental because it allows us to conclude that if an integral of a function multiplied by an arbitrary function (the variation $\delta\boldsymbol{q}$) is always zero, then the function itself must be identically zero. This is a direct consequence of the fact that variations $\delta\boldsymbol{q}$ can be chosen freely within the class of smooth functions that vanish at the endpoints.

Interpretation and Physical Meaning

The system of equations in Equation 10.6 is known as the **Euler-Lagrange equations**. These are the fundamental equations that govern the motion of mechanical systems within the framework of Lagrangian mechanics.

Intuitively, the fundamental lemma tells us that since the variation $\delta\boldsymbol{q}(t)$ is arbitrary (except for vanishing at the endpoints), the only way to ensure that the integral always evaluates to zero is to have the coefficient of $\delta\boldsymbol{q}$ vanish at every instant in time. This leads directly to the equations of motion.

Thus, we can summarize our findings in the following theorem:

Theorem 10.2 — Theorem: Euler-Lagrange Equations. The physical trajectory $\boldsymbol{q}(t)$ of a mechanical system described by a Lagrangian $L(\boldsymbol{q},\dot{\boldsymbol{q}},t)$ satisfies the system of differential equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L^{\mathsf{T}}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L^{\mathsf{T}}}{\partial \boldsymbol{q}} = 0.$$

These equations provide the necessary conditions for the action functional S to be stationary under small variations of the path $\boldsymbol{q}(t)$.

Notice that

1. We derived the variation of the action integral and expressed it in terms of $\delta\boldsymbol{q}$ and $\delta\dot{\boldsymbol{q}}$.
2. Through integration by parts, we isolated the term involving $\delta\boldsymbol{q}$ inside the integral and discarded the boundary terms by assuming that $\delta\boldsymbol{q}(t_1) = \delta\boldsymbol{q}(t_2) = 0$.
3. We applied the Fundamental Lemma of the Calculus of Variations to conclude that the coefficient of $\delta\boldsymbol{q}$ must be identically zero, leading to the Euler-Lagrange equations.

This approach not only provides a rigorous derivation but also highlights the deep connection between variational principles and the fundamental laws of motion.

10.5 Principle of Minimal Action and Coordinate Independence

The following theorem expresses the fundamental idea of the *principle of stationary action*:

Theorem 10.3 — Minimal Action Principle. A curve

$$\boldsymbol{q} : t \mapsto \boldsymbol{q}(t), \quad \text{for} \quad t \in [t_1, t_2]$$

is an extremal of the functional S in the space of curves that pass through the points

$$\boldsymbol{q}(t_1) = \boldsymbol{q}_1 \quad \text{and} \quad \boldsymbol{q}(t_2) = \boldsymbol{q}_2$$

if and only if it satisfies the Euler-Lagrange equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0.$$

In practice the physical trajectory of a system is one that makes the action S stationary. In other words, among all possible paths that connect two fixed endpoints, the actual path followed by the system is the one that satisfies the Euler-Lagrange equations.

10.5.1 Coordinate Invariance of the Equations of Motion

A crucial feature of the Euler-Lagrange equations is their *coordinate-free* nature. This means that the equations remain valid regardless of the specific coordinate system chosen to describe the motion. The principle of least action, and thus the resulting equations of motion, do not depend on how we label points in space but rather on the intrinsic physics of the system.

To illustrate this, consider the case of a free particle moving under gravity, analyzed in two different coordinate systems.

■ **Example 10.4 — Motion of a Free Mass Point in Different Coordinates.** We examine the equations of motion for a particle of mass m subject to gravity in both Cartesian and polar coordinates.

Cartesian Coordinates

In Cartesian coordinates, the equations of motion follow directly from the Euler-Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = m\ddot{q} + mge_2 = 0. \tag{10.7}$$

Here, e_2 is the unit vector in the vertical direction.

Polar Coordinates

In polar coordinates $(\tilde{q}_1, \tilde{q}_2)$, where $\tilde{q}_1$ represents the radial coordinate and $\tilde{q}_2$ the angular coordinate, the equations take a different form:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\tilde{q}}_1} - \frac{\partial L}{\partial \tilde{q}_1} &= m\ddot{\tilde{q}}_1 + mg\sin(\tilde{q}_2), \\ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\tilde{q}}_2} - \frac{\partial L}{\partial \tilde{q}_2} &= m\tilde{q}_1^2\ddot{\tilde{q}}_2 + 2m\tilde{q}_1\dot{\tilde{q}}_1\dot{\tilde{q}}_2 + mg\tilde{q}_1\cos(\tilde{q}_2) = 0. \end{aligned} \tag{10.8}$$

Even though the equations appear different in Cartesian and polar coordinates, they describe the same physical motion. The coordinate transformation between the two representations maintains the fundamental structure of the equations, underscoring the coordinate-free nature of Lagrangian mechanics.

■

10.5.2 Beyond Conservative Systems: Generalized Forces

One limitation of the Euler-Lagrange formulation, as presented so far, is that it applies strictly to *conservative systems*, where the forces derive from a potential energy function. However, many physical systems include *non conservative forces*, such as friction or external driving forces, which do not fit within the standard Lagrangian framework.

To extend the formulation to non conservative forces, we introduce **D'Alembert's principle**, which leads to a modification of the Euler-Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L^\mathsf{T}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L^\mathsf{T}}{\partial \boldsymbol{q}} = \boldsymbol{\tau}.$$

Here, $\boldsymbol{\tau}$ represents the generalized forces that account for non conservative effects such as:

- **Dissipative forces**, e.g., friction or air resistance,
- **Time-dependent external forces**, such as driving forces in oscillatory systems,
- **Constraint forces**, which arise from non holonomic constraints.

This generalized equation provides a more complete description of mechanical systems, making the Lagrangian formalism applicable even when energy is not conserved.

Notice that:

1. **The principle of stationary action** determines the trajectory of a system by making the action integral stationary, leading to the **Euler-Lagrange equations**.
2. **The equations of motion are coordinate-invariant**, meaning the same physical motion can be described in different coordinate systems without altering the underlying physics.
3. **Nonconservative forces** can be incorporated into the Euler-Lagrange framework using D'Alembert's principle, leading to an extended form of the equations of motion.

This powerful formulation unifies mechanics into a single, elegant framework that applies across different coordinate systems and physical settings.

10.6 Mechanical Equations of Motion

While the Euler-Lagrange equations provide a general framework, it is often helpful to consider explicit representations for mechanical systems with specific characteristics. For systems where

- the kinetic energy $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a quadratic form in the generalized velocities $\dot{\boldsymbol{q}}$,

we will derive explicit expressions for the equations of motion. This assumption is broadly applicable, as any mechanical system with a finite number of degrees of freedom can be described by a quadratic kinetic energy.

10.6.1 Kinetic Energy Representation

From this foundational assumption, many interesting properties emerge. For a more comprehensive treatment, refer to [21]. The kinetic energy can be written as:

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} m_{jk}(\boldsymbol{q}) \dot{q}_j \dot{q}_k = \frac{1}{2} \dot{\boldsymbol{q}}^{\mathsf{T}} \boldsymbol{M}(\boldsymbol{q}) \dot{\boldsymbol{q}},$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is the symmetric positive definite matrix known as the *mass matrix*. The entries $m_{jk}(\boldsymbol{q})$ of $\boldsymbol{M}(\boldsymbol{q})$ are referred to as the *generalized masses* or *generalized inertias*.

Remark 10.4 The matrix $\boldsymbol{M}(\boldsymbol{q})$ is symmetric because the quadratic form associated with a matrix depends only on its symmetric part.

10.6.2 Deriving the Equations of Motion

We now aim to derive the explicit equations of motion for mechanical systems that exhibit *quadratic* kinetic energy. This process begins with the Euler-Lagrange equations, given in Equation 10.6.

First Term: Time Derivative of the Kinetic Energy

Consider the term:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{q}_i}$$

This term can be expressed as:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{q}_i} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial T}{\partial \dot{q}_i} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial}{\partial \dot{q}_i} \left(\frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} m_{jk}(\boldsymbol{q}) \dot{q}_j \dot{q}_k \right),$$

for $i = 1, \ldots, n$. By exploiting the linearity of summation, we can interchange the order of the partial derivative and the summation:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} m_{jk}(\boldsymbol{q})\frac{\partial \dot{q}_j}{\partial \dot{q}_i}\dot{q}_k + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} m_{jk}(\boldsymbol{q})\dot{q}_j\frac{\partial \dot{q}_k}{\partial \dot{q}_i}\right)\\ &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{2}\sum_{k=1}^{n} m_{ik}(\boldsymbol{q})\dot{q}_k + \frac{1}{2}\sum_{j=1}^{n} m_{ji}(\boldsymbol{q})\dot{q}_j\right)\\ &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\dot{q}_j\right).\end{aligned} \tag{10.9}$$

In the last step, we used the fact that:

$$\frac{\partial \dot{q}_j}{\partial \dot{q}_i} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

and the symmetry condition $m_{ij}(\boldsymbol{q}) = m_{ji}(\boldsymbol{q})$ for $i, j = 1, \ldots, n$.
Taking the time derivative of this expression yields:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} = \sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial m_{ij}(\boldsymbol{q})}{\partial q_k}\dot{q}_k\dot{q}_j. \tag{10.10}$$

Second Term: Derivative of Potential Energy

Next, we compute the term $\frac{\partial L}{\partial q_i}$, for $i = 1, \ldots, n$:

$$\frac{\partial L}{\partial q_i} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial m_{jk}(\boldsymbol{q})}{\partial q_i}\dot{q}_k\dot{q}_j - \frac{\partial U(\boldsymbol{q},t)}{\partial q_i}. \tag{10.11}$$

We define the following:

$$g_i(\boldsymbol{q}) := \frac{\partial U(\boldsymbol{q},t)}{\partial q_i},$$

for $i = 1, \ldots, n$, and the vector of generalized forces:

$$\boldsymbol{g}(\boldsymbol{q}) := (g_1(\boldsymbol{q}), \ldots, g_n(\boldsymbol{q})) \in \mathbb{R}^n.$$

Equations of Motion

Combining Equation 10.10 and Equation 10.11, we obtain the componentwise equations of motion:

$$\begin{aligned}\sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial m_{ij}(\boldsymbol{q})}{\partial q_k}\dot{q}_j\dot{q}_k \\ - \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial m_{jk}(\boldsymbol{q})}{\partial q_i}\dot{q}_j\dot{q}_k + g_i(\boldsymbol{q}) = \tau_i,\end{aligned} \tag{10.12}$$

where τ_i represents the generalized forces acting on the i-th degree of freedom.

Introducing Inertia Terms

To simplify, we define the following term:

$$h_{ijk}(\boldsymbol{q}) := \frac{\partial m_{ij}(\boldsymbol{q})}{\partial q_k} - \frac{1}{2}\frac{\partial m_{jk}(\boldsymbol{q})}{\partial q_i}. \tag{10.13}$$

Substituting this into Equation 10.12, we get a more compact form of the equations of motion:

$$\sum_{j=1}^{n} m_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{q})\dot{q}_j\dot{q}_k + g_i(\boldsymbol{q}) = \tau_i, \tag{10.14}$$

Matrix Formulation

The equations of motion can be rewritten in compact matrix notation:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau},$$

where $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the matrix that captures the centrifugal and Coriolis effects. The exact form of $\boldsymbol{C}$ depends on the system's geometry and dynamics.

Remark 10.5 It's crucial to note that the number of degrees of freedom in a system cannot be assumed to be $n-m$ without certain regularity conditions on the constraint function $\boldsymbol{\phi}(\boldsymbol{q},t)$. Specifically, if:

$$\operatorname{rank}\left(\frac{\partial \boldsymbol{\phi}(\boldsymbol{q},t)}{\partial \boldsymbol{q}}\right) = m,$$

then the system has exactly $n-m$ degrees of freedom, assuming that $\boldsymbol{\phi}$ is differentiable.

10.7 Constrained Systems and Lagrange Multipliers

For mechanical systems subject to constraints, we use the method of Lagrange multipliers to handle the constraint equation

$$\boldsymbol{\phi}(\boldsymbol{q},t) = 0.$$

The *modified action* $\bar{S}$ is defined as:

$$\bar{S} := \int_{t_1}^{t_2} \left(L(\boldsymbol{q},\dot{\boldsymbol{q}},t) + \boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{q},t)\right) \mathrm{d}t,$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of Lagrange multipliers. This formulation is critical for analyzing the dynamics of constrained systems.

Theorem 10.4 — Stationary Action for Constrained Systems. The motion of mechanical systems transitioning from configuration $\boldsymbol{q}_1$ to $\boldsymbol{q}_2$ over the time interval $t \in [t_1,t_2]$ corresponds to the extremal curves of the functional defined in the equation above.

In other words, for small variations in the curve $\boldsymbol{q}(\cdot)$, the modified action $\bar{S}$ becomes stationary:

$$\delta\bar{S} = 0.$$

Remark 10.6 It is important to note that we implicitly assume that the initial and final configurations, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, satisfy the constraint equations:

$$\boldsymbol{\phi}(\boldsymbol{q}_1,t_1) = \boldsymbol{\phi}(\boldsymbol{q}_2,t_2) = 0.$$

Thus, we can perform variations as we did previously for systems with a minimal set of coordinates. However, in this case, $\bar{S}$ also depends on the Lagrange multipliers. The variational operator δ applies variations to $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\boldsymbol{\lambda}$.

The result of the variation is:

$$\begin{aligned}\delta\bar{S} &= \int_{t_1}^{t_2} \delta L(\boldsymbol{q},\dot{\boldsymbol{q}},t) + \delta\boldsymbol{\lambda}^\top\boldsymbol{\phi}(\boldsymbol{q},t) + \boldsymbol{\lambda}^\top\delta\boldsymbol{\phi}(\boldsymbol{q},t)\,\mathrm{d}t \\ &= \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \boldsymbol{q}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} + \boldsymbol{\lambda}^\top\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\right)\delta\boldsymbol{q} + \delta\boldsymbol{\lambda}^\top\boldsymbol{\phi}(\boldsymbol{q},t)\,\mathrm{d}t + \left[\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\delta\boldsymbol{q}\right]_{t_1}^{t_2} \\ &= \int_{t_1}^{t_2} \begin{bmatrix}\frac{\partial L}{\partial \boldsymbol{q}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} + \boldsymbol{\lambda}^\top\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} & \boldsymbol{\phi}^\top\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{q}\\ \delta\boldsymbol{\lambda}\end{bmatrix}\mathrm{d}t = 0.\end{aligned}$$

We use the fact that the variations at the endpoints are zero: $\delta\boldsymbol{q}_1 = \delta\boldsymbol{q}_2 = 0$.

Finally, applying the *fundamental lemma of calculus of variations* to the above expression gives:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L^\top}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L^\top}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^\top\boldsymbol{\lambda},$$

$$\boldsymbol{\phi}(\boldsymbol{q},t) = 0.$$

The equations above describe the dynamics of a mechanical system subject to holonomic constraints. The term

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^\top\boldsymbol{\lambda}$$

acts as a generalized force ensuring that the constraints $\boldsymbol{\phi}(\boldsymbol{q},t) = 0$ are satisfied. Unlike the equations derived from the Euler-Lagrange equations, this system involves both *differential* and *algebraic* equations, making it a *Differential Algebraic Equation* (DAE). Additionally, the system does not provide a direct method for determining the Lagrange multipliers.

When external non-conservative forces are present, the equations of motion modify to:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L^\top}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L^\top}{\partial \boldsymbol{q}} &= \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^\top\boldsymbol{\lambda} + \boldsymbol{\tau}, \\ \boldsymbol{\phi}(\boldsymbol{q},t) &= 0,\end{aligned}$$

where $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of generalized non-conservative forces.

■ **Example 10.5 — Pendulum System.** Let us consider a simple mechanical system: the pendulum, as illustrated in Figure 10.1.

The pendulum has one degree of freedom, represented by the angular coordinate $\theta \in (-\pi,\pi)$. The relationships between the angular coordinate and the Cartesian coordinates $(x,y) \in \mathbb{R}^2$ are given by:

$$x = r\sin\theta, \quad y = -r\cos\theta.$$

The kinetic and potential energies can be expressed as:

$$T(\theta) = \frac{1}{2}m(\dot{x}^2+\dot{y}^2) = \frac{1}{2}mr^2\dot{\theta}^2,$$

$$U(\theta) = mgy = -mgr\cos\theta.$$

Thus, the Lagrangian for the system is:

$$L = \frac{1}{2}mr^2\dot{\theta}^2 + mgr\cos\theta.$$

Applying the Euler-Lagrange equations, we obtain:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = m r^2 \ddot{\theta} + m g r \sin\theta = 0,$$

which simplifies to:

$$\ddot{\theta} = -\frac{g}{r}\sin\theta.$$

Next, we consider describing the pendulum using a set of redundant coordinates, such as *natural coordinates*. Let the generalized coordinate be $\boldsymbol{q} = (x; y) \in \mathbb{R}^2$. The Lagrangian function can be written as:

$$L = \frac{1}{2} m(\dot{\boldsymbol{q}} \cdot \dot{\boldsymbol{q}}) - m g(\boldsymbol{e}_2 \cdot \boldsymbol{q}),$$

where $\boldsymbol{e}_2 \in \mathbb{R}^2$ is the unit vector $\boldsymbol{e}_2 = (0; 1)$. To ensure the mass m moves along the circular path, we introduce the constraint equation:

$$\phi(\boldsymbol{q}) = x^2 + y^2 - r^2 = (\boldsymbol{q} \cdot \boldsymbol{q}) - r^2 = 0.$$

Using the equations from above, the resulting system of equations becomes:

$$\begin{aligned} m\ddot{\boldsymbol{q}} + mg\boldsymbol{e}_2 &= 2\boldsymbol{q}\lambda, \\ (\boldsymbol{q} \cdot \boldsymbol{q}) - r^2 &= 0. \end{aligned}$$

or more explicitly

$$\begin{aligned} m x'' &= 2x\lambda \\ m y'' &= 2y\lambda - mg \\ x^2 + y^2 &= r^2 \end{aligned}$$

■



11. Numerical Integration of DAE

As pointed out in [17], both from a theoretical and numerical point of view, DAEs are much more challenging than ordinary differential equations (ODEs). Some DAEs can be solved using numerical methods for stiff systems, but others cannot, and they are very sensitive to the step size, which may lead to large errors in the solution and numerical instability. For DAEs arising from constrained mechanical systems, a fairly general form is the following:

$$\begin{cases} \boldsymbol{M}(\boldsymbol{q},\boldsymbol{q}',t)\boldsymbol{q}'' - \dfrac{\partial \boldsymbol{\phi}(\boldsymbol{q},t)}{\partial \boldsymbol{q}}^T \boldsymbol{\lambda} = \boldsymbol{n}(\boldsymbol{q},\boldsymbol{q}',t), \\ \boldsymbol{\phi}(\boldsymbol{q},t) = 0 \end{cases}$$

where the term

$$\boldsymbol{n}(\boldsymbol{q},\boldsymbol{q}',t) \in \mathbb{R}^n$$

collects all the contributions coming from Coriolis and centrifugal effects, conservative and non-conservative forces. It can be defined as follows:

$$\boldsymbol{n}(\boldsymbol{q},\boldsymbol{q}',t) = \boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q},\boldsymbol{q}')\boldsymbol{q}' - \boldsymbol{g}(\boldsymbol{q}).$$

It is well known that the equations of motion for a mechanical system are a system of second-order (possibly non-linear) differential equations. However, for clarity and generality, it is useful to consider a *first-order representation*. For this purpose, let us define $\boldsymbol{v} \in \mathbb{R}^n$ as *generalized velocities*, i.e.,

$$\boldsymbol{v} := \boldsymbol{q}'.$$

Thus, substituting the above equation into the general form of the equation of motion results in:

$$\begin{cases} \boldsymbol{q}' = \boldsymbol{v}, \\ \boldsymbol{M}(\boldsymbol{q},\boldsymbol{v},t)\boldsymbol{v}' - \dfrac{\partial \boldsymbol{\phi}(\boldsymbol{q},t)}{\partial \boldsymbol{q}}^T \boldsymbol{\lambda} = \boldsymbol{n}(\boldsymbol{q},\boldsymbol{v},t), \\ \boldsymbol{\phi}(\boldsymbol{q},t) = 0. \end{cases} \tag{11.1}$$

Notice that the vector

$$\begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{pmatrix} \in \mathbb{R}^{2n}$$

represents the *state* of a mechanical system, namely the pair of *generalized position* and *generalized velocity*.

11.1 Reduction to ODE

Now, a good question is how to solve the problem in the equation above. The issue is how to determine the value for the Lagrange multiplier $\boldsymbol{\lambda}$. One possible approach is to transform the problem into a system of pure differential equations. In order to do so, we need to differentiate the constraint $\boldsymbol{\phi}(\boldsymbol{q},t)$, which yields:

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{v} + \frac{\partial \boldsymbol{\phi}}{\partial t} = \boldsymbol{0}.$$

Notice that when the constraint does not depend on time, this expression indicates that the generalized velocity $\boldsymbol{v}$ must be orthogonal to the gradient of the constraint. This also provides the opportunity to note that the constraint $\boldsymbol{\phi}(\boldsymbol{q},t)$ imposes relationships at the velocity and acceleration levels, restricting the set of feasible velocities. The constraints that we can obtain by differentiating the constraint are sometimes called *hidden constraints*.

Finally, notice that the equation above is still algebraic in the $\boldsymbol{v}$ coordinate. Thus, in order to obtain a differential relationship, let us differentiate again:

$$\frac{\mathrm{d}^2\boldsymbol{\phi}(\boldsymbol{q},t)}{\mathrm{d}t^2} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{v}\right) + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{v}' + \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} = \boldsymbol{0}.$$

Notice that now this expression does not impose any algebraic constraint on the state of the mechanical system $(\boldsymbol{q};\boldsymbol{v})$. Substituting the constraint $\boldsymbol{\phi}(\boldsymbol{q},t) = \boldsymbol{0}$ with the one derived above results in the following modified problem:

$$\begin{cases} \boldsymbol{q}' = \boldsymbol{v}, \\ \boldsymbol{M}(\boldsymbol{q},\boldsymbol{v},t)\boldsymbol{v}' - \dfrac{\partial \boldsymbol{\phi}(\boldsymbol{v},t)}{\partial \boldsymbol{q}}^T \boldsymbol{\lambda} = \boldsymbol{n}(\boldsymbol{q},\boldsymbol{v},t), \\ -\dfrac{\partial \boldsymbol{\phi}(\boldsymbol{v},t)}{\partial \boldsymbol{q}}\boldsymbol{v}' = \boldsymbol{m}(\boldsymbol{q},\boldsymbol{v},t). \end{cases} \tag{11.2}$$

Here, to simplify the notation, we introduce the map

$$\boldsymbol{m}(\boldsymbol{q},\boldsymbol{v},t) := \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{v}\right) + \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2}, \quad \boldsymbol{m}(\boldsymbol{q},\boldsymbol{v},t) \in \mathbb{R}^m.$$

Even though differentiating the constraint seems promising, the problem in the above equations and the one derived earlier are *not equivalent*. This difference leads to the so-called **drift effect**, which will be discussed in the next section. For now, let's continue along the current path. The system of equations can also be written in compact matrix notation as follows:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}(\boldsymbol{q},\boldsymbol{v},t) & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T(\boldsymbol{q},t) \\ \boldsymbol{0} & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}(\boldsymbol{q},t) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}' \\ \boldsymbol{v}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{n}(\boldsymbol{q},\boldsymbol{v},t) \\ \boldsymbol{m}(\boldsymbol{q},\boldsymbol{v},t) \end{bmatrix}.$$

Now, it's easy to see that if the matrix on the left-hand side is invertible, we can express the system in explicit form and use this representation to perform numerical simulations. The matrix is invertible if and only if the sub-block:

$$\begin{bmatrix} \boldsymbol{M} & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T \\ -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)},$$

is non-singular (i.e., full rank).

Recall that $\boldsymbol{M}$ is positive definite (and thus non-singular), so that

$$\begin{bmatrix} \boldsymbol{M} & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T \\ -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{M}^{-1} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}^{-1}\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

where

$$\boldsymbol{D} = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T$$

and thus the block is non-singular if $\boldsymbol{D}$ is non-singular.

This is not generally true, but assuming that the constraints are locally linearly independent (i.e., $\forall \boldsymbol{q} \in \mathbb{R}^n, \forall t$):

$$\operatorname{rank}\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\right) = m,$$

then $\boldsymbol{D}$ is non-singular. Thus, the system can be expressed in *explicit first-order form* as follows:

$$\begin{bmatrix} \boldsymbol{q}' \\ \boldsymbol{v}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \boldsymbol{M}(\boldsymbol{q},\boldsymbol{v},t) & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T(\boldsymbol{q},t) \\ 0 & -\dfrac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}(\boldsymbol{q},t) & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{n}(\boldsymbol{q},\boldsymbol{v},t) \\ \boldsymbol{m}(\boldsymbol{q},\boldsymbol{v},t) \end{bmatrix}.$$

An explicit expression for the inverse matrix can be obtained through tedious calculations (a calculation is done when numerical simulation are done). Regardless of the specific expression for the solution, the important point to emphasize is that by differentiating the constraint, we were able to find an explicit expression for the Lagrange multipliers. For this purpose, we differentiated twice and obtained the equation:

$$\boldsymbol{\lambda} = -\left(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{M}^{-1}\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}^T\right)^{-1}\left(\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{v} + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}}\boldsymbol{M}^{-1}\boldsymbol{n}(\boldsymbol{q},\boldsymbol{v},t) + \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2}\right).$$

Technically, mechanical systems subject to holonomic constraints are systems of differential-algebraic equations (DAEs) of *differential index* 3. The differential index is one plus the number of differentiations of the constraint that are needed in order to eliminate the Lagrange multipliers. Alternatively, it is the number of times that we need to differentiate the constraint in order to obtain a differential equation for the Lagrange multiplier.

11.2 The drift effect

In this section we want investigate the effects of the differentiating process done to transform problem in Equation 11.1 into problem Equation 11.2. Unfortunately this method is *not* priceless, indeed formally we substitute an algebraic constraint with its second derivative w.r.t time. To better understand the consequences let us consider a fictitious constraint $\widetilde{\boldsymbol{\phi}}(\boldsymbol{q},t)$ of the form,

$$\widetilde{\boldsymbol{\phi}}(\boldsymbol{q},t) := \boldsymbol{\phi}(\boldsymbol{q},t) + \boldsymbol{a}_0 + \boldsymbol{a}_1 t.$$

It is easy to see that consider a mechanical system subject to constraint $\phi(\boldsymbol{q},t)$ or $\widetilde{\boldsymbol{\phi}}(\boldsymbol{q},t)$ differentiating twice is exactly the same, i.e.,

$$\frac{\mathrm{d}^2\boldsymbol{\phi}}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\widetilde{\boldsymbol{\phi}}}{\mathrm{d}t^2}.$$

This fact produces the *drift effect*, i.e., the original constraint $\boldsymbol{\phi}(\boldsymbol{q},t)$ is violated with an error growing with time. This error grows in the best case linearly, but it can be faster due to *numerical* errors. The reason of this behaviour is the term $\boldsymbol{a}_1 t$ which does not have any effect in the formulation Equation 11.2. Note also that this is not an undesired effect coming from not enough precision of the numerical scheme, but it is intrinsic effect due to substitution of the constraint $\boldsymbol{\phi}$ with its second derivative w.r.t time. A simple solution to this undesired effect is the Baumgarte's stabilization that we will discuss in the next section.

■ **Example 11.1** Continuing Example 11.1 we provide a first order representation for the pendulum; setting $\boldsymbol{v} = \boldsymbol{q}'$ results,

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{q}', \\ m\boldsymbol{v}' + mg\boldsymbol{e}_2 &= 2\boldsymbol{q}\lambda, \\ \boldsymbol{q}^T\boldsymbol{q} - \ell^2 &= 0. \end{aligned}$$

Then in order to eliminate the Lagrange multiplier let us differentiate the constraint,

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = 2\boldsymbol{q}^T\boldsymbol{v} = 0.$$

Notice that as expected velocity $\boldsymbol{v}$ and position $\boldsymbol{q}$ are orthogonal each other. Differentiating again we obtain

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}t^2} = 2(\boldsymbol{q}')^T\boldsymbol{v} + 2\boldsymbol{q}^T\boldsymbol{v}' = 2\boldsymbol{v}^T\boldsymbol{v} + 2\boldsymbol{q}^T\boldsymbol{v}' = 0.$$

Now we can write the problem in shape of Equation 11.3,

$$\begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & m\boldsymbol{I} & -2\boldsymbol{q} \\ 0 & -2\boldsymbol{q}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{q}' \\ \boldsymbol{v}' \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ -mg\boldsymbol{e}_2 \\ 2\boldsymbol{v}^T\boldsymbol{v} \end{bmatrix}, \tag{11.3}$$

■

where $\boldsymbol{I} \in \mathbb{R}^{2\times 2}$ is the identity matrix of dimension 2. The explicit value for the Lagrange multipliers is the following

$$\lambda = -\frac{m}{2\boldsymbol{q}^T\boldsymbol{q}}\left(\boldsymbol{v}^T\boldsymbol{v} - g\boldsymbol{q}^T\boldsymbol{e}_2\right) = -\frac{m}{2\ell^2}\left(\boldsymbol{v}^T\boldsymbol{v} - gq_2\right). \tag{11.4}$$

To better appreciate the drift effect we can simulate Equation 11.3 using an ODE solver. The results are showed in Figure 11.1

Simulation has been performed for 50 seconds with the following parameters,

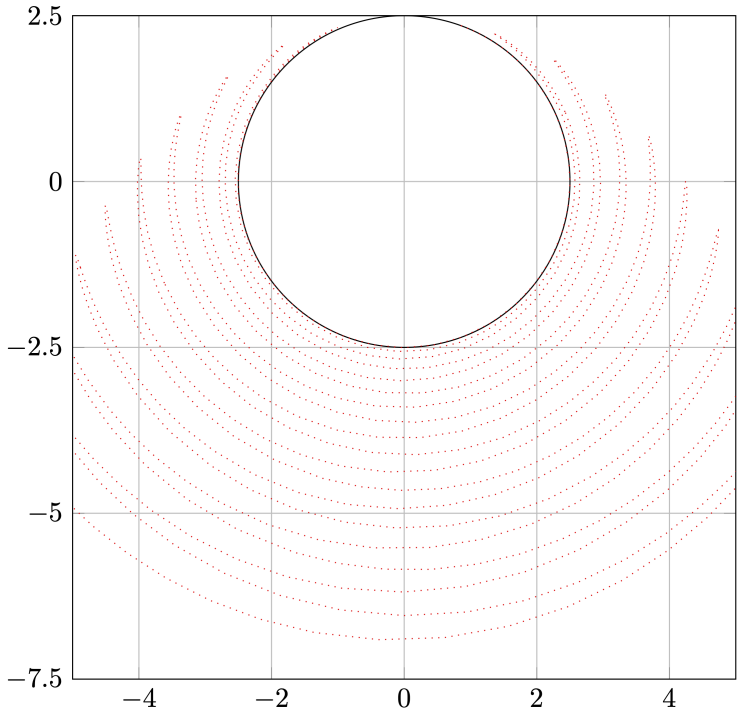


Figure 11.1: Pendulum simulation with drift effect.

- $g = 9.81$
- $m = 2$
- $\ell = 2.5$

and initial conditions

- $\boldsymbol{q}_0 = (0.86; 2.35) \in \mathbb{R}^2$,
- $\boldsymbol{q}_0' = \boldsymbol{v}_0 = (0,0) \in \mathbb{R}^2$
- and $\lambda_0 = 3.69$

which are all consistent with the constraint and with Equation 11.4.

V Index Reduction

12 Index reduction of a Linear DAE 119
12.1 Steps to Compute the Index 119
12.2 Practical construction of Q_a and Q_d 121
12.4 Example 1 122
12.3 Index Reduction Algorithm for linear DAEs . . . 123
12.5 Example 2 125
12.6 Example 3 128

13 Index reduction of a NonLinear DAE . 131
13.1 General DAE 131
13.2 Semi-Explicit Form of DAEs 131
13.3 Matrix Representation 132
13.4 Steps to Compute the Index of a nonlinear DAE 132

14 Index Reduction of DAEs Using Dummy Variables 135
14.1 Introduction 135
14.2 System Requirements 135
14.3 Preprocessing 136
14.4 Index reduction algorithm 137

15 Linear algebra tools used in the index reduction process 141
15.1 Index Reduction through Kernel Computation 141

16 Examples of index reduction 145
16.1 Example 1 145
16.2 Example 2 147
16.3 Example 3 150

Differential-Algebraic Equations (DAEs) frequently arise in the modeling of physical, chemical, and engineering systems where constraints coexist with dynamic behavior. One of the fundamental characteristics of a DAE system is its index. The index of a differential-algebraic equation (DAE) system represents the minimum number of times the algebraic constraints must be differentiated to rewrite the system explicitly as an ordinary differential equation (ODE), i.e., it measures how far the system is from an explicit Ordinary Differential Equation (ODE) form. High-index DAEs are often challenging to solve numerically due to issues such as stiffness, sensitivity to perturbations, and the need for consistent initial conditions.

Index reduction refers to the process of transforming a high-index DAE into an equivalent system with a lower index to facilitate stable and accurate numerical solution. It involves linear transformations and differentiations to isolate the algebraic constraints and convert them into differential form to reformulate the system into a semi-explicit or even explicit form. In particular, at each iteration of the reduction:

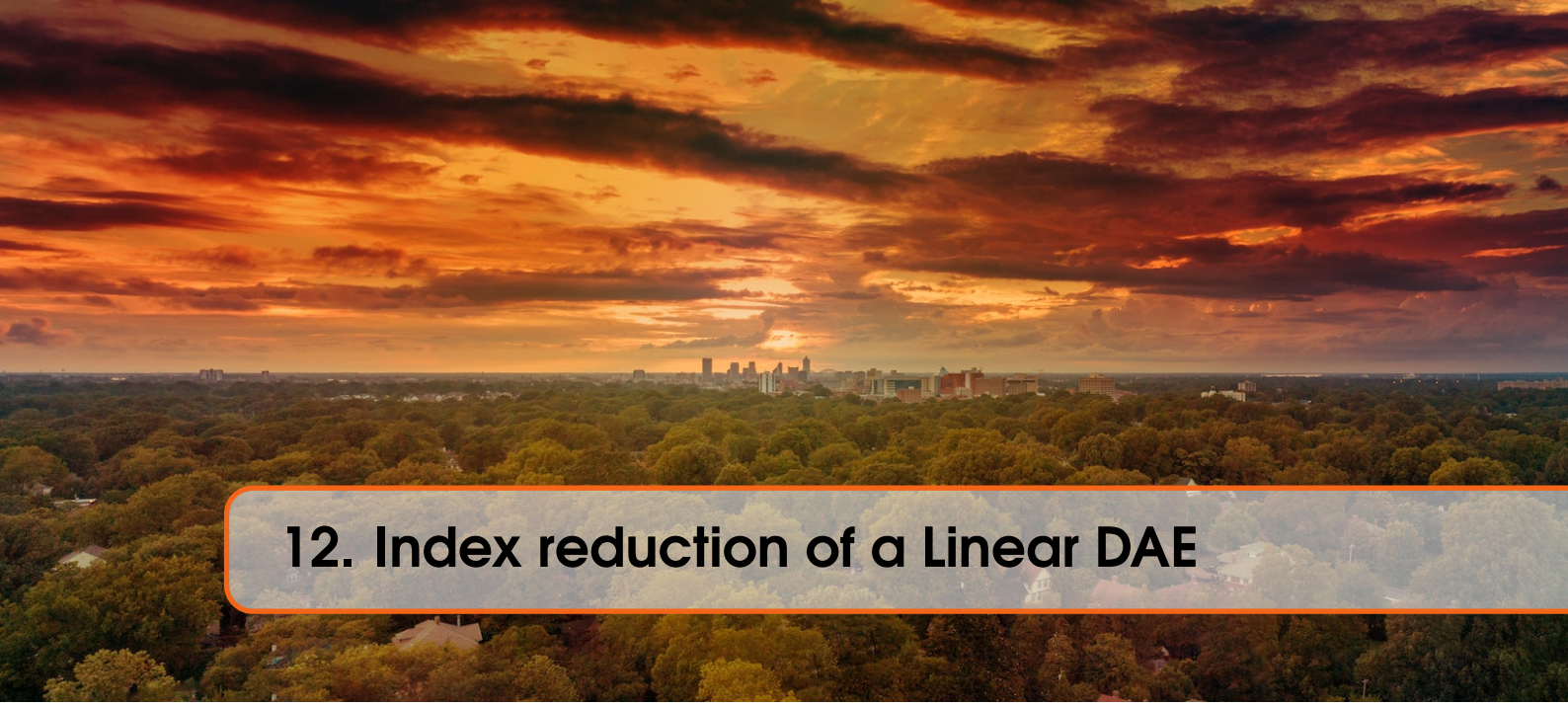
- We separate the algebraic part from the differential part finding the *algebraic invariants* or *hidden constraints* of the final ODE, i.e., conditions that must hold for all time and must be respected by the numerical solution.
- We differentiate the algebraic equations, reducing the index by one until the algebraic part is empty.

Typically, it is sufficient to reduce the DAE to index 1, since index-1 DAEs can be solved using standard numerical integrators.

In this chapter, we explore the concept of index in DAEs, the motivation for index reduction, and various techniques used to perform it, such as the Pantelides algorithm [16] and dummy derivative method [14]. These techniques are essential tools in the preprocessing pipeline for simulation, control design, and optimization of constrained dynamical systems.

Learning objectives

After this part, you should be able to separate differential and algebraic variables, carry out an index-reduction step, and retain the resulting hidden constraints as conditions that the initial data and numerical solution must satisfy.



12. Index reduction of a Linear DAE

A Differential-Algebraic Equation (DAE) is a system of equations that consists of both differential and algebraic equations. Unlike ordinary differential equations (ODEs), DAEs may contain constraints that restrict the solution space. The **(differential) index** of a DAE measures how many times the system must be differentiated to convert it into an explicit ODE.

This tutorial explains how to compute the index of a linear DAE systematically, assuming constant coefficient matrices and avoiding the Kronecker canonical form.

12.1 Steps to Compute the Index

First recall that a linear DAE with constant coefficients is given by:

$$\boldsymbol{E}\boldsymbol{x}' + \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}(t)$$

where:

- $\boldsymbol{E}$ and $\boldsymbol{A}$ are constant matrices,
- $\boldsymbol{x}(t)$ is the state variable,
- $\boldsymbol{f}(t)$ is an input function,
- $\boldsymbol{x}'$ is the time derivative of $\boldsymbol{x}$.

The following steps provide a rough outline of the approach, offering a general understanding of the index reduction procedure.

Step 1: Check if $\boldsymbol{E}$ is Singular

Compute $\det(\boldsymbol{E})$:

- If $\det(\boldsymbol{E}) \neq 0$, then $\boldsymbol{E}$ is invertible, and the system reduces to an ODE (**index 0**). Terminate procedure.
- If $\det(\boldsymbol{E}) = 0$, then $\boldsymbol{E}$ is singular, indicating the presence of algebraic constraints. Proceed to the next step.

Step 2: Decompose the System into Differential and Algebraic Parts

Determine the **rank** of $\boldsymbol{E} \in \mathbb{R}^{n\times n}$, denoted as $r = \operatorname{rank}(\boldsymbol{E})$. Find:

- A **basis** $\boldsymbol{Q}_a \in \mathbb{R}^{n\times(n-k)}$ for the **null space** $\mathcal{N}(\boldsymbol{E}^T)$, which identifies the algebraic constraints. The basis satisfy $\boldsymbol{E}^T\boldsymbol{Q}_a = \boldsymbol{0}$ or transposing $\boldsymbol{Q}_a^T\boldsymbol{E} = \boldsymbol{0}$.
- A matrix $\boldsymbol{Q}_d \in \mathbb{R}^{n\times k}$ that complete $\boldsymbol{Q}_a$ to a basis. In practice $\boldsymbol{Q}_d$ make the matrix $\boldsymbol{Q} = (\boldsymbol{Q}_d, \boldsymbol{Q}_a)$ non singular.

Decomposing the system: $\boldsymbol{E}\boldsymbol{x}' + \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}(t)$ by multiplying to the left with $\boldsymbol{Q}^T$:

- **Differential part**: $\boldsymbol{E}_d\boldsymbol{x}' + \boldsymbol{A}_d\boldsymbol{x} = \boldsymbol{f}_d(t)$ where

$$\boldsymbol{E}_d = \boldsymbol{Q}_d^T\boldsymbol{E}, \quad \boldsymbol{A}_d = \boldsymbol{Q}_d^T\boldsymbol{A}, \quad \boldsymbol{f}_d(t) = \boldsymbol{Q}_d^T\boldsymbol{f}(t)$$

- **Algebraic part**: $\boldsymbol{A}_a\boldsymbol{x} = \boldsymbol{f}_a(t)$ where

$$\boldsymbol{A}_a = \boldsymbol{Q}_a^T\boldsymbol{A}, \qquad \boldsymbol{f}_a(t) = \boldsymbol{Q}_a^T\boldsymbol{f}(t)$$

since $\boldsymbol{Q}_a^T\boldsymbol{E} = \boldsymbol{0}$.

Remark 12.1 The decomposition used to separate differential and algebraic part seem quite cumbersome and difficult at first. However, in practice, it is straightforward to compute using standard tools from linear algebra. For instance, the LU decomposition can be applied to the matrix $\boldsymbol{E}$ to efficiently compute $\boldsymbol{Q}_A$ and $\boldsymbol{Q}_b$. The detail will be discussed forward.

Step 3: Handle the Algebraic Constraints

The algebraic constraints from Step 2 are:

$$\boldsymbol{A}_a\boldsymbol{x} = \boldsymbol{f}_a(t)).$$

Differentiate these constraints:

$$\boldsymbol{A}_a\boldsymbol{x}' = \boldsymbol{f}_a'(t)$$

Now the DAE take the form

$$\begin{pmatrix}\boldsymbol{E}_d\\ \boldsymbol{A}_a\end{pmatrix}\boldsymbol{x}' + \begin{pmatrix}\boldsymbol{A}_d\\ \boldsymbol{0}\end{pmatrix}\boldsymbol{x} = \begin{pmatrix}\boldsymbol{f}_d(t)\\ \boldsymbol{f}_a'(t)\end{pmatrix}$$

or

$$\widetilde{\boldsymbol{E}}\boldsymbol{x}' + \widetilde{\boldsymbol{A}}\boldsymbol{x} = \widetilde{\boldsymbol{f}}(t)$$

where

$$\widetilde{\boldsymbol{E}} = \begin{pmatrix}\boldsymbol{E}_d\\ \boldsymbol{A}_a\end{pmatrix} \quad \widetilde{\boldsymbol{A}} = \begin{pmatrix}\boldsymbol{A}_d\\ \boldsymbol{0}\end{pmatrix} \quad \widetilde{\boldsymbol{f}}(t) = \begin{pmatrix}\boldsymbol{f}_d(t)\\ \boldsymbol{f}_a'(t)\end{pmatrix}$$

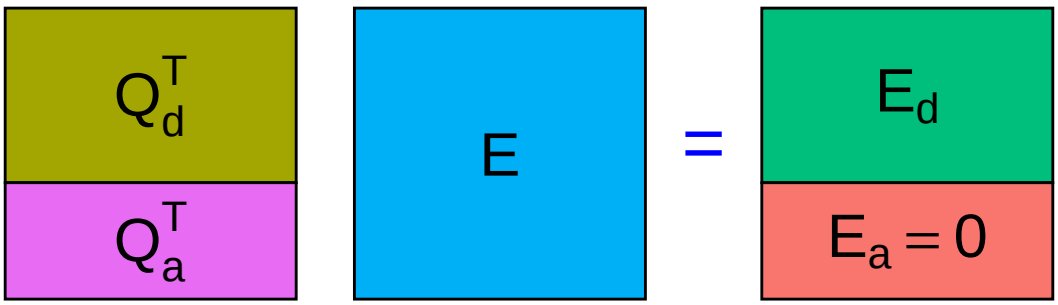


Figure 12.1: The effect of multiply $\boldsymbol{E}$ by $\boldsymbol{Q}^T$

Return to step 1. When termninate the procedure the index is the number of time Step 3 (differentiation) is performed.

12.2 Practical construction of Q_a and Q_d

For the systematic construction of the matrices Q_a and Q_d, we leverage the (generalized) LU decomposition, which provides a structured way to separate the differential and algebraic components of a Differential-Algebraic Equation (DAE).

Theorem 12.1 — see (8), (22). Let $A \in \mathbb{R}^{n\times m}$ of rank $k \leqslant \min(n,m)$, then there exists two permutation matrices P_L and P_R such that

$$P_L A P_R = \begin{pmatrix} L & 0 \\ M & I \end{pmatrix} \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix}$$

with

- $L \in \mathbb{R}^{k\times k}$ lower triangular with 1 on the main diagonal.
- $U \in \mathbb{R}^{k\times k}$ upper triangular with nonzero element on the main diagonal.
- $M \in \mathbb{R}^{n-k\times k}$
- $N \in \mathbb{R}^{k\times m-k}$

Using this theorem for E:

$$P_L E P_R = \begin{pmatrix} L & 0 \\ M & I \end{pmatrix} \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix}$$

and (using the fact that for a permutation matrix $P^{-1} = P^T$)

$$E = P_L^T \begin{pmatrix} L & 0 \\ M & I \end{pmatrix} \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix} P_R^T$$

so that defining

$$Q^T = \begin{pmatrix} L & 0 \\ M & I \end{pmatrix}^{-1} P_L = \begin{pmatrix} L^{-1} & 0 \\ -ML^{-1} & I \end{pmatrix} P_L$$

we have

$$Q^T E = \underbrace{\begin{pmatrix} L^{-1} & 0 \\ -ML^{-1} & I \end{pmatrix} P_L}_{Q^T} \underbrace{P_L^T \begin{pmatrix} L & 0 \\ M & I \end{pmatrix} \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix} P_R^T}_{E}$$

$$= \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix} P_R^T$$

12.2.1 Simplification

we can discard the colum permutation P_R defining

$$\begin{pmatrix} E_d \\ 0 \end{pmatrix} = \begin{pmatrix} U & N \\ 0 & 0 \end{pmatrix} P_R^T$$

so that

$$E = P_L^T \begin{pmatrix} L & 0 \\ M & I \end{pmatrix} \begin{pmatrix} S \\ 0 \end{pmatrix}, \quad Q^T E = \begin{pmatrix} E_d \\ 0 \end{pmatrix},$$

and applying Q^T to the DAE $Ex' + Ax = f(t)$ results in

$$\boldsymbol{Q}^T\left(\boldsymbol{E}\,\boldsymbol{x}' + \boldsymbol{A}\,\boldsymbol{x}\right) = \boldsymbol{Q}^T\boldsymbol{f}(t)$$

$$\begin{pmatrix}\boldsymbol{E}_d\\ \boldsymbol{0}\end{pmatrix}\boldsymbol{x}' + \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{Q}^T\boldsymbol{f}(t)$$

and there is a practical separation of the algebraic equation from the DAE.

Remark 12.2 — Avoiding explicit construction of Q_a and Q_d. The separation of differential and algebraic part is equivalent to the reduction to echelon form (see Chapter 8). This method is a **systematic** approach to solving linear systems, computing ranks, and checking singularity conditions.

12.4 Example 1

12.4.1 Index Reduction of a Linear DAE

Consider the following linear DAE:

$$\begin{aligned} x_1' + x_2 &= 0,\\ x_1 &= 1, \end{aligned}$$

That written in matrix form become:

$$\boldsymbol{E}\boldsymbol{x}' + \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}(t),$$

with the following matrices and vectors:

$$\boldsymbol{E} = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \quad \boldsymbol{f}(t) = \begin{bmatrix}0\\ 1\end{bmatrix}.$$

12.4.2 First reduction

Step 1: Check if E is Singular

We compute:

$$\det(\boldsymbol{E}) = 0$$

Since $\boldsymbol{E}$ is singular, the system is a DAE.

Step 2: Decompose the System

The basis for the null space of $\boldsymbol{E}^T$ is: $\boldsymbol{Q}_a = \begin{bmatrix}0\\ 1\end{bmatrix}$.

The complementary vector is: $\boldsymbol{Q}_d = \begin{bmatrix}1\\ 0\end{bmatrix}$.

We construct the matrix $\boldsymbol{Q} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$.

Multiplying the system by $\boldsymbol{Q}^T$, we get the new matrices:

12.3 Index Reduction Algorithm for linear DAEs

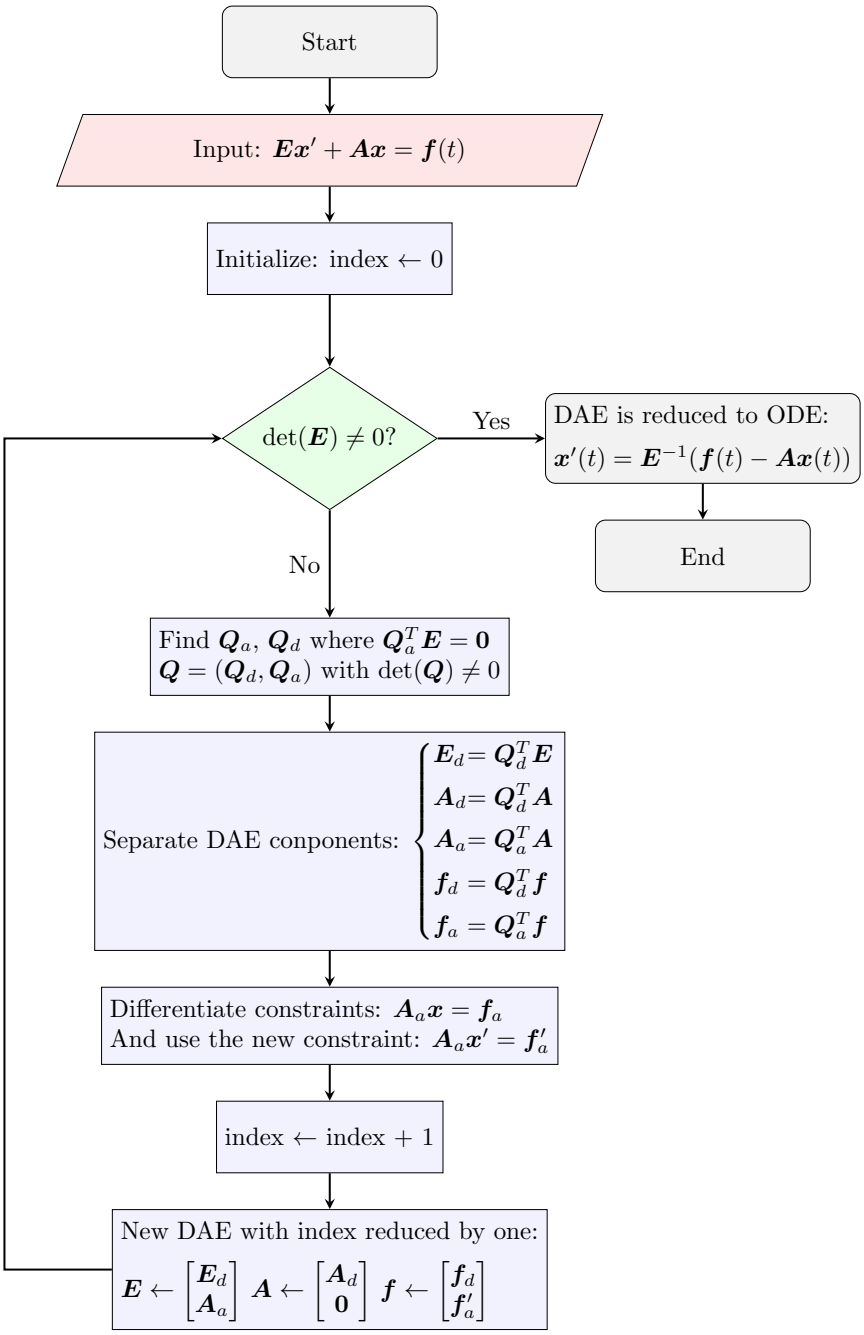


Figure 12.2

$$
\begin{aligned}
\boldsymbol{E}_d = \boldsymbol{Q}_d^T \boldsymbol{E} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\boldsymbol{A}_d = \boldsymbol{Q}_d^T \boldsymbol{A} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\boldsymbol{f}_d(t) = \boldsymbol{Q}_d^T \boldsymbol{f}(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}, \\
\boldsymbol{A}_a = \boldsymbol{Q}_a^T \boldsymbol{A} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\boldsymbol{f}_a(t) = \boldsymbol{Q}_a^T \boldsymbol{f}(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} &&= \begin{bmatrix} 1 \end{bmatrix}.
\end{aligned}
$$

Step 3: Solve the System

The system becomes:

1. **Differential part:** $x_1' + x_2 = 0$.
2. **Algebraic part:** $x_1 = 1$.

Differentiating the algebraic constraint $x_1 = 1$:

$$x_1' = 0.$$

At this point the DAE become:

$$\widetilde{\boldsymbol{E}}\boldsymbol{x}' + \widetilde{\boldsymbol{A}}\boldsymbol{x} = \widetilde{\boldsymbol{f}}(t)$$

with the following matrices and vectors:

$$\widetilde{\boldsymbol{E}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \widetilde{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \widetilde{\boldsymbol{f}}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

12.4.3 Second Reduction

Now reapply the algorithm:

Step 1: Check if $\widetilde{\boldsymbol{E}}$ is Singular

We compute:

$$\det(\widetilde{\boldsymbol{E}}) = 0$$

Since $\widetilde{\boldsymbol{E}}$ is singular, the system is still a DAE.

Step 2: Decompose the System

The basis for the null space of $\widetilde{\boldsymbol{E}}^T$ is: $\widetilde{\boldsymbol{Q}}_a = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

The complementary vector is: $\widetilde{\boldsymbol{Q}}_d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

We construct the matrix $\widetilde{\boldsymbol{Q}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Multiplying the system by $\widetilde{\boldsymbol{Q}}^T$, we get the new matrices:

$$
\begin{aligned}
\widetilde{\boldsymbol{E}}_d = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{E}} &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 2 & 0 \end{bmatrix}, \\
\widetilde{\boldsymbol{A}}_d = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{A}} &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\widetilde{\boldsymbol{f}}_d(t) = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{f}}(t) &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}, \\
\widetilde{\boldsymbol{A}}_a = \widetilde{\boldsymbol{Q}}_a^T \widetilde{\boldsymbol{A}} &= \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\widetilde{\boldsymbol{f}}_a(t) = \widetilde{\boldsymbol{Q}}_a^T \widetilde{\boldsymbol{f}}(t) &= \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}.
\end{aligned}
$$

Step 3: Solve the System

The system becomes:

1. **Differential part:** $2x_1' + x_2 = 0$.
2. **Algebraic part:** $x_2 = 0$.

   Differentiating the algebraic constraint $x_2 = 0$:

$$x_2' = 0.$$

At this point the DAE become:

$$\widetilde{\widetilde{\boldsymbol{E}}}\boldsymbol{x}' + \widetilde{\widetilde{\boldsymbol{A}}}\boldsymbol{x} = \widetilde{\widetilde{\boldsymbol{f}}}(t)$$

with the following matrices and vectors:

$$\widetilde{\widetilde{\boldsymbol{E}}} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \widetilde{\widetilde{\boldsymbol{A}}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \widetilde{\widetilde{\boldsymbol{f}}}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Now reapply the algorithm. At the step 1 $\det(\widetilde{\widetilde{\boldsymbol{E}}}) = 2 \neq 0$ so now we have an ODE:

$$
\begin{aligned}
2x_1' + x_2 &= 0, \\
x_2' &= 0.
\end{aligned}
$$

we have done two derivation, thus the **index of original DAE is 2**.

12.5 Example 2

Consider the DAE

$$
\begin{cases}
x'(t) = z(t) \\
z'(t) = y(t) - t \\
x'(t) = z(t) + x(t) + t + 1
\end{cases}
$$

define

$$\boldsymbol{E} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{A} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & -1 \end{pmatrix}, \qquad \boldsymbol{f}(t) = \begin{pmatrix} 0 \\ -t \\ t+1 \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}\,\boldsymbol{x}'(t) + \boldsymbol{A}\,\boldsymbol{x}(t) = \boldsymbol{f}(t)$$

The table

$$\boldsymbol{T} = \left[\begin{array}{c|c|c} \boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t) \end{array}\right] = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ 1 & 0 & 0 & -1 & 0 & -1 & t+1 \end{array}\right]$$

is used to memorize the operation done on the DAE. Let index $\leftarrow 0$.

1. Multiply table $\boldsymbol{T}$ on the left by $\boldsymbol{L}_1$ where

$$\boldsymbol{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_1 = \boldsymbol{L}_1\boldsymbol{T} = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & 0 & 0 & -1 & 0 & 0 & t+1 \end{array}\right]$$

The last row contain the algebraic equation

$$-x(t) = t+1$$

2. First block now is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 1$.

$$\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

3. Multiply table $\boldsymbol{T}_2$ on the left by $\boldsymbol{L}_2$ where

$$\boldsymbol{L}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_3 = \boldsymbol{L}_2\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right]$$

The last row contain the algebraic equation

$$-z(t) = 1$$

4. First block again is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 2$.

$$\boldsymbol{T}_4 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right]$$

5. Multiply table $\boldsymbol{T}_4$ on the left by $\boldsymbol{L}_4$ where

$$\boldsymbol{L}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_5 = \boldsymbol{L}_4 \boldsymbol{T}_4 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & 0 & 0 & 0 & -1 & 0 & -t \end{array}\right]$$

The last row contain the algebraic equation

$$-y(t) = -t$$

6. First block again is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 3$.

$$\boldsymbol{T}_6 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{array}\right]$$

7. Exchange last two rows and multily by -1 second row

$$\boldsymbol{P}_6 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

multilying $\boldsymbol{T}_6$ by $\boldsymbol{P}_6$:

$$\boldsymbol{P}_6 \boldsymbol{T}_6 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \end{array}\right]$$

- The ODE after index reduction

$$\begin{cases} x'(t) = z(t) \\ y'(t) = 1 \\ z'(t) = y(t) - t \end{cases}$$

- The algebraic (hidden) constraints

$$\begin{array}{l} -x(t) = t + 1 \\ -z(t) = 1 \\ -y(t) = -t \end{array} \implies \begin{cases} x(t) = -t - 1 \\ z(t) = -1 \\ y(t) = t \end{cases}$$

- The **differential index** is 3

Notice that the hiddent constaints determine the complete solution and the initial condition of the ODE $(x(0), y(0), z(0)) = (-1, -1, 0)$.

12.6 Example 3

Consider the DAE

$$\begin{cases} x'(t) = z(t) \\ x'(t) = y(t) - \sin t \\ x'(t) = z(t) + y(t) + \cos t \end{cases}$$

define

$$\boldsymbol{E} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{A} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ 0 & -1 & -1 \end{pmatrix}, \qquad \boldsymbol{f}(t) = \begin{pmatrix} 0 \\ -\sin t \\ \cos t \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}\boldsymbol{x}'(t) + \boldsymbol{A}\boldsymbol{x}(t) = \boldsymbol{f}(t)$$

The table

$$\boldsymbol{T} = \left[\begin{array}{c|c|c} \boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t) \end{array}\right] = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & -\sin t \\ 1 & 0 & 0 & 0 & -1 & -1 & \cos t \end{array}\right]$$

is used to memorize the operation done on the DAE. Let index $\leftarrow 0$.

1. Multiply table $\boldsymbol{T}$ on the left by $\boldsymbol{L}_1$ where

$$\boldsymbol{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_1 = \boldsymbol{L}_1\boldsymbol{T} = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & +1 & -\sin t \\ 0 & 0 & 0 & 0 & -1 & 0 & \cos t \end{array}\right]$$

The last two rows contains the algebraic equations

$$-y(t) + z(t) = -\sin t, \qquad -y(t) = \cos(t)$$

2. First block now is in echelon form. Apply derivation of the last rows (reduce index by one). Let index $\leftarrow 1$.

$$\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ \hline 0 & -1 & +1 & 0 & 0 & 0 & -\cos t \\ 0 & -1 & 0 & 0 & 0 & 0 & -\sin t \end{array}\right]$$

3. Multiply table $\boldsymbol{T}_2$ on the left by $\boldsymbol{L}_2$ where

$$\boldsymbol{L}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_3 = \boldsymbol{L}_2\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & +1 & 0 & 0 & 0 & -\cos t \\ 0 & 0 & -1 & 0 & 0 & 0 & \cos t - \sin t \end{array}\right]$$

- The ODE after index reduction

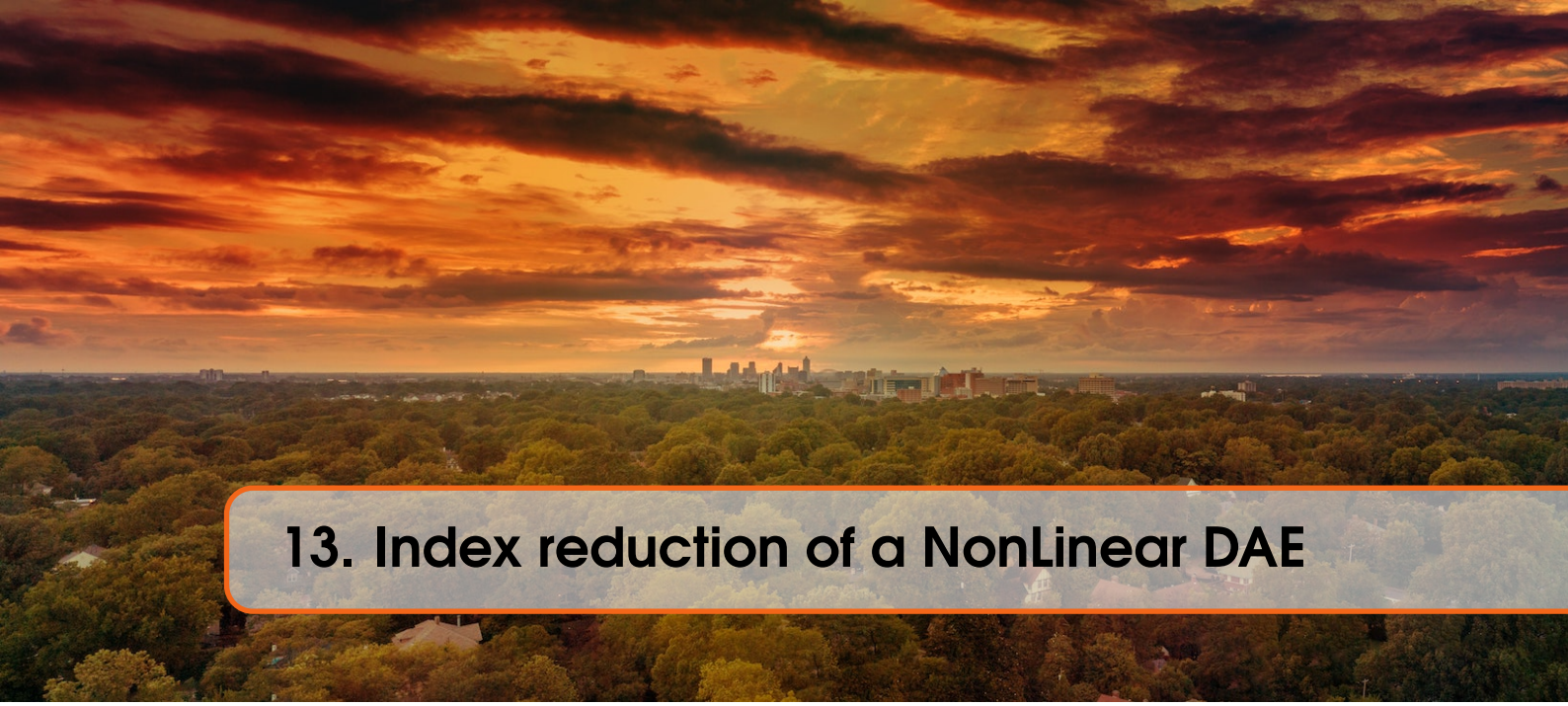
$$\begin{aligned} x'(t) &= z(t) \\ -y'(t) + z'(t) &= -\cos t \\ -z'(t) &= \cos t - \sin t \end{aligned} \quad \Longrightarrow \quad \begin{cases} x'(t) = z(t) \\ y'(t) = \sin t \\ z'(t) = \sin t - \cos t \end{cases}$$

- The algebraic (hidden) constraints

$$\begin{aligned} -y(t) + z(t) &= -\sin t, \\ -y(t) &= \cos t \end{aligned} \quad \Longrightarrow \quad \begin{cases} y(t) = -\cos t \\ z(t) = \cos t - \sin t \end{cases}$$

- The **differential index** is 2

Notice that the hiddent constaints determine **part** of complete solution and the initial condition of the ODE $(x(0), y(0), z(0)) = (\text{free}, -1, 0)$. Thus, only $z(0)$ must be set to determine the solution.



13. Index reduction of a NonLinear DAE

This tutorial explains how to compute the index of a non-linear DAE systematically.

13.1 General DAE

A general **Differential-Algebraic equation (DAE)** takes the form:

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{x}',t)=\boldsymbol{0},$$

where

- $\boldsymbol{x}$ represents the state variables
- $\boldsymbol{x}'$ denotes their derivatives with respect to time t
- $\boldsymbol{f}$ is a system of equations involving both algebraic and differential components.

The defining characteristic of a DAE is that the Jacobian matrix

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{x}',t)}{\partial \boldsymbol{x}'}$$

is **singular**, meaning that the system does not explicitly define all derivatives in terms of the state variables. This is in contrast to **ordinary differential equations (ODEs)**, where the system can be solved explicitly for $\boldsymbol{x}'$.

13.2 Semi-Explicit Form of DAEs

Introducing the auxiliary variable $\boldsymbol{y}=\boldsymbol{x}'$, the DAE can be rewritten as the system:

$$\begin{cases}\boldsymbol{x}'=\boldsymbol{y},\\ \boldsymbol{0}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t).\end{cases}$$

This is a special case of the **semi-explicit DAE** of the form:

$$\begin{cases}\boldsymbol{x}'=\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t),\\ \boldsymbol{0}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t),\end{cases}$$

where $\boldsymbol{g}$ defines an explicit relationship for the differential variables.

13.3 Matrix Representation

The semi-explicit DAE system can be expressed in a compact **matrix form**:

$$\underbrace{\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}}_{\mathscr{E}(z,t)} \underbrace{\begin{pmatrix} x' \\ y' \end{pmatrix}}_{\ddagger(t)} = \underbrace{\begin{pmatrix} g(x,y,t) \\ f(x,y,t) \end{pmatrix}}_{\mathscr{G}(z,t)}.$$

This formulation aligns with the **general form of a DAE system**:

$$\mathscr{E}(z,t)\,z(t) = \mathscr{G}(z,t),$$

where $\mathscr{E}(z,t)$ is a singular matrix, reinforcing the differential-algebraic nature of the system.

13.4 Steps to Compute the Index of a nonlinear DAE

The **differential index** of a differential-algebraic equation (DAE) provides insight into the complexity of solving the system. The following steps outline an approach to determine the index and perform index reduction when necessary.

Step 1: Express the DAE in Matrix Form

A nonlinear DAE can be written as:

$$E(x,t)x' = g(x,t),$$

where:

- $E(x,t)$ is an $n \times n$ matrix,
- $x(t)$ represents the state variables,
- x' is the time derivative of x,
- $g(x,t)$ is a given function.

This formulation allows us to analyze the structure of the system and determine if it contains purely algebraic constraints.

Step 2: Check if $E(x,t)$ is Singular

Compute the determinant:

- If $\det(E(x,t)) \neq 0$, then $E(x,t)$ is invertible, and the system reduces to an **ODE (index 0)**. **Terminate the procedure.**
- If $\det(E(x,t)) = 0$, then $E(x,t)$ is singular, indicating that algebraic constraints are present. **Proceed to the next step.**

Remark 13.1 — Constant Rank Assumption. We assume that $E(x,t)$ has constant rank in the domain of interest (x,t). If the rank varies, numerical methods must adapt accordingly, and the system's index may change dynamically.

Step 3: Decompose the System into Differential and Algebraic Parts

Determine the **rank** of $E(x,t)$, denoted as $r = \operatorname{rank}(E)$. Define:

- A **basis** $Q_a(x,t) \in \mathbb{R}^{n\times(n-k)}$ for the **null space** $\mathcal{N}(E(x,t)^T)$, which identifies the algebraic constraints. This basis satisfies:

$$E(x,t)^T Q_a(x,t) = 0, \quad \text{or equivalently,} \quad Q_a(x,t)^T E(x,t) = 0.$$

- A matrix $Q_d(x,t) \in \mathbb{R}^{n\times k}$ that completes $Q_a(x,t)$ to a full basis, ensuring that:

$$Q(x,t) = \big(Q_d(x,t), Q_a(x,t)\big)$$

is non-singular.

By multiplying the original system by $\boldsymbol{Q}(\boldsymbol{x},t)^T$, we obtain:

- **Differential part:**

$$\boldsymbol{E}_d(\boldsymbol{x},t)\boldsymbol{x}' = \boldsymbol{g}_d(\boldsymbol{x},t),$$

where:

$$\boldsymbol{E}_d(\boldsymbol{x},t) = \boldsymbol{Q}_d(\boldsymbol{x},t)^T\boldsymbol{E}(\boldsymbol{x},t), \quad \boldsymbol{g}_d(\boldsymbol{x},t) = \boldsymbol{Q}_d^T(\boldsymbol{x},t)\boldsymbol{g}(\boldsymbol{x},t).$$

- **Algebraic part:**

$$\boldsymbol{g}_a(\boldsymbol{x},t) = \boldsymbol{0},$$

where:

$$\boldsymbol{g}_a(\boldsymbol{x},t) = \boldsymbol{Q}_a(\boldsymbol{x},t)^T\boldsymbol{g}(\boldsymbol{x},t).$$

Since $\boldsymbol{Q}_a(\boldsymbol{x},t)^T\boldsymbol{E}(\boldsymbol{x},t) = \boldsymbol{0}$, no derivatives appear in this equation.

Step 4: Differentiate the Algebraic Constraints

The algebraic constraints must be differentiated to reveal hidden differential equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}_a(\boldsymbol{x},t) = \boldsymbol{g}_a^t(\boldsymbol{x},t) + \boldsymbol{g}_a^x(\boldsymbol{x},t)\boldsymbol{x}'(t) = \boldsymbol{0}.$$

Rewriting, the system becomes:

$$\begin{pmatrix} \boldsymbol{E}_d(\boldsymbol{x},t) \\ -\boldsymbol{g}_a^x(\boldsymbol{x},t) \end{pmatrix}\boldsymbol{x}' = \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x},t) \\ \boldsymbol{g}_a^t(\boldsymbol{x},t) \end{pmatrix}.$$

Defining:

$$\widetilde{\boldsymbol{E}}(\boldsymbol{x},t) = \begin{pmatrix} \boldsymbol{E}_d(\boldsymbol{x},t) \\ -\boldsymbol{g}_a^x(\boldsymbol{x},t) \end{pmatrix}, \quad \widetilde{\boldsymbol{g}}(\boldsymbol{x},t) = \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x},t) \\ \boldsymbol{g}_a^t(\boldsymbol{x},t) \end{pmatrix},$$

we obtain a new DAE system:

$$\widetilde{\boldsymbol{E}}(\boldsymbol{x},t)\boldsymbol{x}' = \widetilde{\boldsymbol{g}}(\boldsymbol{x},t).$$

Step 5: Repeat Until $\widetilde{\boldsymbol{E}}(\boldsymbol{x},t)$ Becomes Non-Singular

Repeat **Steps 2–4** until $\widetilde{\boldsymbol{E}}(\boldsymbol{x},t)$ is invertible. The **differentiation index** is the number of times Step 4 is performed before reaching an ODE system.

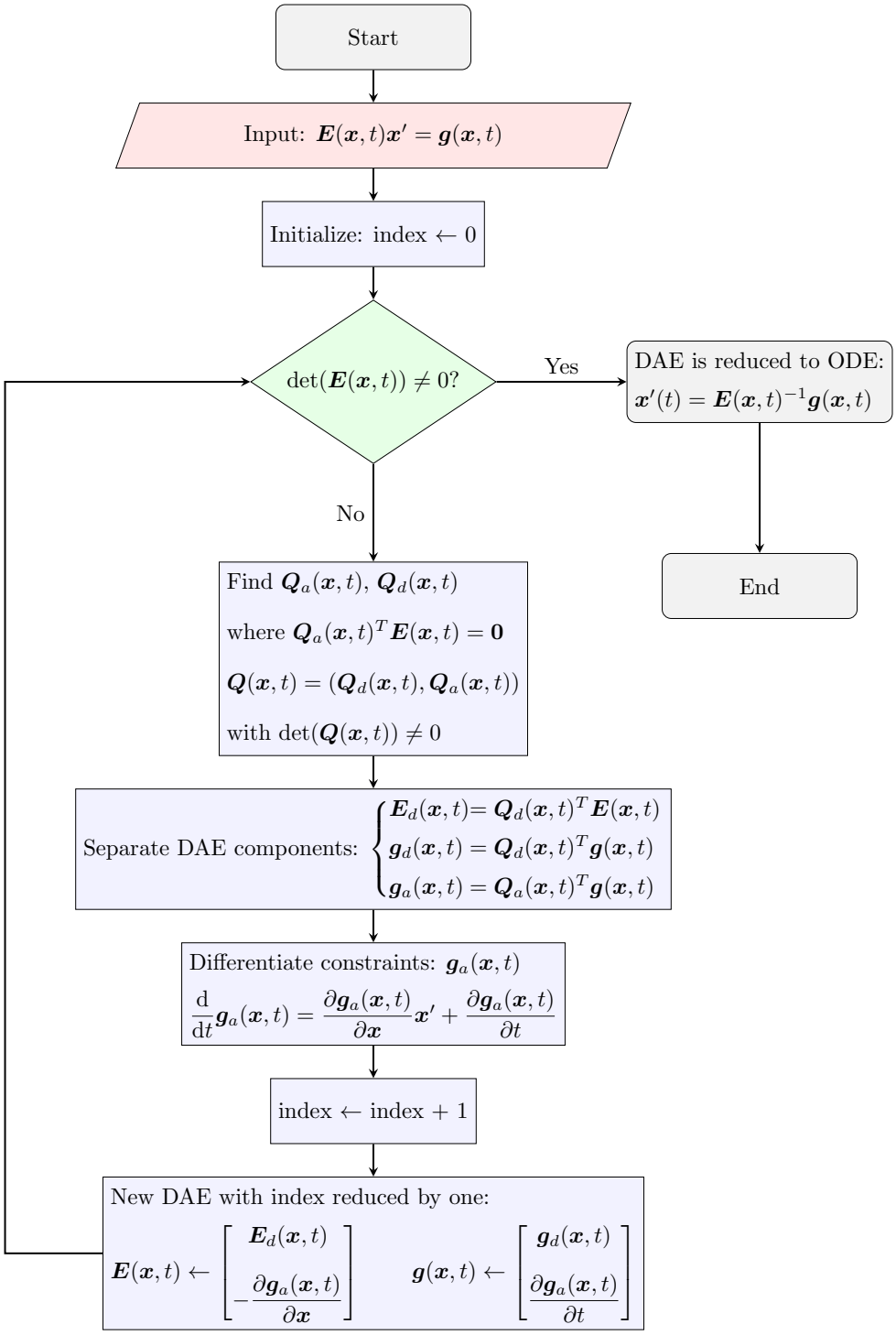
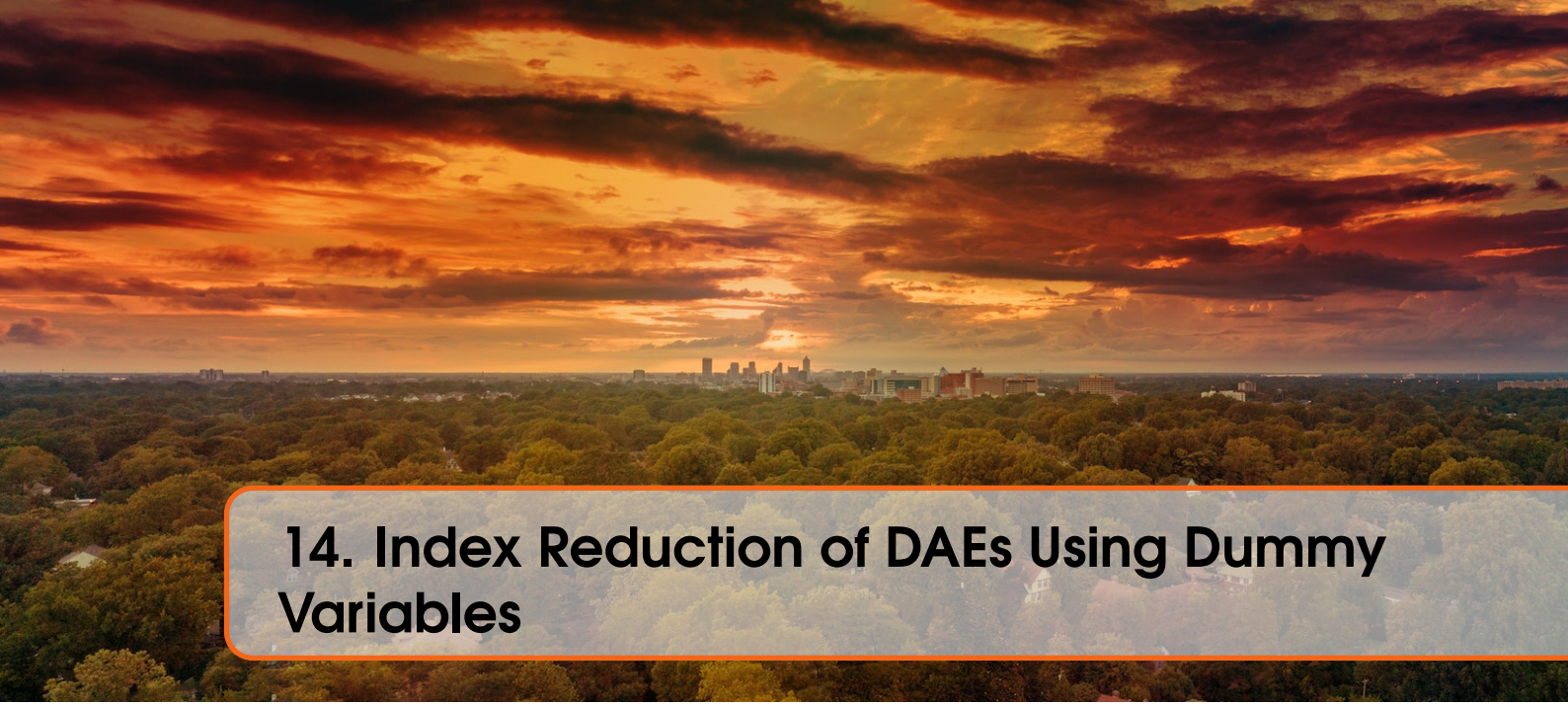


Figure 13.1: Index Reduction Algorithm for Nonlinear DAEs



14. Index Reduction of DAEs Using Dummy Variables

14.1 Introduction

The **dummy variable method** provides a systematic approach for:

- Computing the differential index of nonlinear DAEs
- Transforming high-index problems into solvable forms
- Handling algebraic constraints through variable substitution

14.2 System Requirements

Initial Formulation

Start with a general implicit DAE:

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{x}',t)=\boldsymbol{0},\quad \boldsymbol{x}\in\mathbb{R}^n$$

Required Semi-Explicit Form

Restructure to semi-explicit form (see Chapter 5):

$$\begin{cases}\boldsymbol{x}_d' = \boldsymbol{f}(\boldsymbol{x}_d,\boldsymbol{x}_a,t)\\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x}_d,\boldsymbol{x}_a,t)\end{cases} \tag{14.1}$$

Variables:

- $\boldsymbol{x}_d\in\mathbb{R}^{n_d}$: Differential variables
- $\boldsymbol{x}_a\in\mathbb{R}^{n_a}$: Algebraic variables ($n_d+n_a=n$)

> **Critical Requirement**
>
> The algebraic constraints must be **linear** in $\boldsymbol{x}_a$:
>
> $$\boldsymbol{f}(\boldsymbol{x}_d,\boldsymbol{x}_a,t)\equiv\boldsymbol{f}_0(\boldsymbol{x}_d,t)+\boldsymbol{A}(\boldsymbol{x}_d,t)\boldsymbol{x}_a$$
> $$\boldsymbol{g}(\boldsymbol{x}_d,\boldsymbol{x}_a,t)\equiv\boldsymbol{g}_0(\boldsymbol{x}_d,t)+\boldsymbol{B}(\boldsymbol{x}_d,t)\boldsymbol{x}_a$$

Variable Classification

Partition algebraic variables $\boldsymbol{x}_a$ of Equation 14.1

Type	Notation	Description
Linear	$\boldsymbol{x}_l$	Appears linearly/absent in constraints
Nonlinear	$\boldsymbol{x}_n$	Appears nonlinearly in constraints

14.3 Preprocessing

For $\boldsymbol{x}_n \neq \emptyset$:

1. Introduce dummy derivatives:
$$\boldsymbol{x}_n' = \boldsymbol{v}_n$$
2. Reclassify variables:
   - **New differential**: $(\boldsymbol{x}_d, \boldsymbol{x}_n)$
   - **New algebraic**: $(\boldsymbol{x}_l, \boldsymbol{v}_n)$

Now all the algebraic variables appears linearly. In fact consider the DAE before the preprocessing

$$\begin{cases} \boldsymbol{x}_d' = \boldsymbol{f}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t) + \boldsymbol{A}(\boldsymbol{x}_d, \boldsymbol{x}_n, t)\boldsymbol{x}_l \\ \boldsymbol{0} = \boldsymbol{g}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t) + \boldsymbol{B}(\boldsymbol{x}_d, \boldsymbol{x}_n, t)\boldsymbol{x}_l \end{cases}$$

introducing the dummy variables $\boldsymbol{x}_n' = \boldsymbol{v}_n$ the DAE become

$$\begin{cases} \begin{pmatrix} \boldsymbol{x}_d' \\ \boldsymbol{x}_n' \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t) \\ \boldsymbol{0} \end{pmatrix} + \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}_d, \boldsymbol{x}_n, t) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_l \\ \boldsymbol{v}_n \end{pmatrix} \\ \\ \boldsymbol{0} = \boldsymbol{g}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t) + \boldsymbol{B}(\boldsymbol{x}_d, \boldsymbol{x}_n, t)\boldsymbol{x}_l \end{cases}$$

and defining

- **New differential**: $\widetilde{\boldsymbol{x}}_d = (\boldsymbol{x}_d, \boldsymbol{x}_n)$
- **New algebraic**: $\widetilde{\boldsymbol{x}}_a = (\boldsymbol{x}_l, \boldsymbol{v}_n)$
- $\widetilde{\boldsymbol{f}}_0(\widetilde{\boldsymbol{x}}_d, t) = \begin{pmatrix} \boldsymbol{f}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t) \\ \boldsymbol{0} \end{pmatrix}$
- $\widetilde{\boldsymbol{A}}(\widetilde{\boldsymbol{x}}_d, t) = \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}_d, \boldsymbol{x}_n, t) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}$
- $\widetilde{\boldsymbol{g}}_0(\widetilde{\boldsymbol{x}}_d, t) = \boldsymbol{g}_0(\boldsymbol{x}_d, \boldsymbol{x}_n, t)$
- $\widetilde{\boldsymbol{B}}(\widetilde{\boldsymbol{x}}_d, t) = \begin{pmatrix} \boldsymbol{B}(\boldsymbol{x}_d, \boldsymbol{x}_n, t) & \boldsymbol{0} \end{pmatrix}$

the DAE become

$$\begin{cases} \widetilde{\boldsymbol{x}}_d' = \widetilde{\boldsymbol{f}}_0(\widetilde{\boldsymbol{x}}_d, t) + \widetilde{\boldsymbol{A}}(\widetilde{\boldsymbol{x}}_d, t)\widetilde{\boldsymbol{x}}_a \\ \\ \boldsymbol{0} = \widetilde{\boldsymbol{g}}(\widetilde{\boldsymbol{x}}_d, t) + \widetilde{\boldsymbol{B}}(\widetilde{\boldsymbol{x}}_d, t)\widetilde{\boldsymbol{x}}_a \end{cases}$$

■ **Example 14.1** Consider the following DAE system:

$$\begin{aligned} x' &= u \\ y' &= v \\ 2u' + v' + 2tx\lambda &= t \\ u' + 2v' - \lambda &= 1 - t \\ tx^2 - y^2 &= 1 \end{aligned}$$

Here:

- $\{x, y, u, v\}$ are **differential variables**
- λ is an **algebraic variable** that appears only **linearly**

- $\boldsymbol{x}_n = \emptyset$, $\boldsymbol{x}_l = \{\lambda\}$

This system satisfies the requirements of the dummy variable method.

■

After preprocessing we can assume that DAE can be written as

$$\begin{cases} \boldsymbol{x}_d' = \boldsymbol{f}(\boldsymbol{x}_d,t) + \boldsymbol{A}(\boldsymbol{x}_d,t)\boldsymbol{x}_a \\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x}_d,t) + \boldsymbol{B}(\boldsymbol{x}_d,t)\boldsymbol{x}_a \end{cases}$$

where we have separation of differential and algebraic variables and algebraic variables appears linearly.

14.4 Index reduction algorithm

The index reduction algorithm now work as for the index reduction for nonlinear DAE (see Section 13.4) but working with the matrix $\boldsymbol{B}(\boldsymbol{x}_d,t)$.

Step 1: Check if $\boldsymbol{B}(\boldsymbol{x}_d,t)$ is Singular

The constraints are written as:

$$\boldsymbol{B}(\boldsymbol{x}_d,t)\boldsymbol{x}_a = -\boldsymbol{g}(\boldsymbol{x}_d,t),$$

Compute the determinant:

- If $\det(\boldsymbol{B}(\boldsymbol{x}_d,t)) \neq 0$, then $\boldsymbol{B}(\boldsymbol{x}_d,t)$ is invertible
  - the DAE is of **index 1**: $\boldsymbol{x}_a = -\boldsymbol{B}(\boldsymbol{x}_d,t)^{-1}\boldsymbol{g}(\boldsymbol{x}_d,t)$
  - **Terminate the procedure.**
- If $\det(\boldsymbol{B}(\boldsymbol{x}_d,t)) = 0$
  - $\boldsymbol{B}(\boldsymbol{x}_d,t)$ is singular
  - the DAE is of index > 1.
  - **Proceed to the next step.**

Step 2: Decompose the linear system

Determine the **rank** of $\boldsymbol{B}(\boldsymbol{x}_d,t)$, denoted as $r = \operatorname{rank}(\boldsymbol{B})$ and define:

- A **basis** $\boldsymbol{Q}_a(\boldsymbol{x},t) \in \mathbb{R}^{n_a \times (n_a-k)}$ for the **null space** $\mathcal{N}(\boldsymbol{B}(\boldsymbol{x},t)^T)$, which identifies the algebraic constraints that can be differentiated. This basis satisfies:
  - $\boldsymbol{B}(\boldsymbol{x}_d,t)^T\boldsymbol{Q}_a(\boldsymbol{x}_d,t) = \boldsymbol{0}$ or equivalently
  - $\boldsymbol{Q}_a(\boldsymbol{x}_d,t)^T\boldsymbol{B}(\boldsymbol{x}_d,t) = \boldsymbol{0}$.
- A matrix $\boldsymbol{Q}_d(\boldsymbol{x}_d,t) \in \mathbb{R}^{n_a \times k}$ that completes $\boldsymbol{Q}_a(\boldsymbol{x}_d,t)$ to a full basis, ensuring that:

$$\boldsymbol{Q}(\boldsymbol{x}_d,t) = \big(\boldsymbol{Q}_d(\boldsymbol{x}_d,t), \boldsymbol{Q}_a(\boldsymbol{x}_d,t)\big)$$

  is non-singular.

By multiplying the original constraints by $\boldsymbol{Q}(\boldsymbol{x}_d,t)^T$, we obtain:

- **Part that contains $\boldsymbol{x}_a$:**

$$\boldsymbol{B}_d(\boldsymbol{x}_d,t)\boldsymbol{x}_a = -\boldsymbol{g}_d(\boldsymbol{x}_d,t),$$

  where:

$$\boldsymbol{B}_d(\boldsymbol{x}_d,t) = \boldsymbol{Q}_d(\boldsymbol{x}_d,t)^T\boldsymbol{B}(\boldsymbol{x}_d,t),$$
$$\boldsymbol{g}_d(\boldsymbol{x}_d,t) = \boldsymbol{Q}_d(\boldsymbol{x}_d,t)^T\boldsymbol{g}(\boldsymbol{x}_d,t).$$

- **Part free of $\boldsymbol{x}_a$:**

$$\boldsymbol{g}_a(\boldsymbol{x},t) = \boldsymbol{0},$$

  where:

$$\boldsymbol{g}_a(\boldsymbol{x}_d,t) = \boldsymbol{Q}_a(\boldsymbol{x}_d,t)^T\boldsymbol{g}(\boldsymbol{x}_d,t).$$

  Since $\boldsymbol{Q}_a(\boldsymbol{x}_d,t)^T\boldsymbol{B}(\boldsymbol{x}_d,t) = \boldsymbol{0}$, no algebraic variables appear in this equation.

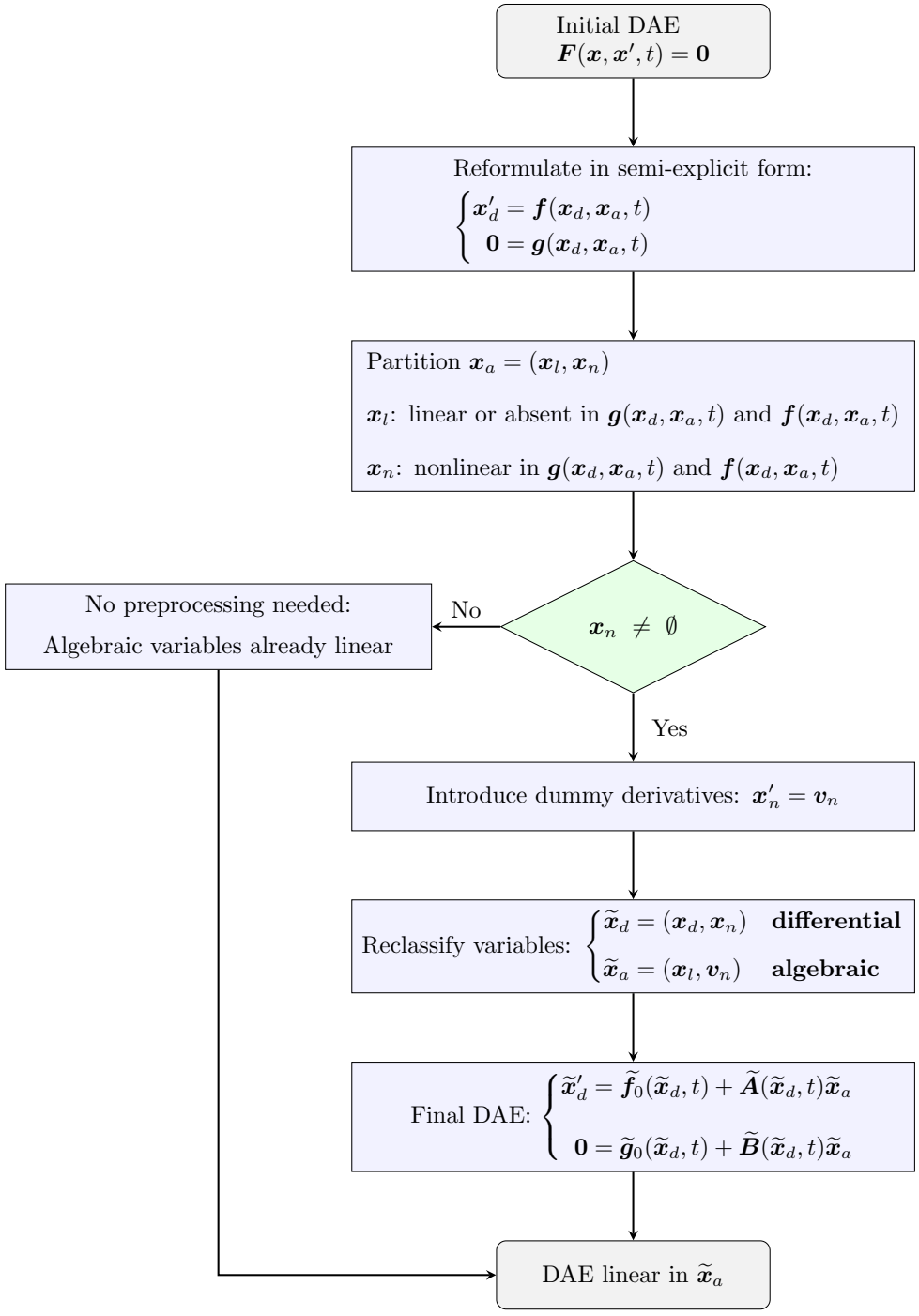


Figure 14.1: Preprocessing of the DAEs

Step 4: Differentiate the Algebraic Constraints

The algebraic constraints must be differentiated to reveal hidden differential equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}_a(\boldsymbol{x}_d,t) = \frac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial t} + \frac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d}\boldsymbol{x}_d'(t) = \boldsymbol{0}.$$

Rewriting, the constraints becomes:

$$\begin{pmatrix} \boldsymbol{B}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d} \end{pmatrix} \boldsymbol{x}_d' = - \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial t} \end{pmatrix}.$$

and using $\boldsymbol{x}_d' = \boldsymbol{f}(\boldsymbol{x}_d,t) + \boldsymbol{A}(\boldsymbol{x}_d,t)\boldsymbol{x}_a$ can be rewritten as

$$\begin{pmatrix} \boldsymbol{B}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d} \end{pmatrix} \boldsymbol{A}(\boldsymbol{x}_d,t)\boldsymbol{x}_a = - \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial t} \end{pmatrix} - \begin{pmatrix} \boldsymbol{B}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d} \end{pmatrix} \boldsymbol{f}(\boldsymbol{x}_d,t)$$

or better

$$\begin{pmatrix} \boldsymbol{B}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d} \end{pmatrix} \boldsymbol{A}(\boldsymbol{x}_d,t)\boldsymbol{x}_a = - \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x}_d,t) + \boldsymbol{B}_d(\boldsymbol{x}_d,t)\boldsymbol{f}(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial t} + \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d}\boldsymbol{f}(\boldsymbol{x}_d,t) \end{pmatrix}$$

Defining:

$$\widetilde{\boldsymbol{B}}(\boldsymbol{x}_d,t) = \begin{pmatrix} \boldsymbol{B}_d(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d} \end{pmatrix} \boldsymbol{A}(\boldsymbol{x}_d,t),$$

$$\widetilde{\boldsymbol{g}}(\boldsymbol{x}_d,t) = \begin{pmatrix} \boldsymbol{g}_d(\boldsymbol{x}_d,t) + \boldsymbol{B}_d(\boldsymbol{x}_d,t)\boldsymbol{f}(\boldsymbol{x}_d,t) \\ \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial t} + \dfrac{\partial \boldsymbol{g}_a(\boldsymbol{x}_d,t)}{\partial \boldsymbol{x}_d}\boldsymbol{f}(\boldsymbol{x}_d,t) \end{pmatrix}$$

we obtain a new constraint part of the DAE:

$$\widetilde{\boldsymbol{B}}(\boldsymbol{x}_d,t)\boldsymbol{x}_a = -\widetilde{\boldsymbol{g}}(\boldsymbol{x}_d,t).$$

notice that the DAE is still in the form with linear algebraic constraints.

Go to step 1.

The **differentiation index** is the number of times Step 4 is performed plus one before reaching an index 1 DAE.

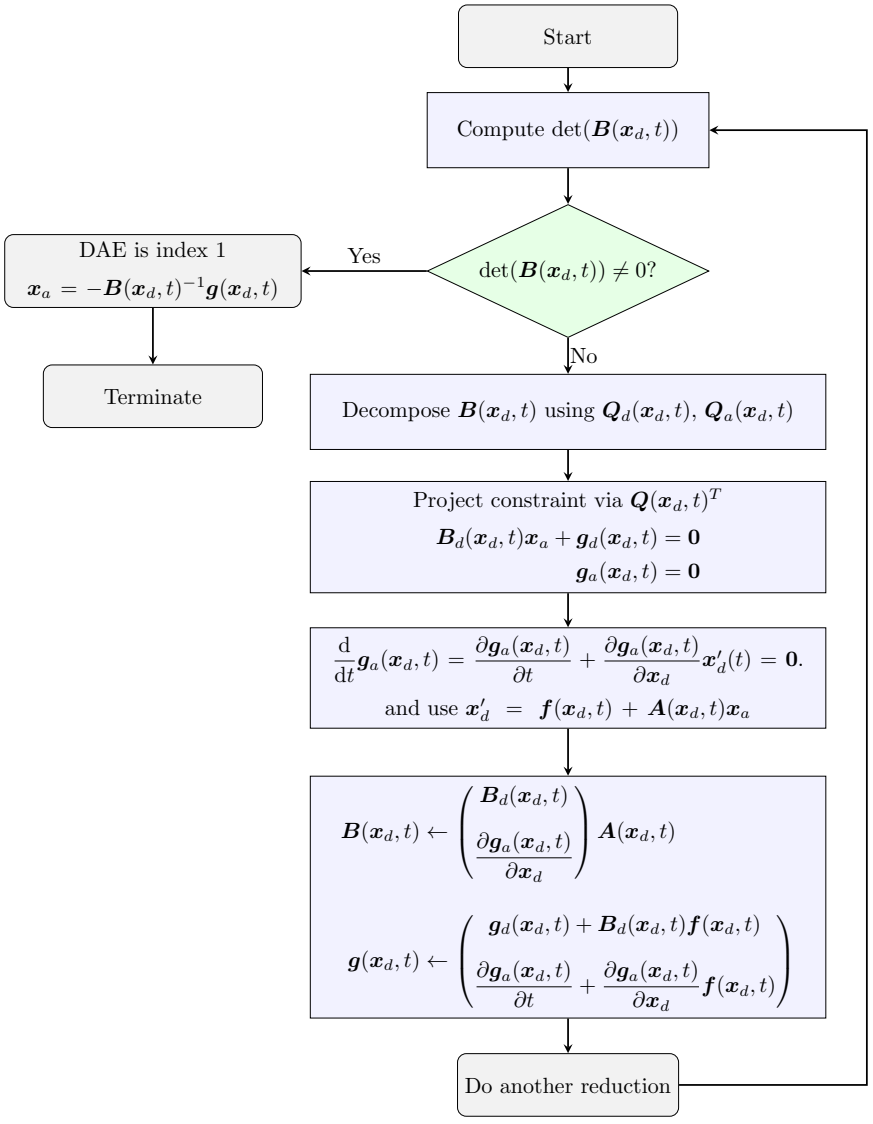
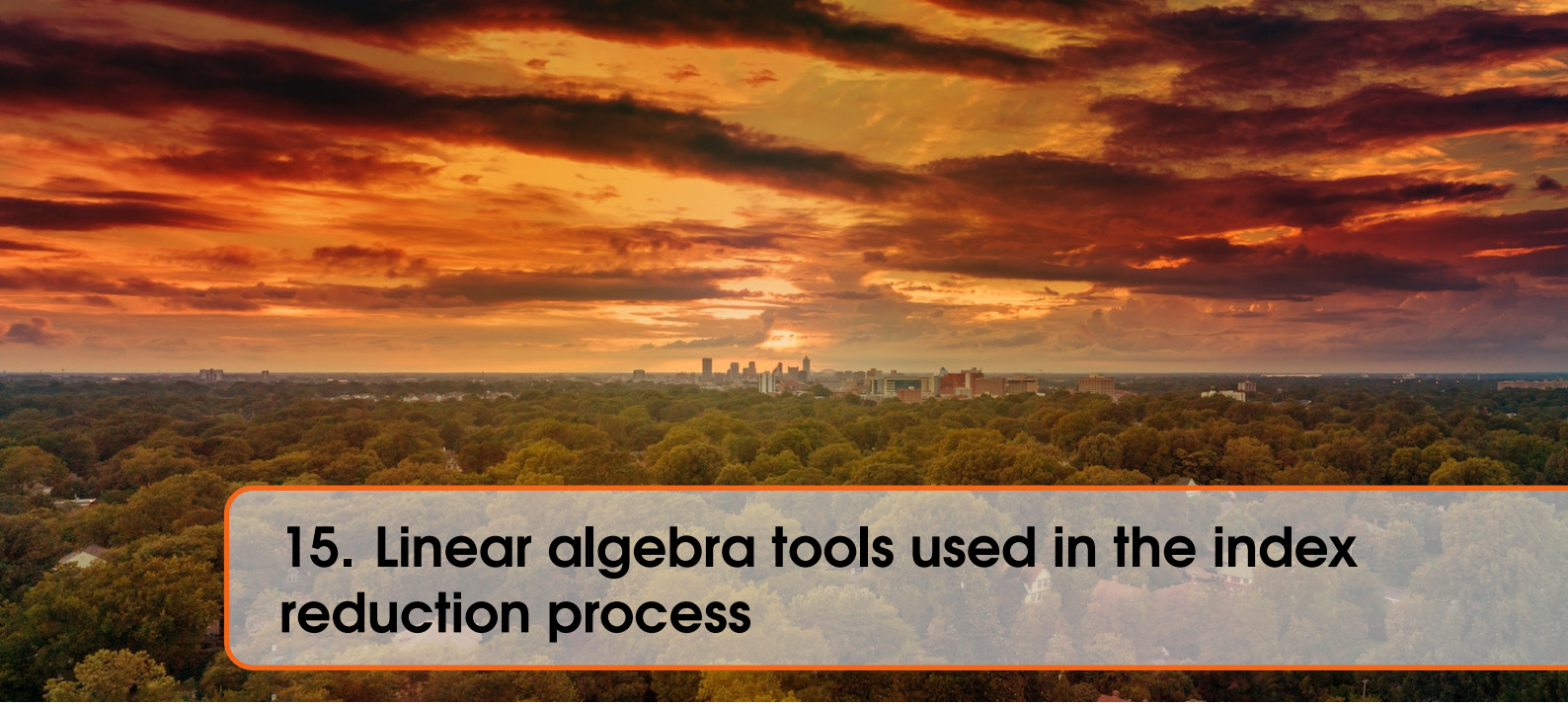


Figure 14.2: Index reduction by dummy variables



15. Linear algebra tools used in the index reduction process

In the index reduction process, to separate the differential and algebraic components, we need to compute the kernel (null space) of $\boldsymbol{E}$, as it represents the algebraic constraints. There are different approaches available to compute it:

- **LU decomposition/factorization**: a decomposition that factors a matrix A as the product of two matrices, $\boldsymbol{A} = \boldsymbol{LU}$, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ is an upper triangular matrix;
- **Singular Value Decomposition (SVD)**: a factorization that factors a matrix as the product of three matrices $\boldsymbol{A} = \boldsymbol{UDV}^T$ where the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal and the matrix $\boldsymbol{D}$ is diagonal with positive real entries, to analyze matrix rank and separate the space into differential and algebraic components.

15.1 Index Reduction through Kernel Computation

When the matrix $\boldsymbol{E}$ is singular, there exists a nonzero vector $\boldsymbol{v} \neq \boldsymbol{0}$ such that

$$\boldsymbol{Ev} = \boldsymbol{0}.$$

All such vectors form a linear subspace called the *kernel* of $\boldsymbol{E}$, denoted as Kernel$(\boldsymbol{E})$. This subspace can be spanned by a set of basis vectors $\{\boldsymbol{v}_i\}$. Similarly, there exists a nonzero vector $\boldsymbol{w} \neq \boldsymbol{0}$ such that

$$\boldsymbol{w}^T\boldsymbol{E} = \boldsymbol{0}^T,$$

which defines the kernel of the transpose of $\boldsymbol{E}$, denoted Kernel$(\boldsymbol{E}^T)$. This subspace is spanned by basis vectors $\{\boldsymbol{w}_i\}$.

We focus on the kernel of the transpose, cokernel,

$$\text{Kernel}(\boldsymbol{E}^T) = \{\boldsymbol{w} \mid \boldsymbol{w}^T\boldsymbol{E} = \boldsymbol{0}\} = \text{Span}\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_p\} = \boldsymbol{K}.$$

Denoting $\boldsymbol{K} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_p)$, it follows that

$$\boldsymbol{K}^T\boldsymbol{E} = \boldsymbol{0}.$$

If we can compute this kernel matrix $\boldsymbol{K}(t, \boldsymbol{y})$ for the time- and state-dependent matrix $\boldsymbol{E}(t, \boldsymbol{y})$, we can identify an invariant condition for the DAE.

Consider the DAE system (**??**), and define

$$\boldsymbol{K}(t,\boldsymbol{y}) = \text{Kernel}(\boldsymbol{E}(t,\boldsymbol{y})^T) = \text{Cokernel}(\boldsymbol{E}).$$

Multiplying the original DAE by $\boldsymbol{K}(t,\boldsymbol{y})^T$, we get:

$$\boldsymbol{K}(t,\boldsymbol{y})^T \boldsymbol{E}(t,\boldsymbol{y})\boldsymbol{y}' = \boldsymbol{K}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}).$$

Since $\boldsymbol{K}^T\boldsymbol{E} = \boldsymbol{0}$, the left side vanishes, yielding

$$\boldsymbol{0} = \boldsymbol{K}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}).$$

This relation defines an invariant that the solution must satisfy.

Next, we extend the kernel matrix $\boldsymbol{K}(t,\boldsymbol{y})$ by finding a complementary matrix $\boldsymbol{N}(t,\boldsymbol{y})$ such that the combined matrix

$$\boldsymbol{M}(t,\boldsymbol{y}) = \begin{bmatrix}\boldsymbol{K}(t,\boldsymbol{y}) & \boldsymbol{N}(t,\boldsymbol{y})\end{bmatrix}$$

is nonsingular (this process is often called the *completion of the kernel*).

Multiplying the original DAE by $\boldsymbol{M}(t,\boldsymbol{y})^T$ separates the system into algebraic and differential parts:

$$\boldsymbol{M}(t,\boldsymbol{y})^T \boldsymbol{E}(t,\boldsymbol{y})\boldsymbol{y}' = \boldsymbol{M}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}),$$

which can be rewritten as

$$\begin{bmatrix}\boldsymbol{K}(t,\boldsymbol{y})^T \\ \boldsymbol{N}(t,\boldsymbol{y})^T\end{bmatrix} \boldsymbol{E}(t,\boldsymbol{y})\boldsymbol{y}' = \begin{bmatrix}\boldsymbol{K}(t,\boldsymbol{y})^T \\ \boldsymbol{N}(t,\boldsymbol{y})^T\end{bmatrix} \boldsymbol{G}(t,\boldsymbol{y}).$$

Manipulating the left side using $\boldsymbol{K}^T\boldsymbol{E} = \boldsymbol{0}$ gives

$$\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{N}(t,\boldsymbol{y})^T \boldsymbol{E}(t,\boldsymbol{y})\end{bmatrix} \boldsymbol{y}' = \begin{bmatrix}\boldsymbol{K}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}) \\ \boldsymbol{N}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y})\end{bmatrix}.$$

Define

$$\tilde{\boldsymbol{G}}_1 = \boldsymbol{N}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}), \quad \tilde{\boldsymbol{G}}_2 = \boldsymbol{K}(t,\boldsymbol{y})^T \boldsymbol{G}(t,\boldsymbol{y}).$$

Differentiating $\tilde{\boldsymbol{G}}_2$ with respect to time,

$$\frac{d}{dt}\tilde{\boldsymbol{G}}_2(t,\boldsymbol{y}) = \frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial t} + \frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial \boldsymbol{y}}\boldsymbol{y}'.$$

The reduced DAE system becomes

$$\begin{bmatrix}\boldsymbol{M}_1(t,\boldsymbol{y}) \\ -\frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial \boldsymbol{y}}\end{bmatrix} \boldsymbol{y}' = \begin{bmatrix}\tilde{\boldsymbol{G}}_1(t,\boldsymbol{y}) \\ \frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial t}\end{bmatrix},$$

where $\boldsymbol{M}_1(t,\boldsymbol{y}) = \boldsymbol{N}(t,\boldsymbol{y})^T \boldsymbol{E}(t,\boldsymbol{y})$.

Thus, the DAE can be expressed as the system

$$\begin{cases}\boldsymbol{E}_2(t,\boldsymbol{y})\boldsymbol{y}' = \boldsymbol{G}_2(t,\boldsymbol{y}), \\ \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y}) = \boldsymbol{0},\end{cases}$$

with

$$\boldsymbol{E}_2(t,\boldsymbol{y}) = \begin{bmatrix}\boldsymbol{M}_1(t,\boldsymbol{y}) \\ -\frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial \boldsymbol{y}}\end{bmatrix}, \quad \boldsymbol{G}_2(t,\boldsymbol{y}) = \begin{bmatrix}\tilde{\boldsymbol{G}}_1(t,\boldsymbol{y}) \\ \frac{\partial \tilde{\boldsymbol{G}}_2(t,\boldsymbol{y})}{\partial t}\end{bmatrix}.$$

This reduction process decreases the index of the DAE by one.
In summary, the algorithm proceeds as follows:

- Compute the kernel of $\boldsymbol{E}^T$.
- Complete the kernel to obtain a nonsingular matrix $\boldsymbol{N}$.
- Separate the algebraic and differential parts.
- Differentiate the algebraic constraints.
- Repeat as needed to reduce the index.

15.1.1 Kernel computation methods

The goal is to find a matrix $\mathbf{K}$ whose columns form a basis of Kernel$(\mathbf{E}^T)$, i.e., they satisfy

$$\mathbf{K}^T\mathbf{E} = \mathbf{0},$$

and such that the augmented matrix $(\mathbf{K},\mathbf{N})$ is nonsingular (invertible), where $\mathbf{N}$ complements $\mathbf{K}$ to form a full-rank basis of the space.

Two classical methods to compute $\mathbf{K}$ are described below.

15.1.1.1 Kernel computation using LU decomposition

The LU decomposition factorizes the matrix $\mathbf{E}$ into a product of a lower triangular matrix $\mathbf{L}$ and an upper trapezoidal matrix $\mathbf{U}$, up to row and column permutations given by permutation matrices $\mathbf{P}$ and $\mathbf{Q}$:

$$\mathbf{PEQ} = \mathbf{LU}.$$

Here, $\mathbf{P}$ and $\mathbf{Q}$ are chosen to ensure numerical stability and reveal the rank structure of $\mathbf{E}$.
Rearranging, the original matrix $\mathbf{E}$ can be expressed as

$$\mathbf{E} = \mathbf{P}^T\mathbf{LUQ}^T.$$

Define

$$\mathbf{M} = \mathbf{UQ}^T,$$

so that

$$\mathbf{E} = \mathbf{P}^T\mathbf{LM}.$$

Because $\mathbf{U}$ is upper trapezoidal, $\mathbf{M}$ has a block structure of the form

$$\mathbf{M} = \begin{bmatrix}\mathbf{M}_1 \\ \mathbf{0}\end{bmatrix},$$

where $\mathbf{M}_1$ is full rank.
Multiplying both sides by $\mathbf{L}^{-1}\mathbf{P}$, we obtain

$$\mathbf{L}^{-1}\mathbf{PE} = \mathbf{M} = \begin{bmatrix}\mathbf{M}_1 \\ \mathbf{0}\end{bmatrix}.$$

Now, by pre-multiplying with the block row selectors $(\mathbf{I},\mathbf{0})$ and $(\mathbf{0},\mathbf{I})$, where $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ the zero matrix, we isolate the nonzero and zero parts respectively:

$$(\mathbf{I},\mathbf{0})\mathbf{L}^{-1}\mathbf{PE} = \mathbf{M}_1,$$

$$(\mathbf{0},\mathbf{I})\mathbf{L}^{-1}\mathbf{PE} = \mathbf{0}.$$

From this, we define the matrices $\mathbf{K}$ and $\mathbf{N}$ as

$$(\mathbf{K},\mathbf{N}) := \mathbf{P}^T\mathbf{L}^{-T}\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix},$$

so that

$$\mathbf{N}^T = (\mathbf{I},\mathbf{0})\mathbf{L}^{-1}\mathbf{P}, \quad \text{and} \quad \mathbf{K}^T = (\mathbf{0},\mathbf{I})\mathbf{L}^{-1}\mathbf{P}.$$

By construction, $\mathbf{K}^T\mathbf{E} = \mathbf{0}$, and $(\mathbf{K},\mathbf{N})$ is nonsingular. Thus, the LU decomposition provides a practical way to compute the *kernel* of $\mathbf{E}$.

15.1.1.2 Kernel computation using Singular Value Decomposition (SVD)

Another standard approach is to use the Singular Value Decomposition (SVD), which factorizes $\mathbf{E}$ into orthogonal matrices $\mathbf{U}$, $\mathbf{V}$, and a diagonal matrix $\blacksquare$:

$$\mathbf{E} = \mathbf{U}\blacksquare\mathbf{V}^T,$$

where

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}, \quad \text{and} \quad \blacksquare = \operatorname{diag}(\sigma_1,\ldots,\sigma_r,0,\ldots,0).$$

Partitioning the matrices according to the rank r of $\mathbf{E}$, we write

$$\mathbf{E} = \left[\begin{array}{c|c}\mathbf{U}_{11} & \mathbf{U}_{12}\\ \hline \mathbf{U}_{21} & \mathbf{U}_{22}\end{array}\right]\left[\begin{array}{c|c}\blacksquare_r & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{0}\end{array}\right]\left[\begin{array}{c|c}\mathbf{V}_{11}^T & \mathbf{V}_{12}^T\\ \hline \mathbf{V}_{21}^T & \mathbf{V}_{22}^T\end{array}\right].$$

Multiplying both sides by $\mathbf{U}^T$ and then selecting the lower block with $(\mathbf{0},\mathbf{I})$, we get

$$(\mathbf{0},\mathbf{I})\mathbf{U}^T\mathbf{E} = (\mathbf{0},\mathbf{I})\left[\begin{array}{c|c}\blacksquare_r & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{0}\end{array}\right]\left[\begin{array}{c|c}\mathbf{V}_{11}^T & \mathbf{V}_{12}^T\\ \hline \mathbf{V}_{21}^T & \mathbf{V}_{22}^T\end{array}\right] = \mathbf{0}.$$

This directly shows that the columns of $(\mathbf{0},\mathbf{I})\mathbf{U}^T$ form a basis of the *kernel* of $\mathbf{E}$.
Defining

$$(\mathbf{K},\mathbf{N}) = \mathbf{U}\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix},$$

we have

$$\mathbf{N}^T = (\mathbf{I},\mathbf{0})\mathbf{U}^T, \quad \text{and} \quad \mathbf{K}^T = (\mathbf{0},\mathbf{I})\mathbf{U}^T,$$

and thus $\mathbf{K}^T\mathbf{E} = \mathbf{0}$.

While the SVD method is robust and numerically stable, it is computationally more expensive than the LU approach and therefore less suited for large-scale or real-time applications. Also for symbolic computations, it is preferable to use LU decomposition.



16. Examples of index reduction

16.1 Example 1

Consider the following linear DAE:

$$\begin{aligned} x_1' + x_2 &= 0, \\ x_1 &= 1, \end{aligned}$$

That written in matrix form become:

$$\boldsymbol{E}\boldsymbol{x}' + \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}(t),$$

with the following matrices and vectors:

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{f}(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

16.1.1 First reduction

Step 1: Check if $\boldsymbol{E}$ is singular

We compute:

$$\det(\boldsymbol{E}) = 0$$

Since $\boldsymbol{E}$ is singular, the system is a DAE.

Step 2: Decompose the system

The basis for the null space of $\boldsymbol{E}^T$ is: $\boldsymbol{Q}_a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

The complementary vector is: $\boldsymbol{Q}_d = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

We construct the matrix $\boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Multiplying the system by $\boldsymbol{Q}^T$, we get the new matrices:

$$
\begin{aligned}
\boldsymbol{E}_d = \boldsymbol{Q}_d^T \boldsymbol{E} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\boldsymbol{A}_d = \boldsymbol{Q}_d^T \boldsymbol{A} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\boldsymbol{f}_d(t) = \boldsymbol{Q}_d^T \boldsymbol{f}(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}, \\
\boldsymbol{A}_a = \boldsymbol{Q}_a^T \boldsymbol{A} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\
\boldsymbol{f}_a(t) = \boldsymbol{Q}_a^T \boldsymbol{f}(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} &&= \begin{bmatrix} 1 \end{bmatrix}.
\end{aligned}
$$

Step 3: Solve the system

The system becomes:

1. **Differential part:** $x_1' + x_2 = 0$.
2. **Algebraic part:** $x_1 = 1$.

   Differentiating the algebraic constraint $x_1 = 1$:

$$x_1' = 0.$$

   At this point the DAE become:

$$\widetilde{\boldsymbol{E}}\boldsymbol{x}' + \widetilde{\boldsymbol{A}}\boldsymbol{x} = \widetilde{\boldsymbol{f}}(t)$$

   with the following matrices and vectors:

$$\widetilde{\boldsymbol{E}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \widetilde{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \widetilde{\boldsymbol{f}}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

16.1.2 Second Reduction

Now reapply the algorithm:

Step 1: Check if $\widetilde{\boldsymbol{E}}$ is singular

We compute:

$$\det(\widetilde{\boldsymbol{E}}) = 0$$

Since $\widetilde{\boldsymbol{E}}$ is singular, the system is still a DAE.

Step 2: Decompose the system

The basis for the null space of $\widetilde{\boldsymbol{E}}^T$ is: $\widetilde{\boldsymbol{Q}}_a = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

The complementary vector is: $\widetilde{\boldsymbol{Q}}_d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

We construct the matrix $\widetilde{\boldsymbol{Q}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Multiplying the system by $\widetilde{\boldsymbol{Q}}^T$, we get the new matrices:

$$
\begin{aligned}
\widetilde{\boldsymbol{E}}_d = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{E}} &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} &&= \begin{bmatrix} 2 & 0 \end{bmatrix}, \\
\widetilde{\boldsymbol{A}}_d = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{A}} &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\widetilde{\boldsymbol{f}}_d(t) = \widetilde{\boldsymbol{Q}}_d^T \widetilde{\boldsymbol{f}}(t) &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}, \\
\widetilde{\boldsymbol{A}}_a = \widetilde{\boldsymbol{Q}}_a^T \widetilde{\boldsymbol{A}} &= \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} &&= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
\widetilde{\boldsymbol{f}}_a(t) = \widetilde{\boldsymbol{Q}}_a^T \widetilde{\boldsymbol{f}}(t) &= \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} &&= \begin{bmatrix} 0 \end{bmatrix}.
\end{aligned}
$$

Step 3: Solve the system

The system becomes:

1. **Differential part:** $2x_1' + x_2 = 0$.
2. **Algebraic part:** $x_2 = 0$.

Differentiating the algebraic constraint $x_2 = 0$:

$$x_2' = 0.$$

At this point the DAE become:

$$\widetilde{\widetilde{\boldsymbol{E}}}\boldsymbol{x}' + \widetilde{\widetilde{\boldsymbol{A}}}\boldsymbol{x} = \widetilde{\widetilde{\boldsymbol{f}}}(t)$$

with the following matrices and vectors:

$$\widetilde{\widetilde{\boldsymbol{E}}} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \widetilde{\widetilde{\boldsymbol{A}}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \widetilde{\widetilde{\boldsymbol{f}}}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Now reapply the algorithm. At the step 1 $\det(\widetilde{\widetilde{\boldsymbol{E}}}) = 2 \neq 0$ so now we have an ODE:

$$
\begin{aligned}
2x_1' + x_2 &= 0, \\
x_2' &= 0.
\end{aligned}
$$

we have done two derivation, thus the **index of original DAE is 2**.

Avoiding explicit construction of $\boldsymbol{Q}_a$ and $\boldsymbol{Q}_d$

The separation of differential and algebraic part is equivalent to the reduction to echelon form (see Section 8.2).

16.2 Example 2

Consider the DAE

$$
\begin{cases}
x'(t) = z(t) \\
z'(t) = y(t) - t \\
x'(t) = z(t) + x(t) + t + 1
\end{cases}
$$

define

$$\boldsymbol{E} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & -1 \end{pmatrix}, \quad \boldsymbol{f}(t) = \begin{pmatrix} 0 \\ -t \\ t+1 \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}\,\boldsymbol{x}'(t)+\boldsymbol{A}\,\boldsymbol{x}(t)=\boldsymbol{f}(t)$$

The table

$$\boldsymbol{T}=\left[\begin{array}{c|c|c}\boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t)\end{array}\right]=\left[\begin{array}{ccc|ccc|c}1&0&0&0&0&-1&0\\0&0&1&0&-1&0&-t\\1&0&0&-1&0&-1&t+1\end{array}\right]$$

is used to memorize the operation done on the DAE. Let index $\leftarrow 0$.

1. Multiply table $\boldsymbol{T}$ on the left by $\boldsymbol{L}_1$ where

$$\boldsymbol{L}_1=\begin{pmatrix}1&0&0\\0&1&0\\-1&0&1\end{pmatrix}$$

so that

$$\boldsymbol{T}_1=\boldsymbol{L}_1\boldsymbol{T}=\left[\begin{array}{ccc|ccc|c}1&0&0&0&0&-1&0\\0&0&1&0&-1&0&-t\\\hline 0&0&0&-1&0&0&t+1\end{array}\right]$$

The last row contain the algebraic equation

$$-x(t)=t+1$$

2. First block now is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 1$.

$$\boldsymbol{T}_2=\left[\begin{array}{ccc|ccc|c}1&0&0&0&0&-1&0\\0&0&1&0&-1&0&-t\\\hline -1&0&0&0&0&0&1\end{array}\right]$$

3. Multiply table $\boldsymbol{T}_2$ on the left by $\boldsymbol{L}_2$ where

$$\boldsymbol{L}_2=\begin{pmatrix}1&0&0\\0&1&0\\1&0&1\end{pmatrix}$$

so that

$$\boldsymbol{T}_3=\boldsymbol{L}_2\boldsymbol{T}_2=\left[\begin{array}{ccc|ccc|c}1&0&0&0&0&-1&0\\0&0&1&0&-1&0&-t\\\hline 0&0&0&0&0&-1&1\end{array}\right]$$

The last row contain the algebraic equation

$$-z(t)=1$$

4. First block again is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 2$.

$$\boldsymbol{T}_4=\left[\begin{array}{ccc|ccc|c}1&0&0&0&0&-1&0\\0&0&1&0&-1&0&-t\\\hline 0&0&-1&0&0&0&0\end{array}\right]$$

5. Multiply table $\boldsymbol{T}_4$ on the left by $\boldsymbol{L}_4$ where

$$\boldsymbol{L}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_5 = \boldsymbol{L}_4 \boldsymbol{T}_4 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & 0 & 0 & 0 & -1 & 0 & -t \end{array}\right]$$

The last row contain the algebraic equation

$$-y(t) = -t$$

6. First block again is in echelon form. Apply derivation of the last row (reduce index by one). Let index $\leftarrow 3$.

$$\boldsymbol{T}_6 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \\ \hline 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{array}\right]$$

7. Exchange last two rows and multily by -1 second row

$$\boldsymbol{P}_6 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

multilying $\boldsymbol{T}_6$ by $\boldsymbol{P}_6$:

$$\boldsymbol{P}_6 \boldsymbol{T}_6 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & -t \end{array}\right]$$

- The ODE after index reduction

$$\begin{cases} x'(t) = z(t) \\ y'(t) = 1 \\ z'(t) = y(t) - t \end{cases}$$

- The algebraic (hidden) constraints

$$\begin{array}{l} -x(t) = t + 1 \\ -z(t) = 1 \\ -y(t) = -t \end{array} \implies \begin{cases} x(t) = -t - 1 \\ z(t) = -1 \\ y(t) = t \end{cases}$$

- The **differential index** is 3

Notice that the hiddent constaints determine the complete solution and the initial condition of the ODE $(x(0), y(0), z(0)) = (-1, -1, 0)$.

16.3 Example 3

Consider the DAE

$$\begin{cases} x'(t) = z(t) \\ x'(t) = y(t) - \sin t \\ x'(t) = z(t) + y(t) + \cos t \end{cases}$$

define

$$\boldsymbol{E} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ 0 & -1 & -1 \end{pmatrix}, \quad \boldsymbol{f}(t) = \begin{pmatrix} 0 \\ -\sin t \\ \cos t \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}\,\boldsymbol{x}'(t) + \boldsymbol{A}\,\boldsymbol{x}(t) = \boldsymbol{f}(t)$$

The table

$$\boldsymbol{T} = \left[\begin{array}{c|c|c} \boldsymbol{E} & \boldsymbol{A} & \boldsymbol{f}(t) \end{array}\right] = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & -\sin t \\ 1 & 0 & 0 & 0 & -1 & -1 & \cos t \end{array}\right]$$

is used to memorize the operation done on the DAE. Let index $\leftarrow 0$.

1. Multiply table $\boldsymbol{T}$ on the left by $\boldsymbol{L}_1$ where

$$\boldsymbol{L}_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_1 = \boldsymbol{L}_1\boldsymbol{T} = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & +1 & -\sin t \\ 0 & 0 & 0 & 0 & -1 & 0 & \cos t \end{array}\right]$$

The last two rows contains the algebraic equations

$$-y(t) + z(t) = -\sin t, \qquad -y(t) = \cos(t)$$

2. First block now is in echelon form. Apply derivation of the last rows (reduce index by one). Let index $\leftarrow 1$.

$$\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ \hline 0 & -1 & +1 & 0 & 0 & 0 & -\cos t \\ 0 & -1 & 0 & 0 & 0 & 0 & -\sin t \end{array}\right]$$

3. Multiply table $\boldsymbol{T}_2$ on the left by $\boldsymbol{L}_2$ where

$$\boldsymbol{L}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_3 = \boldsymbol{L}_2\boldsymbol{T}_2 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & +1 & 0 & 0 & 0 & -\cos t \\ 0 & 0 & -1 & 0 & 0 & 0 & \cos t - \sin t \end{array}\right]$$

- The ODE after index reduction

$$\begin{aligned} x'(t) &= z(t) \\ -y'(t) + z'(t) &= -\cos t \\ -z'(t) &= \cos t - \sin t \end{aligned} \quad \Longrightarrow \quad \begin{cases} x'(t) = z(t) \\ y'(t) = \sin t \\ z'(t) = \sin t - \cos t \end{cases}$$

- The algebraic (hidden) constraints

$$\begin{aligned} -y(t) + z(t) &= -\sin t, \\ -y(t) &= \cos t \end{aligned} \quad \Longrightarrow \quad \begin{cases} y(t) = -\cos t \\ z(t) = \cos t - \sin t \end{cases}$$

- The **differential index** is 2

Notice that the hiddent constaints determine **part** of complete solution and the initial condition of the ODE $(x(0), y(0), z(0)) = (\text{free}, -1, 0)$. Thus, only $z(0)$ must be set to determine the solution.

VI Numerical Method for DAE

17 Numerical methods for linear DAE . . . 157
17.1 Consistent initial values . . . 157
17.2 Explicit Euler . . . 158
17.3 Implicit Euler . . . 158
17.4 Numerical result . . . 158
17.5 Damped stabilization (Baumgarte) . . . 160
17.6 Numerical result . . . 161

18 Baumgarte's stabilization . . . 163
18.1 Differential-Algebraic Equations (DAEs) in Mechanics . . . 163
18.2 Numerical Solution of DAEs . . . 164
18.3 Example: Pendulum in Redundant Coordinates 165
18.4 Numerical Solution . . . 166

19 Baumgarte's stabilization . . . 169
19.1 Pendulum DAE reduced to first order . . . 174
19.2 Integration of the DAE without stabilization . . . 176
19.3 Baumgarte stabilization . . . 181

20 Projection method . . . 183
20.1 Projection Methods in general . . . 184
20.2 Mathematical Formulation . . . 185
20.3 Numerical Integration with Projection . . . 186
20.4 Generic Minimization Problem with Constraints 186

21 Projection Algorithm for index 1,2 and 3 189
21.1 Index-1 DAE Projection Algorithm . . . 189
21.2 Index 2 DAE projection algorithm . . . 191
21.3 Index 3 DAE projection algorithm . . . 195

22 Projection with implicit numerical scheme 199
22.1 Index-1 DAE . . . 199
22.2 Index-3 DAE Special Case . . . 200

23 Coordinate Partitioning for Pendulum in cartesian coordinates . . . 203
23.1 Step 1 choose independent coordinates . . . 203
23.2 Step 2a propagate constraints . . . 203
23.3 Step 2b propagate constraints . . . 204
23.4 Step 2c propagate constraints . . . 204
23.5 Step 2d propagate constraints . . . 204
23.6 Collatz . . . 204
23.7 Runge-Kutta 4th order . . . 205

This chapter presents a selection of numerical methods for the simulation of differential-algebraic equations (DAEs). The focus is not on developing a complete solver, but rather on exploring basic strategies to handle the numerical integration of DAE systems.

In particular, we discuss how to check for constraint drift—i.e., the gradual violation of algebraic conditions during numerical integration—and introduce simple techniques for stabilization and projection to maintain consistency. These methods aim to preserve the essential structure of the system while keeping the implementation accessible and practical.

Learning objectives

After this part, you should be able to measure constraint drift, explain the trade-off introduced by Baumgarte stabilization, and choose between projection and stabilization for the class of DAE under consideration.



17. Numerical methods for linear DAE

Consider the DAE

$$\begin{aligned} x(t) &= \sin(t) \\ x'(t) + y(t) &= 0 \end{aligned}$$

with exact solution

$$\begin{aligned} x(t) &= \sin(t) \\ y(t) &= -\cos(t) \end{aligned}$$

Is a linear DAE $\boldsymbol{F}\boldsymbol{x}'(t) + \boldsymbol{G}\boldsymbol{x}(t) = \boldsymbol{h}(t)$ where

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sin(t) \\ 0 \end{pmatrix}$$

Using Kronecker/Weierstrass normal form with

$$\boldsymbol{U} = \boldsymbol{V} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{N} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

the DAE is written as

$$\boldsymbol{N} \begin{pmatrix} y' \\ x' \end{pmatrix} + \boldsymbol{I} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ \sin(t) \end{pmatrix}$$

because $\boldsymbol{N} \neq \boldsymbol{0}$ and $\boldsymbol{N}^2 = \boldsymbol{0}$ the DAE is of index 2. No initial conditions are required because block $\boldsymbol{C}$ is absent.

17.1 Consistent initial values

Reconsider

$$\begin{aligned} x(t) &= \sin(t) \\ x'(t) + y(t) &= 0 \end{aligned}$$

I initial values x_0 and y_0 are consistent if

$$x(t) = \sin(t)$$

the hidden constraint is obtained by deriving the algebraic part

$$-y(t) = x'(t) = \cos(t)$$

and thus consistent initial condition must satisfy:

$$\begin{aligned} x_0 &= \sin(0) \\ y_0 &= -\cos(0) \end{aligned}$$

17.2 Explicit Euler

Consider the ODE $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},t)$, explicit Euler take the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h\boldsymbol{f}(\boldsymbol{x}_k, t_k)$$

applied to the DAE become ($t_k = kh$)

$$\begin{aligned} 0 &= h(\sin(t_k) - x_k) \\ x_{k+1} &= x_k - hy_k \end{aligned}$$

Thus, the method cant be applied (y_{k+1} do not appear in the equations).

17.3 Implicit Euler

For $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},t)$, implicit Euler take the form

$$\boldsymbol{x}_{k+1} - h\boldsymbol{f}(\boldsymbol{x}_{k+1}, t_{k+1}) = \boldsymbol{x}_k$$

Applied to the DAE result in ($t_k = k\,h$)

$$\begin{aligned} h(\sin(t_{k+1}) - x_{k+1}) &= 0 \\ x_{k+1} + hy_{k+1} &= x_k \end{aligned}$$

so that

$$\begin{aligned} x_{k+1} &= \sin(t_{k+1}) \\ y_{k+1} &= (x_k - \sin(t_{k+1})/h \\ &= (\sin(t_k) - \sin(t_{k+1}))/h = -\cos(t_{k+1}) + \mathcal{O}(h) \end{aligned}$$

17.4 Numerical result

17.4.1 Crank-Nicolson

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{h} = \frac{\boldsymbol{f}(\boldsymbol{x}_{k+1}, t_{k+1}) + \boldsymbol{f}(\boldsymbol{x}_k, t_k)}{2}$$

Applied to the DAE result in

$$\begin{aligned} \sin(t_{k+1}) - x_{k+1} &= x_k - \sin(t_k) \\ x_{k+1} - \frac{h}{2}y_{k+1} &= x_k + \frac{h}{2}y_k \end{aligned}$$

so that

$$\begin{aligned} x_{k+1} &= \sin(t_{k+1}) + \sin(t_k) - x_k \\ y_{k+1} &= \frac{2}{h}(x_k - x_{k+1}) - y_k \end{aligned}$$

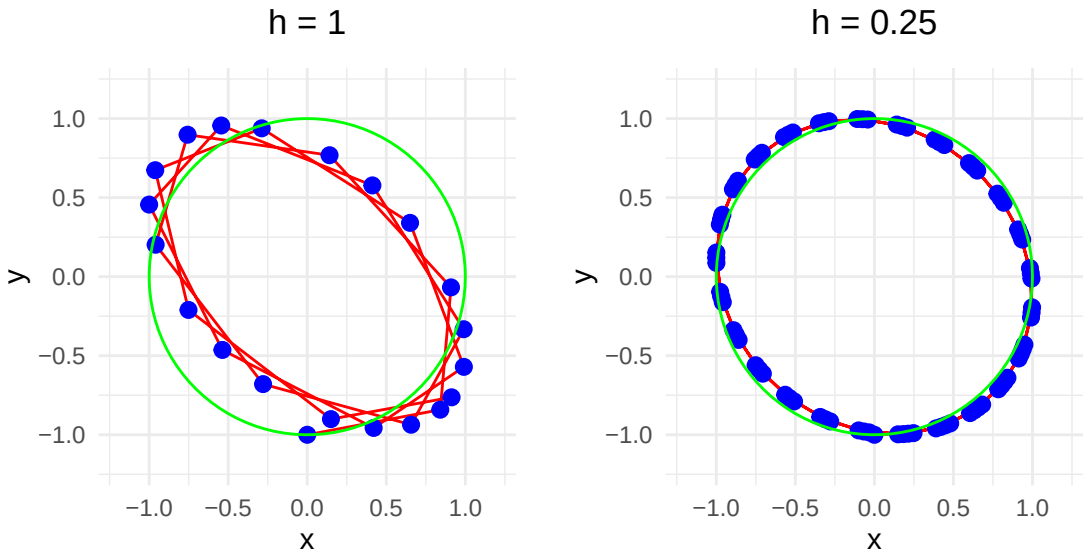


Figure 17.1: Implicit Euler. Consistent initial data.

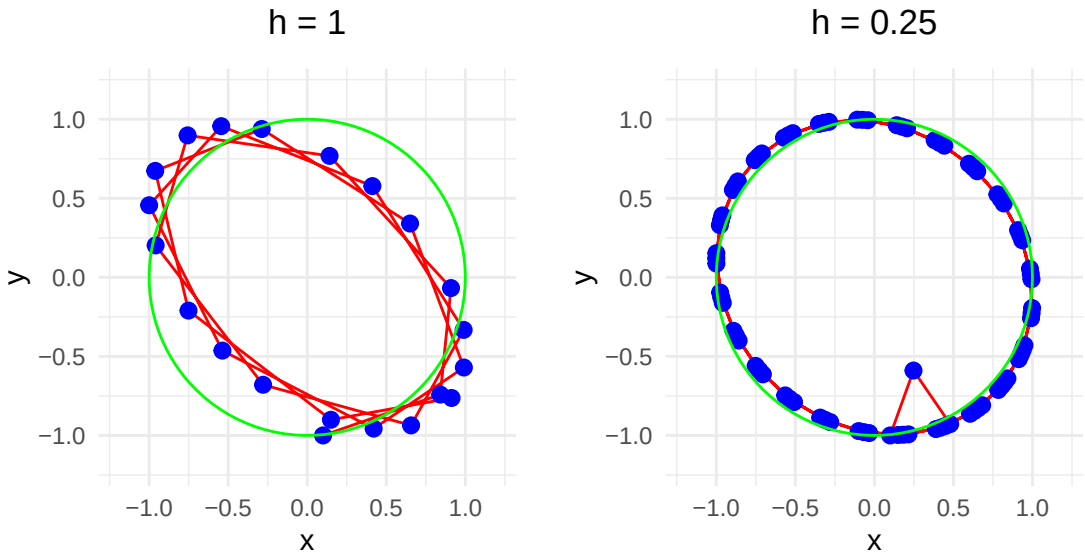


Figure 17.2: Implicit Euler. Inconsistent initial data.

solving for x_k

$$x_k = \sin(t_k) + (-1)^k x_0$$

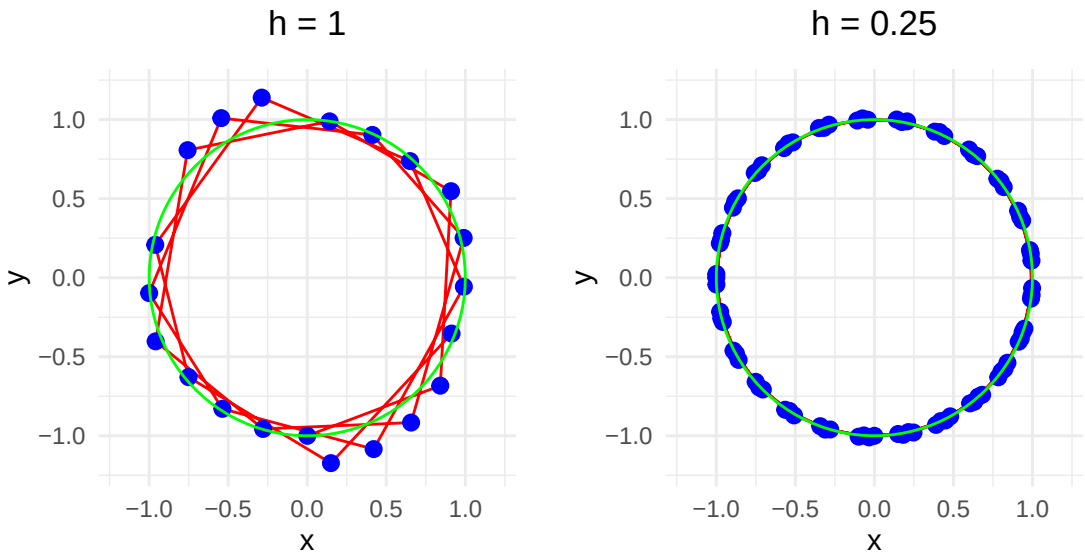


Figure 17.3: Consistent initial data $x(0) = -1$, $y(0) = 0$

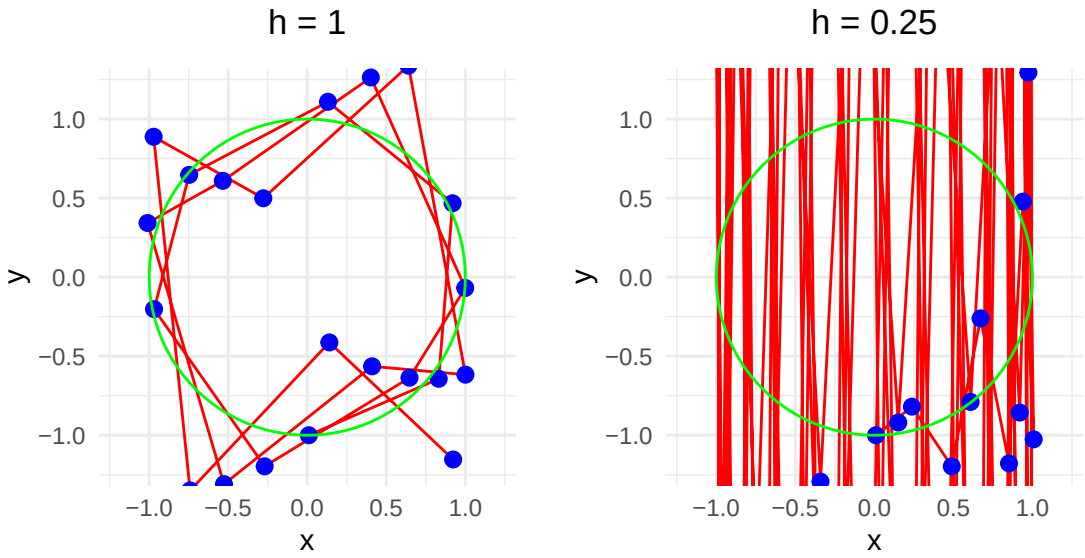


Figure 17.4: Inconsistent initial data $x(0) = -1$, $y(0) = 0.01$

17.5 Damped stabilization (Baumgarte)

Re-consider the DAE

$$\begin{aligned} x(t) &= \sin(t) \\ x'(t) + y(t) &= 0 \end{aligned}$$

stabilize the constraint $x(t) - \sin(t) = 0$ using Baumgarte

$$\begin{aligned} \tau(x(t) - \sin(t))' + (x(t) - \sin(t)) &= 0 \\ x'(t) + y(t) &= 0 \end{aligned}$$

Observation

The ODE $\tau z'(t) + z(t) = 0$ has solution

$$z(t) = z(0) \exp\left(-\frac{t}{\tau}\right)$$

and $\lim_{t\to\infty} z(t) = 0$

Using Crank-Nicolson

$$\frac{\tau}{h}(x_{k+1} - x_k) - \frac{\tau}{h}(\sin(t_{k+1}) - \sin(t_k)) +$$

$$\frac{1}{2}(x_{k+1} + x_k) - \frac{1}{2}(\sin(t_{k+1}) + \sin(t_k)) = 0$$

$$\frac{x_{k+1} - x_k}{h} + \frac{y_{k+1} + y_k}{2} = \quad 0$$

solving for x_{k+1} and y_{k+1}

$$x_{k+1} = \frac{2\tau - h}{2\tau + h}\big(x_k - \sin(t_k)\big) + \sin(t_{k+1})$$

$$y_{k+1} = \frac{\sin(t_k)(4\tau - 2h) - \sin(t_{k+1})(4\tau + 2h) + 4hx_k}{h(2\tau + h)} - y_k$$

17.6 Numerical result

- Implicit Euler appears to work also with inconsistent data.
- Crank–Nicolson does not work with inconsistent data.
- Baumgarte stabilization works well, but there is no general recipe for determining stabilization parameters (τ in this case).

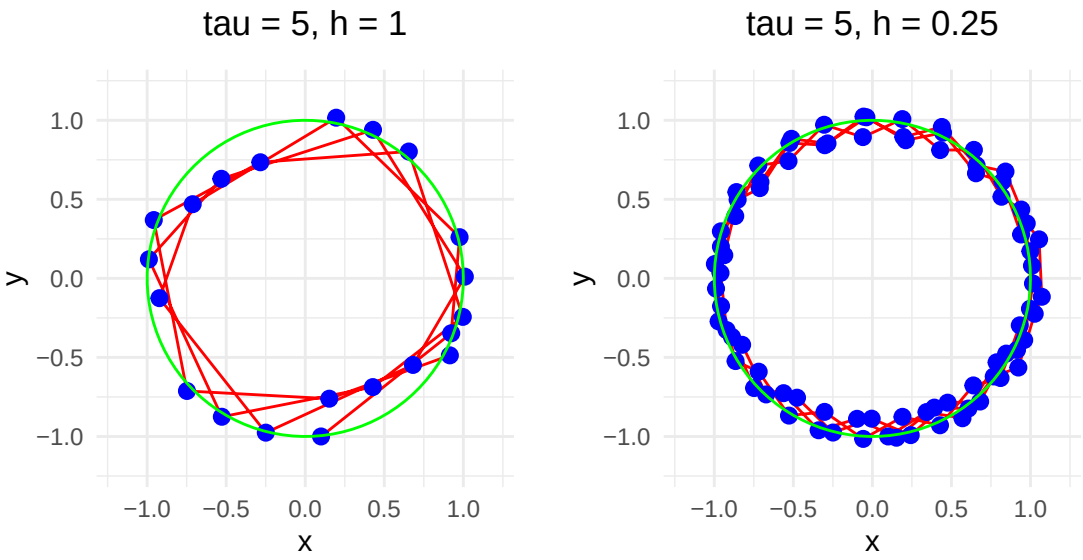


Figure 17.5: Inconsistent initial data $x(0) = -1$, $y(0) = 0.1$, $\tau = 5$

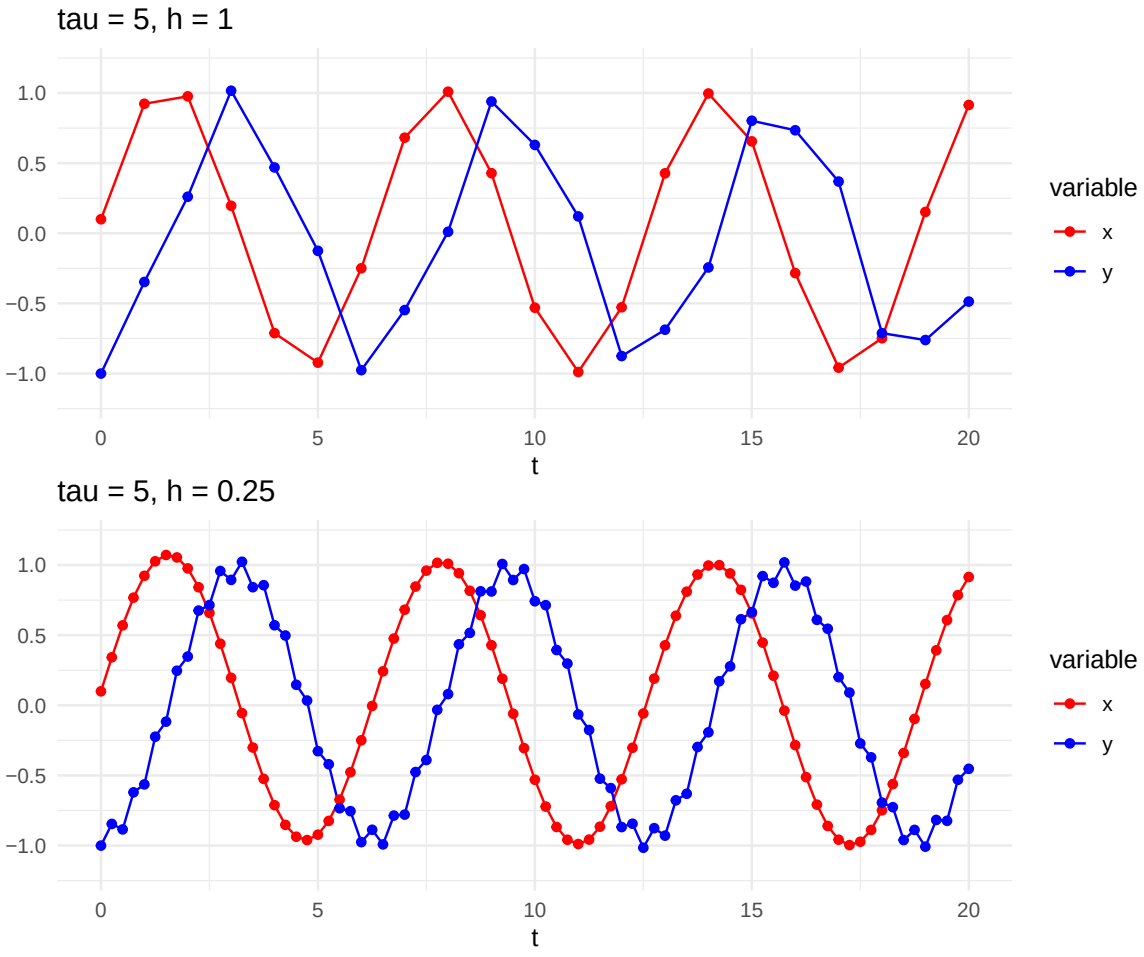


Figure 17.6: Inconsistent initial data $x(0) = -1$, $y(0) = 0.1$, $\tau = 5$



18. Baumgarte's stabilization

18.1 Differential-Algebraic Equations (DAEs) in Mechanics

This document offers an in-depth exploration of how Differential-Algebraic Equations (DAEs) are formulated and numerically solved, particularly within the context of mechanical systems. It emphasizes systems that involve constraints, which commonly appear in areas like multibody dynamics, robotics, and structural mechanics.

18.1.1 Notation and Setup

Consider a mechanical system with the following quantities:

- **Generalized (position) coordinates**: $\boldsymbol{p} \in \mathbb{R}^n$, representing the configuration of the system.
- **Kinetic energy**: $T(\boldsymbol{p}, \boldsymbol{p}', t)$, where $\boldsymbol{p}'$ is the time derivative of $\boldsymbol{p}$.
- **Potential energy**: $V(\boldsymbol{p}, t)$.
- **External generalized forces**: $\boldsymbol{Q}_{ex}(\boldsymbol{p}, t) \in \mathbb{R}^n$.
- **Constraint equations**: $\boldsymbol{\Phi}(\boldsymbol{p}, t) \in \mathbb{R}^m$, which enforce m constraints on the system.
- **Lagrangian**:

$$L(\boldsymbol{p}, \boldsymbol{p}', t) = T(\boldsymbol{p}, \boldsymbol{p}', t) - V(\boldsymbol{p}, t).$$

The **equations of motion** are derived from Lagrange's equations with constraints:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \boldsymbol{p}'}\right) - \frac{\partial L}{\partial \boldsymbol{p}} + \left(\frac{\partial \boldsymbol{\Phi}(\boldsymbol{p}, t)}{\partial \boldsymbol{p}}\right)^T \boldsymbol{\lambda} = \boldsymbol{Q}_{ex}(\boldsymbol{p}, \boldsymbol{p}', t),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ are the **Lagrange multipliers** enforcing the constraints.

18.1.2 Reformulating as a DAE

Kinetic Energy and Mass Matrix

The kinetic energy is often expressed in terms of the mass matrix $\boldsymbol{M}(\boldsymbol{p}, \boldsymbol{p}', t)$:

$$T(\boldsymbol{p}, \boldsymbol{p}', t) = \frac{1}{2}(\boldsymbol{p}')^T \boldsymbol{M}(\boldsymbol{p}, \boldsymbol{p}', t)\boldsymbol{p}'.$$

Substituting this into the equations of motion yields:

$$\boldsymbol{M}(\boldsymbol{p}, \boldsymbol{p}', t)\boldsymbol{p}'' + \left(\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{p}}\right)^T \boldsymbol{\lambda} = \boldsymbol{Q}(\boldsymbol{p}, \boldsymbol{p}', t),$$

where $\boldsymbol{Q}(\boldsymbol{p}, \boldsymbol{p}', t)$ includes external forces, potential energy gradients, and inertial terms.

Constraint Equations

The constraints $\boldsymbol{\Phi}(\boldsymbol{p},t) = 0$ must be satisfied at all times. Combining these with the equations of motion, we obtain the **full DAE system**:

$$\begin{cases} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{p}'t)\boldsymbol{p}'' + \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T\boldsymbol{\lambda} = \boldsymbol{Q}(\boldsymbol{p},\boldsymbol{p}',t), \\ \boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{0}. \end{cases}$$

This system has $n+m$ equations and $n+m$ unknowns ($\boldsymbol{p}''$ and $\boldsymbol{\lambda}$), making it well-posed.

18.2 Numerical Solution of DAEs

Solving DAEs requires specialized numerical methods due to the presence of algebraic constraints. Below, we outline a common approach based on time discretization and constraint differentiation.

18.2.1 Time Discretization

A simple numerical scheme can be derived using a **Taylor expansion** in time:

$$\boldsymbol{p}'(t+\Delta t) = \boldsymbol{p}'(t) + \boldsymbol{p}''(t)\Delta t + \mathcal{O}(\Delta t^2),$$

$$\boldsymbol{p}(t+\Delta t) = \boldsymbol{p}(t) + \boldsymbol{p}'(t)\Delta t + \frac{1}{2}\boldsymbol{p}''(t)\Delta t^2 + \mathcal{O}(\Delta t^3).$$

Neglecting higher-order terms, the updates for position and velocity become:

$$\boldsymbol{p}'_{k+1} = \boldsymbol{p}'_k + \boldsymbol{p}''_k\Delta t,$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{p}'_k\Delta t + \frac{1}{2}\boldsymbol{p}''_k\Delta t^2.$$

This requires computing $\boldsymbol{p}''_k$, which involves solving for $\boldsymbol{\lambda}$.

18.2.2 Computing $\boldsymbol{p}''$ and $\boldsymbol{\lambda}$

If the mass matrix $\boldsymbol{M}(\boldsymbol{p},\boldsymbol{q}',t)$ is **nonsingular**, we can express $\boldsymbol{p}''$ as:

$$\boldsymbol{p}'' = \boldsymbol{M}(\boldsymbol{p},\boldsymbol{q}',t)^{-1}\left[\boldsymbol{Q}(\boldsymbol{p},\boldsymbol{p}',t) - \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T\boldsymbol{\lambda}\right].$$

However, $\boldsymbol{\lambda}$ is still unknown. To resolve this, we differentiate the constraint equations to incorporate $\boldsymbol{p}''$.

Differentiating Constraints

Differentiating the constraints $\boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{0}$ once with respect to time yields:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{\Phi}_t + \boldsymbol{\Phi}_{\boldsymbol{p}}\boldsymbol{p}' = \boldsymbol{0}.$$

Differentiating a second time gives:

$$\boldsymbol{\Phi}_{tt} + 2\boldsymbol{\Phi}_{t\boldsymbol{p}}\boldsymbol{p}' + \boldsymbol{\Phi}_{\boldsymbol{pp}}(\boldsymbol{p}',\boldsymbol{p}') + \boldsymbol{\Phi}_{\boldsymbol{p}}\boldsymbol{p}'' = \boldsymbol{0}.$$

Rewriting in matrix form:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{q}',t) & \boldsymbol{\Phi}_{\boldsymbol{p}}^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}'' \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}(\boldsymbol{p},\boldsymbol{p}',t) \\ \boldsymbol{c}(\boldsymbol{p},\boldsymbol{p}',t) \end{pmatrix},$$

where:

$$\boldsymbol{c}(\boldsymbol{p},\boldsymbol{p}',t) = -\boldsymbol{\Phi}_{tt} - 2\boldsymbol{\Phi}_{t\boldsymbol{p}}\boldsymbol{p}' - (\boldsymbol{p}')^T\boldsymbol{\Phi}_{\boldsymbol{pp}}\boldsymbol{p}'.$$

This system can be solved for $\boldsymbol{p}''$ and $\boldsymbol{\lambda}$.

18.2.3 Numerical Integration Algorithm

Initialization

Given initial values $t_0 = 0$, $\boldsymbol{p}_0$, and $\boldsymbol{p}'_0$.

Time Stepping

For each time step $k = 0, 1, \ldots$:

1. **Solve for $\boldsymbol{p}''_k$ and $\boldsymbol{\lambda}_k$:**

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p}_k,\boldsymbol{p}'_k,t_k) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p}_k,t_k)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p}_k,t_k) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}''_k \\ \boldsymbol{\lambda}_k \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}(\boldsymbol{p}_k,\boldsymbol{p}'_k,t_k) \\ \boldsymbol{c}(\boldsymbol{p}_k,\boldsymbol{p}'_k,t_k) \end{pmatrix}.$$

2. **Update position and velocity**:

$$\boldsymbol{p}'_{k+1} = \boldsymbol{p}'_k + \Delta t\, \boldsymbol{p}''_k$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \Delta t\, \boldsymbol{p}'_k + \frac{1}{2}\Delta t^2 \boldsymbol{p}''_k.$$

18.2.4 Handling Constraint Drift

Over time, numerical errors can cause the system to drift away from the constraints $\boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{0}$. To mitigate this, we redefine the constraints as:

$$\widetilde{\boldsymbol{\Phi}}(\boldsymbol{p},t) = \boldsymbol{\Phi}(\boldsymbol{p},t) + \boldsymbol{a}_0 + \boldsymbol{a}_1 t,$$

where $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$ are chosen to satisfy the initial conditions:

$$\widetilde{\boldsymbol{\Phi}}(\boldsymbol{p}(0),0) = \boldsymbol{0}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{p}(0),0) = \boldsymbol{0}.$$

If $\boldsymbol{a}_0 = \boldsymbol{a}_1 = \boldsymbol{0}$, the constraints are perfectly satisfied initially. However, small errors can accumulate, leading to drift over long simulations.

18.3 Example: Pendulum in Redundant Coordinates

This section demonstrates the application of Differential-Algebraic Equations (DAEs) to model a pendulum using redundant coordinates. The pendulum is a classic example of a mechanical system with constraints, making it an ideal case study for DAE formulation and numerical solution.

18.3.1 System Description

Kinetic and Potential Energy

The pendulum consists of a mass m attached to a rigid, massless rod of length 1. The system is described using Cartesian coordinates (x, y), where x and y represent the horizontal and vertical positions of the mass, respectively.

- **Kinetic energy**:

$$T(t) = \frac{1}{2}m\left(x'(t)^2 + y'(t)^2\right).$$

- **Potential energy**:

$$V(t) = mgy(t),$$

where g is the gravitational acceleration.

Constraint Equation

The pendulum is constrained to move along a circular path of radius 1. This constraint is expressed as:

$$\Phi(x,y) = x^2 + y^2 - 1 = 0.$$

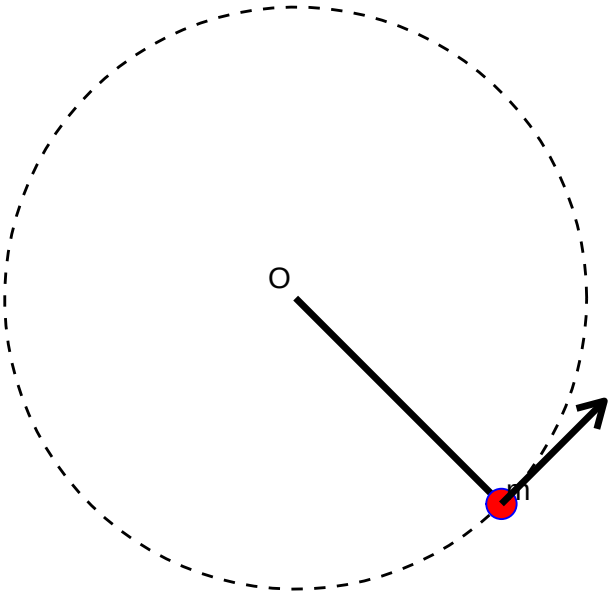


Figure 18.1: Schematic

18.3.2 Equations of Motion

Using Lagrange's equations with constraints, the equations of motion for the pendulum are:

$$\begin{cases} mx''(t) + 2x(t)\lambda(t) = 0, \\ my''(t) + 2y(t)\lambda(t) = -mg, \\ x(t)^2 + y(t)^2 = 1, \end{cases} \tag{18.1}$$

where $\lambda(t)$ is the Lagrange multiplier enforcing the constraint.

18.4 Numerical Solution

To solve the system numerically Equation 18.1, we differentiate the constraint equation twice to incorporate the acceleration terms $x''(t)$ and $y''(t)$.

First Derivative of the Constraint

Differentiating $\Phi(x,y) = 0$ once with respect to time yields:

$$2x(t)x'(t) + 2y(t)y'(t) = 0.$$

Second Derivative of the Constraint

Differentiating again gives:

$$2x(t)x''(t) + 2y(t)y''(t) + 2x'(t)^2 + 2y'(t)^2 = 0.$$

This equation introduces the acceleration terms $x''(t)$ and $y''(t)$, allowing us to solve for $\lambda(t)$.

The system of equations can be written in matrix form as:

$$\begin{pmatrix} m & 0 & 2x(t) \\ 0 & m & 2y(t) \\ 2x(t) & 2y(t) & 0 \end{pmatrix} \begin{pmatrix} x''(t) \\ y''(t) \\ \lambda(t) \end{pmatrix} = \begin{pmatrix} 0 \\ -mg \\ -2x'(t)^2 - 2y'(t)^2 \end{pmatrix}.$$

18.4.1 Time Discretization

Given initial conditions x_0, y_0, x'_0, and y'_0, we solve the system iteratively using the following steps:

1. Solve for x''_k, y''_k, and λ_k:

$$\begin{pmatrix} m & 0 & 2x_k \\ 0 & m & 2y_k \\ 2x_k & 2y_k & 0 \end{pmatrix} \begin{pmatrix} x''_k \\ y''_k \\ \lambda_k \end{pmatrix} = \begin{pmatrix} 0 \\ -mg \\ -2(x'_k)^2 - 2(y'_k)^2 \end{pmatrix}.$$

2. Update position and velocity:

$$\begin{aligned} x_{k+1} &= x_k + \Delta t\, x'_k + \frac{\Delta t^2}{2} x''_k, \\ y_{k+1} &= y_k + \Delta t\, y'_k + \frac{\Delta t^2}{2} y''_k, \\ x'_{k+1} &= x'_k + \Delta t\, x''_k, \\ y'_{k+1} &= y'_k + \Delta t\, y''_k. \end{aligned}$$

Numerical solution shows that the numerical method do not work well

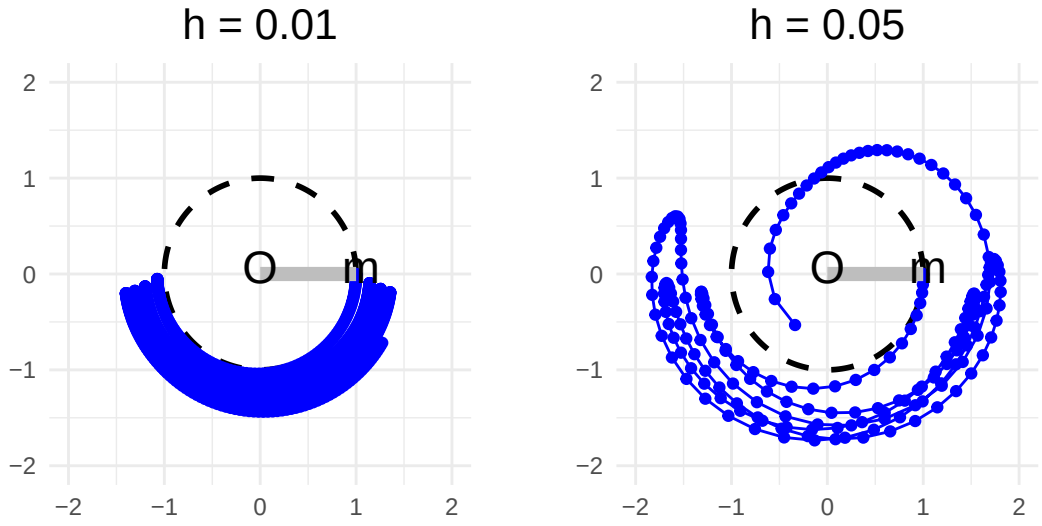


Figure 18.2: **Consistent Initial Conditions**: $x(0) = 1$, $y(0) = u(0) = v(0) = 0$. The system remains stable for short times but exhibits drift over longer simulations.

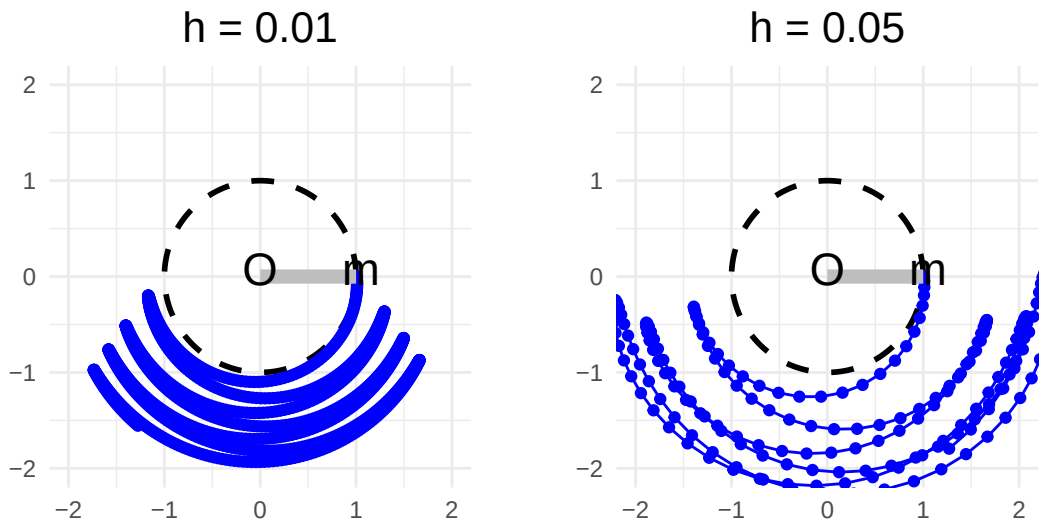


Figure 18.3: **Inconsistent Initial Conditions**: $x(0) = 1$, $y(0) = v(0) = 0$, $u(0) = 0.1$:. The system quickly diverges from the constraint.



19. Baumgarte's stabilization

A very popular way to stabilize the constraint $\boldsymbol{\phi}(\boldsymbol{p},t)=0$ and thus avoid the drift effect is using Baumgarte's stabilization, see [3]. This idea is well presented with other techniques in the survey paper [2]. The key idea is to substitute the constraint with a *differential equation* that *asymptotically* satisfies $\boldsymbol{\phi}(\boldsymbol{p},t)=0$. Obviously this technique is effective only if you are interested in the *long term* behaviour, since during transients the constraint may be violated. Since constrained mechanical systems require to differentiations to eliminate the Lagrange multipliers, we consider a linear second order stable differential equation of the form,

$$\frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2}+2\xi\omega_\mathrm{n}\frac{\mathrm{d}z(t)}{\mathrm{d}t}+\omega_\mathrm{n}^2 z(t)=0, \tag{19.1}$$

where $z(t)\in\mathbb{R}$. In this form we recognize the well known equation for second order linear mechanical or electrical systems, such as the *RLC circuit* or the *mass spring damper*. In these framework $\xi\in\mathbb{R}_{>0}$ is the *damping ratio* and $\omega_\mathrm{n}\in\mathbb{R}_{>0}$ is the *natural frequency*. Depending on the value of ξ different cases can arise:

- $0<\xi<1$: solution z is *underdamped*,
- $\xi=1$: solution z is *critically damped*,
- $\xi>1$: solution z is *overdamped*.

The general form for the solution is the following

$$z(t)=k_1 e^{\left(-\xi+\sqrt{\xi^2-1}\right)\omega_\mathrm{n}t}+k_2 e^{\left(-\xi-\sqrt{\xi^2-1}\right)\omega_\mathrm{n}t},$$

where $k_1,k_2\in\mathbb{R}$ are constants determined by the initial conditions $z_0=z(t=0)$ and $\dot{z}_0=\dot{z}(t=0)$. Solutions for the different cases can be arranged in different ways, here we show some of them. For a deeper treatment see for example [19]. Let us start from the underdamped case; solution takes the form

$$z(t)=c_1 e^{-\xi\omega_\mathrm{n}t}\cos\left(\sqrt{1-\xi^2}\omega_\mathrm{n}t-c_2\right), \tag{19.2}$$

where constants $c_1,c_2\in\mathbb{R}$ are given by,

$$c_1=\frac{\sqrt{z_0^2\omega_\mathrm{n}^2+\dot{z}_0^2+2z_0\dot{z}_0\xi\omega_\mathrm{n}}}{\sqrt{1-\xi^2}\omega_\mathrm{n}}$$
$$c_2=\arctan\left(\frac{\dot{z}_0+\xi\omega_\mathrm{n}z_0}{x_0\omega_\mathrm{n}\sqrt{1-\xi^2}}\right).$$

As we can observe in Equation 19.2 solution is a cosine function modulated by a decreasing exponential function. For the critically damped case a linear term appears and the solution takes the form,

$$z(t) = (c_1 + c_2 t)e^{-\omega_n t}, \tag{19.3}$$

where $c_1, c_2 \in \mathbb{R}$ are given by $c_1 = z_0$ and $c_2 = \dot{z}_0 + \omega_n z_0$. Finally for the overdamped case the solution takes the form

$$z(t) = c_1 e^{\left(-\xi+\sqrt{\xi^2-1}\right)\omega_n t} + c_2 e^{\left(-\xi-\sqrt{\xi^2-1}\right)\omega_n t}, \tag{19.4}$$

with constants,

$$c_1 = \frac{x_0\omega_n\left(\xi+\sqrt{\xi^2-1}\right)+\dot{x}_0}{2\omega_n\sqrt{\xi^2-1}},$$
$$c_2 = \frac{-x_0\omega_n\left(\xi-\sqrt{\xi^2-1}\right)-\dot{x}_0}{2\omega_n\sqrt{\xi^2-1}}.$$

However, regardless of the specific form of the solution, all the three cases satisfy a *global exponential stability* property, i.e., $\forall(z_0,\dot{z}_0) \in \mathbb{R}\times\mathbb{R}$ the solution $z(t,z_0,\dot{z}_0)$ satisfies

$$\lim_{t\to\infty} z(t,z_0,\dot{z}_0) = 0.$$

The notation $z(t,z_0,\dot{z}_0)$ is just to stress that the solution depends on time but also from initial conditions $(z_0,\dot{z}_0)$. This consideration suggests to substitute the constraint $\boldsymbol{\phi}(\boldsymbol{p},t)$ with an ODE of the form Equation 19.1. In this way, despite the initial conditions, the solution will asymptotically satisfies $\boldsymbol{\phi}(\boldsymbol{p},t) = 0$. Thus the explicit *Baumgarte's stabilization* for DAEs of index three takes the form,

$$\frac{\mathrm{d}^2\boldsymbol{\phi}}{\mathrm{d}t^2} + 2\xi\omega_n\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} + \omega_n^2\boldsymbol{\phi} = 0.$$

Computing explicitly the terms for mechanical systems the stabilization results,

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{p}}\dot{\boldsymbol{p}} + \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{p}}\ddot{\boldsymbol{p}} + \frac{\partial^2\boldsymbol{\phi}}{\partial t^2} + 2\xi\omega_n\frac{\partial\phi}{\partial\boldsymbol{p}}\dot{\boldsymbol{p}} + 2\xi\omega_n\frac{\partial\boldsymbol{\phi}}{\partial t} + \omega_n^2\boldsymbol{\phi} = 0.$$

which can be re-arranged in the following form

$$\begin{aligned} -\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{p}}\dot{\boldsymbol{v}} &= \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{p}}\boldsymbol{v} + \frac{\partial^2\boldsymbol{\phi}}{\partial t^2} + 2\xi\omega_n\frac{\partial\phi}{\partial\boldsymbol{p}}\boldsymbol{v} + 2\xi\omega_n\frac{\partial\boldsymbol{\phi}}{\partial t} + \omega_n^2\boldsymbol{\phi} \\ &= \boldsymbol{m}(\boldsymbol{p},\boldsymbol{v},t). \end{aligned} \tag{19.5}$$

Now Equation 19.5 can be easy implemented without any modification in Equation 11.3; indeed it is sufficient to redefine $\boldsymbol{m}(\boldsymbol{p},\boldsymbol{v},t)$ as in Equation 19.5. However a question is still open, how can we tune the parameters ω_n and ξ?. Unfortunately there is not general rule, but it is useful to notice that the velocity of convergence to zero of the solution depends on bothξ and ω_n. Therefore it is a good idea to select them in order to obtain an high decay rate compatibly with the *numerical stability*. This means that the dynamics associated to the constraint $\boldsymbol{\phi}(\boldsymbol{p},t)$ cannot be too fast compared to other ones, otherwise the system could be very hard (stiff) to simulate numerically. Indeed systems where two or more very different time scales are involved are usually called *stiff problems* or *stiff systems**. These may cause some numerical problems, since stiff equations require an integration step extremely small and are numerically unstable. In general it is hard to

provide a unique definition of stiffness, but the main idea is that the differential equations includes some terms that can lead to rapid variation in the solution.

■ **Example 19.1** Continuing Example 19.1} we want use the Baumgarte's echnique to stabilize the pendulum. Therefore the idea is to substitute the constraint

$$2\boldsymbol{v}^{\mathsf{T}}\boldsymbol{v} + 2\boldsymbol{p}\dot{\boldsymbol{v}} = 0$$

with

$$-2(\boldsymbol{p}\cdot\dot{\boldsymbol{v}}) = 2(\boldsymbol{v}\cdot\boldsymbol{v}) + 4\omega_{\mathrm{n}}\xi(\boldsymbol{p}\cdot\boldsymbol{v}) + \omega_{\mathrm{n}}^2(\boldsymbol{p}\cdot\boldsymbol{p} - \ell^2).$$

Thus final equations appears as,

$$\begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & m\boldsymbol{I} & -2\boldsymbol{p} \\ 0 & -2\boldsymbol{p}^T & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{v}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ -mg\boldsymbol{e}_2 \\ 2(\boldsymbol{v}\cdot\boldsymbol{v}) + 4\omega_{\mathrm{n}}\xi(\boldsymbol{p}\cdot\boldsymbol{v}) + \omega_{\mathrm{n}}^2(\boldsymbol{p}\cdot\boldsymbol{p} - \ell^2) \end{bmatrix}. \tag{19.6}$$

Again to appreciate the effects of Baumgarte's stabilization we can simulate Equation 19.6 with same parameters as Example 19.1 and stabilization parameters $\xi = 0.8$, $\omega_{\mathrm{n}} = 2$. The result is showed in Figure 19.1.

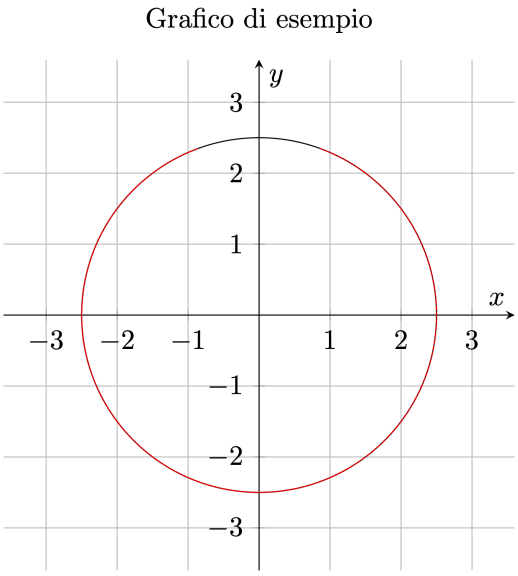


Figure 19.1: Pendulum simulation with Baumgarte's stabilization

■

19.0.1 Baumgarte details

The **Baumgarte method** stabilizes the constraints by introducing damping into the system. Consider the following ODE:

$$z''(t) + 2\zeta\omega z'(t) + \omega^2 z(t) = 0,$$

which has the solution:

$$z(t) = A(t)e^{-\zeta\omega t},$$

where $A(t)$ depends on the initial conditions. For $0 \leq \zeta \leq 1$, the solution decays exponentially without oscillation.

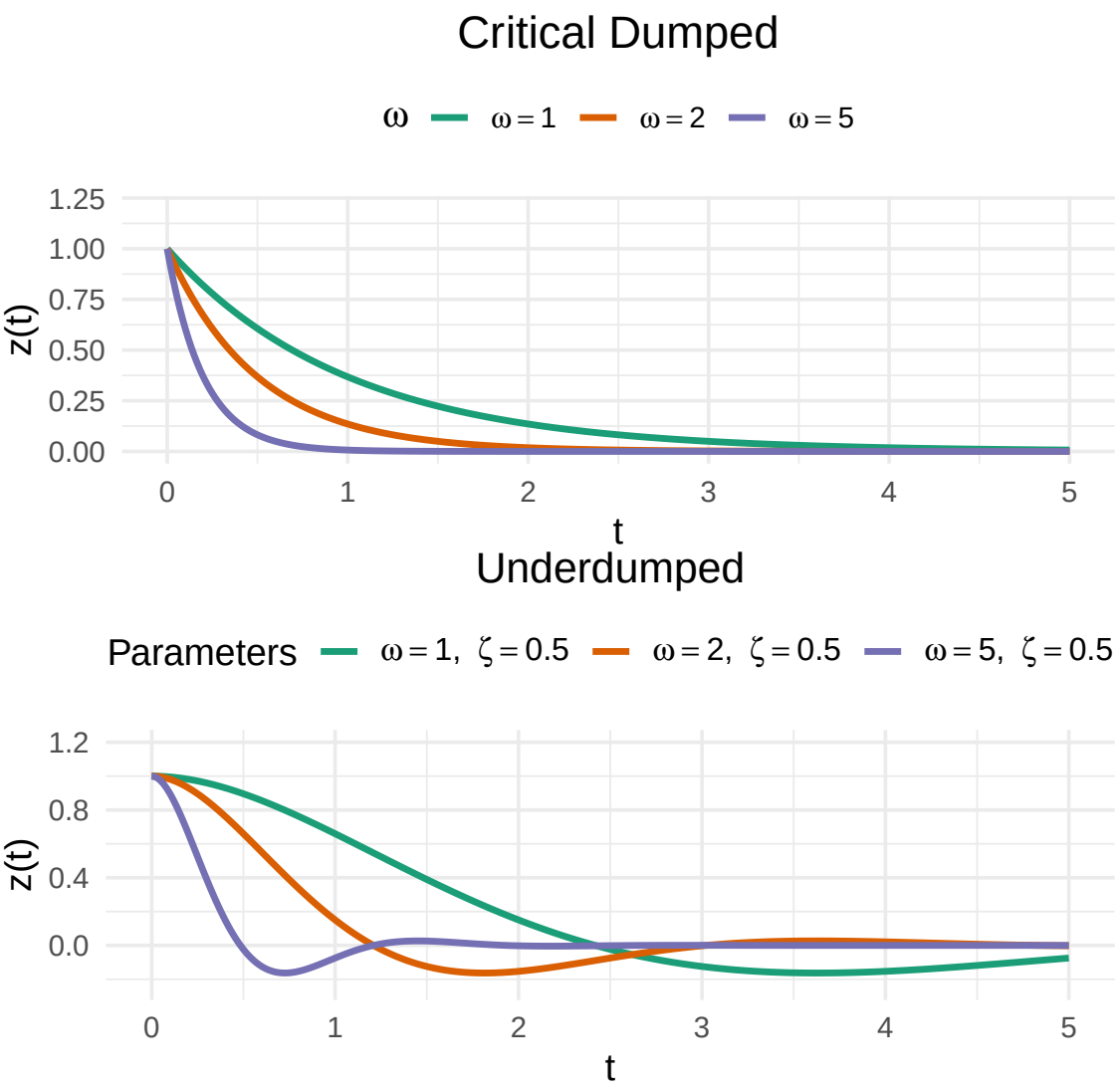


Figure 19.2: Solution example for some values of ω and ζ.

Over time, numerical errors can cause the system to drift away from the constraint $\Phi(x,y)$ of Equation 18.1 where

$$\Phi(x,y) = x^2 + y^2 - 1 = 0 \tag{19.7}$$

To mitigate this, we use the **Baumgarte stabilization** method, which modifies the constraint equation to include damping terms.

The stabilized constraint equation is:

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\Phi(x,y) + 2\zeta\omega\frac{\mathrm{d}}{\mathrm{d}t}\Phi(x,y) + \omega^2\Phi(x,y) = 0,$$

where ζ and ω are damping parameters. For Equation 19.7 this becomes:

$$2xx'' + 2yy'' + 2x'^2 + 2y'^2 + 4\zeta\omega(xx' + yy') + \omega^2(x^2 + y^2 - 1) = 0.$$

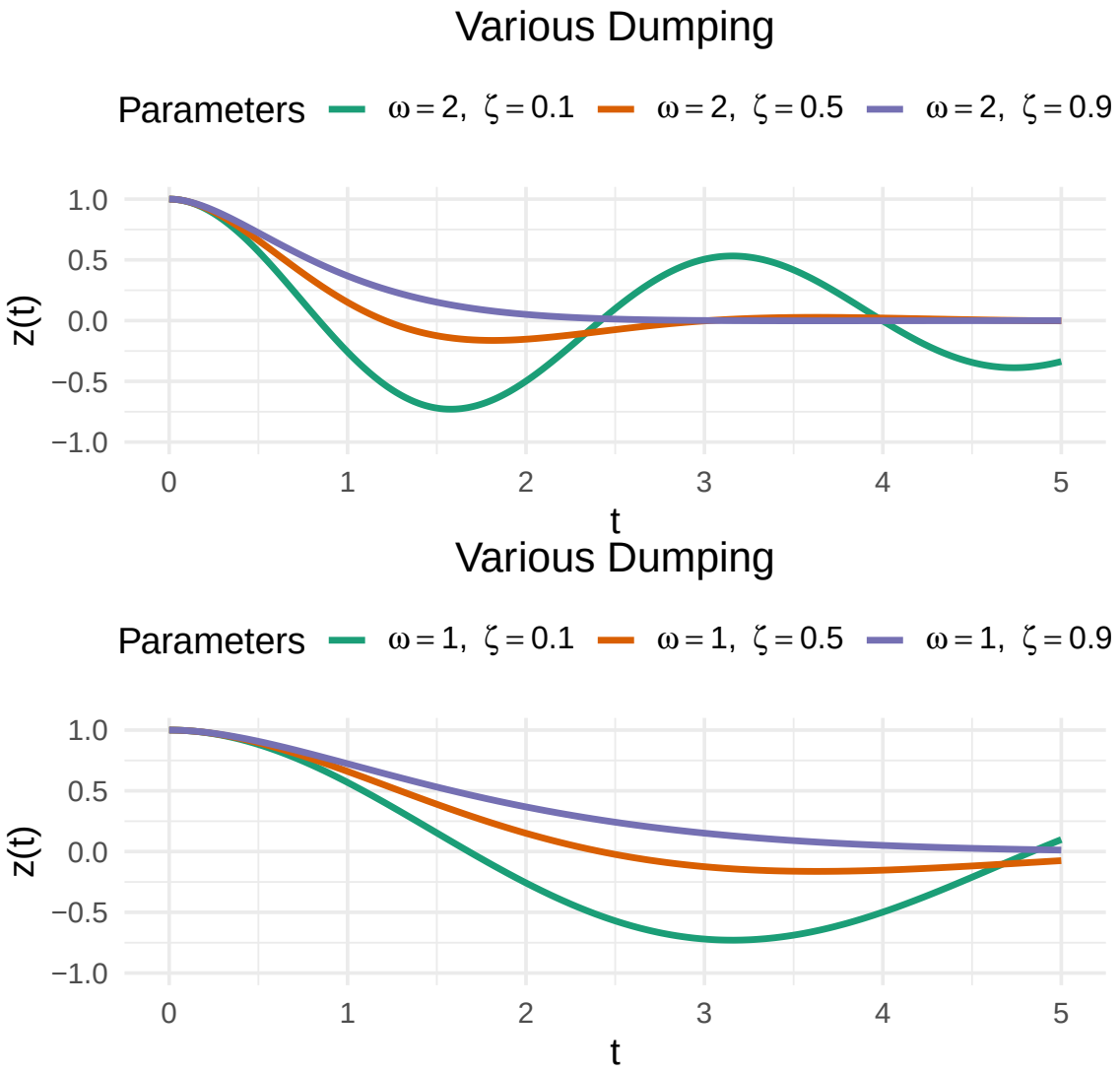


Figure 19.3: Solution example for some values of ω and ζ.

19.0.2 Modified Linear System

The stabilized system is solved using the same time-stepping approach, but with the right-hand side of the linear system modified to include the stabilization terms:

$$r_k = 2\left[(x_k')^2 + (y_k')^2\right] + 4\zeta\omega(x_k x_k' + y_k y_k') + \omega^2(x_k^2 + y_k^2 - 1).$$

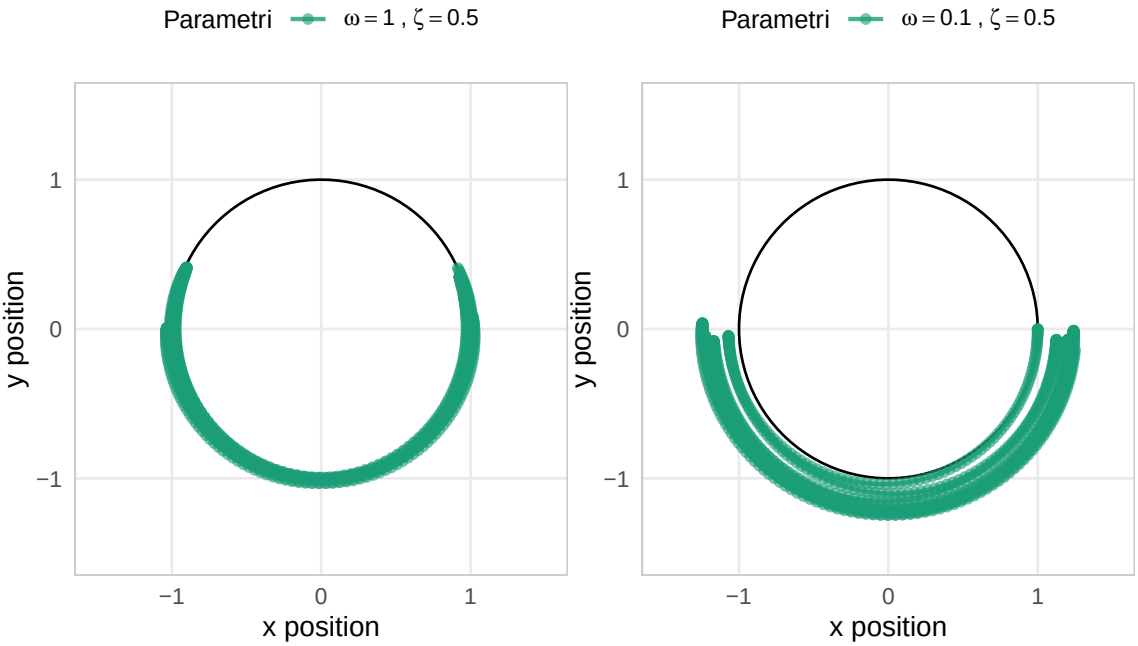


Figure 19.4: **Consistent Initial Conditions**: The system remains stable even for long simulation times. Energy is not conserved. Initial condition $x(0) = 1$, $y(0) = u(0) = v(0) = 0$, time step $\Delta t = 0.01$ final time $t_{\max} = 10$. Stabilization parameters are $\zeta = 0.5$ and $\omega = 1$ on the left and $\omega = 0.1$ on the right.

The pendulum example illustrates the challenges of solving DAEs numerically, particularly the issue of constraint drift. The Baumgarte stabilization method effectively mitigates this drift, ensuring long-term stability of the solution. Proper choice of damping parameters ζ and ω is critical for achieving optimal performance.

19.1 Pendulum DAE reduced to first order

Start from the motion equations

$$\begin{cases} mx'' + 2x\lambda = 0 \\ my'' + 2y\lambda + mg = 0 \\ x^2 + y^2 = 1 \end{cases}$$

divide by m and set $\mu = -2\lambda/m$

$$\begin{cases} x'' = \mu x \\ y'' = \mu y - g \\ x^2 + y^2 = r^2 \end{cases}$$

set $u = x'$ and $v = y'$ and transform to first order

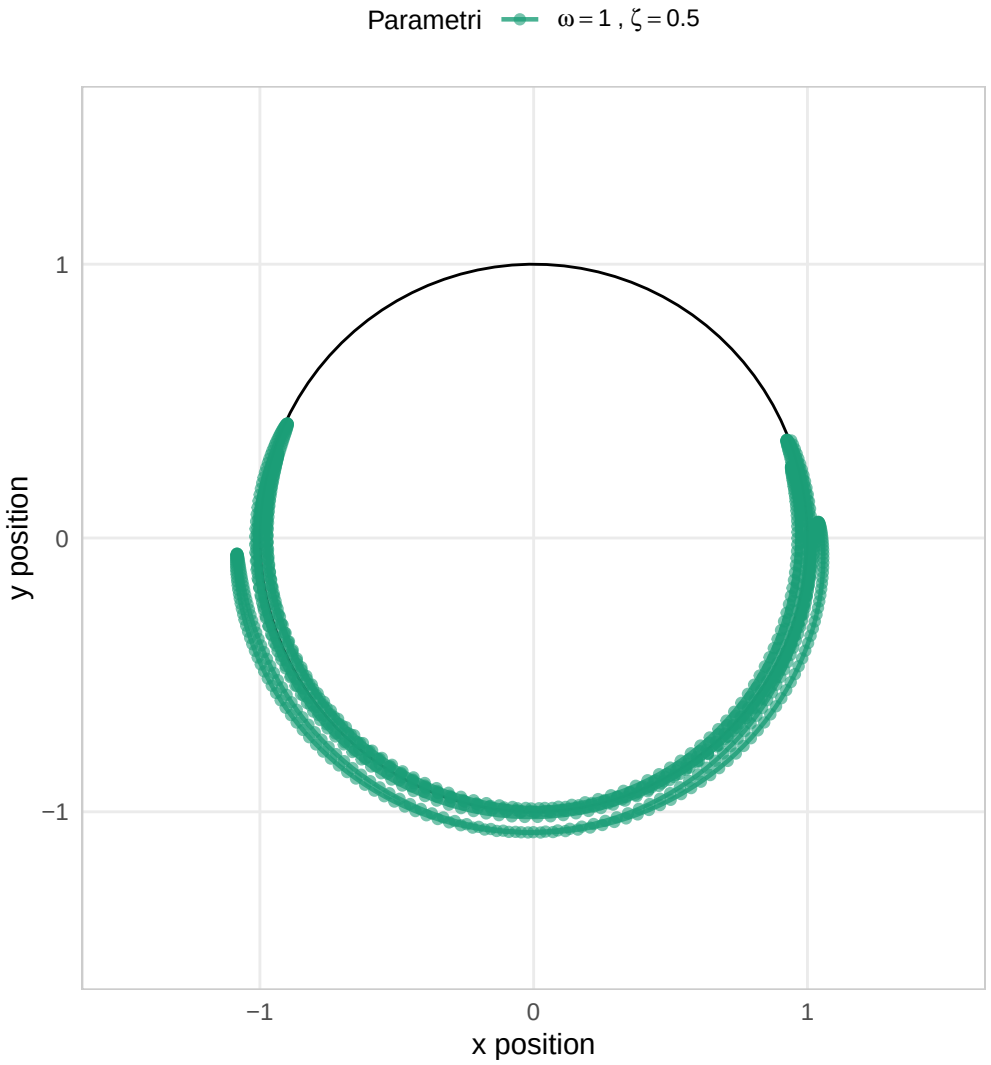


Figure 19.5: **Inconsistent Initial Conditions**: The system converges back to the constraint due to the damping terms. Initial condition $x(0) = 1$, $y(0) = v(0) = 0$, $u(0) = 0.1$ time step $\Delta t = 0.01$ final time $t_{\max} = 10$. Stabilization parameters are $\zeta = 0.5$ and $\omega = 1$ on the left and $\omega = 0.1$ on the right.

$$\begin{cases} u' = \mu x \\ v' = \mu y - g \\ x' = u \\ y' = v \\ x^2 + y^2 = r^2 \end{cases}$$

Derive one time the constraint

$$2xx' + 2yy' = 0 \quad \Longrightarrow \quad xu + yv = 0$$

Derive another time the constraint

$$\begin{aligned} x\dot{u} + y\dot{v} + \dot{x}u + \dot{y}v = 0 \quad &\Longrightarrow \\ \mu(x^2 + y^2) - gy + (u^2 + v^2) = 0 \quad &\Longrightarrow \quad r^2\mu - gy + (u^2 + v^2) = 0 \end{aligned}$$

Derive a third time the constraint obtaining a differential equation for μ:

$$\begin{aligned} r^2\dot{\mu} - gv + 2(u\dot{u} + v\dot{v}) = 0 \quad &\Longrightarrow \\ r^2\dot{\mu} - gv + 2\mu(ux + vy) - 2gv = 0 \quad &\Longrightarrow \\ r^2\dot{\mu} - 3gv = 0 \end{aligned}$$

The DAE has index 3

Summary

Original DAE

$$\begin{cases} u' = \mu x \\ v' = \mu y - g \\ x' = u \\ y' = v \\ x^2 + y^2 = 1 \end{cases}$$

hidden constraints

$$\begin{aligned} xu + yv = 0 \\ \mu - gy + (u^2 + v^2) = 0 \end{aligned}$$

constraint differential equation

$$\mu' - 3gv = 0$$

19.2 Integration of the DAE without stabilization

Consider the ODE without stabilization

$$\begin{cases} u' = \mu x \\ v' = \mu y - g \\ x' = u \\ y' = v \\ \mu' = 3gv/r^2 \end{cases}$$

Integration with implicit Euler:

$$\begin{aligned} \frac{u_{k+1} - u_k}{h} &= \mu_{k+1} x_{k+1} \\ \frac{v_{k+1} - v_k}{h} &= \mu_{k+1} y_{k+1} - g \\ \frac{x_{k+1} - x_k}{h} &= u_{k+1} \\ \frac{y_{k+1} - y_k}{h} &= v_{k+1} \\ \frac{\mu_{k+1} - \mu_k}{h} &= \frac{3g v_{k+1}}{r^2} \end{aligned}$$

Solving the first four equations for x_{k+1}, y_{k+1}, u_{k+1} and v_{k+1}

$$\begin{aligned} c_{k+1} &= 1 - h^2 \mu_{k+1} \\ x_{k+1} &= (x_k + h u_k)/c_{k+1} \\ y_{k+1} &= (y_k + h v_k - g h^2)/c_{k+1} \\ u_{k+1} &= (u_k + h \mu_{k+1} x_k)/c_{k+1} \\ v_{k+1} &= (v_k + h \mu_{k+1} y_k - g h)/c_{k+1} \end{aligned}$$

substituting into the multiplier equation and set $\delta = \mu_{k+1} - \mu_k$ a quadratic in δ is obtained:

$$r^2 h^2 \delta^2 + (3gh^2 y_k + r^2 (h^2 \mu_k - 1))\delta + 3gh(v_k + y_k \mu_k h - gh) = 0$$

Choose for the solution the δ closer to the esplicit step, i.e. $\delta \approx 3hgv_k/r^2$

Implicit Eulero implicito without stabilization do not work. The constraint $x^2 + y^2 = 1$ is violated in few steps:

onsider the DAE without stabilization

$$\begin{cases} u' = \mu x \\ v' = \mu y - g \\ x' = u \\ y' = v \\ x^2 + y^2 = r^2 \end{cases}$$

Using implicit Euler:

$$\begin{aligned}
\frac{u_{k+1}-u_k}{h} &= \mu_{k+1}x_{k+1}\\
\frac{v_{k+1}-v_k}{h} &= \mu_{k+1}y_{k+1}-g\\
\frac{x_{k+1}-x_k}{h} &= u_{k+1}\\
\frac{y_{k+1}-y_k}{h} &= v_{k+1}\\
x_{k+1}^2+y_{k+1}^2 &= 1
\end{aligned}$$

Solving the first four equation for x_{k+1}, y_{k+1}, u_{k+1} and v_{k+1}

$$\begin{aligned}
c_{k+1} &= 1-h^2\mu_{k+1}\\
x_{k+1} &= (x_k+hu_k)/c_{k+1}\\
y_{k+1} &= (y_k+hv_k-gh^2)/c_{k+1}\\
u_{k+1} &= (u_k+h\mu_{k+1}x_k)/c_{k+1}\\
v_{k+1} &= (v_k+h\mu_{k+1}y_k-gh)/c_{k+1}
\end{aligned}$$

substituting into constraint a quadratic equation in μ_{k+1} is obtained:

$$x_k^2+y_k^2-r^2+2h(x_ku_k+y_0v_k)+h^2(u_k^2+v_k^2)+g^2h^4-2gh^2(y_k+hv_k)+2h^2\mu_{k+1}-h^4\mu_{k+1}^2=0$$

$$\begin{aligned}
&\underbrace{[x_k^2+y_k^2-r^2]}_{A}+2h\underbrace{(x_ku_k+y_0v_k)}_{B}+h^2(u_k^2+v_k^2)\\
&+g^2h^4-2gh^2(y_k+hv_k)+2h^2\mu_{k+1}-h^4\mu_{k+1}^2=0
\end{aligned}$$

On the solution block (A) is zero so it is removed. Block (B) is the constraint derivative not 0 in the discretization. Thus,

$$2(x_ku_k+y_0v_k)+h(u_k^2+v_k^2)+g^2h^3-2gh(y_k+hv_k)+2h\mu_{k+1}-h^3\mu_{k+1}^2=0$$

Solving the quadratic

$$\mu_{k+1}=\frac{r\pm\sqrt{r^2+A}}{h^2r}$$

$$A=g^2h^4+2h(x_ku_k+y_kv_k)-2gh^2(y_k-hv_k)+h^2(u_k^2+v_k^2)$$

μ represent centripetal force maintaining mass on the constraint. Thus μ must be negative and solving trying to avoid cancellation error:

$$A=g^2h^4+2h(x_ku_k+y_kv_k)-2gh^2(y_k-hv_k)+h^2(u_k^2+v_k^2)$$

$$\mu_{k+1}=\frac{-A}{h^2(r^2+r\sqrt{r^2+A})}=\frac{-A/r^2}{h^2(1+\sqrt{1+A/r^2})}$$

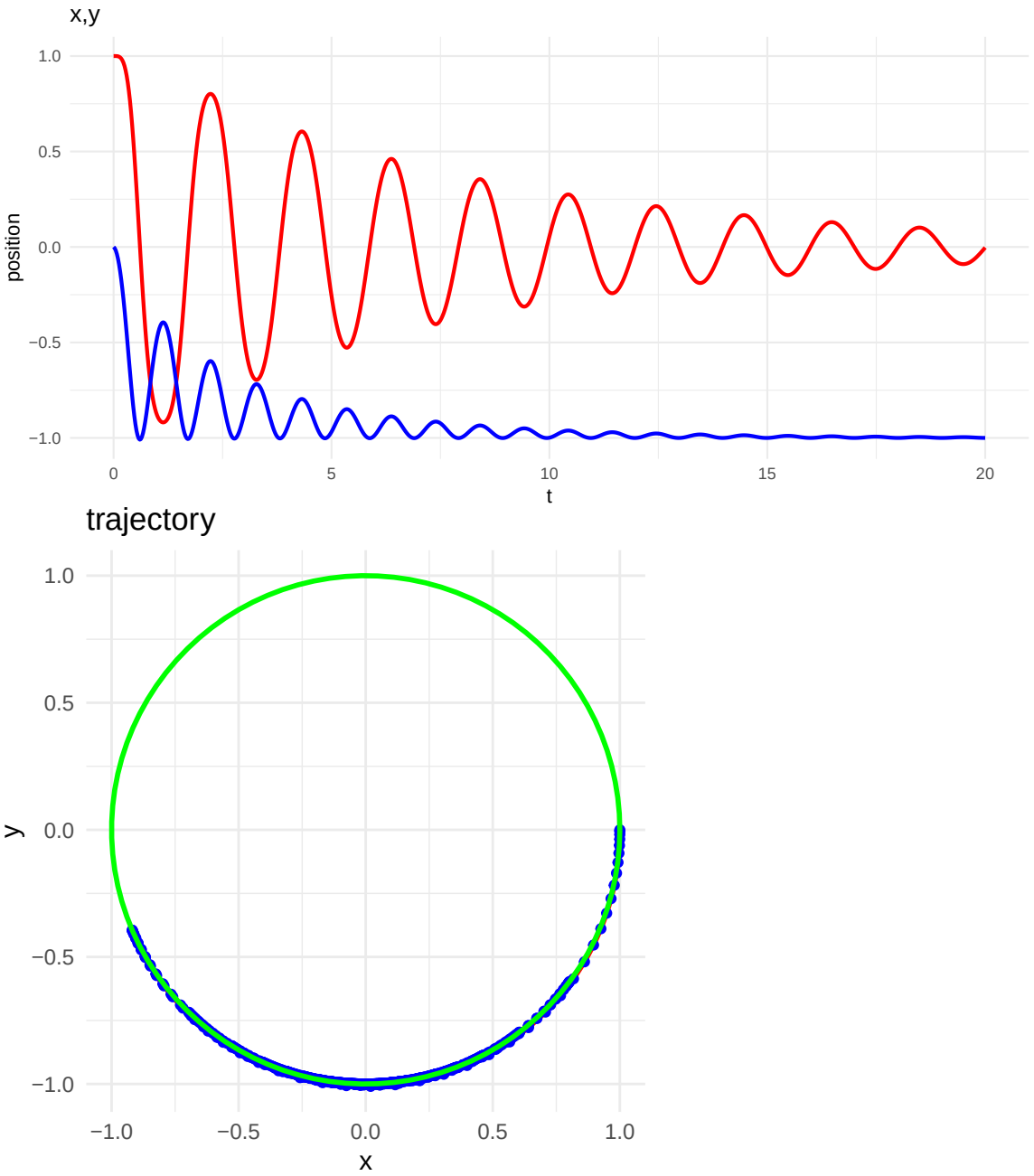


Figure 19.6: **Consistent** initial data $x_0 = 1$, $y_0 = u_0 = v_0 = 0$. Implicit Euler, no stabilization $\Delta t = 0.025$, $t_{\max} = 20$.

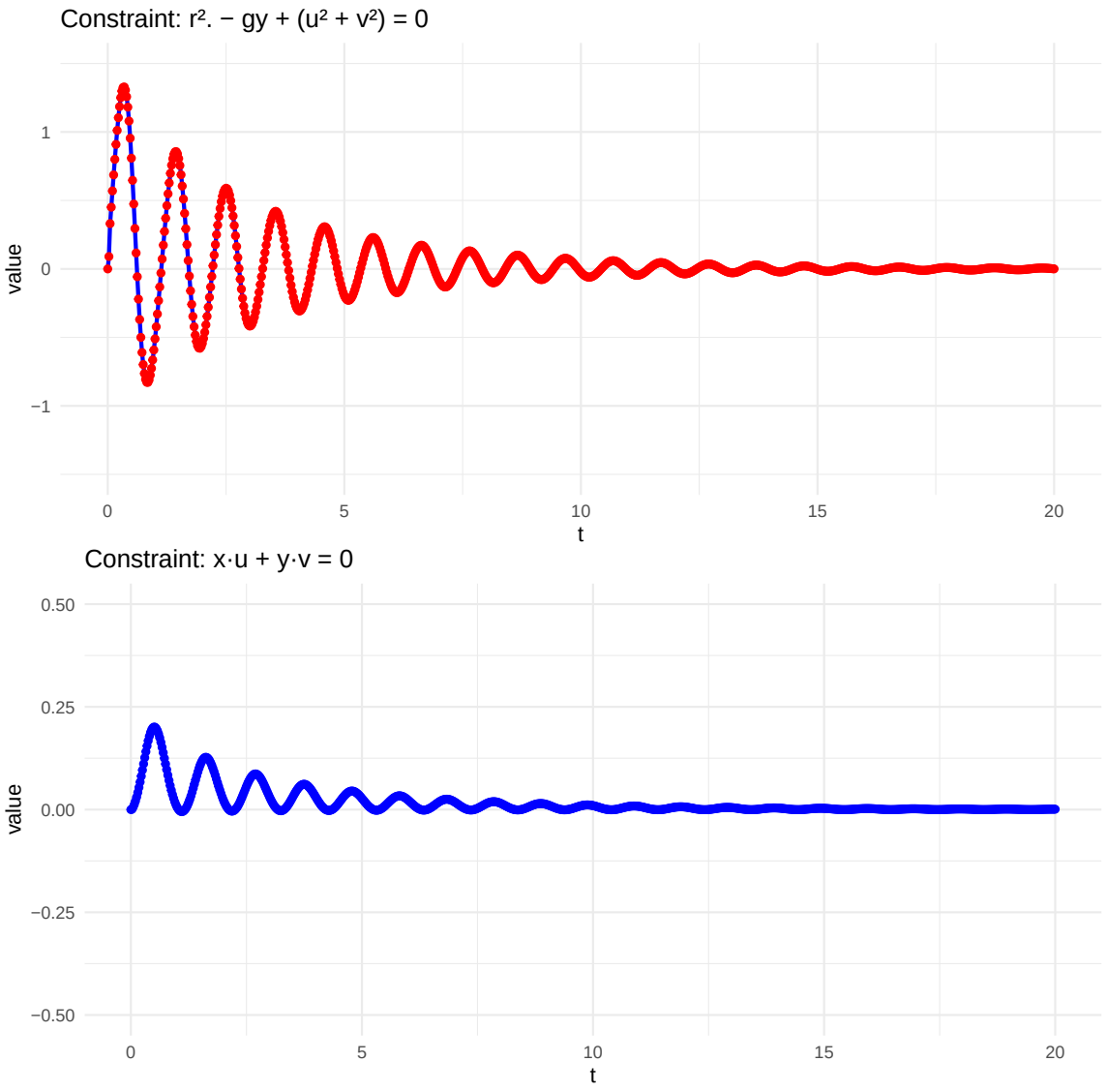


Figure 19.7: **Consistent** initial data $x_0 = 1$, $y_0 = u_0 = v_0 = 0$. Implicit Euler, no stabilization $\Delta t = 0.025$, $t_{\max} = 20$.

19.2.1 Numerical Results

19.3 Baumgarte stabilization

Inplicit Euler work but it is too dissipative. Cranck-Nicolson needs stabilization. DAE transformed to ODE need a third order ODE to stabilize as follows:

$$\frac{\mathrm{d}^3V(x,y)}{\mathrm{d}t^3} + a\frac{\mathrm{d}^2V(x,y)}{\mathrm{d}t^2} + b\frac{\mathrm{d}V(x,y)}{\mathrm{d}t} + cV(x,y) = 0$$

where $V(x,y) = x^2 + y^2 - 1$. And a, b, c are chosen in such a way

$$z'''(t) + az''(t) + bz'(t) + cz(t) = 0$$

satisfy $\lim_{t\to\infty} z(t) = 0$.

19.3.1 Results with Crank-Nicolson

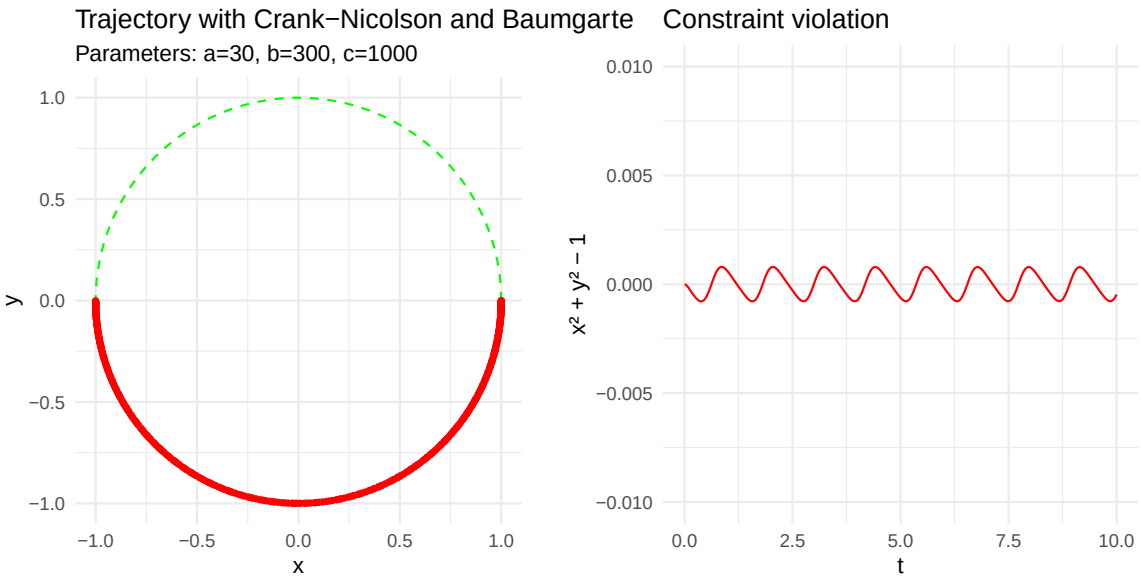


Figure 19.8: Initial **consistent** data and stabilizing values $a = 30$, $b = 300$, $c = 1000$.



20. Projection method

Another way to stabilize the constraint

$$\boldsymbol{\phi}(\boldsymbol{p},t)=0$$

and avoid the drift effect is using the **projection method**. The idea is to firstly reduce the index of the DAE, thus obtaining an ODE and a list of **hidden constraints**. Then the resulting ODE can be numerically integrated, leading to a generic intermediate solution

$$\widetilde{\boldsymbol{p}}(t_n).$$

This temporary solution is projected to the **hidden constraints** in order to minimize the error between the numerical integration solution and the actual system constraints, which is a **nonlinear constrained least squares problem** because of the norm chosen.

It should be noticed that unlike the Baumgarte's stabilization method this technique is useful and particularly effective if you are interested in the **short term** behavior, since even during transients the constraint is always satisfied. Different information from the original and reduced system can be used for various projection methods. For example, we can decide to get an advantage by first projecting the ODE solution

$$\widetilde{\boldsymbol{p}}(t_n)$$

to the **position** constraints and than to the **velocity** constraints.

▪ Example 20.1

Continuing Example 19.1 we want use the Projection technique to stabilize the pendulum. The numerical scheme for integrating the reduced DAE is that in Equation 11.3. We want now to project the results of the generic ODE numerical integrator to the following *hidden constraints*,

$$\frac{\mathrm{d}\phi}{\mathrm{d}t}=2(\boldsymbol{p}\cdot\boldsymbol{v})=0.$$

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}t^2}=2(\dot{\boldsymbol{p}}\cdot\boldsymbol{v})+2(\boldsymbol{p}\cdot\dot{\boldsymbol{v}})=2(\boldsymbol{v}\cdot\boldsymbol{v})+2(\boldsymbol{p}\cdot\dot{\boldsymbol{v}})=\boldsymbol{0}.$$

At each time step, the resulting orthogonal projection to the constraints is equal to

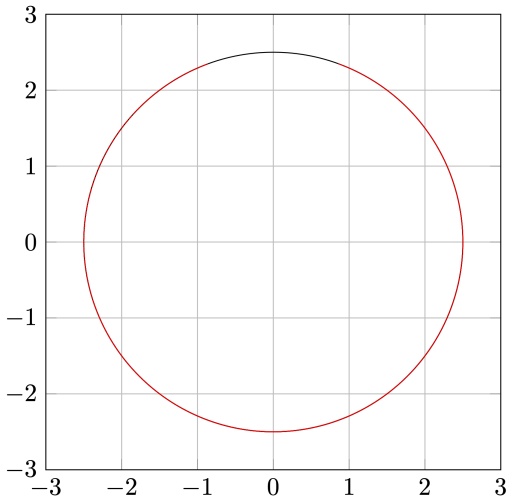


Figure 20.1: Pendulum simulation with Projection method.

$$\text{minimize:} \quad \frac{1}{2}\|\widetilde{\boldsymbol{p}} - \boldsymbol{p}\|^2$$

$$\text{subject to:} \quad \boldsymbol{\phi}(\boldsymbol{p},t) = \boldsymbol{p}\cdot\boldsymbol{p} - \ell^2 = 0$$

for the position coordinates projection, and

$$\text{minimize:} \quad \frac{1}{2}\|\widetilde{\boldsymbol{v}} - \boldsymbol{v}\|^2$$

$$\text{subject to:} \quad \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t}(\boldsymbol{p},t) = 2(\boldsymbol{v}\cdot\boldsymbol{p}) = 0$$

for the velocity coordinates projection.

■

20.1 Projection Methods in general

Projection methods aim to leverage well-established numerical techniques to efficiently solve DAEs while ensuring consistency with their inherent constraints. The core idea is to exploit the robustness and accuracy of existing solvers for ordinary differential equations (ODEs) by transforming the problem into an ODE system with constraints and subsequently applying a projection step to maintain consistency.

20.1.1 Key Principles of the Projection Method

1. **Utilization of existing ODE Solvers**:
   - Many sophisticated and highly accurate solvers are available for ODEs.
   - By reformulating a DAE as an ODE, one can take advantage of these mature numerical techniques.
2. **Constraint Satisfaction via Projection**:
   - Directly solving a transformed ODE may lead to constraint violation over time due to numerical errors, a phenomenon known as *drift*.

- To mitigate drift, a projection step is introduced to enforce consistency with the hidden constraints at each step or whenever the drift becomes non-negligible.

Projection Method Workflow

The projection method follows a straightforward two-step paradigm:

1. **ODE Step**: Advance the solution using any standard numerical solver for ODEs.

2. **Projection Step**: Adjust the computed solution by projecting it onto the manifold defined by the constraints, thereby ensuring consistency.

This approach provides a balance between computational efficiency and constraint preservation, making it an attractive method for solving DAEs.

20.1.2 Requirements for the Projection Method

The projection method relies on the following essential components:

- **Index Reduction**:
  - DAEs often have a higher differential index, which makes them difficult to solve directly.
  - Index reduction techniques are applied to convert the DAE into an ODE system with explicit algebraic constraints.
- **Collection of Constraints and Jacobians**:
  - To perform projection, the constraints and their Jacobians must be determined.
  - This involves symbolic computations to extract hidden constraints and their derivatives.

Thus, while the projection method simplifies numerical integration, it does not eliminate the need for symbolic preprocessing:

- Index reduction typically requires symbolic manipulations proportional to the index of the DAE.
- The projection step involves a smaller symbolic computation effort, primarily related to evaluating constraint Jacobians, whose complexity depends on the chosen projection algorithm.

20.2 Mathematical Formulation

Consider a general DAE system:

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

where $\boldsymbol{x}$ represents the state variables, and $\boldsymbol{F}$ defines the differential-algebraic structure.

To facilitate numerical integration, the system is transformed into an ODE system with explicit constraints using index reduction:

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{G}(\boldsymbol{x},t), && \text{[ODE]} \\ \boldsymbol{0} &= \boldsymbol{H}(\boldsymbol{x},t), && \text{[Invariants or hidden constraints]} \end{aligned}$$

where:

- $\boldsymbol{G}(\boldsymbol{x},t)$ defines the transformed r.h.s. of the ODE system.
- $\boldsymbol{H}(\boldsymbol{x},t)$ represents the constraints that must be satisfied throughout the integration process.

20.3 Numerical Integration with Projection

Step 1: ODE Integration

Using an explicit numerical scheme, such as the explicit Euler method, we compute a preliminary update:

$$\widetilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_k + \Delta t\, \boldsymbol{G}(\boldsymbol{x}_k, t_k)$$

where $\widetilde{\boldsymbol{x}}_{k+1}$ is the intermediate solution, which may not satisfy the constraints due to numerical errors.

Step 2: Projection onto the Constraint Manifold

To restore consistency, the next step is to compute the corrected state $\boldsymbol{x}_{k+1}$ by solving the constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{z} - \widetilde{\boldsymbol{x}}_{k+1}\|^2 \\ \text{subject to the constraint:} \quad & \boldsymbol{H}(\boldsymbol{z}, t_{k+1}) = \boldsymbol{0}. \end{aligned}$$

then set $\boldsymbol{x}_{k+1} = \boldsymbol{z}$.

This optimization step ensures that $\boldsymbol{x}_{k+1}$ remains on the constraint manifold, effectively eliminating the drift introduced by numerical integration.

20.4 Generic Minimization Problem with Constraints

In many numerical applications, we encounter the problem of minimizing a function while satisfying a set of constraints. The general form of such a constrained optimization problem is:

$$\text{minimize} \quad f(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{h}(\boldsymbol{z}) = \boldsymbol{0}$$

where:

- $f(\boldsymbol{z})$ is the objective function to be minimized.
- $\boldsymbol{h}(\boldsymbol{z})$ represents the set of constraints that must be satisfied.

Here is shown a sketch in matlab for the solution of constrained minimization problem using function `fmincon`:

```
% Dimensione di z
n = ...;

% objective function f(z)
objective = @(z) ...;

% constraint h(z)
nonlcon = @(z) ...;

% Initial point
z0 = xtilde;

% Solve using fmincon
options = optimoptions(  'fmincon', ...
  'Display',             'iter',   ...
  'Algorithm',           'sqp',    ...
```

```
  'MaxIterations',       100,     ...
  'OptimalityTolerance', 1e-8     ...
);

z_sol = fmincon( objective, z0, ...
                 [], [], [], [], [], [], ...
                 nonlcon, options);
```

Solution Using Lagrange Multipliers

To find the solution, we introduce the **Lagrangian function**:

$$\mathscr{L}(z,\lambda) = f(z) - \lambda \cdot h(z)$$

where:

- λ is the vector of **Lagrange multipliers**, which enforces the constraints.

The necessary conditions for a solution (also called the **Karush-Kuhn-Tucker conditions**) correspond to the stationary points of the Lagrangian:

$$\begin{cases} \dfrac{\partial \mathscr{L}(z,\lambda)}{\partial z} = 0, \\ \dfrac{\partial \mathscr{L}(z,\lambda)}{\partial \lambda} = 0. \end{cases}$$

Expanding these derivatives explicitly:

$$\begin{cases} \nabla f(z) - h_z(z)^T \lambda = 0, \\ h(z) = 0. \end{cases}$$

where h_z is the Jacobian matrix of the constraints:

$$h_z = \frac{\partial h(z)}{\partial z}.$$

This system of nonlinear equations must be solved to find the optimal solution (z,λ).

Linearization of the Minimization Problem

Since the nonlinear system above is often difficult to solve directly, we employ a **linearization technique** around a current estimate z. This leads to an iterative solution approach.

By performing a first-order Taylor expansion of the necessary conditions with respect to z, we obtain:

$$\begin{cases} \nabla f(z) + \nabla^2 f(z)\delta z - h_z(z)^T \lambda = \mathscr{O}(\|\delta z\|^2), \\ h(z) + h_z(z)\delta z = \mathscr{O}(\|\delta \lambda\|^2). \end{cases}$$

Ignoring the higher-order terms $\mathscr{O}(\|\delta z\|^2)$, we derive an iterative **Newton-like method** for solving the optimization problem.

Iterative Algorithm for the Constrained Minimization Problem

1. Initialize with an initial guess $z_0 = \widetilde{x}_{k+1}$.

2. Update the solution iteratively:

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{z}_\ell + \delta \boldsymbol{z}, \quad \ell = 0, 1, \dots \quad \text{(until convergence)}.$$

3. Solve for the update $\delta \boldsymbol{z}$ by solving the following **linear system**:

$$\begin{pmatrix} \nabla^2 f(\boldsymbol{z}_\ell) & -\boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}_\ell)^T \\ -\boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}_\ell) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{z} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla f(\boldsymbol{z}_\ell) \\ \boldsymbol{h}(\boldsymbol{z}_\ell) \end{pmatrix}$$

This system provides the correction $\delta \boldsymbol{z}$ for the next iteration step.

Special Case: Quadratic Objective Function

A particularly important case is when the objective function is **quadratic**, given by:

$$f(\boldsymbol{z}) = \frac{1}{2} \|\boldsymbol{z} - \widetilde{\boldsymbol{x}}_{k+1}\|^2.$$

For this choice of $f(\boldsymbol{z})$, the gradient and Hessian simplify to:

$$\nabla f(\boldsymbol{z}) = \boldsymbol{z} - \widetilde{\boldsymbol{x}}_{k+1}, \qquad \nabla^2 f(\boldsymbol{z}) = \boldsymbol{I}.$$

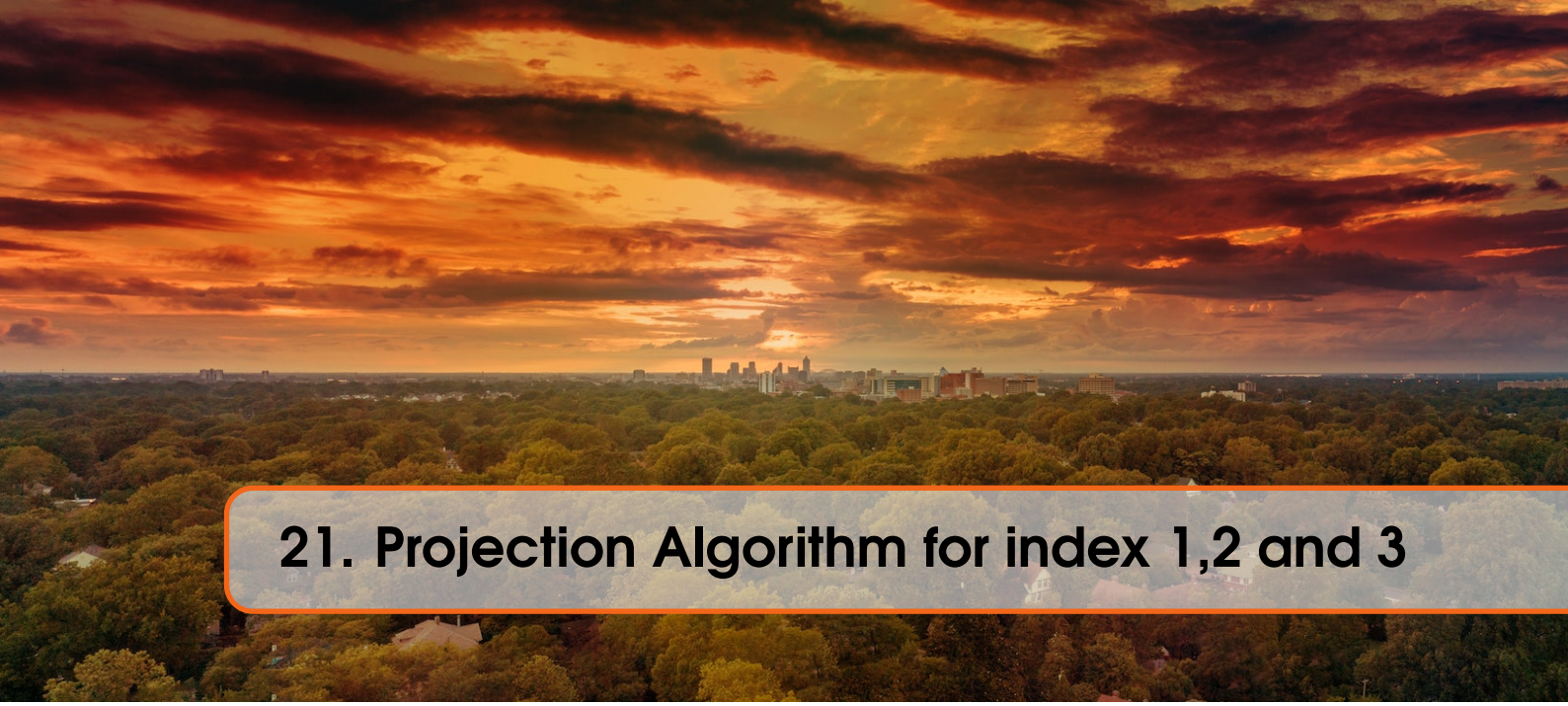
Thus, the linear system simplifies to:

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}_\ell)^T \\ -\boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}_\ell) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{z} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{z}_\ell \\ \boldsymbol{h}(\boldsymbol{z}_\ell) \end{pmatrix}$$

This case is particularly useful in applications like **projection methods** for constrained dynamics, where $\widetilde{\boldsymbol{x}}_{k+1}$ represents an unconstrained update that needs to be corrected to satisfy the constraints $\boldsymbol{h}(\boldsymbol{z}) = \boldsymbol{0}$.

A sketch in matlab for the projection solution is shown here:

```
n      = ...; % dimension of the problem
m      = ...; % number of constraints
xt     = ...; % point to be projected
H_fun  = @(z) ...; % The constraints m x 1 matrix
Hz_fun = @(z) ...; % Jacobian of the constraints m x n matrix
z      = xt; % initial point
for k=1:max_iter
  Hz  = Hz_fun( z );
  H   = H_fun( z );
  M   = [ eye(n), Hz.' ; Hz, zeros(n,n) ];
  b   = [ xt - z; H ];
  sol = M \ b;
  dz  = sol(1:n);
  if norm(dz,inf) < tolerance
    break;
  end
  z = z+dz;
end
```



21. Projection Algorithm for index 1,2 and 3

21.1 Index-1 DAE Projection Algorithm

Consider an index-1 Differential-Algebraic Equation (DAE) system of the form:

$$\begin{cases} \boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) \\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t) \end{cases} \tag{21.1}$$

where:

- $\boldsymbol{x}$: differential variables
- $\boldsymbol{y}$: algebraic variables
- $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)$: differential dynamics
- $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t) = \boldsymbol{0}$: algebraic constraints

Reformulate with Dummy Variables by introducing a dummy derivative $\boldsymbol{y}_{\text{dot}}(t)$:

$$\begin{cases} \boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{y}(t),t) \\ \boldsymbol{y}'(t) = \boldsymbol{y}_{\text{dot}}(t) \\ \quad \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{y}_{\text{dot}},t) \\ \quad \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t) \end{cases}$$

Define:

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{y}_{\text{dot}},t) &= \frac{d}{dt}\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t) \\ &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\boldsymbol{x}' + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\boldsymbol{y}' + \frac{\partial \boldsymbol{g}}{\partial t} \\ &= \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)\,\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)\,\boldsymbol{y}_{\text{dot}} + \boldsymbol{g}_t(\boldsymbol{x},\boldsymbol{y},t) \end{aligned}$$

Using:

$$\boldsymbol{g}_{\boldsymbol{x}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}, \quad \boldsymbol{g}_{\boldsymbol{y}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}, \quad \boldsymbol{g}_t = \frac{\partial \boldsymbol{g}}{\partial t}$$

we solve for $\boldsymbol{y}_{\text{dot}}$:

$$\boldsymbol{y}_{\text{dot}} = -\boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)^{-1}\left[\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)(\boldsymbol{x},\boldsymbol{y},t)\,\boldsymbol{f} + \boldsymbol{g}_t(\boldsymbol{x},\boldsymbol{y},t)\right]$$

Define:

$$\boldsymbol{e}(\boldsymbol{x},\boldsymbol{y},t) = -\boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)^{-1}\left[\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)\,\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{g}_t(\boldsymbol{x},\boldsymbol{y},t)\right]$$

and obtain the ODE + constraint system:

$$\begin{cases} \boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) \\ \boldsymbol{y}' = \boldsymbol{e}(\boldsymbol{x},\boldsymbol{y},t) \\ \mathbf{0} = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t) \end{cases}$$

The projection method advances the ODE using any numerical integration scheme (Euler, RK, etc.) and then **projects** onto the constraints (**red part**).

21.1.1 Projection Phase

The system is advanced with a time integrator (e.g., Euler, Runge–Kutta), then projected onto the constraint manifold.

E.g. using Explicit Euler step:

$$\begin{cases} \widetilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_k + \Delta t\, \boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{y}_k,t_k) \\ \widetilde{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_k + \Delta t\, \boldsymbol{e}(\boldsymbol{x}_k,\boldsymbol{y}_k,t_k) \end{cases}$$

The true solution will be

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \widetilde{\boldsymbol{x}}_{k+1} + \delta\boldsymbol{x} \\ \boldsymbol{y}_{k+1} &= \widetilde{\boldsymbol{y}}_{k+1} + \delta\boldsymbol{y} \end{aligned}$$

Index-1 Projection as Minimization

Project the point $(\widetilde{\boldsymbol{x}}_{k+1}, \widetilde{\boldsymbol{y}}_{k+1})$ onto the constraint manifold:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \frac{1}{2}\left\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1}\right\|^2 + \frac{1}{2}\left\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1}\right\|^2$$

$$\text{subject to} \quad \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t_{k+1}) = \mathbf{0}$$

Using **Lagrange multipliers**, the candidate solutions are the stationary points of the Lagrangian:

$$\mathscr{L}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1}\|^2 + \frac{1}{2}\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1}\|^2 - \boldsymbol{\lambda}\cdot\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t_{k+1})$$

which leads to the stationery conditions:

$$\begin{cases} \boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t_{k+1})^T\boldsymbol{\lambda} = \mathbf{0} \\ \boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t_{k+1})^T\boldsymbol{\lambda} = \mathbf{0} \\ \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},t_{k+1}) = \mathbf{0} \end{cases}$$

Newton's Method for Projection

Use Newton (like) iterations to solve the nonlinear system. Set

$$\boldsymbol{x}_{k+1} = \widetilde{\boldsymbol{x}}_{k+1} + \delta\boldsymbol{x} \quad \text{and} \quad \boldsymbol{y}_{k+1} = \widetilde{\boldsymbol{y}}_{k+1} + \delta\boldsymbol{y}$$

after linearization neglecting high order term the Newton like iterative method become:

1. Initialization $\ell \leftarrow 0$

$$\begin{aligned} \boldsymbol{x}^{(\ell)} &= \widetilde{\boldsymbol{x}}_{k+1} \\ \boldsymbol{y}^{(\ell)} &= \widetilde{\boldsymbol{y}}_{k+1} \end{aligned}$$

2. Solve:

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{g}_{\boldsymbol{x}}^T \\ \boldsymbol{0} & \boldsymbol{I} & -\boldsymbol{g}_{\boldsymbol{y}}^T \\ -\boldsymbol{g}_{\boldsymbol{x}} & -\boldsymbol{g}_{\boldsymbol{y}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{x} \\ \delta \boldsymbol{y} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^{(\ell)} \\ \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{y}^{(\ell)} \\ \boldsymbol{g}(\boldsymbol{x}^{(\ell)}, \boldsymbol{y}^{(\ell)}, t_{k+1}) \end{pmatrix}$$

3. Update:

$$\begin{aligned} \boldsymbol{x}^{(\ell+1)} &= \boldsymbol{x}^{(\ell)} + \delta \boldsymbol{x}, \\ \boldsymbol{y}^{(\ell+1)} &= \boldsymbol{y}^{(\ell)} + \delta \boldsymbol{y} \end{aligned}$$

4. $\ell \leftarrow \ell + 1$ and repeat step 2 and 3 until convergence.

Addendum

For Equation 21.1 **it is not convenient to use projection**, instead solve the ODE

$$\boldsymbol{x}' = \boldsymbol{F}(\boldsymbol{x}, t)$$

where

$$\boldsymbol{F}(\boldsymbol{x}, t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}(x,t), t)$$

and $\boldsymbol{y}(x,t)$ is the solution of

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t) = 0 \qquad (\star)$$

how to solve $(\star)$?
For example using explicit Euler

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{y}_k, t_k)$$

$\boldsymbol{y}_{k+1}$ is the solution of $\boldsymbol{c}(\boldsymbol{z}) = \boldsymbol{0}$ where $\boldsymbol{c}(\boldsymbol{z}) = \boldsymbol{g}(\boldsymbol{x}_{k+1}, \boldsymbol{z}, t_{k+1})$ that can be solved using Newton-Raphson method

$$\boldsymbol{z}^{(0)} = \boldsymbol{y}_k, \qquad \boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^{(\ell)} - \left(\frac{\partial \boldsymbol{c}(\boldsymbol{z}^{(\ell)})}{\partial \boldsymbol{z}} \right)^{-1} \boldsymbol{c}(\boldsymbol{z}^{(\ell)})$$

or better, start with $\boldsymbol{y}_{k+1}^{(0)} = \boldsymbol{y}_k$ and then

$$\boldsymbol{y}_{k+1}^{(\ell+1)} = \boldsymbol{y}_{k+1}^{(\ell)} - \boldsymbol{g}_{\boldsymbol{y}}(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}^{(\ell)}, t_{k+1})^{-1} \boldsymbol{g}(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}^{(\ell)}, t_{k+1})$$

set $\boldsymbol{y}_{k+1} = \boldsymbol{y}_{k+1}^{(L)}$ with L accepted value.

21.2 Index 2 DAE projection algorithm

An index-2 differential-algebraic equation (DAE) can be written as:

$$\begin{cases} \boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) \\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},t) \end{cases}$$

where:

- $\boldsymbol{x}$ and $\boldsymbol{y}$ are the differential and algebraic variables;
- $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)$ is the right-hand side of the differential equations;
- $\boldsymbol{g}(\boldsymbol{x},t)$ encodes the algebraic constraints.

21.2.1 Constraint Differentiation

To regularize the system, differentiate the constraint and introduce a dummy derivative for $\boldsymbol{y}$:

$$\begin{cases} \boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{y}(t),t) \\ \boldsymbol{y}'(t) = \boldsymbol{y}_{\text{dot}}(t) \\ \quad \boldsymbol{0} = \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},t)\boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)\boldsymbol{y}_{\text{dot}} \\ \quad \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) \\ \quad \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},t) \end{cases}$$

where

- First hidden constraint

$$\begin{aligned} \boldsymbol{0} &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(\boldsymbol{x},t) \\ &= \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},t)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{g}_t(\boldsymbol{x},t) \\ &= \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) \end{aligned}$$

- Second hidden constraint

$$\begin{aligned} \boldsymbol{0} &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) \\ &= \underbrace{\boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{h}_t(\boldsymbol{x},\boldsymbol{y},t)}_{\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)} + \underbrace{\boldsymbol{h}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)}_{\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},t)\boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)}\ \boldsymbol{y}_{\text{dot}} \end{aligned}$$

Solving for $\boldsymbol{y}_{\text{dot}}$ gives the regularized system:

$$\begin{cases} \boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) \\ \boldsymbol{y}'(t) = -\left[\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x},t)\boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)\right]^{-1}\boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t) \\ \quad \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},t) \\ \quad \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{x},t) \end{cases}$$

an ODE with hidden constraints.

21.2.2 Numerical Solution

The system is solved using a numerical method (e.g., Euler or Runge-Kutta) followed by a projection step to enforce the constraints.

Explicit Euler Step Example

1. **Advance solution**

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t\, \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{y}_k, t_k) \\ \boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \Delta t \left[\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}_k, t_k) \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x}_k, \boldsymbol{y}_k, t_k) \right]^{-1} \boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x}_k, \boldsymbol{y}_k, t_k) \end{cases}$$

2. **Project onto Constraints:**
   Solve the constrained minimization problem:

$$\text{minimize} \qquad \frac{1}{2} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1}\|^2 + \frac{1}{2} \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1}\|^2$$

$$\text{subject to} \qquad \begin{cases} \boldsymbol{g}(\boldsymbol{x}, t_{k+1}) = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1}) = \boldsymbol{0} \end{cases}$$

   This is solved using Lagrange multipliers and linearization.

Projection algorithm

Using Lagrange multiplier:

$$\begin{aligned} \mathscr{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2} \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1}\|^2 + \frac{1}{2} \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1}\|^2 \\ - \boldsymbol{\lambda} \cdot \boldsymbol{g}(\boldsymbol{x}, t_{k+1}) - \boldsymbol{\mu} \cdot \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1}) \end{aligned}$$

or the solution of the non linear system:

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{x}} = \boldsymbol{0}, \quad \frac{\partial \mathscr{L}}{\partial \boldsymbol{y}} = \boldsymbol{0}, \quad \frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}} = \boldsymbol{0}, \quad \frac{\partial \mathscr{L}}{\partial \boldsymbol{\mu}} = \boldsymbol{0}.$$

that written explicitly is

$$\begin{cases} \boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}, t_{k+1})^T \boldsymbol{\lambda} - \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1})^T \boldsymbol{\mu} = \boldsymbol{0} \\ \boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{h}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1})^T \boldsymbol{\mu} = \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{x}, t_{k+1}) = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1}) = \boldsymbol{0} \end{cases}$$

after linearization do until convergence ($\ell = 0, 1, \ldots$) the following iterations

1. Initialization $\ell \leftarrow 0$

$$\begin{aligned} \boldsymbol{x}^{(\ell)} &= \widetilde{\boldsymbol{x}}_{k+1}, \\ \boldsymbol{y}^{(\ell)} &= \widetilde{\boldsymbol{y}}_{k+1} \end{aligned}$$

2. Solve

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{g}_{\boldsymbol{x}}^T & -\boldsymbol{h}_{\boldsymbol{x}}^T \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{h}_{\boldsymbol{y}}^T \\ -\boldsymbol{g}_{\boldsymbol{x}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{h}_{\boldsymbol{x}} & -\boldsymbol{h}_{\boldsymbol{y}} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{x} \\ \delta \boldsymbol{y} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^{(\ell)} \\ \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{y}^{(\ell)} \\ \boldsymbol{g}(\boldsymbol{x}^{(\ell)}, t_{k+1}) \\ \boldsymbol{h}(\boldsymbol{x}^{(\ell)}, \boldsymbol{y}^{(\ell)}, t_{k+1}) \end{pmatrix}$$

3. Advance:

$$\begin{aligned} \boldsymbol{x}^{(\ell+1)} &= \boldsymbol{x}^{(\ell)} + \delta \boldsymbol{x}, \\ \boldsymbol{y}^{(\ell+1)} &= \boldsymbol{y}^{(\ell)} + \delta \boldsymbol{y}, \end{aligned}$$

4. $\ell \leftarrow \ell + 1$ and repeat step 2 and 3 until convergence.

Approximate Projection algorithm

To speed up computation we can project separately position and velocity solving two consecutive project:

- **x projection:**

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1}\|^2 \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}, t_{k+1}) = \boldsymbol{0} \end{aligned}$$

- **y projection:**

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1}\|^2 \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}_{k+1}, \boldsymbol{y}, t_{k+1}) = \boldsymbol{0} \end{aligned}$$

Using Lagrange multiplier we obtain the two nonlinear system to be solved

- **x problem:**

$$\begin{cases} \boldsymbol{x} - \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{x}, t_{k+1})^T \boldsymbol{\lambda} = \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{x}, t_{k+1}) = \boldsymbol{0} \end{cases}$$

1. Initialization $\ell \leftarrow 0$

$$\boldsymbol{x}^{(\ell)} = \widetilde{\boldsymbol{x}}_{k+1}$$

2. Solve

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{g}_{\boldsymbol{x}}^T \\ -\boldsymbol{g}_{\boldsymbol{x}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{x} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}^{(\ell)} \\ \boldsymbol{g}(\boldsymbol{x}^{(\ell)}, t_{k+1}) \end{pmatrix}$$

3. Advance:

$$\boldsymbol{x}^{(\ell+1)} = \boldsymbol{x}^{(\ell)} + \delta\boldsymbol{x}$$

4. $\ell \leftarrow \ell + 1$ and repeat step 2 and 3 until convergence.

- **y problem:**

$$\begin{cases} \boldsymbol{y} - \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{h}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1})^T \boldsymbol{\mu} = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1}) = \boldsymbol{0} \end{cases}$$

1. Initialization $\ell \leftarrow 0$

$$\boldsymbol{y}^{(\ell)} = \widetilde{\boldsymbol{y}}_{k+1}$$

2. Solve

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{h}_{\boldsymbol{y}}^T \\ -\boldsymbol{h}_{\boldsymbol{y}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{y} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{y}}_{k+1} - \boldsymbol{y}^{(\ell)} \\ \boldsymbol{h}(\boldsymbol{x}^{(\ell)}, \boldsymbol{y}^{(\ell)}, t_{k+1}) \end{pmatrix}$$

3. Advance:

$$\boldsymbol{y}^{(\ell+1)} = \boldsymbol{y}^{(\ell)} + \delta\boldsymbol{y},$$

4. $\ell \leftarrow \ell + 1$ and repeat step 2 and 3 until convergence.

21.3 Index 3 DAE projection algorithm

An **Index-3** DAE typically arises in mechanical systems with constraints, such as:

$$\begin{cases} \boldsymbol{p}' = \boldsymbol{v} \\ \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v}' + \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T\boldsymbol{\lambda} = \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ \boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{0} \end{cases}$$

where:

- $\boldsymbol{p}$ and $\boldsymbol{v}$ are the position and velocity variables,
- $\boldsymbol{M}$ is the mass matrix,
- $\boldsymbol{\Phi}$ represents the constraints,
- $\boldsymbol{\lambda}$ are the Lagrange multipliers.

To handle the higher-index constraints, we introduce dummy variables and differentiate the constraints

$$\begin{cases} \boldsymbol{p}'(t) = \boldsymbol{v}(t), \\ \boldsymbol{v}'(t) = \boldsymbol{v}_{\text{dot}} \\ \\ \quad \text{Index one part (solve for } \boldsymbol{v}_{\text{dot}} \text{ and } \boldsymbol{\lambda}) \\ \boldsymbol{0} = \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v}_{\text{dot}} + \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T\boldsymbol{\lambda} - \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ \boldsymbol{0} = \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)\boldsymbol{v}_{\text{dot}} + \boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) \\ \\ \quad \text{Hidden constraints (projection for } \boldsymbol{p} \text{ and } \boldsymbol{v}) \\ \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{p},\boldsymbol{v},t), \\ \boldsymbol{0} = \boldsymbol{\Phi}(\boldsymbol{p},t) \end{cases}$$

where

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{p},\boldsymbol{v},t) &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(\boldsymbol{p},t) \\ &= \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{x},t)\boldsymbol{v} + \boldsymbol{\Phi}_t(\boldsymbol{p},t) \end{aligned}$$

$$\boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) = \boldsymbol{h}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v} + \boldsymbol{h}_t(\boldsymbol{p},\boldsymbol{v},t)$$

By solving for $\boldsymbol{v}_{\text{dot}}$ and $\boldsymbol{\lambda}$ we obtain the reduced ODE system:

$$\begin{cases} \boldsymbol{p}' = \boldsymbol{v} \\ \boldsymbol{v}' = \boldsymbol{v}_{\text{dot}}(\boldsymbol{p},\boldsymbol{v},t) \end{cases}$$

where $\boldsymbol{v}_{\text{dot}}$ is obtained from $\boldsymbol{p}$ and $\boldsymbol{v}$ by solving the linear system:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ -\boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) \end{pmatrix}$$

The invariants

$$\begin{cases} \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{p},\boldsymbol{v},t) \\ \boldsymbol{0} = \boldsymbol{\Phi}(\boldsymbol{p},t) \end{cases}$$

are used in the projection step.

Index-3 Advancing phase

Projection method advance the ODE using any numerical method (Euler, RK, ...) and next project onto the constraints (red and blue part).

E.g. using Explicit Euler step:

$$\begin{cases} \boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \Delta t\, \boldsymbol{v}_k \\ \boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \Delta t\, (\boldsymbol{v}_{\text{dot}})_k \end{cases}$$

where $\boldsymbol{v}_{\text{dot}}$ is obtained from $\boldsymbol{p}_k$ and $\boldsymbol{v}_k$ by solving the linear system:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p}_k, \boldsymbol{v}_k, t_k) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p}_k, t_k)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p}_k, t_k) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{p}_k, \boldsymbol{v}_k, t_k) \\ -\boldsymbol{a}(\boldsymbol{p}_k, \boldsymbol{v}_k, t_k) \end{pmatrix}$$

Index-3 Projection phase

The projection phase solve the problem

$$\text{minimize} \qquad \frac{1}{2}\|\boldsymbol{p} - \widetilde{\boldsymbol{p}}_{k+1}\|^2 + \frac{1}{2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}_{k+1}\|^2$$

$$\text{subject to} \qquad \begin{cases} \boldsymbol{\Phi}(\boldsymbol{p}, t_{k+1}) = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{p}, \boldsymbol{v}, t_{k+1}) = \boldsymbol{0} \end{cases}$$

using Lagrange multiplier:

$$\mathscr{L}(\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{p} - \widetilde{\boldsymbol{p}}_{k+1}\|^2 + \frac{1}{2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}_{k+1}\|^2$$

$$- \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\boldsymbol{x}, t_{k+1}) - \boldsymbol{\mu} \cdot \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, t_{k+1})$$

or the solution of the non linear system:

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{p}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{v}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{\mu}} = \boldsymbol{0}.$$

Use Newton (like) iterations to solve the nonlinear system. Set

$$\boldsymbol{p}_{k+1} = \widetilde{\boldsymbol{p}}_{k+1} + \delta \boldsymbol{p} \quad \text{and} \quad \boldsymbol{v}_{k+1} = \widetilde{\boldsymbol{v}}_{k+1} + \delta \boldsymbol{v}$$

after linearization neglecting high order term the Newton like iterative method become:

1. Initialization $\ell \leftarrow 1$

$$\boldsymbol{p}^{(\ell)} = \widetilde{\boldsymbol{p}}_{k+1} \quad \text{and} \quad \boldsymbol{v}^{(\ell)} = \widetilde{\boldsymbol{v}}_{k+1}$$

2. Solve:

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{\Phi}_{\boldsymbol{p}}^T & -\boldsymbol{h}_{\boldsymbol{p}}^T \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{h}_{\boldsymbol{v}}^T \\ -\boldsymbol{\Phi}_{\boldsymbol{p}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{h}_{\boldsymbol{p}} & -\boldsymbol{h}_{\boldsymbol{v}} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{p} \\ \delta \boldsymbol{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{p}}_{k+1} - \boldsymbol{p}^{(\ell)} \\ \widetilde{\boldsymbol{v}}_{k+1} - \boldsymbol{v}^{(\ell)} \\ \boldsymbol{\Phi}(\boldsymbol{p}^{(\ell)}, t_{k+1}) \\ \boldsymbol{h}(\boldsymbol{p}^{(\ell)}, \boldsymbol{v}^{(\ell)}, t_{k+1}) \end{pmatrix}$$

3. Update:

$$\begin{aligned} \boldsymbol{p}^{(\ell+1)} &= \boldsymbol{p}^{(\ell)} + \delta \boldsymbol{p}, \\ \boldsymbol{v}^{(\ell+1)} &= \boldsymbol{v}^{(\ell)} + \delta \boldsymbol{v} \end{aligned}$$

4. Repeat step 2 and 3 until convergence.

Index 3 DAE approximate projection algorithm

Projection phase: position

$$\begin{aligned} &\text{minimize} && \frac{1}{2}||\boldsymbol{p}-\widetilde{\boldsymbol{p}}_{k+1}||^2 \\ &\text{subject to} && \boldsymbol{\Phi}(\boldsymbol{p},t_{k+1})=\boldsymbol{0} \end{aligned}$$

using Lagrange multiplier:

$$\mathscr{L}(\boldsymbol{p},\boldsymbol{\lambda})=\frac{1}{2}||\boldsymbol{p}-\widetilde{\boldsymbol{p}}_{k+1}||^2-\boldsymbol{\lambda}\cdot\boldsymbol{\Phi}(\boldsymbol{p},t_{k+1})$$

or find the solution of the non linear system:

$$\frac{\partial\mathscr{L}}{\partial\boldsymbol{p}}=\boldsymbol{0},\quad \frac{\partial\mathscr{L}}{\partial\boldsymbol{\lambda}}=\boldsymbol{0}.$$

Projection phase: velocity

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{v}-\widetilde{\boldsymbol{v}}_{k+1}\|^2 \\ &\text{subject to} && \boldsymbol{h}(\boldsymbol{p}_{k+1},\boldsymbol{v},t_{k+1})=\boldsymbol{0} \end{aligned}$$

using Lagrange multiplier:

$$\mathscr{L}(\boldsymbol{v},\boldsymbol{\mu})=\frac{1}{2}|\boldsymbol{v}-\widetilde{\boldsymbol{v}}_{k+1}||^2-\boldsymbol{\mu}\cdot\boldsymbol{h}(\boldsymbol{p}_{k+1},\boldsymbol{v},t_{k+1})$$

or the solution of the non linear system:

$$\frac{\partial\mathscr{L}}{\partial\boldsymbol{v}}=\boldsymbol{0},\qquad \frac{\partial\mathscr{L}}{\partial\boldsymbol{\mu}}=\boldsymbol{0}.$$

$$\boldsymbol{p}^{(0)}=\widetilde{\boldsymbol{p}}_{k+1},\qquad \boldsymbol{p}^{(\ell+1)}=\boldsymbol{p}^{(\ell)}+\delta\boldsymbol{p},$$

$$\boldsymbol{v}^{(0)}=\widetilde{\boldsymbol{v}}_{k+1},\qquad \boldsymbol{v}^{(\ell+1)}=\boldsymbol{v}^{(\ell)}+\delta\boldsymbol{v},$$

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{\Phi}_{\boldsymbol{p}}^T \\ -\boldsymbol{\Phi}_{\boldsymbol{p}} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \delta\boldsymbol{p} \\ \boldsymbol{\lambda} \end{pmatrix}=\begin{pmatrix} \widetilde{\boldsymbol{p}}_{k+1}-\boldsymbol{p}^{(\ell)} \\ \boldsymbol{\Phi}(\boldsymbol{p}^{(\ell)},t_{k+1}) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{h}_{\boldsymbol{v}}^T \\ -\boldsymbol{h}_{\boldsymbol{v}} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \delta\boldsymbol{v} \\ \boldsymbol{\mu} \end{pmatrix}=\begin{pmatrix} \widetilde{\boldsymbol{v}}_{k+1}-\boldsymbol{v}^{(\ell)} \\ \boldsymbol{h}(\boldsymbol{p}_{k+1},\boldsymbol{v}^{(\ell)},t_{k+1}) \end{pmatrix}$$

where $\boldsymbol{h}_{\boldsymbol{v}}=\boldsymbol{h}_{\boldsymbol{v}}(\boldsymbol{p}_{k+1},\boldsymbol{v}^{(\ell)},t_{k+1})$ and $\boldsymbol{\Phi}_{\boldsymbol{p}}=\boldsymbol{\Phi}(\boldsymbol{p}^{(\ell)},t_{k+1})$.



22. Projection with implicit numerical scheme

22.1 Index-1 DAE

Consider a semi-explicit DAE (with index > 1):

$$\begin{cases} \boldsymbol{z}' = \boldsymbol{f}(\boldsymbol{z},\boldsymbol{w},t) \\ \boldsymbol{0} = \boldsymbol{g}(\boldsymbol{z},\boldsymbol{w},t) \end{cases}$$

This system can be transformed into an index-1 DAE by index reduction, along with invariants:

$$\text{Index-1 DAE: } \begin{cases} \boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t) \\ \boldsymbol{0} = \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t) \end{cases} \qquad \underbrace{\boldsymbol{b}(\boldsymbol{x},\boldsymbol{y},t) = \boldsymbol{0}}_{\text{invariants}}$$

The index-1 system is then solved as an implicitly defined ODE:

$$\boldsymbol{x}' = \boldsymbol{F}(\boldsymbol{x},t)$$

where $\boldsymbol{F}(\boldsymbol{x},t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t)$ and $\boldsymbol{y}(\boldsymbol{x},t)$ is the solution to the equation $\boldsymbol{b}(\boldsymbol{x},\boldsymbol{y},t) = \boldsymbol{0}$.

22.1.1 Advancing the ODE: Explicit Methods

Explicit methods, such as the explicit Euler method, do not require complex handling. The update step is given by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \boldsymbol{F}(\boldsymbol{x}_k, t_k)$$

The evaluation of $\boldsymbol{F}(\boldsymbol{x}_k, t_k)$ proceeds as follows:

1. Solve $\boldsymbol{h}(\boldsymbol{y}) = \boldsymbol{a}(\boldsymbol{x}_k, \boldsymbol{y}, t_k) = \boldsymbol{0}$ to compute $\boldsymbol{y}_k$.
2. Compute $\boldsymbol{F}(\boldsymbol{x}_k, t_k) = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{y}_k, t_k)$.

22.1.2 Advancing the ODE: Implicit Methods

For implicit methods, such as implicit Euler, the update step is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \boldsymbol{F}(\boldsymbol{x}_{k+1}, t_{k+1})$$

This requires solving the implicit equation:

$$\boldsymbol{G}(\boldsymbol{z}) = \boldsymbol{z} - \Delta t \boldsymbol{F}(\boldsymbol{z}, t_{k+1}) - \boldsymbol{x}_k = \boldsymbol{0}$$

The solution of $\boldsymbol{G}(\boldsymbol{z}) = 0$ requires an iterative method such as Newton-Raphson:

$$\boldsymbol{z}^{(\ell+1)} = \boldsymbol{z}^{(\ell)} - \left(\frac{\partial \boldsymbol{G}(\boldsymbol{z}^{(\ell)})}{\partial \boldsymbol{z}} \right)^{-1} \boldsymbol{G}(\boldsymbol{z}^{(\ell)})$$

This method requires the Jacobian of $\boldsymbol{F}(\boldsymbol{x},t)$:

$$\frac{\partial \boldsymbol{G}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \boldsymbol{I} - \Delta t \frac{\partial \boldsymbol{F}(\boldsymbol{z}, t_{k+1})}{\partial \boldsymbol{z}}$$

22.1.3 RHS Jacobian

Given the system

$$\boldsymbol{F}(\boldsymbol{x},t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t), \qquad \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t) = \boldsymbol{0},$$

we can derive the following expressions for the Jacobian:

$$\begin{cases} \dfrac{\partial \boldsymbol{F}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \dfrac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{x}} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}} \dfrac{\partial \boldsymbol{y}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \\ \\ \boldsymbol{0} = \dfrac{\partial \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{x}} + \dfrac{\partial \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y},t)}{\partial \boldsymbol{y}} \dfrac{\partial \boldsymbol{y}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \end{cases}$$

From this, we can express the derivative of $\boldsymbol{y}(\boldsymbol{x},t)$ with respect to $\boldsymbol{x}$ as:

$$\frac{\partial \boldsymbol{y}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = -\boldsymbol{a}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)^{-1} \boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t),$$

and the Jacobian of $\boldsymbol{F}(\boldsymbol{x},t)$ becomes:

$$\boxed{\frac{\partial \boldsymbol{F}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t) - \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t) \boldsymbol{a}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)^{-1} \boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)}$$

22.1.4 Practical Computation of the Jacobian

In practice, the Jacobian should be computed as follows:

1. Start with the system:

$$\boldsymbol{F}(\boldsymbol{x},t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t), \qquad \boldsymbol{a}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x},t),t) = \boldsymbol{0}.$$

2. Solve for $\boldsymbol{Z}$ by solving the linear system:

$$\boldsymbol{a}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)\boldsymbol{Z} = -\boldsymbol{a}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t)$$

3. Compute the Jacobian:

$$\frac{\partial \boldsymbol{F}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y},t) + \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{y},t)\boldsymbol{Z}$$

22.2 Index-3 DAE Special Case

We begin with the following system of differential-algebraic equations (DAEs):

$$\begin{cases} \boldsymbol{p}' = \boldsymbol{v} \\ \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v}' + \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \boldsymbol{\lambda} = \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ \boldsymbol{\Phi}(\boldsymbol{p},t) = \boldsymbol{0} \end{cases}$$

Transformation to ODE with Implicit RHS and Invariants

We aim to transform this system into a set of ordinary differential equations (ODEs) with implicit right-hand sides (RHS) and invariants:

$$\begin{cases} \boldsymbol{p}' = \boldsymbol{v} \\ \boldsymbol{v}' = \boldsymbol{v}_{\text{dot}}(\boldsymbol{p},\boldsymbol{v},t) \end{cases} \qquad \begin{cases} \mathbf{0} = \boldsymbol{h}(\boldsymbol{p},\boldsymbol{v},t) = \dfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(\boldsymbol{p},t) \\ \mathbf{0} = \boldsymbol{\Phi}(\boldsymbol{p},t) \end{cases}$$

Here, $\boldsymbol{v}_{\text{dot}}$ is derived from the system by solving the following linear system:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ -\boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) \end{pmatrix}$$

Additionally, we define:

$$\boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) = \boldsymbol{h}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v} + \boldsymbol{h}_t(\boldsymbol{p},\boldsymbol{v},t)$$

Defining Functions for RHS and Invariants

Let us introduce the following functions to represent the system:

- **Function F:**

$$\boldsymbol{F}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) = \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t)\boldsymbol{v}_{\text{dot}} + \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T\boldsymbol{\lambda} - \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t)$$

- **Function G:**

$$\boldsymbol{G}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) = \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)\boldsymbol{v}_{\text{dot}} + \boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t)$$

Thus, the system can be expressed as:

$$\boldsymbol{F}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}}(\boldsymbol{p},\boldsymbol{v},t),\boldsymbol{\lambda}(\boldsymbol{p},\boldsymbol{v},t),t) = \mathbf{0}$$

$$\boldsymbol{G}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}}(\boldsymbol{p},\boldsymbol{v},t),t) = \mathbf{0}$$

Jacobian Derivatives of Functions F and G

Taking derivatives of $\boldsymbol{F}$ and $\boldsymbol{G}$ with respect to $\boldsymbol{p}$, we obtain the following system of equations:

$$\boldsymbol{F}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) + \boldsymbol{F}_{\boldsymbol{v}_{\text{dot}}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t)\frac{\partial \boldsymbol{v}_{\text{dot}}}{\partial \boldsymbol{p}} + \boldsymbol{F}_{\boldsymbol{\lambda}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t)\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{p}} = \mathbf{0}$$

$$\boldsymbol{G}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) + \boldsymbol{G}_{\boldsymbol{v}_{\text{dot}}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t)\frac{\partial \boldsymbol{v}_{\text{dot}}}{\partial \boldsymbol{p}} = \mathbf{0}$$

Expressions for the Jacobians of Functions F and G

From the definition of the functions, we have the following expressions for the Jacobian derivatives:

$$\boldsymbol{F}_{\boldsymbol{v}_{\text{dot}}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) = \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t)$$

$$\boldsymbol{F}_{\boldsymbol{\lambda}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) = \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T$$

$$\boldsymbol{G}_{\boldsymbol{v}_{\text{dot}}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) = \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)$$

Final System of Equations for the Jacobians

The system of Jacobian equations can now be written as:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \boldsymbol{0} \end{pmatrix} \frac{\partial}{\partial \boldsymbol{p}} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{F}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) \\ -\boldsymbol{G}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) \end{pmatrix}$$

Similarly, for derivatives with respect to $\boldsymbol{v}$:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \boldsymbol{0} \end{pmatrix} \frac{\partial}{\partial \boldsymbol{v}} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{F}_{\boldsymbol{v}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) \\ -\boldsymbol{G}_{\boldsymbol{v}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) \end{pmatrix}$$

Final Transformed Index-3 DAE

Thus, the transformed system of index-3 DAEs is:

$$\begin{cases} \boldsymbol{p}' = \boldsymbol{v} \\ \boldsymbol{v}' = \boldsymbol{v}_{\text{dot}}(\boldsymbol{p},\boldsymbol{v},t) \end{cases} \qquad \begin{cases} \boldsymbol{0} = \boldsymbol{h}(\boldsymbol{p},\boldsymbol{v},t) = \dfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(\boldsymbol{p},t) \\ \boldsymbol{0} = \boldsymbol{\Phi}(\boldsymbol{p},t) \end{cases}$$

The RHS and its Jacobians are obtained from:

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{p},\boldsymbol{v},t) \\ -\boldsymbol{a}(\boldsymbol{p},\boldsymbol{v},t) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{M}(\boldsymbol{p},\boldsymbol{v},t) & \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t)^T \\ \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{p},t) & \boldsymbol{0} \end{pmatrix} \frac{\partial}{\partial \boldsymbol{p}} \begin{pmatrix} \boldsymbol{v}_{\text{dot}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{F}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},\boldsymbol{\lambda},t) \\ -\boldsymbol{G}_{\boldsymbol{p}}(\boldsymbol{p},\boldsymbol{v},\boldsymbol{v}_{\text{dot}},t) \end{pmatrix}$$



23. Coordinate Partitioning for Pendulum in cartesian coordinates

23.1 Step 1 choose independent coordinates

Consider redundant coordinate pendulum DAE

$$\begin{aligned} u' &= \mu x \\ v' &= \mu y - g \\ x' &= u \\ y' &= v \\ x^2 + y^2 &= r^2 \end{aligned}$$

Solve y as a function of x (use negative branch)

$$y(x) = -\sqrt{r^2 - x^2}$$

constraint can be eliminated from the DAE by introducing $y \equiv y(x)$.

23.2 Step 2a propagate constraints

$$\begin{aligned} u' &= \mu x \\ v' &= \mu y(x) - g \\ x' &= u \\ y' &= v \\ y(x) &= -\sqrt{r^2 - x^2} \end{aligned}$$

Deriving constraint $x^2 + y^2 - r^2 = 0$ derivative of $y'(x)$ is deduced

$$x\dot{x} + y\dot{y} = 0 \quad \Longrightarrow \quad \dot{y} = -x\dot{x}/y \quad \Longrightarrow \quad v = -xu/y$$

so that

$$\dot{y} = v(x,u) = -\frac{xu}{y(x)}$$

23.3 Step 2b propagate constraints

Substitute $\dot{y}$ with $v(x,u)$

$$\begin{aligned} u' &= \mu x \\ v' &= \mu y(x) - g \\ x' &= u \\ v(x,u) &= -\frac{xu}{y(x)} \\ y(x) &= -\sqrt{r^2 - x^2} \end{aligned}$$

23.4 Step 2c propagate constraints

Deriving two time the constraint $x^2 + y^2 - r^2 = 0$ derivative $\ddot{y} = \dot{v}$ is deduced:

$$xu' + y\dot{v} + x'u + y'v = 0 \quad \Longrightarrow$$

$$\mu(x^2 + y^2) - gy + (u^2 + v^2) = 0$$

so that μ can be written as a function of x and u

$$\mu(x,u) = \frac{gy(x) - (u^2 + v(x,u)^2)}{r^2}$$

thus $\mu = \mu(x,u)$ and

$$v'(x,u) = \mu(x,u)y(x) - g$$

can be eliminated as redundant algebraic equation.

23.5 Step 2d propagate constraints

Substitute $\mu = \mu(x,u)$

$$\begin{aligned} u' &= \mu(x,u)x \\ x' &= u \\ \mu(x,u) &= \frac{gy(x) - (u^2 + v(x,u)^2)}{r^2} \\ v(x,u) &= -\frac{xu}{y(x)} \\ y(x) &= -\sqrt{r^2 - x^2} \end{aligned}$$

in practice an ODE with 2 equations and $\mu(x,u)$ computed recursively.

23.6 Collatz

Using Collatz explicit integrator

$$\begin{aligned} u_{k+1/2} &= u_k + (h/2)\mu(x_k, u_k)x_k \\ x_{k+1/2} &= x_k + (h/2)u_k \end{aligned}$$

$$\begin{aligned} u_{k+1} &= u_k + h\mu(x_{k+1/2}, u_{k+1/2})x_{k+1/2} \\ x_{k+1} &= x_k + h u_{k+1/2} \end{aligned}$$

where $\mu(x,u)$

$$y_k = -\sqrt{r^2 - x_k^2}$$
$$v_k = -\frac{x_k u_k}{y_k}$$
$$\mu(x_k,u_k) = \frac{g y_k - (u_k^2 + v_k{}^2)}{r^2}$$

$T_{\max} = 10$, $x(0) = 0.9$, $u(0) = 0$. The remaining value $y(0) = \sqrt{r^2 - x(0)^2}$ and $v(0) = -x(0)u(0)/y(0)$.

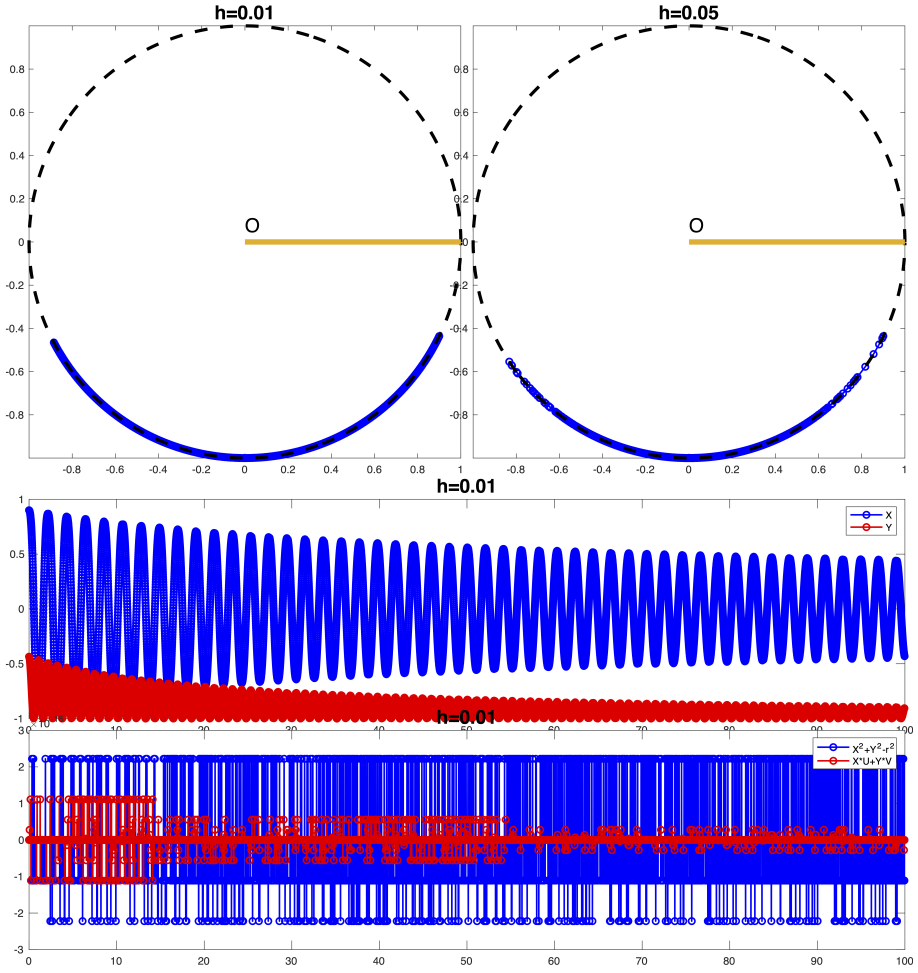


Figure 23.1: Trajectory and constraint violation:

23.7 Runge-Kutta 4th order

$$K_1 = hu_k \qquad L_1 = h\mu\left(x_k,u_k\right)x_k$$
$$K_2 = h\left(u_k + \frac{L_1}{2}\right) \qquad L_2 = h\mu\left(x_k + \frac{K_1}{2}, u_k + \frac{L_1}{2}\right)\left(x_k + \frac{K_1}{2}\right)$$
$$K_3 = h\left(u_k + \frac{L_2}{2}\right) \qquad L_3 = h\mu\left(x_k + \frac{K_2}{2}, u_k + \frac{L_2}{2}\right)\left(x_k + \frac{K_2}{2}\right)$$
$$K_4 = h\left(u_k + L_3\right) \qquad L_4 = h\mu\left(x_k + K_3, u_k + L_3\right)\left(x_k + K_3\right)$$

$$x_{k+1} = x_k + \frac{K_1 + 2K_2 + 2K_3 + K_4}{6}$$
$$u_{k+1} = u_k + \frac{L_1 + 2L_2 + 2L_3 + L_4}{6}$$

$T_{\max} = 10$, $x(0) = 0.9$, $u(0) = 0$. The remaining value $y(0) = \sqrt{r^2 - x(0)^2}$ and $v(0) = -x(0)u(0)/y(0)$.

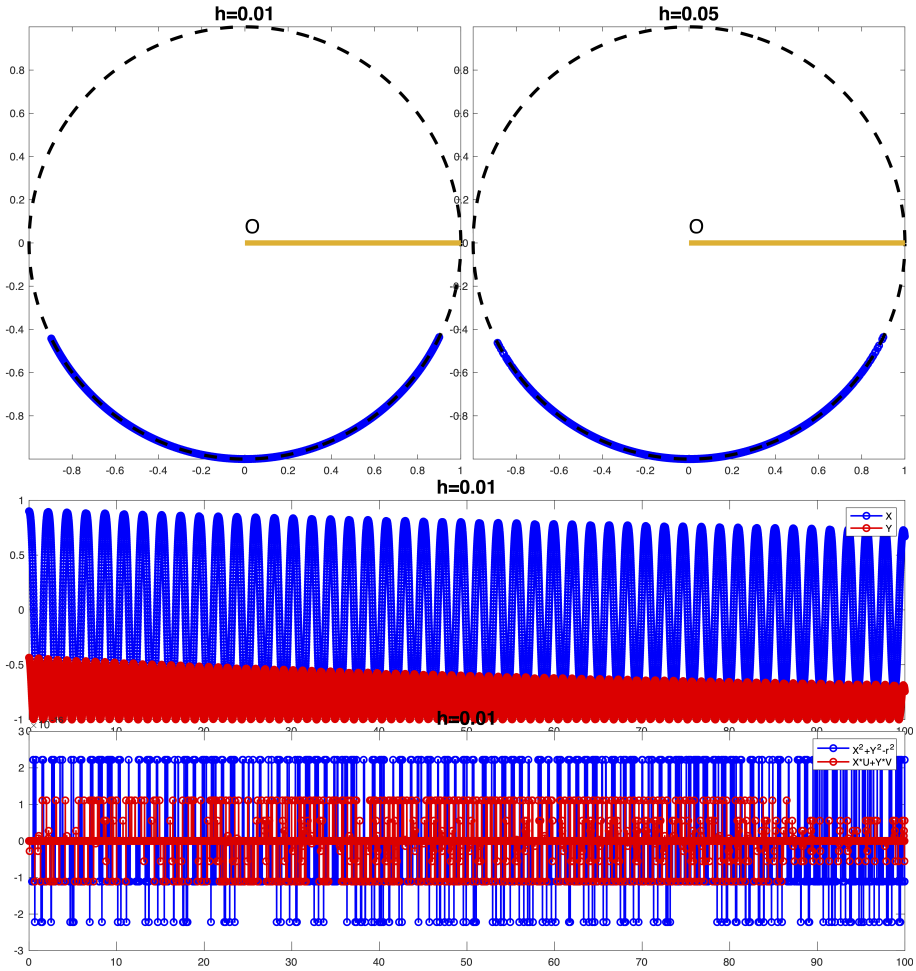


Figure 23.2: Trajectory and constraint violation:

VII Examples

24 Example 1 211
24.1 Index reduction 211
24.2 Index reduction using dummy variables 215

25 Example 2 221
25.1 3D Pendulum 221
25.2 Index reduction 222
25.3 Index reduction using dummy variable 224
25.4 Stabilization of the DAE System via Hidden Constraints 225

26 Example 3 229
26.1 Index reduction 229
26.2 Index reduction using dummy variable 234

27 Example 4 237
27.1 Double Pendulum 237
27.2 Index reduction 239
27.3 Index reduction using dummy variable 244

28 Example 5 251
28.1 A Constrained Van der Pol System and a Dummy-Variable Reformulation 251
28.2 Formulation as a Stiff DAE 251
28.3 Baumgarte Stabilization 253
28.4 Numerical Examples 254

29 Example 6 255
29.1 Introduction 255
29.2 Baumgarte Stabilization 256

30 Example 7 259
30.1 Circuit Description 259
30.2 Baumgarte Stabilization 261
30.3 Simulation Results 262

31 Example 8 263
31.1 The DAE 263

32 Example 9 273
32.1 Double Pendulum System with Connecting Bar 274

References 281

In this section, we present a concrete example of how to reduce the index of a differential-algebraic equation (DAE) system and solve it numerically. High-index DAEs often cannot be integrated directly due to hidden constraints or inconsistent derivatives, so index reduction is an essential preprocessing step.

We illustrate the key steps: identifying the structure of the system, performing symbolic or numerical differentiation of constraints if necessary, and rewriting the system in a form suitable for time integration. The example highlights typical issues such as constraint consistency and numerical drift, and demonstrates how basic stabilization or projection methods can help maintain the validity of the solution.

How to use these examples

For each example, first identify the variables and constraints, then verify the required consistent initial data before following the reduction. Finally, use the plotted constraint residuals to compare an unstabilized integration with a stabilized or projected one.



24. Example 1

Consider the DAE

$$\begin{cases} x'(t) = u(t) \\ y'(t) = v(t) \\ 2u'(t) + v'(t) + 2t\,x(t)\,\lambda(t) = t \\ u'(t) + 2v'(t) - \lambda(t) = 1 - t \\ t\,x(t)^2 - y(t) = 1 \end{cases}$$

define

$$\boldsymbol{Z} = \begin{pmatrix} x \\ y \\ u \\ v \\ \lambda \end{pmatrix}, \quad \boldsymbol{E}(\boldsymbol{Z},t) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{G}(\boldsymbol{Z},t) = \begin{pmatrix} u \\ v \\ t - 2t\,x\,\lambda \\ 1 - t + \lambda \\ 1 + y - t\,x^2 \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}(\boldsymbol{Z},t)\,\boldsymbol{Z}'(t) = \boldsymbol{G}(\boldsymbol{Z},t)$$

24.1 Index reduction

The table

$$\boldsymbol{T} = \left[\; \boldsymbol{E}(\boldsymbol{Z},t) \;\middle|\; \boldsymbol{G}(\boldsymbol{Z},t) \;\right] = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 2 & 1 & 0 & t - 2t\,x\,\lambda \\ 0 & 0 & 1 & 2 & 0 & 1 - t + \lambda \\ 0 & 0 & 0 & 0 & 0 & 1 + y - t\,x^2 \end{array}\right]$$

is used to memorize the operation done on the DAE. The reduction is done by using linear algebra and avoid explicit construction of LU decomposition.

24.1.1 First reduction of the index by one

1. Exchange row 3 and 4 by using

$$\boldsymbol{P}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_1 = \boldsymbol{P}_1\boldsymbol{T} = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 2 & 0 & 1-t+\lambda \\ 0 & 0 & 2 & 1 & 0 & t-2tx\lambda \\ 0 & 0 & 0 & 0 & 0 & 1+y-tx^2 \end{array}\right]$$

2. Subtract 2 times row 3 to row 4

$$\boldsymbol{L}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_2 = \boldsymbol{L}_2\boldsymbol{T}_1 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 2 & 0 & 1-t+\lambda \\ 0 & 0 & 0 & -3 & 0 & 3t-2(\lambda(1+tx)+1) \\ \hline 0 & 0 & 0 & 0 & 0 & 1+y-tx^2 \end{array}\right]$$

3. Scale 4 the row by $-1/3$

$$\boldsymbol{T}_2 = \boldsymbol{L}_2\boldsymbol{T}_1 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 2 & 0 & 1-t+\lambda \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 0 & 0 & 0 & 0 & 0 & 1+y-tx^2 \end{array}\right]$$

Now algebraic constraint is separated and is

$$[\text{alg1}] = 1+y-tx^2 = 0$$

$$\Longrightarrow \quad [\text{alg2}] = \frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = y' - x^2 - 2txx' = 0$$

apply derivation to the algebraic constraint (reduce the index by one)

$$\boldsymbol{T}_3 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 2 & 0 & 1-t+\lambda \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline -2tx & 1 & 0 & 0 & 0 & x^2 \end{array}\right]$$

24.1.2 Second reduction of the index by one

1. Eliminate $-2tx$ and 1 of the last rpw by multiplying

$$\boldsymbol{L}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 2tx & -1 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_4 = \boldsymbol{L}_3\boldsymbol{T}_3 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 2 & 0 & 1-t+\lambda \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 0 & 0 & 0 & 0 & 0 & x^2+2txu-v \end{array}\right]$$

2. Eliminate also 2 my multiplying

$$\boldsymbol{U}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_5 = \boldsymbol{U}_4\boldsymbol{T}_4 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t-\frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 0 & 0 & 0 & 0 & 0 & x^2+2txu-v \end{array}\right]$$

The algebraic constraint is

$$[\text{alg2}] = x^2+2txu-v = 0$$

$$\Longrightarrow \quad [\text{alg3}] = \frac{\mathrm{d}[\text{alg2}]}{\mathrm{d}t} = 2(x+tu)x'+2xu+2txu'-v' = 0$$

apply derivation to the algebraic constraint (reduce the index by one)

$$\boldsymbol{T}_6 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t-\frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 2(x+tu) & 0 & 2tx & -1 & 0 & -2xu \end{array}\right]$$

24.1.3 Third reduction of the index by one

1. Apply elimination for the last row

$$\boldsymbol{L}_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -2(x+tu) & 0 & -2tx & 1 & 1 \end{pmatrix}$$

so that

$$\boldsymbol{T}_7 = \boldsymbol{L}_6\boldsymbol{T}_6 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t-\frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 0 & 0 & 0 & 0 & 0 & \frac{(8\lambda x^2-6x)t^2}{3}+\frac{((4\lambda+2)x-6u^2-3)t}{3}-4xu+\frac{2\lambda}{3}+\frac{2}{3} \end{array}\right]$$

2. Scale last row by $3/2$

$$\boldsymbol{S}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{3}{2} \end{pmatrix}$$

so that

$$\boldsymbol{T}_8 = \boldsymbol{S}_7\boldsymbol{T}_7 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t-\frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline 0 & 0 & 0 & 0 & 0 & [\text{alg3}] \end{array}\right]$$

where

$$[\text{alg3}] = \left(4x^2\lambda - 3x\right)t^2 + \left((2\lambda+1)x - 3u^2 - \frac{3}{2}\right)t - 6xu + \lambda + 1$$

The algebraic constraint [alg3] is derived and become

$$\frac{\mathrm{d}[\text{alg3}]}{\mathrm{d}t} = (A)x' + (B)u' + (C)\lambda' + (D), \qquad \begin{cases} (A) = (8x\lambda - 3)t^2 + (2\lambda+1)t - 6u \\ (B) = -6(tu+x) \\ (C) = (4xt+2)xt + 1 \\ (D) = 8tx^2\lambda + (1-6t+2\lambda)x - 3u^2 - \dfrac{3}{2} \end{cases}$$

reducing the index by one

$$\boldsymbol{T}_9 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t-\frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1)-t \\ \hline (A) & 0 & (B) & 0 & (C) & (D) \end{array}\right]$$

24.1.4 DAE reduced to an ODE

$(D) \neq 0$ so that DAE is reduced to an ODE.

1. Eliminating in the last row and scaling

$$\boldsymbol{T}_9 = \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & u \\ 0 & 1 & 0 & 0 & 0 & v \\ 0 & 0 & 1 & 0 & 0 & t - \frac{1}{3}(1+\lambda(1+4tx)) \\ 0 & 0 & 0 & 1 & 0 & \frac{2}{3}(\lambda(1+tx)+1) - t \\ \hline 0 & 0 & 0 & 0 & 1 & (E)/(C) \end{array}\right]$$

where

$$\begin{aligned}(E) &= (D) - x\cdot(A) - u\cdot(B) \\ &= ((16x\lambda - 9)t + 4\lambda + 3)tu + 16x^2\lambda t + (4\lambda + 3 - 12t)x - \frac{3}{2} - 9u^2\end{aligned}$$

The DAE reduced to ODE become:

$$\begin{cases} x' = u \\ y' = v \\ u' = t - \dfrac{1}{3}(1+\lambda(1+4tx)) \\ v' = \dfrac{2}{3}(\lambda(1+tx)+1) - t \\ \lambda' = \dfrac{((16x\lambda - 9)t + 4\lambda + 3)tu + 16x^2\lambda t + (4\lambda + 3 - 12t)x - \frac{3}{2} - 9u^2}{(4xt+2)xt + 1} \end{cases}$$

The **differential index** is 3 and the invariants are

$$\begin{aligned}[\text{alg1}] &= 1 + y - tx^2 = 0 \\ [\text{alg2}] &= x^2 + 2txu - v = 0 \\ [\text{alg3}] &= \left(4x^2\lambda - 3x\right)t^2 + \left((2\lambda+1)x - 3u^2 - \frac{3}{2}\right)t - 6xu + \lambda + 1 = 0\end{aligned}$$

notice that

$$\frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = [\text{alg2}]$$

but

$$\frac{\mathrm{d}[\text{alg2}]}{\mathrm{d}t} \neq [\text{alg3}]$$

and Baumgarte stabilization cannot be used.

24.2 Index reduction using dummy variables

This DAE system exhibits mixed differential and algebraic characteristics. To analyze its structure, we first convert it to semi-explicit form.

Introduce dummy variables:

- $u'(t) = u_{\text{dot}}(t)$
- $v'(t) = v_{\text{dot}}(t)$

The system decomposes into differential (blue) and algebraic (red) components:

$$\begin{cases} x' = u \\ y' = v \\ u' = u_{\text{dot}} \\ v' = v_{\text{dot}} \end{cases} \oplus \begin{cases} 0 = 2u_{\text{dot}} + v_{\text{dot}} + 2tx\lambda - t & \text{(Constraint 1)} \\ 0 = u_{\text{dot}} + 2v_{\text{dot}} - \lambda - 1 + t & \text{(Constraint 2)} \\ 0 = tx^2 - y - 1 & \text{(Constraint 3)} \end{cases}$$

where blue equations are the ODE part explicited and the red part are the algebraic constraints. If the algebraic constraints determine a solution for $u_{\text{dot}}(t)$, $v_{\text{dot}}(t)$ and λ the DAE is of index 1 and we can stop here because we can write (formally) the solution of the algebraic part as

- $u_{\text{dot}} = u_{\text{dot}}(x, y, u, v)$
- $v_{\text{dot}} = v_{\text{dot}}(x, y, u, v)$
- $\lambda = \lambda(x, y, u, v)$

24.2.1 Variable classification

The variable are classified, in practice the variable that appears with a derivative are differential variable, the other are algebraic:

- **Differential variables**: x, y, u, v
- **Algebraic variables**: λ, u_{dot}, v_{dot}

Notice that algebraic variables appears linearly in the DAE.

24.2.2 Index Reduction Procedure

We attempt to solve for u_{dot} and v_{dot}, but due to the presence of λ, the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Now we must derivate the constraints in order to reduce the index. Here we want to differentiate **only** the algebraic equation that can reduce the index and do not introduce new differential equation. Here we want to avoid to introduce derivative of the algebraic variable unless is strictly necessary.

We construct a dependency table to identify which constraints to differentiate:

	u_{dot}	v_{dot}	λ
(Constraint 1)	1	1	1
(Constraint 2)	1	1	1
(Constraint 3)	0	0	0

a 0 means the the function do not depend on the variable on the column. A 1 means dependence. (Constraint 3) is the optimal candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus (Constraint 3) and substitute known variable in differetial form using the ODE part

$$(\text{Constraint 3})' = x^2 + 2txx' - y' = x^2 + 2txu - v$$

the new equation (Constraint 3)$'$ **is algebraic** and thus is used to replace (Constraint 3). The DAE thus become:

$$\begin{cases} x' = u \\ y' = v \\ u' = u_{\text{dot}} \\ v' = v_{\text{dot}} \end{cases} \oplus \begin{cases} 0 = 2u_{\text{dot}} + v_{\text{dot}} + 2tx\lambda - t & \text{(Constraint 1)} \\ 0 = u_{\text{dot}} + 2v_{\text{dot}} - \lambda - 1 + t & \text{(Constraint 2)} \\ 0 = x^2 + 2txu - v & \text{(Constraint 3)}' \\ \\ 0 = t\,x^2 - y - 1 & \text{(Constraint 3)} \end{cases}$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine the algebraic variable and must be differentiated again.

Step 2: Differentiate (again) the algebraic constraints

To find the right cadidate(s) here we write a table of the dependencies

	u_{dot}	v_{dot}	λ
(Constraint 1)	1	1	1
(Constraint 2)	1	1	1
(Constraint 3)$'$	0	0	0

(Constraint 3)$'$ is the right candidate because it do not depend on algebraic variable and its derivative do not introduce extra differential equation. Differentiate thus (Constraint 3)$'$ and substitute known variable in differential form using the ODE part

$$\begin{aligned} \text{(Constraint 3)}'' &= 2xx' + 2xu + 2tx'u + 2txu' - v' \\ &= 4xu + 2tu^2 + 2txu_{\text{dot}} - v_{\text{dot}} \end{aligned}$$

the new equation (Constraint 3)$''$ **is algebraic** and thus is used to replace (Constraint 3)$'$. The DAE thus become:

$$\begin{cases} x' = u \\ y' = v \\ u' = u_{\text{dot}} \\ v' = v_{\text{dot}} \end{cases} \oplus \begin{cases} 0 = 2u_{\text{dot}} + v_{\text{dot}} + 2tx\lambda - t & \text{(Constraint 1)} \\ 0 = u_{\text{dot}} + 2v_{\text{dot}} - \lambda - 1 + t & \text{(Constraint 2)} \\ 0 = 4xu + 2tu^2 + 2txu_{\text{dot}} - v_{\text{dot}} & \text{(Constraint 3)}'' \\ \\ 0 = t\,x^2 - y - 1 & \text{(Constraint 3)} \\ 0 = x^2 + 2txu - v & \text{(Constraint 3)}' \end{cases}$$

Te red part is a linear system in the algebraic variable:

$$\begin{pmatrix} 2 & 1 & 2tx \\ 1 & 2 & -1 \\ 2tx & -1 & 0 \end{pmatrix} \begin{pmatrix} u_{\text{dot}} \\ v_{\text{dot}} \\ \lambda \end{pmatrix} = \begin{pmatrix} t \\ 1-t \\ -2u(2x+tu) \end{pmatrix}$$

Check if solvable

$$\begin{aligned} \begin{vmatrix} 2 & 1 & 2tx \\ 1 & 2 & -1 \\ 2tx & -1 & 0 \end{vmatrix} &= 2tx \begin{vmatrix} 1 & 2tx \\ 2 & -1 \end{vmatrix} + \begin{vmatrix} 2 & 2tx \\ 1 & -1 \end{vmatrix} \\ &= 2tx(-1-4tx) - 2 - 2tx \\ &= -4tx - 8(tx)^2 - 2 \end{aligned}$$

Let $z = tx$ then the roots of the polynomial $-2-4z-8z^2$ are

$$z_{1,2} = \frac{-1-\imath\sqrt{3}}{4}$$

that are complex, thus the linear system is always solvable for tx real. Thus blue part plus red part constitute a DAE of index one. We have done two differentiation and this implies that the original DAE is of **index 3**.

24.2.3 DAE reduced to ODE

The index-1 DAE reduced to ODE become

$$\begin{cases} x' = u \\ y' = v \\ u' = u_{\text{dot}}(t,x,y,u,v) \\ v' = v_{\text{dot}}(t,x,y,u,v) \end{cases}$$

where $u_{\text{dot}}(t,x,y,u,v)$ and $v_{\text{dot}}(t,x,y,u,v)$ are the solution of the linear system

$$\begin{pmatrix} 2 & 1 & 2tx \\ 1 & 2 & -1 \\ 2tx & -1 & 0 \end{pmatrix} \begin{pmatrix} u_{\text{dot}}(t,x,y,u,v) \\ v_{\text{dot}}(t,x,y,u,v) \\ \lambda(t,x,y,u,v) \end{pmatrix} = \begin{pmatrix} t \\ 1-t \\ -2u(2x+tu) \end{pmatrix} \tag{24.1}$$

there are also two hidden constraints

$$\begin{aligned} (\text{Constraint 3}) &= t\,x^2 - y - 1 = 0, \\ (\text{Constraint 3})' &= x^2 + 2txu - v = 0. \end{aligned}$$

24.2.4 Baumgarte stabilization

For this DAE the hidden constraint can be used to stabilize the algebric constraint: (Constraint 3):

$$(\text{Constraint 3})'' + 2\eta\omega(\text{Constraint 3})' + \omega^2(\text{Constraint 3}) = 0$$

that become

$$4xu + 2tu^2 + 2txu_{\text{dot}} - v_{\text{dot}} + 2\eta\omega(x^2 + 2txu - v) + \omega^2(t\,x^2 - y - 1) = 0$$

using this stabilized constraint in the linear system Equation 24.1 become

$$\begin{pmatrix} 2 & 1 & 2tx \\ 1 & 2 & -1 \\ 2tx & -1 & 0 \end{pmatrix} \begin{pmatrix} u_{\text{dot}}(t,x,y,u,v) \\ v_{\text{dot}}(t,x,y,u,v) \\ \lambda(t,x,y,u,v) \end{pmatrix} = \begin{pmatrix} t \\ 1-t \\ -2u(2x+tu) - (\star) \end{pmatrix} \tag{24.2}$$

$$(\star) = 2\eta\omega(x^2 + 2txu - v) + \omega^2(t\,x^2 - y - 1)$$

24.2.5 Jacobians

For implicit solvers we require derivative of algebraic variables. Let

$$\boldsymbol{A}(t,x) = \begin{pmatrix} 2 & 1 & 2tx \\ 1 & 2 & -1 \\ 2tx & -1 & 0 \end{pmatrix},$$

$$\boldsymbol{X}(t,x,y,u,v) = \begin{pmatrix} u_{\mathrm{dot}}(t,x,y,u,v) \\ v_{\mathrm{dot}}(t,x,y,u,v) \\ \lambda(t,x,y,u,v) \end{pmatrix},$$

$$\boldsymbol{B}(t,x,y,u,v) = \begin{pmatrix} t \\ 1-t \\ -2u(2x+tu) - 2\eta\,\omega(x^2+2txu-v) - \omega^2(t\,x^2-y-1) \end{pmatrix}$$

Deriving $\boldsymbol{A}(t,x)\boldsymbol{X}(t,x,y,u,v) = \boldsymbol{B}(t,x,y,u,v)$

$$\boldsymbol{A}(t,x)\partial_t\boldsymbol{X}(t,x,y,u,v) = \partial_t\boldsymbol{B}(t,x,y,u,v) - \partial_t\boldsymbol{A}(t,x)\boldsymbol{X}(t,x,y,u,v)$$

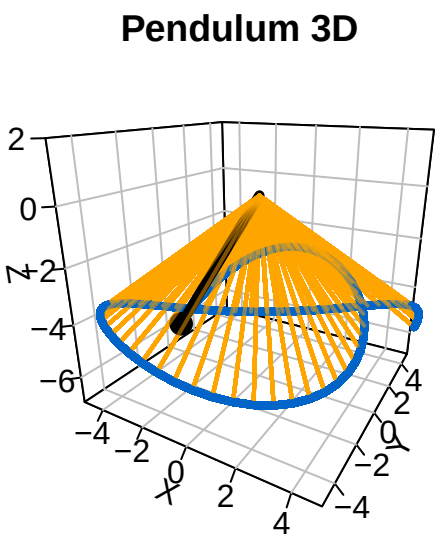
$$\boldsymbol{A}(t,x)\partial_t\boldsymbol{X}(t,x,y,u,v) = \begin{pmatrix} 1 \\ -1 \\ -2u^2 - 4xu\eta\,\omega - \omega^2x^2 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 2x \\ 0 & 0 & 0 \\ 2x & 0 & 0 \end{pmatrix}\boldsymbol{X}(t,x,y,u,v)$$

$$\partial_t\boldsymbol{X}(t,x,y,u,v) = \boldsymbol{A}(t,x)^{-1}\left(\begin{pmatrix} 1 \\ -1 \\ -2u^2 - 4xu\eta\,\omega - \omega^2x^2 \end{pmatrix} - 2x\begin{pmatrix} \lambda(t,x,y,u,v) \\ 0 \\ u_{\mathrm{dot}}(t,x,y,u,v) \end{pmatrix}\right)$$

analogously all the other derivatives can be obtained.

25. Example 2

25.1 3D Pendulum



Given the following definitions:

$$T = \frac{m}{2}(v_x^2 + v_y^2 + v_z^2) \qquad \text{(Kinetic energy)}$$

$$V = -mgz \qquad \text{(Potential energy)}$$

$$\Phi(x,y,z) = x^2 + y^2 + z^2 - \ell^2 = 0 \qquad \text{(Holonomic constraint)}$$

The constraint enforces that the pendulum mass lies on the surface of a sphere of radius ℓ, centered at the origin.

The corresponding system of Differential-Algebraic Equations (DAE) is:

$$\begin{cases} x'(t) = v_x(t) \\ y'(t) = v_y(t) \\ z'(t) = v_z(t) \\ m\,v_x'(t) = 2\,x(t)\,\lambda(t) \\ m\,v_y'(t) = 2\,y(t)\,\lambda(t) \\ m\,v_z'(t) = 2\,z(t)\,\lambda(t) - mg \\ x^2 + y^2 + z^2 = \ell^2 \end{cases}$$

This is a **DAE of index 3** due to the algebraic constraint $\Phi(x,y,z) = 0$, and the fact that the constraint must be differentiated **three times** to obtain an explicit expression for the highest derivative (λ').

We define

$$\boldsymbol{Z} = \begin{pmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ \lambda \end{pmatrix}, \quad \boldsymbol{E}(\boldsymbol{Z},t) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{G}(\boldsymbol{Z},t) = \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \\ \frac{2}{m}x(t)\,\lambda(t) \\ \frac{2}{m}y(t)\,\lambda(t) \\ \frac{2}{m}z(t)\,\lambda(t) - g \\ x^2 + y^2 + z^2 - \ell^2 \end{pmatrix}$$

so that the DAE can be written as

$$\boldsymbol{E}(\boldsymbol{Z},t)\,\boldsymbol{Z}'(t) = \boldsymbol{G}(\boldsymbol{Z},t)$$

25.2 Index reduction

The table

$$\boldsymbol{T} = \left[\begin{array}{c|c} \boldsymbol{E}(\boldsymbol{Z},t) & \boldsymbol{G}(\boldsymbol{Z},t) \end{array}\right] = \left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & v_x(t) \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & v_y(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & v_z(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{2}{m}x(t)\,\lambda(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \frac{2}{m}y(t)\,\lambda(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{2}{m}z(t)\,\lambda(t) - g \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x^2 + y^2 + z^2 - \ell^2 \end{array}\right]$$

is used to memorize the operation done on the DAE. Let index $\leftarrow 0$.

1. The algebraic constraint is

$$[\text{alg1}] = x^2 + y^2 + z^2 - \ell^2 = 0$$

2. Reduce by one the index by differentiating [alg1] (index $\leftarrow= 1$)

$$[\text{alg2}] = \frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = 2\left(xx' + yy' + zz'\right) = 2\left(xv_x + yv_y + zv_z\right)$$

3. Reduce by one the index by differentiating [alg2] (index $\leftarrow= 2$)

$$\begin{aligned}[\text{alg3}] = \frac{\mathrm{d}[\text{alg2}]}{\mathrm{d}t} &= 2\left(x'v_x + y'v_y + z'v_z\right) + 2\left(xv_x' + yv_y' + zv_z'\right) \\ &= 2\left(v_x^2 + v_y^2 + v_z^2\right) + \frac{4\lambda}{m}\left(x^2 + y^2 + z^2\right) - 2gz\end{aligned}$$

Now λ can be solved from $[\text{alg3}] = 0$

$$\lambda = \frac{m}{2}\frac{gz - \left(v_x^2 + v_y^2 + v_z^2\right)}{x^2 + y^2 + z^2}$$

DAE reduced to index 1 DAE

using this solution the index 1 DAE (after the reduction) can be reduced to an ODE by substituting λ as follows:

$$\begin{cases} x' = v_x \\ y' = v_y \\ z' = v_z \\ v_x' = x\left[\dfrac{gz - \left(v_x^2 + v_y^2 + v_z^2\right)}{x^2 + y^2 + z^2}\right] \\ v_y' = y\left[\dfrac{gz - \left(v_x^2 + v_y^2 + v_z^2\right)}{x^2 + y^2 + z^2}\right] \\ v_z' = z\left[\dfrac{gz - \left(v_x^2 + v_y^2 + v_z^2\right)}{x^2 + y^2 + z^2}\right] - mg \end{cases}$$

DAE reduced to ODE

If you want to include also λ in the reduced ODE derive $[\text{alg3}] = 0$ so that index $\leftarrow= 3$

$$\begin{aligned}\frac{\mathrm{d}[\text{alg3}]}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left[2\left(v_x^2 + v_y^2 + v_z^2\right) + \frac{4\lambda}{m}\left(x^2 + y^2 + z^2\right) - 2gz\right] \\ &= 4\left(v_xv_x' + v_yv_y' + v_zv_z'\right) + \frac{4\lambda'}{m}\left(x^2 + y^2 + z^2\right) \\ &+ \frac{8\lambda}{m}\left(xx' + yy' + zz'\right) - 2gz' \\ &= \frac{8\lambda}{m}\left(xv_x + yv_y + zv_z\right) - 4v_zg + \frac{4\lambda'}{m}\left(x^2 + y^2 + z^2\right) \\ &+ \frac{8\lambda}{m}\left(xv_x + yv_y + zv_z\right) - 2v_zg \\ &= \frac{16\lambda}{m}\left(xv_x + yv_y + zv_z\right) - 6v_zg + \frac{4\lambda'}{m}\left(x^2 + y^2 + z^2\right) = 0\end{aligned}$$

Solving for λ'

$$\lambda' = \frac{1}{2}\frac{3mv_zg - 8\lambda\left(xv_x + yv_y + zv_z\right)}{x^2 + y^2 + z^2}$$

The DAE reduced to the ODE become

$$\begin{cases} x' = v_x \\ y' = v_y \\ z' = v_z \\ v_x' = \dfrac{2}{m} x\lambda \\ v_y' = \dfrac{2}{m} y\lambda \\ v_z' = \dfrac{2}{m} z\lambda - g \\ \lambda' = \dfrac{1}{2} \dfrac{3 m v_z g - 8\lambda \left(x v_x + y v_y + z v_z\right)}{x^2 + y^2 + z^2} \end{cases}$$

with the hidden constraints

$$\text{[alg1]} \qquad x^2 + y^2 + z^2 - \ell^2 = 0$$

$$\text{[alg2]} \qquad 2\left(x v_x + y v_y + z v_z\right) = 0$$

$$\text{[alg3]} \qquad 2\left(v_x^2 + v_y^2 + v_z^2\right) + \frac{4\lambda}{m}\left(x^2 + y^2 + z^2\right) - 2gz = 0$$

25.3 Index reduction using dummy variable

Variable classification

- **Differential variables**: x, y, z, v_x, v_y, v_z
- **Algebraic variables**: λ

We attempt to solve for λ, the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Differentiate (Constraint 1) $= x^2 + y^2 + z^2 - \ell^2$:

$$\frac{\mathrm{d}(\text{Constraint 1})}{\mathrm{d}t} = 2xx' + 2yy' + 2zz' = 0 \tag{25.1}$$

Substitute known differential variables:

$$(\text{Constraint 2}) = 2xv_x + 2yv_y + 2zv_z = 0$$

Still no λ. We differentiate once more.

Step 2: Differentiate algebraic constraint

Differentiate Equation 25.1:

$$\frac{\mathrm{d}(\text{Constraint 2})}{\mathrm{d}t} = 2x'v_x + 2y'v_y + 2z'v_z + 2xv_x' + 2yv_y' + 2zv_z' = 0$$

Substitute known derivatives:

$$(\text{Constraint 3}) = 2(v_x^2+v_y^2+v_z^2)+\frac{4\lambda}{m}(x^2+y^2+z^2)-2zg=0$$

Now λ appears in the DAE. We can solve for λ

$$\lambda = \frac{m}{2}\cdot\frac{zg-(v_x^2+v_y^2+v_z^2)}{x^2+y^2+z^2}$$

now we can substitute λ in the DAE, but this means reduce the DAE by eliminating a variable. Better define

$$\widetilde{\lambda}(x,y,z,v_x,v_y,v_z) = \frac{m}{2}\cdot\frac{zg-(v_x^2+v_y^2+v_z^2)}{x^2+y^2+z^2}$$

and obtain the index 1 DAE

$$\begin{cases} x' = v_x \\ y' = v_y \\ z' = v_z \\ v_x' = \dfrac{2}{m}x\lambda \\ v_y' = \dfrac{2}{m}y\lambda \\ v_z' = \dfrac{2}{m}z\lambda - g \\ \lambda = \widetilde{\lambda}(x,y,z,v_x,v_y,v_z) \end{cases}$$

having done two differetiation to obtain an **index 1** DAE he original DAE was of **index 3**.

25.4 Stabilization of the DAE System via Hidden Constraints

At step 4, we obtained the expression for the Lagrange multiplier:

$$\lambda(x,y,z,v_x,v_y,v_z) = \frac{m}{2}\cdot\frac{gz-(v_x^2+v_y^2+v_z^2)}{x^2+y^2+z^2}.$$

When substituted into the DAE system:

$$\begin{cases} x' = v_x, \\ y' = v_y, \\ z' = v_z, \\ v_x' = \dfrac{2x}{m}\lambda, \\ v_y' = \dfrac{2y}{m}\lambda, \\ v_z' = \dfrac{2z}{m}\lambda - g, \end{cases}$$

this formally reduces the system to an ODE. However, **this does not resolve the numerical drift issue**, as the algebraic constraints are not inherently enforced.

Constraint Dynamics and Stabilization

The Lagrange multiplier λ is derived from the second hidden constraint $[\text{alg3}] = 0$. Defining,

$$\boldsymbol{X} = \begin{pmatrix} x \\ y \\ x \end{pmatrix}, \qquad \boldsymbol{V} = \begin{pmatrix} v_x \\ v_y \\ v_x \end{pmatrix}$$

- position-level constraint
$$[\text{alg1}] = \boldsymbol{X} \cdot \boldsymbol{X} - \ell^2 = 0$$
- velocity-level constraint
$$[\text{alg2}] = 2\boldsymbol{X} \cdot \boldsymbol{V} = 0$$
- acceleration-level constraint
$$[\text{alg3}] = 2\boldsymbol{V} \cdot \boldsymbol{V} + \frac{4\lambda}{m}\boldsymbol{X} \cdot \boldsymbol{X} - 2gz$$

By construction, these constraints satisfy:

$$\frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = [\text{alg2}], \quad \frac{\mathrm{d}^2[\text{alg1}]}{\mathrm{d}t^2} = \frac{\mathrm{d}[\text{alg2}]}{\mathrm{d}t} = [\text{alg3}].$$

Baumgarte Stabilization

To mitigate drift, we replace $[\text{alg3}] = 0$ with a stabilized ODE:

$$\frac{\mathrm{d}^2[\text{alg1}]}{\mathrm{d}t^2} + 2\eta\omega\frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} + \omega^2[\text{alg1}] = 0,$$

where η (damping ratio) and ω (natural frequency) are tuning parameters. Substituting the constraints:

$$[\text{alg3}] + 2\eta\omega[\text{alg2}] + \omega^2[\text{alg1}] = 0.$$

Expanding this yields:

$$2\boldsymbol{V} \cdot \boldsymbol{V} + \frac{4\lambda}{m}\boldsymbol{X} \cdot \boldsymbol{X} - 2gz + 2\eta\omega \cdot 2\boldsymbol{X} \cdot \boldsymbol{V} + \omega^2(\boldsymbol{X} \cdot \boldsymbol{X} - \ell^2) = 0.$$

Solving for λ

Isolate λ to obtain the stabilized multiplier:

$$\lambda = \frac{m}{4} \cdot \frac{2gz - 2(\boldsymbol{V} \cdot \boldsymbol{V}) - 4\eta\omega(\boldsymbol{X} \cdot \boldsymbol{V}) - \omega^2(\boldsymbol{X} \cdot \boldsymbol{X} - \ell^2)}{\boldsymbol{X} \cdot \boldsymbol{X}}$$

Key Observations:

1. **Stabilization Terms**:
   The terms in blue introduce feedback to correct drift:
   - $4\eta\omega(\boldsymbol{X} \cdot \boldsymbol{V})$ penalizes velocity-level deviations.
   - $\omega^2(\boldsymbol{X} \cdot \boldsymbol{X} - \ell^2)$ penalizes position-level deviations.
2. **Parameters**:
   - Choose $\omega \sim 1/\Delta t$ (time step) for critical damping.
   - Set $\eta = 1$ for optimal convergence (overdamping may slow dynamics).
3. **Numerical Behavior**:
   The modified λ ensures constraints are satisfied exponentially, even with numerical errors.

Final Stabilized ODE System

The stabilized system becomes:

$$\begin{cases} x' = v_x, \\ y' = v_y, \\ z' = v_z, \\ v_x' = \dfrac{2x}{m}\lambda_{\text{stab}}(x,y,z,v_x,v_y,v_z), \\ v_y' = \dfrac{2y}{m}\lambda_{\text{stab}}(x,y,z,v_x,v_y,v_z), \\ v_z' = \dfrac{2z}{m}\lambda_{\text{stab}}(x,y,z,v_x,v_y,v_z) - g, \end{cases}$$

where λ_{stab} includes Baumgarte terms:

$$\lambda_{\text{stab}}(x,y,z,v_x,v_y,v_z) = \frac{m}{4} \cdot \frac{2gz - 2(\boldsymbol{V}\cdot\boldsymbol{V}) - 4\eta\omega(\boldsymbol{X}\cdot\boldsymbol{V}) - \omega^2(\boldsymbol{X}\cdot\boldsymbol{X} - \ell^2)}{\boldsymbol{X}\cdot\boldsymbol{X}}$$

This eliminates drift while preserving physical fidelity.

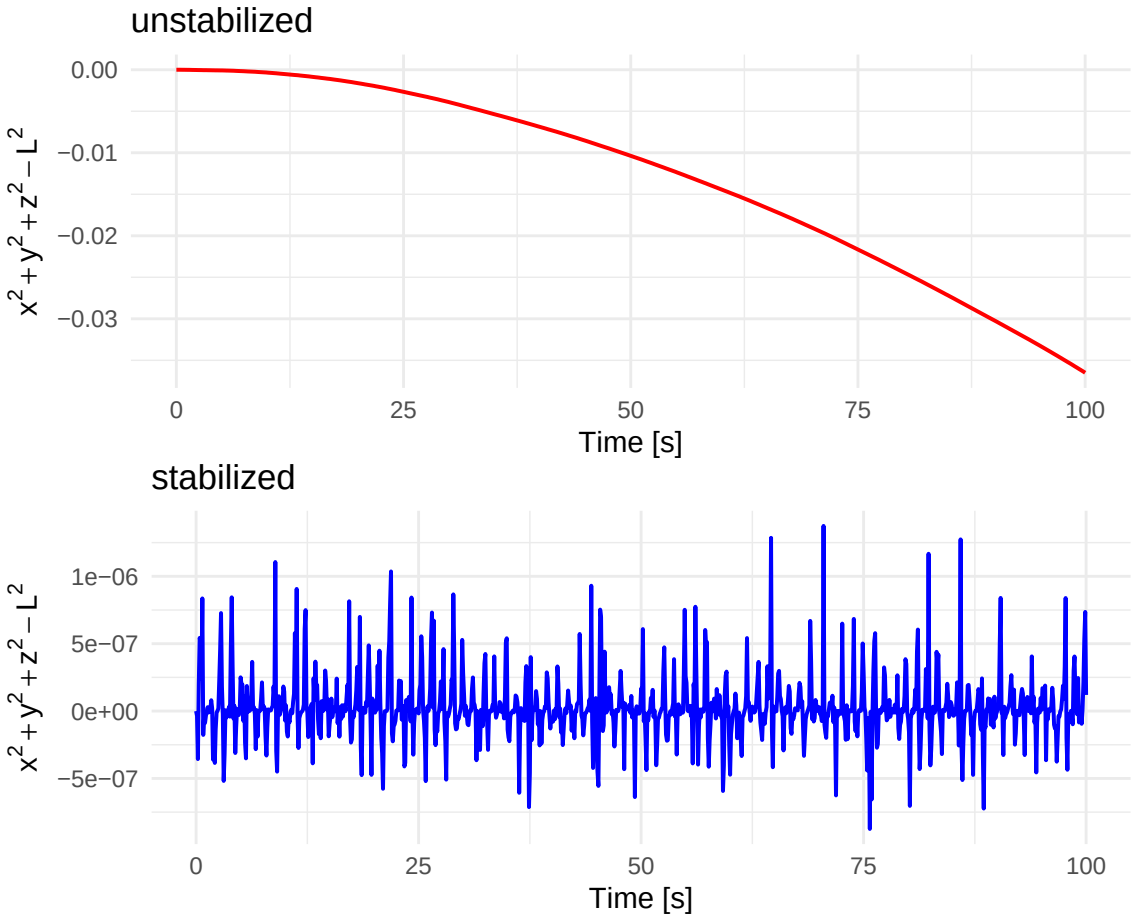
Advantages

- **No Drift**: Constraints are enforced at all levels (position, velocity, acceleration).
- **Tunable Accuracy**: Adjust η, ω to balance stability and numerical stiffness.

Limitations

- Introduces artificial damping if η or ω are too large.
- Requires careful parameter selection for stiff systems.

Implementation Note In practice, use: - $\omega = 10/\Delta t$ (for typical time steps Δt),
- $\eta = 0.7$ (underdamped to avoid overshooting).
This ensures rapid convergence without excessive distortion of the system dynamics.



Constraints satisfaction with and without stabilization.

Parameter	Value	Units	Description
Δt	0.1	s	Time step
ω	100	rad/s	Angular frequency
η	0.7	-	Damping coefficient
g	9.81	m/s^2	Gravitational acceleration
m	1	kg	Mass
$x(0) = L$	1	m	Initial position x
$y(0)$	0	m	Initial position y
$z(0)$	0	m	Initial position z
$v_x(0)$	0	m/s	Initial velocity x
$v_y(0)$	0	m/s	Initial velocity y
$v_z(0)$	0	m/s	Initial velocity z

Figure 25.1



26. Example 3

Consider the following DAE system:

$$\begin{cases} x_6'(t) + x_5(t) = \sin t \\ x_5'(t) + x_4(t) = \sin t \\ x_4'(t) + x_3(t) = \sin t \\ x_3'(t) + x_2 t) = \sin t \\ x_2'(t) + x_1(t) = \sin t \\ x_6(t) = \cos t \end{cases}$$

We define the variable vector and matrices:

$$\boldsymbol{Z} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}, \quad \boldsymbol{E}(\boldsymbol{Z},t) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{G}(\boldsymbol{Z},t) = \begin{pmatrix} \sin t - x_5(t) \\ \sin t - x_4(t) \\ \sin t - x_3(t) \\ \sin t - x_2(t) \\ \sin t - x_1(t) \\ \cos t - x_6(t) \end{pmatrix}$$

So that the DAE reads:

$$\boldsymbol{E}(\boldsymbol{Z},t)\,\boldsymbol{Z}'(t) = \boldsymbol{G}(\boldsymbol{Z},t)$$

26.1 Index reduction

The table

$$\boldsymbol{T} = \left[\begin{array}{c|c} \boldsymbol{E}(\boldsymbol{Z},t) & \boldsymbol{G}(\boldsymbol{Z},t) \end{array}\right] = \left[\begin{array}{cccccc|c} 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos t - x_6(t) \end{array}\right]$$

is used to memorize the operation done on the DAE.

26.1.1 First reduction of the index by one

Reduce index by one

1. Let

$$[\text{alg1}] = \cos t - x_6$$

2. Derive the constraint [alg1]

$$\frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = -\sin t - x_6'$$

3. rewrite the DAE

$$\boldsymbol{T}_1 = \left[\begin{array}{cccccc|c} 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & -\sin t \end{array}\right]$$

26.1.2 Second reduction of the index by one

1. Reorder the equation

$$\boldsymbol{T}_2 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & -\sin t \end{array}\right]$$

2. Subtract row 5 to row 6

$$\boldsymbol{T}_3 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & x_5(t) - 2\sin t \end{array}\right]$$

3. found algebraic constraint

$$[\text{alg2}] = x_5 - 2\sin t$$

Derive constraint [alg2]

$$\frac{\mathrm{d}[\mathrm{alg2}]}{\mathrm{d}t} = x_5' - 2\cos t$$

so that

$$\boldsymbol{T}_4 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & 2\cos t \end{array}\right]$$

26.1.3 Third reduction of the index by one

1. Subtract row 4 to row 6

$$\boldsymbol{T}_5 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & x_4(t) + 2\cos t - \sin t \end{array}\right]$$

2. found algebraic constraint

$$[\mathrm{alg3}] = x_4 + 2\cos t - \sin t$$

3. Reduce index by one by deriving the constraint [alg3]

$$\frac{\mathrm{d}[\mathrm{alg3}]}{\mathrm{d}t} = x_4' - \cos t - 2\sin t$$

so that

$$\boldsymbol{T}_6 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \cos t + 2\sin t \end{array}\right]$$

26.1.4 Fourth reduction of the index by one

1. Subtract row 3 to row 6

$$\boldsymbol{T}_7 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos t + \sin t + x_3(t) \end{array}\right]$$

found algebraic constraint

$$[\text{alg4}] = x_3 + \sin t + \cos t$$

2. Reduce index by one by deriving the constraint [alg4]

$$\frac{\mathrm{d}[\text{alg4}]}{\mathrm{d}t} = x_3' + \cos t - \sin t$$

so that

$$\boldsymbol{T}_8 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - \cos t \end{array}\right]$$

26.1.5 Fifth reduction of the index by one

1. Subtract row 2 to row 6

$$\boldsymbol{T}_9 = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & x_2(t) - \cos t \end{array}\right]$$

2. found algebraic constraint

$$[\text{alg5}] = x_2 - \cos t$$

3. Reduce index by one by deriving the constraint [alg5]

$$\frac{\mathrm{d}[\text{alg5}]}{\mathrm{d}t} = x_2' + \sin t$$

so that

$$\boldsymbol{T}_{10} = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 1 & 0 & 0 & 0 & 0 & -\sin t \end{array}\right]$$

26.1.6 Sixth reduction of the index by one

1. Subtract row 1 to row 6

$$\boldsymbol{T}_{11} = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1(t) - 2\sin t \end{array}\right]$$

2. found algebraic constraint

$$[\text{alg6}] = x_1 - 2\sin t$$

3. Reduce index y one by deriving the constraint [alg6]

$$\frac{\mathrm{d}[\text{alg6}]}{\mathrm{d}t} = x_1' - 2\cos t$$

so that

$$\boldsymbol{T}_{12} = \left[\begin{array}{cccccc|c} 0 & 1 & 0 & 0 & 0 & 0 & \sin t - x_1(t) \\ 0 & 0 & 1 & 0 & 0 & 0 & \sin t - x_2(t) \\ 0 & 0 & 0 & 1 & 0 & 0 & \sin t - x_3(t) \\ 0 & 0 & 0 & 0 & 1 & 0 & \sin t - x_4(t) \\ 0 & 0 & 0 & 0 & 0 & 1 & \sin t - x_5(t) \\ 1 & 0 & 0 & 0 & 0 & 0 & 2\cos t \end{array}\right]$$

26.1.7 DAE reduced to an ODE

Now matrix is non-singular ans the resulting ODE is

$$\begin{cases} x_1'(t) = 2\cos t \\ x_2'(t) = \sin t - x_1(t) \\ x_3'(t) = \sin t - x_2(t) \\ x_4'(t) = \sin t - x_3(t) \\ x_5'(t) = \sin t - x_4(t) \\ x_6'(t) = \sin t - x_5(t) \end{cases}$$

with invariants (or hidden constraints)

$$
\begin{aligned}
[\text{alg1}] &= \cos t - x_6 \\
[\text{alg2}] &= x_5 - 2\sin t \\
[\text{alg3}] &= x_4 + 2\cos t - \sin t \\
[\text{alg4}] &= x_3 + \sin t + \cos t \\
[\text{alg5}] &= x_2 - \cos t \\
[\text{alg6}] &= x_1 - 2\sin t
\end{aligned}
$$

Notice that from the invariants we can deduce

$$
\begin{aligned}
x_1(t) &= 2\sin t, \\
x_2(t) &= \cos t, \\
x_3(t) &= -\sin t - \cos t \\
x_4(t) &= \sin t - 2\cos t, \\
x_5(t) &= 2\sin t, \\
x_6(t) &= \cos t
\end{aligned}
$$

and the general solution in determined without fixing any initial value.

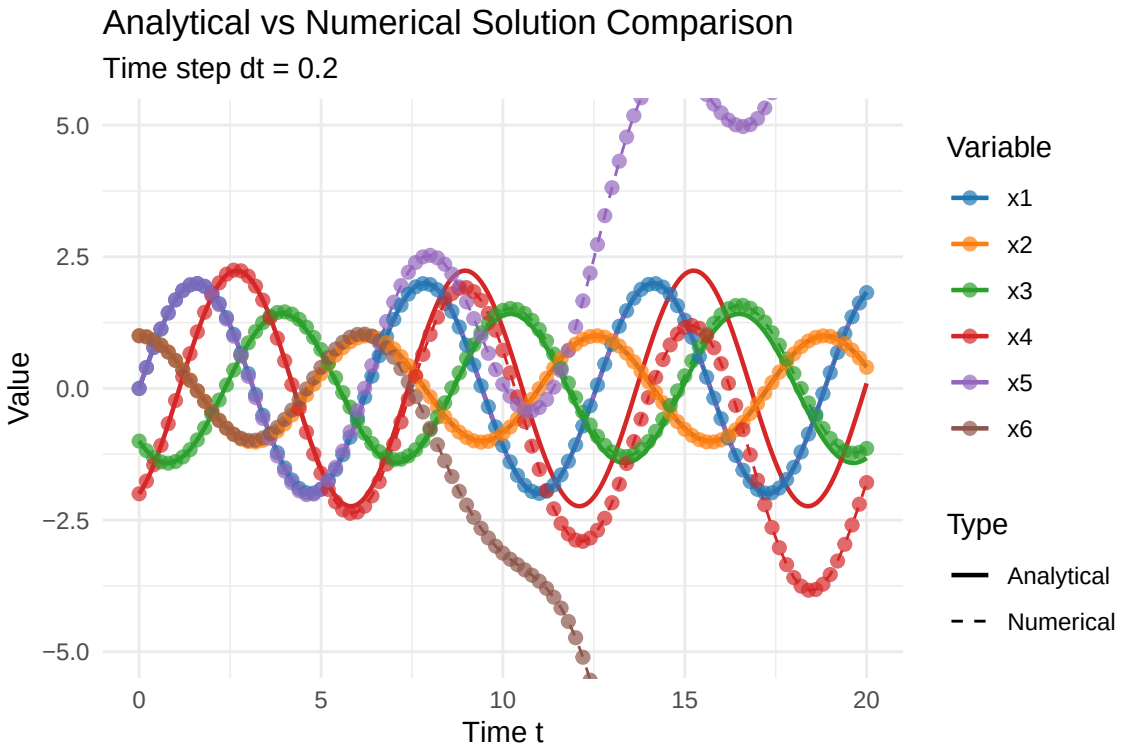


Figure 26.1: Numerical solution witb $\Delta t = 0.2$. Notice the drift for components x_6 and x_5.

26.2 Index reduction using dummy variable

Variable classification

- **Differential variables**: $x_1, x_2, x_3, x_4. x_5$
- **Algebraic variables**: x_6

We attempt to solve for u_6 and v_{dot}, but due to the presence of λ, the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Differentiate (Constraint 1) $= \cos t - x_6$:

$$\frac{\mathrm{d}(\text{Constraint 1})}{\mathrm{d}t} = -\sin t - x_6'$$

Substitute known differential variables:

$$(\text{Constraint 2}) = x_5 - 2\sin t$$

x_1' not found, continue index reduction.

Step 2: Differentiate algebraic constraints

Differentiate (Constraint 2)

$$\frac{\mathrm{d}(\text{Constraint 2})}{\mathrm{d}t} = x_5' - 2\cos t \quad \Longrightarrow \quad (\text{Constraint 3}) = \sin t - 2\cos t - x_4$$

x_1' not found, continue index reduction.

Step 3: Differentiate algebraic constraints

Differentiate (Constraint 3):

$$\frac{\mathrm{d}(\text{Constraint 3})}{\mathrm{d}t} = \cos t + 2\sin t - x_4' \quad \Longrightarrow \quad (\text{Constraint 4}) = \cos t + \sin t + x_3$$

x_1' not found, continue index reduction.

Step 4: Differentiate algebraic constraints

Differentiate (Constraint 4):

$$\frac{\mathrm{d}(\text{Constraint 4})}{\mathrm{d}t} = -\sin t + \cos t + x_3' \quad \Longrightarrow \quad (\text{Constraint 5}) = \cos t - x_2$$

x_1' not found, continue index reduction.

Step 5: Differentiate algebraic constraints

Differentiate (Constraint 5):

$$\frac{\mathrm{d}(\text{Constraint 5})}{\mathrm{d}t} = -\sin t - x_2' \quad \Longrightarrow \quad (\text{Constraint 6}) = -2\sin t + x_1$$

x_1' not found, continue index reduction.

Step 6: Differentiate algebraic constraints

Differentiate (Constraint 6):

$$\frac{\mathrm{d}(\text{Constraint 6})}{\mathrm{d}t} = -2\cos t - 2\sin t + x_1'$$

x_1' found, index reduction complete

$$\begin{cases} x_6' = \sin t - x_5 \\ x_5' = \sin t - x_4 \\ x_4' = \sin t - x_3 \\ x_3' = \sin t - x_2 \\ x_2' = \sin t - x_1 \\ x_1' = 2\sin t \end{cases} \quad \begin{cases} (\text{Constraint 1}) = \cos t - x_6 = 0 \\ (\text{Constraint 2}) = x_5 - 2\sin t = 0 \\ (\text{Constraint 3}) = \sin t - 2\cos t - x_4 = 0 \\ (\text{Constraint 4}) = \sin t + \cos t + x_3 = 0 \\ (\text{Constraint 5}) = \cos t - x_2 = 0 \\ (\text{Constraint 6}) = x_1 - 2\sin t = 0 \end{cases}$$

Applied 6 time differentiation, the **index is 6**.

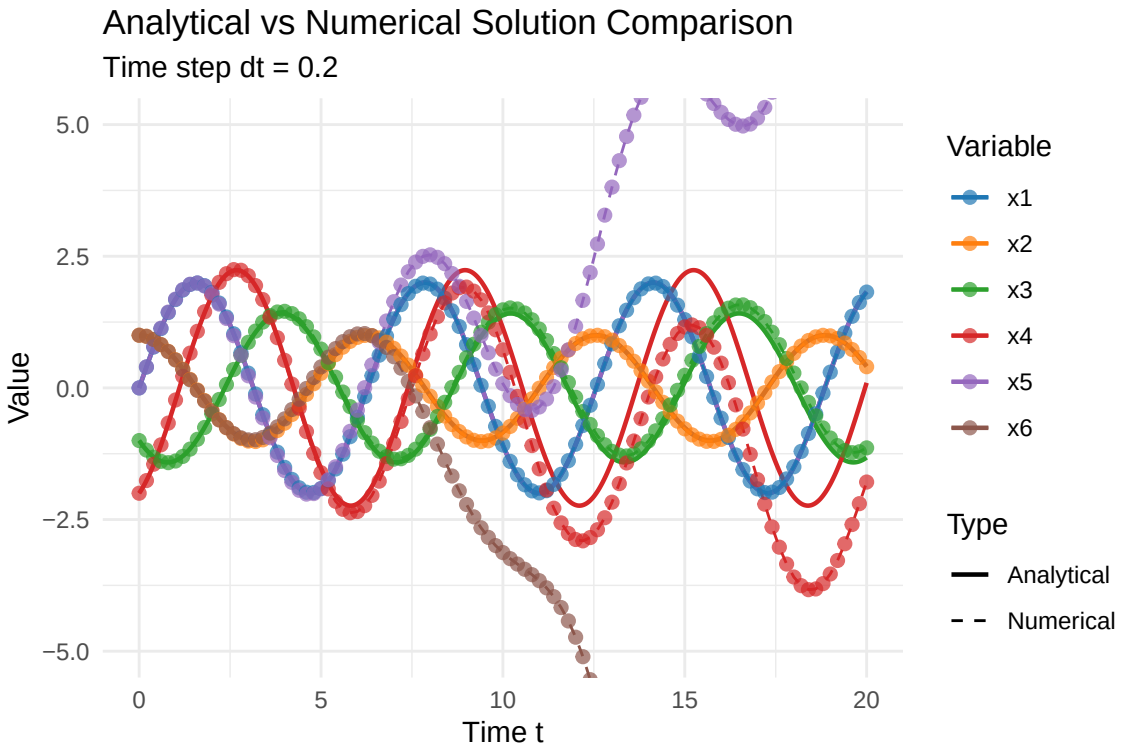


Figure 26.2: Numerical solution witb $\Delta t = 0.2$. Notice the drift for components x_6 and x_5.



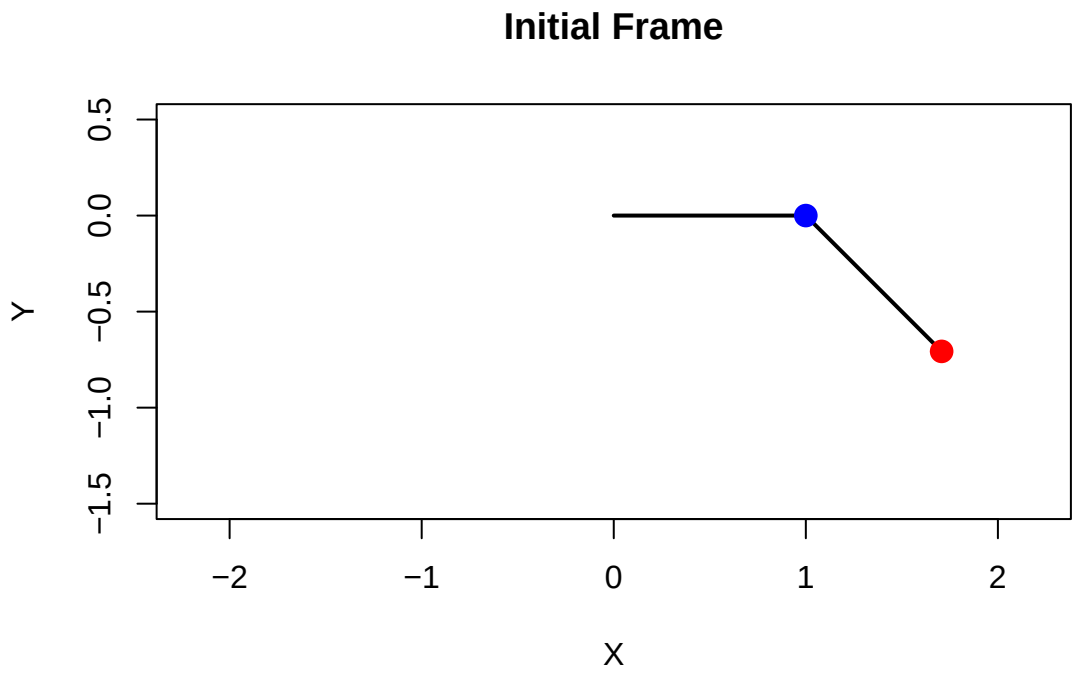
27. Example 4

27.1 Double Pendulum

We consider a constrained formulation of a planar double pendulum as a **differential-algebraic equation** (DAE) system:

$$
\begin{cases}
x_1' = u_1 \\
y_1' = v_1 \\
x_2' = u_2 \\
y_2' = v_2 \\
u_1' = \dfrac{x_1\lambda_1 + (x_1 - x_2)\lambda_2}{m_1} \\
v_1' = \dfrac{y_1\lambda_1 + (y_1 - y_2)\lambda_2}{m_1} - g \\
u_2' = \dfrac{(x_2 - x_1)\lambda_2}{m_2} \\
v_2' = \dfrac{(y_2 - y_1)\lambda_2}{m_2} - g \\
\ell_1^2 = x_1^2 + y_1^2 \\
\ell_2^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2
\end{cases}
$$

We define the variable vector and matrices:



Parameter	Value	Parameter	Value	Parameter	Value
m_1	5	m_2	1	ℓ_1	1
ℓ_2	1	g	9.81	θ_1	$\pi/2$
θ_2	$\pi/4$	ω_1	0	ω_2	-0.1

Figure 27.1: Double pendulum animation with the above parameters

$$\boldsymbol{E}(\boldsymbol{Z},t) = \begin{pmatrix} 1 & . & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . & . \\ . & . & . & 1 & . & . & . & . & . & . \\ . & . & . & . & 1 & . & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . & . \\ . & . & . & . & . & . & 1 & . & . & . \\ . & . & . & . & . & . & . & 1 & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \end{pmatrix}, \quad \boldsymbol{Z} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ u_1 \\ v_1 \\ u_2 \\ v_2 \\ \lambda_1 \\ \lambda_2 \end{pmatrix},$$

$$\boldsymbol{G}(\boldsymbol{Z},t) = \begin{pmatrix} u_1 \\ v_2 \\ u_2 \\ v_2 \\ \dfrac{x_1\lambda_1 + (x_1 - x_2)\lambda_2}{m_1} \\ \dfrac{y_1\lambda_1 + (y_1 - y_2)\lambda_2}{m_1} - g \\ \dfrac{(x_2 - x_1)\lambda_2}{m_2} \\ \dfrac{(y_2 - y_1)\lambda_2}{m_2} - g \\ x_1^2 + y_1^2 - \ell_1^2 \\ (x_1 - x_2)^2 + (y_1 - y_2)^2 - \ell_2^2 \end{pmatrix}$$

So that the DAE reads:

$$\boldsymbol{E}(\boldsymbol{Z},t)\,\boldsymbol{Z}'(t) = \boldsymbol{G}(\boldsymbol{Z},t)$$

27.2 Index reduction

The table

$$\boldsymbol{T} = \left[\ \boldsymbol{E}(\boldsymbol{Z},t)\ \middle|\ \boldsymbol{G}(\boldsymbol{Z},t)\ \right] = \left[\begin{array}{cccccccccc|c} 1 & . & . & . & . & . & . & . & . & . & u_1 \\ . & 1 & . & . & . & . & . & . & . & . & v_2 \\ . & . & 1 & . & . & . & . & . & . & . & u_2 \\ . & . & . & 1 & . & . & . & . & . & . & v_2 \\ . & . & . & . & 1 & . & . & . & . & . & \dfrac{x_1\lambda_1+(x_1-x_2)\lambda_2}{m_1} \\ . & . & . & . & . & 1 & . & . & . & . & \dfrac{y_1\lambda_1+(y_1-y_2)\lambda_2}{m_1}-g \\ . & . & . & . & . & . & 1 & . & . & . & \dfrac{(x_2-x_1)\lambda_2}{m_2} \\ . & . & . & . & . & . & . & 1 & . & . & \dfrac{(y_2-y_1)\lambda_2}{m_2}-g \\ . & . & . & . & . & . & . & . & . & . & x_1^2+y_1^2-\ell_1^2 \\ . & . & . & . & . & . & . & . & . & . & (x_1-x_2)^2+(y_1-y_2)^2-\ell_2^2 \end{array}\right]$$

is used to memorize the operation done on the DAE.

Remark 27.1 — semi-explicit. Notice that

$$\boldsymbol{E}(\boldsymbol{Z},t) = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}$$

this permits to simplify index reduction working only on the last two rows.

27.2.1 First reduction of the index by one

1. Derive the algebraic constraint

$$[\text{alg1}] = \begin{pmatrix} x_1^2+y_1^2-\ell_1^2 \\ (x_1-x_2)^2+(y_1-y_2)^2-\ell_2^2 \end{pmatrix}$$

$$\frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = \begin{pmatrix} 2x_1x_1'+2y_1y_1' \\ 2(x_1-x_2)(x_1'-x_2')+2(y_1-y_2)(y_1'-y_2') \end{pmatrix}$$

2. Update reduced table $\widetilde{\boldsymbol{T}}$ of the last two rows

$$\widetilde{\boldsymbol{T}} \leftarrow \left[\begin{array}{cccccccccc|c} 2x_1 & 2y_1 & . & . & . & . & . & . & . & . & 0 \\ 2(x_1-x_2) & 2(y_1-y_2) & -2(x_1-x_2) & -2(y_1-y_2) & . & . & . & . & . & . & 0 \end{array}\right]$$

apply elimination by subtract a combination of the fist four row to eliminate $2x_1$, $2y_1$ and so on. However this is equivalent to substitute x_1' with u_1, y_1' with v_1 and so on., So that

$$[\text{alg2}] = \frac{\mathrm{d}[\text{alg1}]}{\mathrm{d}t} = \begin{pmatrix} 2x_1u_1+2y_1v_1 \\ 2(x_1-x_2)(u_1-u_2)+2(y_1-y_2)(v_1-v_2) \end{pmatrix}$$

27.2.2 Second reduction of the index by one

1. Derive the constraint and apply substitution (using $\delta x = x_1-x_2$, $\delta y = y_1-y_2$, $\delta u = u_1-u_2$, $\delta v = v_1-v_2$)

$$
\begin{aligned}
[\text{alg3}] = \frac{\mathrm{d}[\text{alg2}]}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} 2x_1u_1 + 2y_1v_1 \\ 2(x_1-x_2)(u_1-u_2) + 2(y_1-y_2)(v_1-v_2) \end{pmatrix} \\
&= \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} 2x_1u_1 + 2y_1v_1 \\ 2\,\delta x\,\delta u_1 + 2\,\delta y\,\delta v \end{pmatrix} \\
&= \begin{pmatrix} 2x_1'u_1 + 2y_1'v_1 + 2x_1u_1' + 2y_1v_1' \\ 2\,\delta x'\,\delta u + 2\,\delta y'\,\delta v + 2\,\delta x\,\delta u' + 2\,\delta y\,\delta v' \end{pmatrix} \\
&= \begin{pmatrix} 2u_1^2 + 2v_1^2 + 2x_1u_1' + 2y_1v_1' \\ 2\,\delta u^2 + 2\,\delta v^2 + 2\,\delta x\,\delta u' + 2\,\delta y\,\delta v' \end{pmatrix}
\end{aligned}
$$

2. defining $1/\widetilde{m} = 1/m_1 + 1/m_2$

$$
\begin{aligned}
\delta u' = u_1' - u_2' &= \frac{x_1\lambda_1 + (x_1-x_2)\lambda_2}{m_1} - \frac{(x_2-x_1)\lambda_2}{m_2} \\
&= \frac{x_1\lambda_1}{m_1} + \delta x\lambda_2\left(\frac{1}{m_1} + \frac{1}{m_2}\right) \\
&= \frac{x_1\lambda_1}{m_1} + \frac{\delta x\lambda_2}{\widetilde{m}} \\
\delta v' = v_1' - v_2' &= \frac{y_1\lambda_1 + (y_1-y_2)\lambda_2}{m_1} - g - \frac{(y_2-y_1)\lambda_2}{m_2} + g \\
&= \frac{y_1\lambda_1}{m_1} + \delta y\lambda_2\left(\frac{1}{m_1} + \frac{1}{m_2}\right) \\
&= \frac{y_1\lambda_1}{m_1} + \frac{\delta y\lambda_2}{\widetilde{m}}
\end{aligned}
$$

3. The new constraint become

$$
\begin{aligned}
[\text{alg3}] &= \begin{pmatrix} [\text{alg3}]_1 \\ [\text{alg3}]_2 \end{pmatrix} \\
[\text{alg3}]_1 &= 2(u_1^2 + v_1^2) + \frac{2}{m_1}\lambda_1(x_1^2 + y_1^2) + \frac{2}{m_1}\lambda_2(x_1\delta x + y_1\delta y) - 2gy_1 \\
[\text{alg3}]_2 &= 2(\delta u^2 + \delta v^2) + \frac{2\lambda_1}{m_1}(x_1\delta x + y_1\delta y) + \frac{2\lambda_2}{\widetilde{m}}\left(\delta x^2 + \delta y^2\right)
\end{aligned}
$$

DAE reduced to index 1 DAE

Remark 27.2 At this point λ_1 and λ_2 appears. Thus after another derivation the DAE is reduced to an ODE and, thus, the index is 3.

The derivation produce a complex ODE and is not reccomeneded. Instead, λ_1 and λ_2 can be solved, or better written as a function of position and velocity.

Elimination of the multiplier

1. Rewrite [alg3] as a linear system

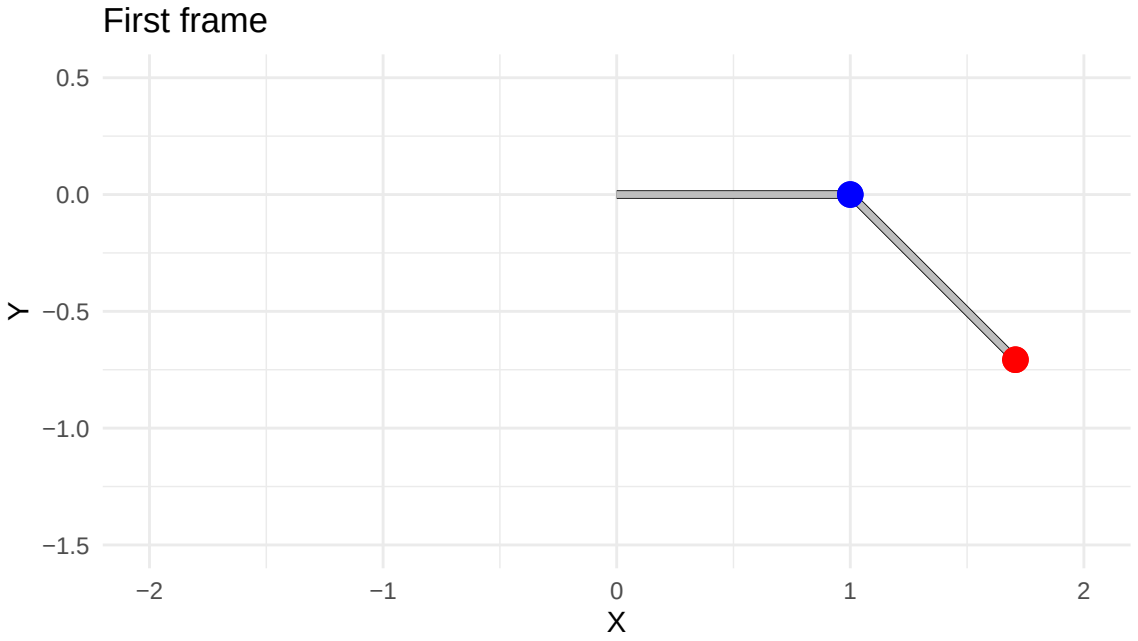
$$
\frac{1}{m_1}\begin{pmatrix} x_1^2 + y_1^2 & x_1\delta x + y_1\delta y \\ x_1\delta x + y_1\delta y & \dfrac{m_1}{\widetilde{m}}(\delta x^2 + \delta y^2) \end{pmatrix}\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} gy_1 - (u_1^2 + v_1^2) \\ -(\delta u^2 + \delta v^2) \end{pmatrix}
$$

2. the solution of the linear system are the function

$$\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} = m_1 \begin{pmatrix} x_1^2+y_1^2 & x_1\delta x+y_1\delta y \\ x_1\delta x+y_1\delta y & \frac{m_1}{\widetilde{m}}(\delta x^2+\delta y^2)\end{pmatrix}^{-1} \begin{pmatrix} gy_1-(u_1^2+v_1^2) \\ -(\delta u^2+\delta v^2)\end{pmatrix}$$

The final ODE is

$$\begin{cases} x_1' = u_1 \\ y_1' = v_1 \\ x_2' = u_2 \\ y_2' = v_2 \\ u_1' = \frac{x_1}{m_1}\lambda_1(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) + \frac{(x_1-x_2)}{m_1}\lambda_2(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) \\ v_1' = \frac{y_1}{m_1}\lambda_1(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) + \frac{(y_1-y_2)}{m_1}\lambda_2(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) - g \\ u_2' = \frac{(x_2-x_1)}{m_2}\lambda_2(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) \\ v_2' = \frac{(y_2-y_1)}{m_2}\lambda_2(x_1,y_1,x_2,y_2,u_1,v_1,u_2,v_2) - g \end{cases}$$



Parameter	Parameter	Parameter
$m_1 = 5$	$m_2 = 1$	$\ell_1 = 1$
$\ell_2 = 1$	$g = 9.81$	$\theta_1 = \pi/2$
$\theta_2 = \pi/4$	$\omega_1 = 0$	$\omega_2 = -0.1$
$x_1(0) = \ell_1 \sin(\theta_1)$	$y_1(0) = -\ell_1 \cos(\theta_1)$	$x_2(0) = x_1(0) + \ell_2 \sin(\theta_2)$
$y_2(0) = x_2(0) - \ell_2 \cos(\theta_2)$	$u_1(0) = 0$	$v_1(0) = 0$
$u_2(0) = 0$	$v_2(0) = 0$	

Figure 27.2: Parameters for the double pendulum solution of the reduced DAE without stabilization

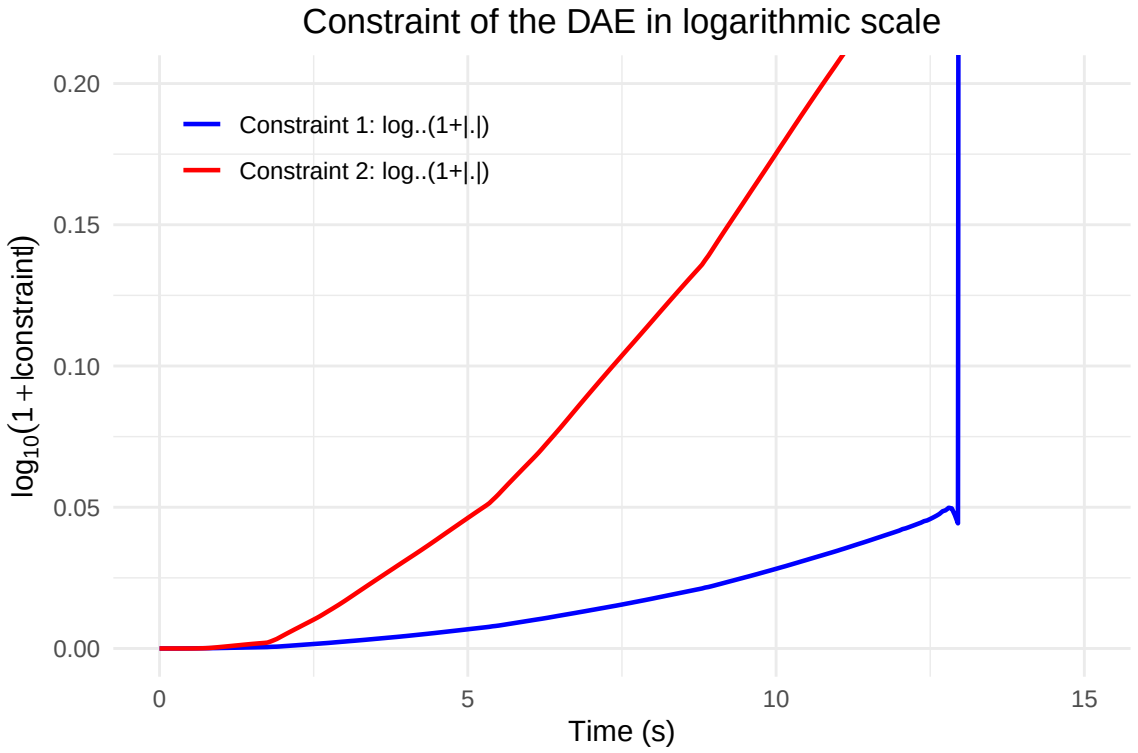


Figure 27.3: Drift of the constraints. Notice that scale is logarithmic and error grows exponentially.

Baumgarte Stabilization

To stabilize the index-3 DAE system of the double pendulum and avoid constraint drift during numerical integration, we introduce Baumgarte stabilization, a method that replaces the original constraints with second-order damped versions of them.

We define the stabilized constraints by enforcing a second-order differential equation for each component:

$$\frac{\mathrm{d}^2[\mathrm{alg1}]}{\mathrm{d}t^2} + 2\eta\omega\frac{\mathrm{d}[\mathrm{alg1}]}{\mathrm{d}t} + \omega^2[\mathrm{alg1}] = \mathbf{0}$$

or equivalently

$$[\mathrm{alg3}] + 2\eta\omega[\mathrm{alg2}] + \omega^2[\mathrm{alg1}] = \mathbf{0}$$

that become

$$2(u_1^2+v_1^2) + \frac{2}{m_1}\lambda_1(x_1^2+y_1^2) + \frac{2}{m_1}\lambda_2(x_1\delta x + y_1\delta y) - 2gy_1 + 2\eta\omega(2x_1u_1 + 2y_1v_1) + \omega^2(x_1^2+y_1^2-\ell_1^2) = 0$$

$$2(\delta u^2+\delta v^2) + \frac{2\lambda_1}{m_1}(x_1\delta x + y_1\delta y) + \frac{2\lambda_2}{\widetilde{m}}\left(\delta x^2+\delta y^2\right) + 2\eta\omega(2\delta x\,\delta u + 2\delta y\,\delta v) + \omega^2(\delta x^2+\delta y^2-\ell_2^2) = 0$$

that become the linear system

$$\frac{1}{m_1}\begin{pmatrix} x_1^2+y_1^2 & x_1\delta x+y_1\delta y \\ x_1\delta x+y_1\delta y & \frac{m_1}{\widetilde{m}}(\delta x^2+\delta y^2)\end{pmatrix}\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}=\begin{pmatrix} r_1\\ r_2\end{pmatrix}$$

where

$$r_1 = gy_1-(u_1^2+v_1^2)-2\eta\omega(x_1u_1+y_1v_1)-\frac{1}{2}\omega^2(x_1^2+y_1^2-\ell_1^2)$$
$$r_2 = -(\delta u^2+\delta v^2)-2\eta\omega(\delta x\,\delta u+\delta y\,\delta v)-\frac{1}{2}\omega^2(\delta x^2+\delta y^2-\ell_2^2)$$

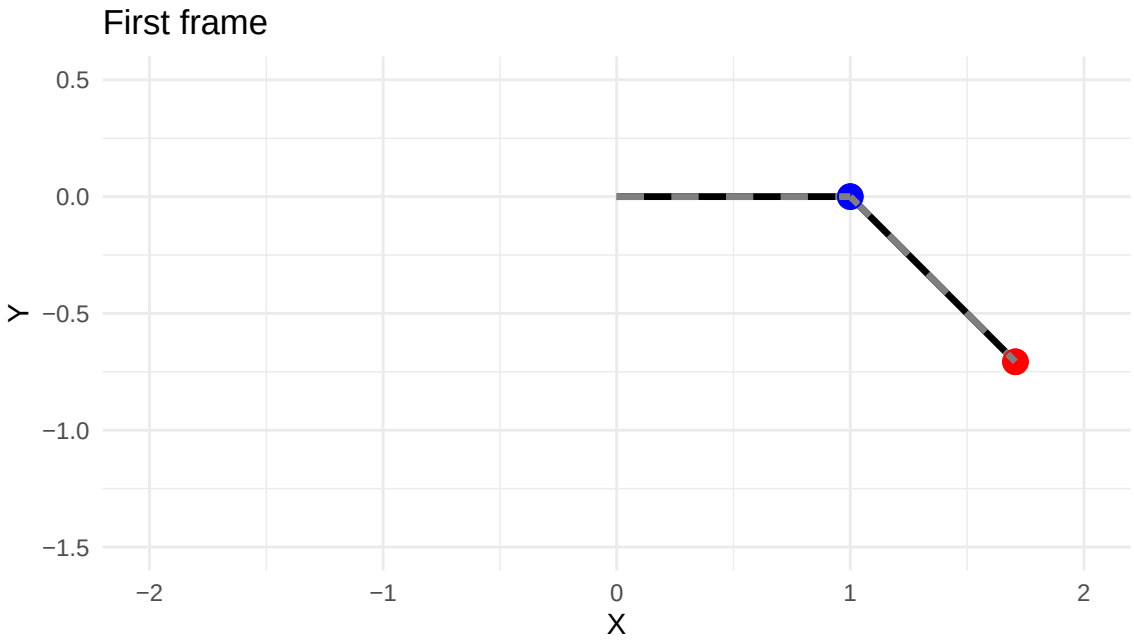


Figure 27.4

Parameters for the double pendulum solution of the reduced DAE with stabilization

Parameter	Parameter
$m_1 = 5$	$m_2 = 1$
$\ell_1 = 1$	$\ell_2 = 1$
$g = 9.81$	$\theta_1 = \pi/2$
$\theta_2 = \pi/4$	$\omega_1 = 0$
$\omega_2 = -0.1$	$x_1(0) = \ell_1\sin(\theta_1)$
$y_1(0) = -\ell_1\cos(\theta_1)$	$x_2(0) = x_1(0)+\ell_2\sin(\theta_2)$
$y_2(0) = x_2(0)-\ell_2\cos(\theta_2)$	$u_1(0) = 0$
$v_1(0) = 0$	$u_2(0) = 0$
$v_2(0) = 0$	$\eta = 0.7$
$\Delta t = 0.05$	$\omega = 2.4/\Delta t$

27.3 Index reduction using dummy variable

Variable classification

- **Differential variables**: $x_1, y_1, x_2, y_2, u_1, v_1, u_2, v_2$

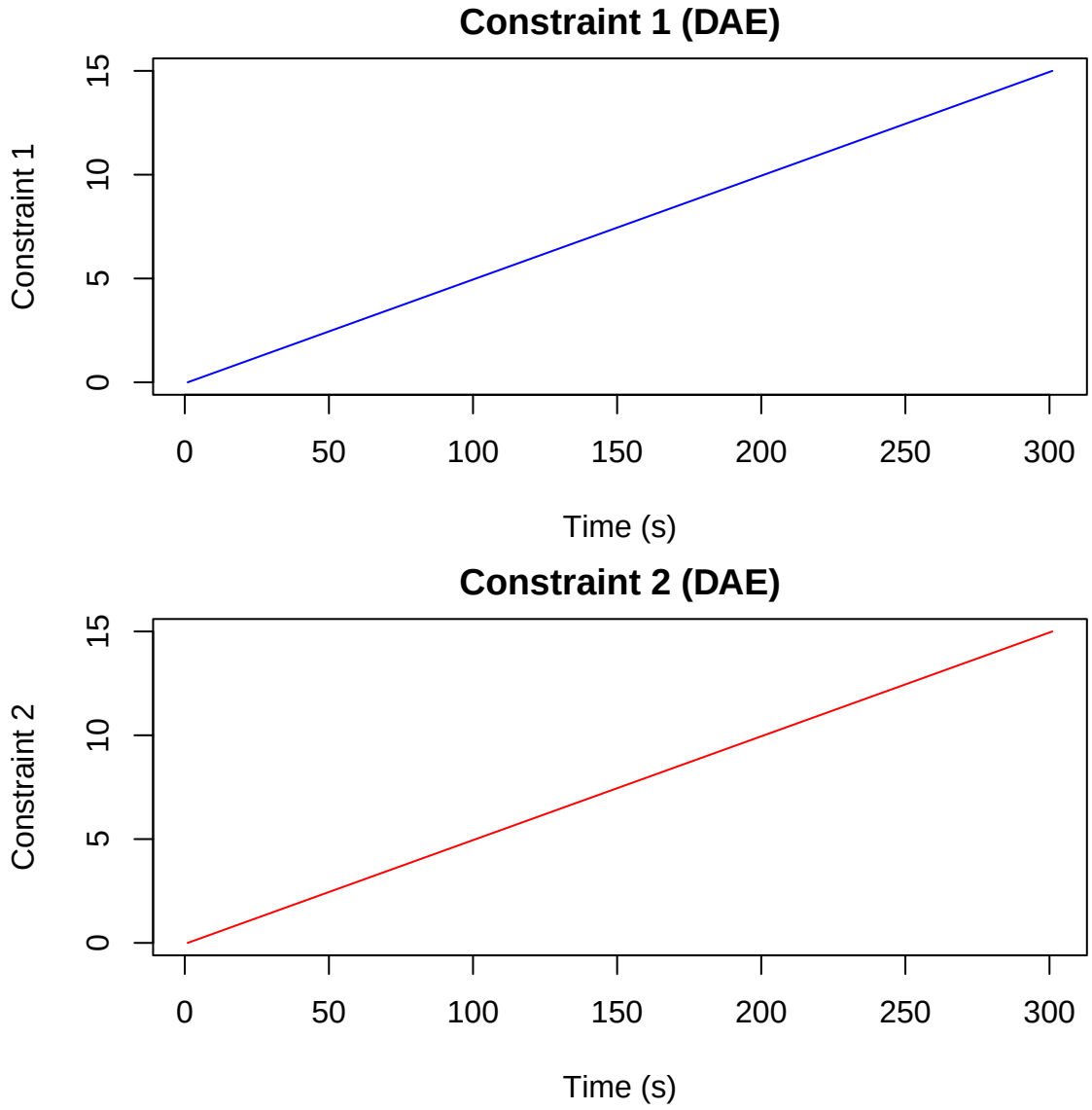


Figure 27.5: Drift of the constraints. Notice that drift do not grow and is maintained close to zero by Baumgarte stabilization.

- **Algebraic variables**: λ_1, λ_2.

We attempt to solve for λ_1 and λ_2, but the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Now we must derivate the constraints in order to reduce the index. Here we want to differentiate **only** the algebraic equation that can reduce the index and do not introduce new differential equation. Here we want to avoid to introduce derivatove of the algebraic variable unless is strictly necessary.

We construct a dependency table to identify which constraints to differentiate:

	λ_1	λ_2
(Constraint 1)	0	0
(Constraint 2)	0	0

a 0 means the the function do not depend on the variable on the column. A 1 means dependence. Both constraints are candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus the constraints and substitute known variable in differential form using the ODE part

$$\begin{aligned}
(\text{Constraint 1})' &= 2x_1x_1' + 2y_1y_1' = 2(x_1u_1 + y_1v_1) \\
(\text{Constraint 2})' &= 2(x_1 - x_2)(x_1' - x_2') + 2(y_1 - y_2)(y_1' - y_2') \\
&= 2((x_1 - x_2)(u_1 - u_2) + (y_1 - y_2)(v_1 - v_2))
\end{aligned}$$

Substitute known differential variables and use shortcuts $\delta x = x_1 - x_2$, $\delta y = y_1 - y_2$, $\delta u = u_1 - u_2$, $\delta v = v_1 - v_2$ so that the DAE become:

$$\begin{cases}
x_1' = u_1 \\
y_1' = v_1 \\
x_2' = u_2 \\
y_2' = v_2 \\
u_1' = \dfrac{x_1\lambda_1 + (x_1 - x_2)\lambda_2}{m_1} \\
v_1' = \dfrac{y_1\lambda_1 + (y_1 - y_2)\lambda_2}{m_1} - g \\
u_2' = \dfrac{(x_2 - x_1)\lambda_2}{m_2} \\
v_2' = \dfrac{(y_2 - y_1)\lambda_2}{m_2} - g
\end{cases}
\oplus
\begin{cases}
0 = 2(x_1u_1 + y_1v_1) & (\text{Constraint 1})' \\
0 = 2(\delta x\delta u + \delta y\delta v) & (\text{Constraint 2})' \\
\\
\ell_1^2 = x_1^2 + y_1^2 & (\text{Constraint 1}) \\
\ell_2^2 = \delta x^2 + \delta y^2 & (\text{Constraint 2})
\end{cases}$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine the algebraic variable and must be differentiated again.

Step 2: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	λ_1	λ_2
(Constraint 1)$'$	0	0
(Constraint 2)$'$	0	0

Both constraints are candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus the constraints and substitute known variable in differential form using the ODE part

Step 3: Differentiate algebraic constraints

$$\begin{aligned}\text{(Constraint 1)}'' &= 2x_1'u_1 + 2y_1'v_1 + 2x_1 + 2y_1v_1' = 0\\ \text{(Constraint 2)}'' &= 2\delta x'\delta u + 2\delta y'\delta v + 2\delta x\delta u' + 2\delta y\delta v' = 0\end{aligned}$$

Substitute known differential variables

$$\begin{aligned}2(u_1^2+v_1^2) + 2\frac{x_1^2\lambda_1 + x_1\delta x\lambda_2}{m_1} + 2\frac{y_1^2\lambda_1 + y_1\delta y\lambda_2}{m_1} - 2y_1g = 0\\ 2(\delta u^2+\delta v^2) + 2\delta x\delta u' + 2\delta y\delta v' = 0\end{aligned}$$

using $1/\widetilde{m} = 1/m_1 + 1/m_2$

$$\begin{aligned}\delta u' &= \frac{x_1\lambda_1 + \delta x\lambda_2}{m_1} + \frac{\delta x\lambda_2}{m_2} = \frac{x_1}{m_1}\lambda_1 + \frac{\delta x}{\widetilde{m}}\lambda_2\\ \delta v' &= \frac{y_1\lambda_1 + \delta y\lambda_2}{m_1} + \frac{\delta y\lambda_2}{m_2} = \frac{y_1}{m_1}\lambda_1 + \frac{\delta y}{\widetilde{m}}\lambda_2\end{aligned}$$

so that constraints after substitution become

$$\begin{aligned}\text{(Constraint 1)}'' &= 2(u_1^2+v_1^2) + \frac{2}{m_1}((x_1^2+y_1^2)\lambda_1 + (x_1\delta x + y_1\delta y)\lambda_2) - 2y_1g = 0\\ \text{(Constraint 2)}'' &= 2(\delta u^2+\delta v^2) + 2\delta x\left(\frac{x_1}{m_1}\lambda_1 + \frac{\delta x}{\widetilde{m}}\lambda_2\right) + 2\delta y\left(\frac{y_1}{m_1}\lambda_1 + \frac{\delta y}{\widetilde{m}}\lambda_2\right) = 0\end{aligned}$$

and now λ_1 and λ_2 appears linearly:

$$\begin{pmatrix} x_1^2+y_1^2 & x_1\delta x + y_1\delta y \\ x_1\delta x + y_1\delta y & \frac{m_1}{\widetilde{m}}(\delta x^2+\delta y^2)\end{pmatrix}\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} = m_1\begin{pmatrix} y_1g - (u_1^2+v_1^2)\\ -(\delta u^2+\delta v^2)\end{pmatrix}$$

27.3.1 The DAE of index 1

Thus DAE now is of index 1 and can be written as:

$$\begin{cases} x_1' = u_1 \\ y_1' = v_1 \\ x_2' = u_2 \\ y_2' = v_2 \\ u_1' = \dfrac{x_1\lambda_1 + (x_1 - x_2)\lambda_2}{m_1} \\ v_1' = \dfrac{y_1\lambda_1 + (y_1 - y_2)\lambda_2}{m_1} - g \\ u_2' = \dfrac{(x_2 - x_1)\lambda_2}{m_2} \\ v_2' = \dfrac{(y_2 - y_1)\lambda_2}{m_2} - g \\ \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \boldsymbol{A}(t,\boldsymbol{Z})^{-1}\boldsymbol{b}(t,\boldsymbol{Z}) \end{cases}$$

where

$$\boldsymbol{Z} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ u_1 \\ v_1 \\ u_2 \\ v_2 \end{pmatrix}, \qquad \boldsymbol{A}(t,\boldsymbol{Z}) = \begin{pmatrix} x_1^2 + y_1^2 & x_1\delta x + y_1\delta y \\ x_1\delta x + y_1\delta y & \dfrac{m_1}{\widetilde{m}}(\delta x^2 + \delta y^2) \end{pmatrix},$$

$$\boldsymbol{b}(t,\boldsymbol{Z}) = m_1 \begin{pmatrix} y_1 g - (u_1^2 + v_1^2) \\ -(\delta u^2 + \delta v^2) \end{pmatrix}$$

Moreover using the right initial condition satisfy the hidden constraints.

$$\begin{cases} 0 = x_1^2 + y_1^2 - \ell_1^2 & \text{(Constraint 1)} \\ 0 = \delta x^2 + \delta y^2 - \ell_2^2 & \text{(Constraint 2)} \\ 0 = 2(x_1u_1 + y_1v_1) & \text{(Constraint 1)}' \\ 0 = 2(\delta x\delta u + \delta y\delta v) & \text{(Constraint 2)}' \end{cases}$$

27.3.2 Baumgarte Stabilization

For this DAE the hidden constraints can be used to stabilize the algebric constraints: (Constraint 1) and (Constraint 2)

$$\begin{aligned} \text{(Constraint 1)}_{\text{stab}} &= \text{(Constraint 1)}'' + 2\eta_1\omega_1\text{(Constraint 1)}' + \omega_1^2\text{(Constraint 1)} = 0 \\ \text{(Constraint 2)}_{\text{stab}} &= \text{(Constraint 2)}'' + 2\eta_2\omega_2\text{(Constraint 2)}' + \omega_2^2\text{(Constraint 2)} = 0 \end{aligned}$$

and substituting (diving also by 2)

$$\text{(Constraint 1)}_{\text{stab}} = u_1^2 + v_1^2 + \frac{(x_1^2 + y_1^2)\lambda_1 + (x_1\delta x + y_1\delta y)\lambda_2}{m_1} - y_1 g + (\star)_1$$

$$\text{(Constraint 2)}_{\text{stab}} = \delta u^2 + \delta v^2 + \frac{(x_1\delta x + y_1\delta y)\lambda_1}{m_1} + \frac{(\delta x^2 + \delta y^2)\lambda_2}{\widetilde{m}} + (\star)_2$$

$$(\star)_1 = 2\eta_1\omega_1(x_1u_1 + y_1v_1) + \omega_1^2\frac{x_1^2 + y_1^2 - \ell_1^2}{2}$$

$$(\star)_2 = 2\eta_2\omega_2(\delta x\delta u + \delta y\delta v) + \omega_2^2\frac{\delta x^2 + \delta y^2 - \ell_2^2}{2}$$

and using the stabilized constraints

$$\boldsymbol{b}_{\text{stab}}(t, \boldsymbol{Z}) = m_1 \begin{pmatrix} y_1 g - (u_1^2 + v_1^2) - (\star)_1 \\ -(\delta u^2 + \delta v^2) - (\star)_2 \end{pmatrix}$$



28. Example 5

28.1 A Constrained Van der Pol System and a Dummy-Variable Reformulation

The Van der Pol equation

$$x'' - \mu(1-x^2)x' + x = 0$$

represents a classic example of a nonlinear oscillator with nonlinear damping, widely used in the study of dynamical systems and electrical circuits.

28.2 Formulation as a Stiff DAE

To construct a constrained DAE system [15], we introduce:

1. The variable $y = x'$ to obtain a first-order system
2. An additional state z

The resulting DAE is:

$$\begin{cases} x' = y \\ y' = \mu(1-x^2)y - x \\ 0 = x^2 - z^2 + 5 \end{cases}$$

In this original formulation, x and y are differential variables and z is algebraic. Since $\partial(x^2 - z^2 + 5)/\partial z = -2z \neq 0$ on the constraint manifold $z^2 = x^2 + 5$, this is a regular semi-explicit **index-1 DAE**. The dummy-variable construction below is used to illustrate how one obtains an ODE representation together with its hidden invariant; it does not change the index of the original DAE.

28.2.1 Dummy-Variable Reformulation

Introduce dummy variables y_{dot}, z_{dot}. In this augmented representation, the system decomposes into differential (blue) and algebraic (red) components:

$$\begin{cases} x' = y \\ y' = y_{\text{dot}} \\ z' = z_{\text{dot}} \end{cases} \quad \oplus \quad \begin{cases} 0 = y_{\text{dot}} - \mu(1-x^2)y + x & \text{(Constraint 1)} \\ 0 = x^2 - z^2 + 5 & \text{(Constraint 2)} \end{cases}$$

For the augmented system we have:

- **Differential variables**: x, y, z.
- **Algebraic variables**: y_{dot}, z_{dot}.

28.2.2 Index Reduction Procedure

The equations determine y_{dot} but not z_{dot}. To obtain the ODE representation, we differentiate the geometric constraint once.

Step 1: Differentiate algebraic constraints

We differentiate only the geometric constraint: this introduces z_{dot} without differentiating an algebraic variable again.

We construct a dependency table to identify which constraints to differentiate:

	y_{dot}	z_{dot}
(Constraint 1)	1	0
(Constraint 2)	0	0

A 0 means that the function does not depend on the variable in the column. A 1 means dependence. The last constraint is the candidate for differentiation because it does not introduce a new algebraic variable. Differentiate it and substitute known derivatives from the differential part:

$$(\text{Constraint 2})' = 2xx' - 2zz' = 2xy - 2zz_{\text{dot}}$$

The augmented DAE becomes:

$$\begin{cases} x' = y \\ y' = y_{\text{dot}} \\ z' = z_{\text{dot}} \end{cases} \oplus \begin{cases} 0 = y_{\text{dot}} - \mu(1-x^2)y + x & \text{(Constraint 1)} \\ 0 = 2xy - 2zz_{\text{dot}} & \text{(Constraint 2)}' \\ \\ 0 = x^2 - z^2 + 5 & \text{(Constraint 2)} \end{cases}$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine the algebraic variable and must be differentiated again.

Step 2: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	y_{dot}	z_{dot}
(Constraint 1)	1	0
(Constraint 2)$'$	0	1

The table shows that the nonlinear system is **structurally** non singular and algebraic constraints can be solved. From (Constraint 2)$'$ solve for z_{dot} and from (Constraint 1) solve for y_{dot}

$$z_{\text{dot}} = \frac{xy}{z}, \qquad y_{\text{dot}} = \mu(1-x^2)y - x$$

28.2.3 The ODE

Thus DAE now can be reduced to an ODE and can be written as:

$$\begin{cases} x' = y \\ y' = y_{\text{dot}} = \mu(1-x^2)y - x \\ z' = z_{\text{dot}} = \dfrac{xy}{z} \end{cases} \quad \bigoplus \quad x^2 - z^2 + 5 = 0, \quad \text{(Constraint 1)}$$

28.3 Baumgarte Stabilization

Baumgarte stabilization is a classical technique used to prevent this drift. Instead of enforcing a constraint like:

$$z(t) = 0$$

we instead modify it to a differential equation:

$$\frac{dz}{dt} + \frac{1}{\tau} z = 0$$

This form ensures that any initial deviation from the constraint decays exponentially with a rate controlled by the parameter τ. The smaller τ is, the stronger the correction applied.

The exact solution is:

$$z(t) = z_0 e^{-t/\tau}$$

In the following example, we simulate the stabilized equation for different values of τ and visualize how quickly the constraint error decays.

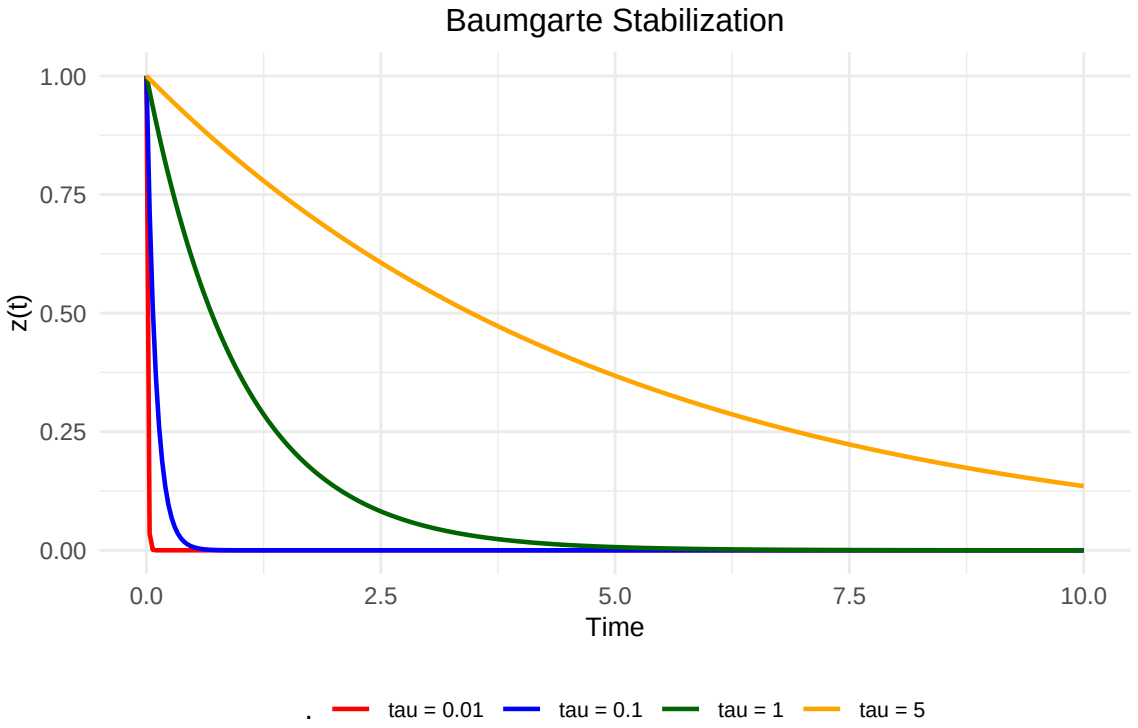


Figure 28.1: Impact of the stabilization parameter τ.

The plot clearly shows the impact of the stabilization parameter τ:

- **Small τ (e.g., 0.01)**: The solution quickly decays to zero, meaning constraint violations are rapidly corrected. However, this can introduce numerical stiffness.

- **Large τ (e.g., 5)**: The decay is slow, allowing more deviation from the constraint over time.

This trade-off is at the heart of Baumgarte stabilization: small τ improves accuracy but increases stiffness and numerical cost, while large τ is easier to integrate but less effective at controlling constraint drift.

In real DAE systems, choosing τ often involves tuning based on system stiffness and desired constraint enforcement.

For this DAE the hidden constraints can be used to stabilize the algebraic constraint (Constraint 1)

$$(\text{Constraint 1})_{\text{stab}} = (\text{Constraint 1})' + \frac{1}{\tau}(\text{Constraint 1}) = 0$$

and substituting (diving also by 2)

$$(\text{Constraint 1})_{\text{stab}} = xy - zz_{\text{dot}} + \frac{1}{\tau}\frac{x^2 - z^2 + 5}{2}$$

and using the stabilized constraints

$$\begin{cases} x' = y \\ y' = \mu(1-x^2)y - x \\ z' = \dfrac{xy}{z} + \dfrac{1}{\tau}\dfrac{x^2 - z^2 + 5}{2z} \end{cases} \quad \bigoplus \quad x^2 - z^2 + 5 = 0, \quad (\text{Constraint 1})$$

28.4 Numerical Examples

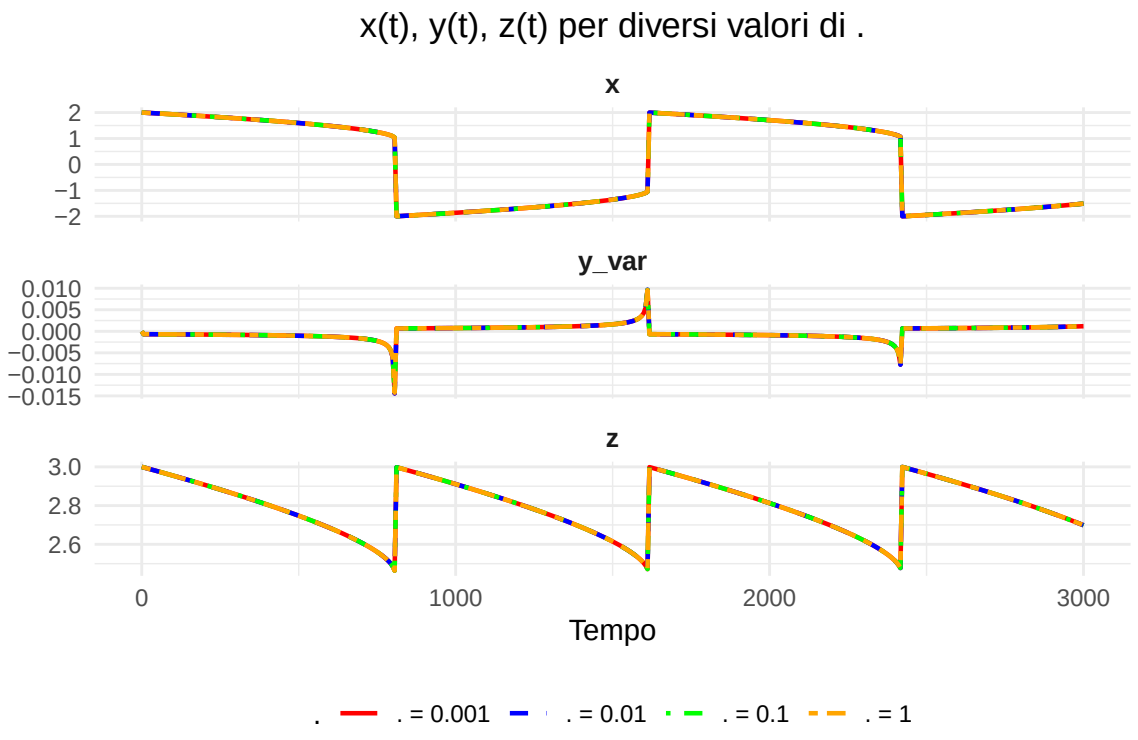


Figure 28.2



29. Example 6

29.1 Introduction

Example from [12] equation (3.1)–(3.3)

$$\begin{cases} y' = t - x \\ y' + z' = 1 - (\alpha x + \eta) y \\ 0 = y(2 - y) + 2z \end{cases}$$

29.1.1 Semi-Explicit Formulation

For the previous DAE system introduce dummy variable x_{dot} and the system decomposes into differential (blue) and algebraic (red) components:

$$\begin{cases} x' = x_{\text{dot}} \\ y' = t - x \\ z' = 1 - t + x - (\alpha x + \eta) y \end{cases} \quad \bigoplus \quad 0 = y(1 - y/2) + z \qquad \text{(Constraint 1)}$$

For this DAE we have:

- **Differential variables**: x, y, z.
- **Algebraic variables**: x_{dot}.

29.1.2 Index Reduction Procedure

We attempt to solve for x_{dot}, but the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Differentiate the constraint

$$\begin{aligned} (\text{Constraint 1})' &= 1 - yy' + z' \\ &= 1 - y(t - x) + 1 - t + x - (\alpha x + \eta) y \\ &= 2 + (x - t)(y + 1) - (\alpha x + \eta) y \end{aligned}$$

so that the DAE become:

$$\begin{cases} x' = x_{\text{dot}} \\ y' = t - x \\ z' = 1 - t + x - (\alpha x + \eta) y \end{cases} \oplus \begin{cases} 0 = 2 + (x-t)(y+1) - (\alpha x + \eta) y & \text{(Constraint 1)}' \\ 0 = y(1 - y/2) + z & \text{(Constraint 1)} \end{cases}$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine x_{dot} the algebraic variable must be differentiated again.

Step 2: Differentiate (again) the algebraic constraint

$$\begin{aligned} (\text{Constraint 1})'' &= (x'-1)(y+1) + (x-t)y' - \alpha(x'y + xy') + \eta y' \\ &= (x_{\text{dot}} - 1)(y+1) + (x-t)(t-x) - \alpha(x_{\text{dot}} y + x(t-x)) + \eta(t-x) \\ &= x_{\text{dot}}(y(1-\alpha)+1) - y - 1 - (x-t)^2 + (\alpha x + \eta)(t-x) = 0 \end{aligned}$$

The dummy variable x_{dot} can be solved:

$$x_{\text{dot}} = \frac{y + 1 + (x - t + \alpha x + \eta)(x-t)}{y(1-\alpha)+1}$$

and the DAE is reduced to an index-1 DAE. Having done two derivation of the constraint the original DAE was of **index 3**. Removing the dummy variable we can obtain an ODE with invariants:

$$\begin{cases} x' = \dfrac{y + 1 + (x - t + \alpha x + \eta)(x-t)}{y(1-\alpha)+1} \\ y' = t - x \\ z' = 1 - t + x - (\alpha x + \eta) y \end{cases} \oplus \begin{cases} 0 = 2 + (x-t)(y+1) - (\alpha x + \eta) y & \text{(Constraint 1)}' \\ 0 = y(1 - y/2) + z & \text{(Constraint 1)} \end{cases}$$

29.2 Baumgarte Stabilization

As usual we want to stabilize the constraints to avoid the drift on the hidden constraints:

$$(\text{Constraints 1})'' + 2\eta\omega(\text{Constraints 1})' + \omega^2(\text{Constraints 1}) = 0$$

and substituting the expression

$$\begin{aligned} & x_{\text{dot}}(y(1-\alpha)+1) - y - 1 - (x-t)^2 + (\alpha x + \eta)(t-x) \\ & + 2\eta\omega(2 + (x-t)(y+1) - (\alpha x + \eta) y) \\ & + \omega^2(y(1 - y/2) + z) = 0 \end{aligned}$$

so that

$$\begin{aligned} x_{\text{dot}} &= f(x,y,z) \\ f(t,x,y,z) &= \frac{a(t,x,y,z) - 2\eta\omega b(t,x,y,z) - \omega^2 c(t,x,y,z)}{y(1-\alpha)+1} \\ a(t,x,y,z) &= y + 1 + (x - t + \alpha x + \eta)(x-t) \\ b(t,x,y,z) &= 2 + (x-t)(y+1) - (\alpha x + \eta) y \\ c(t,x,y,z) &= y(1 - y/2) + z \end{aligned}$$

and the stabilized ODE become

$$\begin{cases} x' = f(t,x,y,z) \\ y' = t - x \\ z' = 1 - t + x - (\alpha x + \eta) y \end{cases} \oplus \begin{cases} 0 = 2 + (x-t)(y+1) - (\alpha x + \eta) y & \text{(Constraint 1)}' \\ 0 = y(1 - y/2) + z & \text{(Constraint 1)} \end{cases}$$



30. Example 7

From [20]

30.1 Circuit Description

We start with a system of differential-algebraic equations (DAE) that describes a circuit. The equations are given by:

$$\begin{pmatrix} C & -C & 0 \\ -C & C & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1' \\ v_3' \\ i \end{pmatrix} + \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1/R & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_3 \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ V(t) \end{pmatrix}$$

30.1.1 Semi-Explicit Formulation

We decompose the system into differential (blue) and algebraic (red) components as follows:

$$\begin{cases} v_1' = v_{1,\text{dot}} \\ v_3' = v_{3,\text{dot}} \end{cases} \oplus \begin{cases} C(v_{1,\text{dot}} - v_{3,\text{dot}}) - i = 0, & \text{(Constraint 1)} \\ RC(v_{1,\text{dot}} - v_{3,\text{dot}}) - v_3 = 0, & \text{(Constraint 2)} \\ v_1 + V(t) = 0, & \text{(Constraint 3)} \end{cases}$$

In this system:

- **Differential variables**: v_1, v_3.
- **Algebraic variables**: $v_{1,\text{dot}}$, $v_{3,\text{dot}}$, i.

We begin by attempting to solve for the variables ($v_{1,\text{dot}}$, $v_{3,\text{dot}}$, and i), but the system is not explicitly solvable at this stage. We need to reduce the index of the system. To do this, we differentiate the algebraic constraints.

30.1.2 Index Reduction Procedure

We attempt to solve for $v_{1,\text{dot}}$, $v_{3,\text{dot}}$ and i_{dot} but the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Now we must derivate the constraints in order to reduce the index. Here we want to differentiate **only** the algebraic equation that can reduce the index and do not introduce new differential equation. Here we want to avoid to introduce derivative of the algebraic variable unless is strictly necessary.

We construct a dependency table to identify which constraints to differentiate:

	$v_{1,\text{dot}}$	$v_{3,\text{dot}}$	i
(Constraint 1)	1	1	1
(Constraint 2)	1	1	0
(Constraint 3)	0	0	0

A 0 means no dependence on the corresponding variable, while a 1 indicates dependence. Notice that Constraint 3 does not depend on $v_{1,\text{dot}}$, $v_{3,\text{dot}}$, or i, making it the most suitable candidate for differentiation.

We differentiate Constraint 3:

$$(\text{Constraint 3})' = v_1' - V'(t) = v_{1,\text{dot}} - V'(t)$$

After differentiation, the system becomes:

$$\begin{cases} v_1' = v_{1,\text{dot}} \\ v_3' = v_{3,\text{dot}} \end{cases} \oplus \begin{cases} C(v_{1,\text{dot}} - v_{3,\text{dot}}) - i = 0, & \text{(Constraint 1)} \\ RC(v_{1,\text{dot}} - v_{3,\text{dot}}) - v_3 = 0, & \text{(Constraint 2)} \\ v_{1,\text{dot}} + V'(t) = 0, & \text{(Constraint 3)}' \\ v_1 + V(t) = 0, & \text{(Constraint 3)} \end{cases}$$

Here, the red equations represent the newly derived algebraic equations, while the green equation represents the original constraint that has now been removed from the system. This hidden constraint must still be satisfied, but it can be accounted for through proper initial conditions.

Step 2: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	$v_{1,\text{dot}}$	$v_{3,\text{dot}}$	i
(Constraint 1)	1	1	1
(Constraint 2)	1	1	0
(Constraint 3)$'$	1	0	0

None of the constraints can be differentiated further at this stage. Therefore, we check the Jacobian of the system to determine whether it is singular. If the Jacobian is non-singular, we can solve the system and the DAE will be reduced to index-1.

We analyze the sparsity pattern of the Jacobian matrix:

$$\boldsymbol{S} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & \cdot \\ 1 & \cdot & \cdot \end{pmatrix}$$

This structure suggests that the matrix is non-singular, but we will use MATLAB's sprank function to check for its rank:

```
sparsity_pattern = [
  1 1 1;  % Constraint 1
  1 1 0;  % Constraint 2
  1 0 0]; % Constraint 3'

fprintf('Estimated rank = %d\n', sprank(sparsity_pattern) )

> Estimated rank = 3
```

Since the matrix is structurally non-singular, we proceed by solving for the derivatives:

$$\begin{aligned} v_{1,\text{dot}} &= -V'(t) \\ v_{3,\text{dot}} &= v_{1,\text{dot}} - v_3/(RC) = -V'(t) - v_3/(RC) \\ i &= C(v_{1,\text{dot}} - v_{3,\text{dot}}) = v_3/R \end{aligned}$$

and thus the DAE is of index-1. The DAE is better written as

$$\begin{cases} v_1' = v_{1,\text{dot}} \\ v_3' = v_{3,\text{dot}} \end{cases} \oplus \begin{cases} \begin{pmatrix} C & -C & -1 \\ RC & -RC & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{1,\text{dot}} \\ v_{2,\text{dot}} \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ v_3 \\ -V'(t) \end{pmatrix}, \\ \\ v_1 + V(t) = 0, \quad \text{(Constraint 3)} \end{cases}$$

where the green part are the hidden constraints that are satisfied by choosing the correct initial conditions.

Note that we have done one derivation on the constraints and the DAE is of index 1, thus, the original DAE is of index 2.

30.2 Baumgarte Stabilization

To avoid drift in the hidden constraints, we apply **Baumgarte stabilization**:

$$(\text{Constraints } 3)' + \frac{1}{\tau}(\text{Constraints } 3) = 0$$

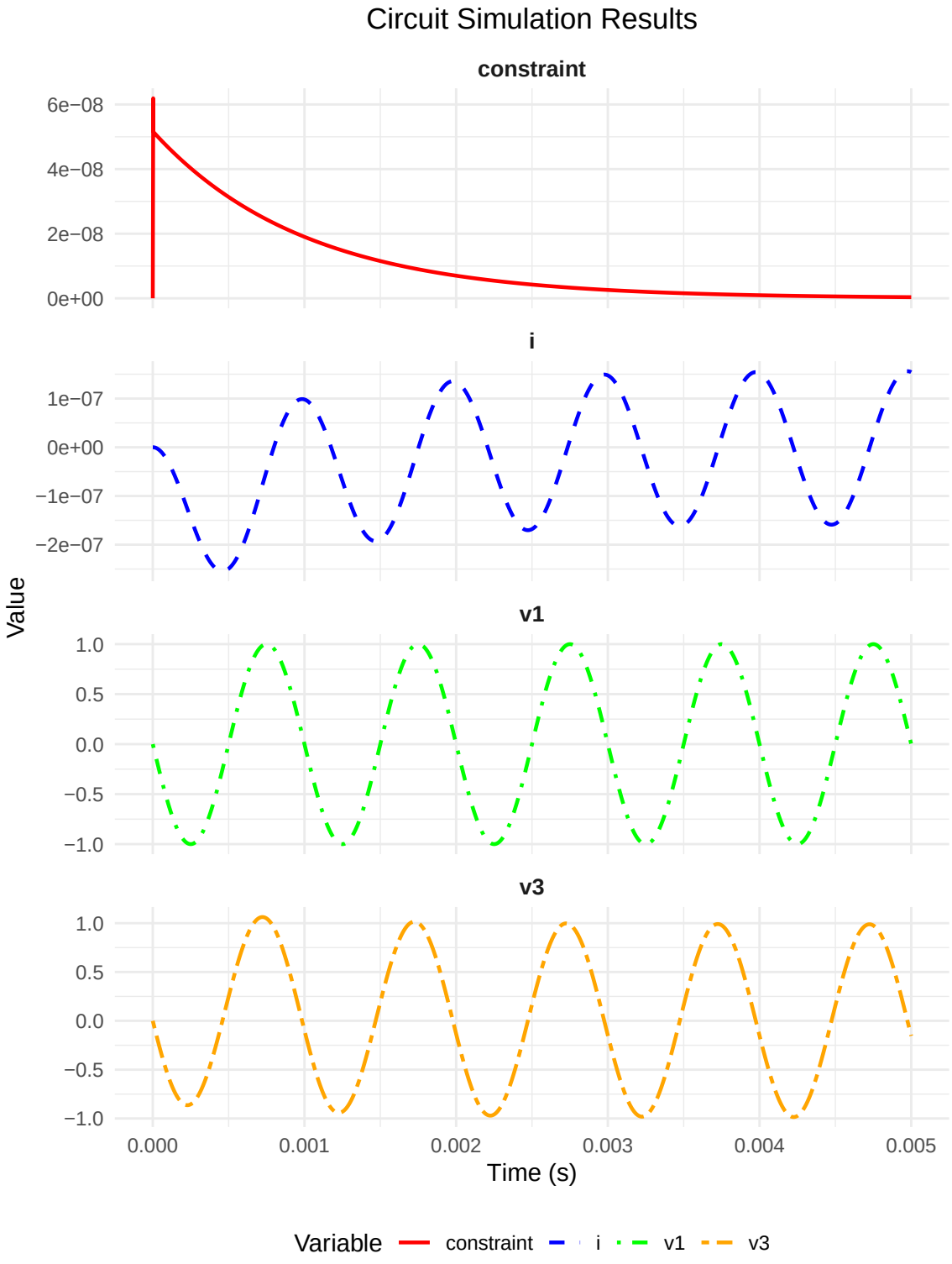
and substituting the expression

$$v_{1,\text{dot}} + V'(t) + \frac{v_1 + V(t)}{\tau} 0$$

The stabilized constraints are:

$$\begin{pmatrix} C & -C & -1 \\ RC & -RC & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{1,\text{dot}} \\ v_{2,\text{dot}} \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ v_3 \\ -V'(t) - (v_1 + V(t))/\tau \end{pmatrix}$$

30.3 Simulation Results



31. Example 8

Example taken from [18] example 2.1.

31.1 The DAE

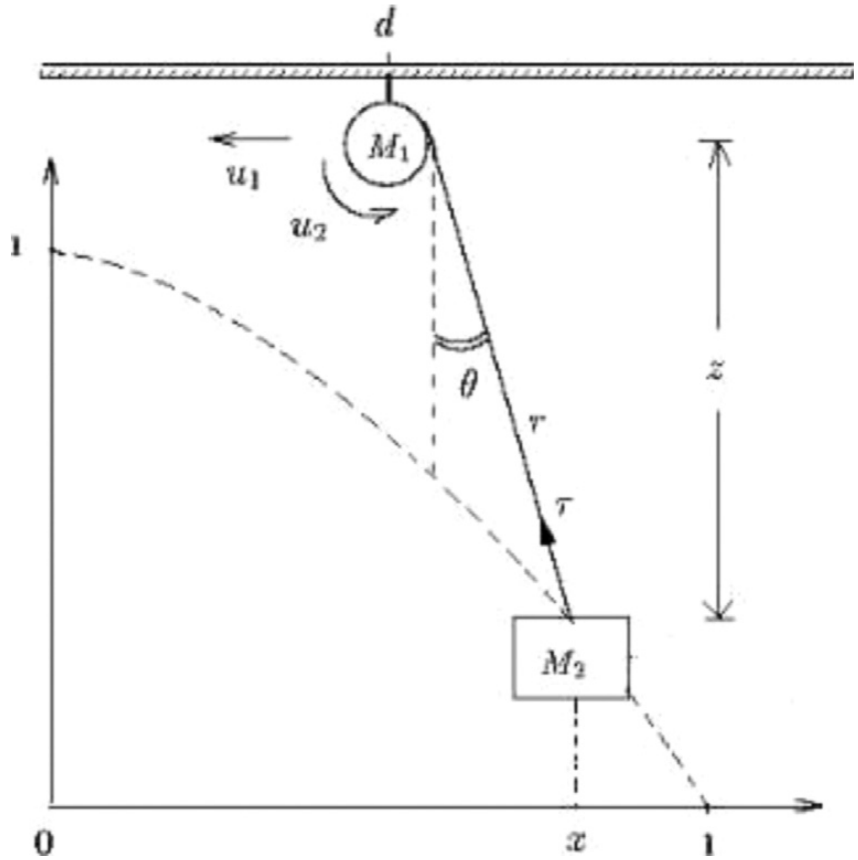


Figure 31.1: Physical model of the DAE

We consider an index-five DAE system that models the planar motion of a crane.
The goal is to determine the horizontal velocity u_1 and the angular velocity u_2 of a winch of mass

M_1, so that the attached load of mass M_2 follows a prescribed path $p_1(t)$ and $p_2(t)$.

The DAEs, derived from Newton's second law, are given by:

$$\begin{cases} x' = v_x \\ z' = v_z \\ d' = v_d \\ r' = v_r \\ M_2 v_x' = -\tau \sin\theta \\ M_2 v_z' = -\tau \cos\theta + mg \\ M_1 v_d' = -C_1 v_d + u_1 + \tau \sin\theta \\ J v_r' = -C_2 v_r - C_3 u_2 + C_3^2 \tau \\ x = r\sin\theta + d \\ z = r\cos\theta \\ x = p_1(t) \\ z = p_2(t) \end{cases}$$

where:

- (x, z) denotes the position of the load,
- τ is the tension in the cable,
- θ is the angle of the cable with the vertical,
- J is the moment of inertia of the winch,
- d is the position of the trolley,
- r is the length of the cable.

The constants satisfy $C_1 > 0$, $C_2 > 0$, $C_3 > 0$, $g > 0$, and $m > 0$.

31.1.1 Semi-Explicit Formulation

For the previous DAE system introduce dummy variables

- $v_{x,\text{dot}}$
- $v_{z,\text{dot}}$
- $v_{d,\text{dot}}$
- $v_{r,\text{dot}}$

and the system decomposes into differential (blue) and algebraic (red) components:

$$\begin{cases} x' = v_x \\ z' = v_z \\ d' = v_d \\ r' = v_r \\ v_x' = v_{x,\text{dot}} \\ v_z' = v_{z,\text{dot}} \\ v_d' = v_{d,\text{dot}} \\ v_r' = v_{r,\text{dot}} \end{cases} \oplus \begin{cases} 0 = M_2 v_{x,\text{dot}} + \tau \sin\theta & \text{(Constraint 1)} \\ 0 = M_2 v_{z,\text{dot}} + \tau \cos\theta - mg & \text{(Constraint 2)} \\ 0 = M_1 v_{d,\text{dot}} + C_1 v_d - u_1 - \tau \sin\theta & \text{(Constraint 3)} \\ 0 = J v_{r,\text{dot}} + C_2 v_r + C_3 u_2 - C_3^2 \tau & \text{(Constraint 4)} \\ 0 = r\sin\theta + d - x & \text{(Constraint 5)} \\ 0 = r\cos\theta - z & \text{(Constraint 6)} \\ 0 = p_1(t) - x & \text{(Constraint 7)} \\ 0 = p_2(t) - z & \text{(Constraint 8)} \end{cases}$$

For this DAE we have:

- **Differential variables**: x, z, d, r, v_x, v_z, v_d, v_r .
- **Algebraic variables**:- $v_{x,\text{dot}}$, $v_{z,\text{dot}}$, $v_{d,\text{dot}}$, $v_{r,\text{dot}}$, u_1, u_2. τ, θ

The algebraic variables do not appears linearly in the constraints. In particular θ appears nonlinear as $\sin\theta$ and $\cos\theta$. Introducing the new dummy variable $\theta_{\text{dot}} = \theta'$ we have for the differential part:

$$\begin{cases} x' = v_x \\ z' = v_z \\ d' = v_d \\ r' = v_r \\ v_x' = v_{x,\text{dot}} \\ v_z' = v_{z,\text{dot}} \\ v_d' = v_{d,\text{dot}} \\ v_r' = v_{r,\text{dot}} \\ \theta' = \theta_{\text{dot}} \end{cases}$$

and now

- **Differential variables**: x, z, d, r, v_x, v_z, v_d, v_r, θ.
- **Algebraic variables**:- $v_{x,\text{dot}}$, $v_{z,\text{dot}}$, $v_{d,\text{dot}}$, $v_{r,\text{dot}}$, u_1, u_2. τ, θ_{dot}

and not constraints are linear in the algebraic variables.

31.1.2 Index Reduction Procedure

We attempt to solve for $v_{x,\text{dot}}$, $v_{z,\text{dot}}$, $v_{d,\text{dot}}$, $v_{r,\text{dot}}$, but the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Now we must derivate the constraints in order to reduce the index. Here we want to differentiate **only** the algebraic equation that can reduce the index and do not introduce new differential equation. Here we want to avoid to introduce derivatove of the algebraic variable unless is strictly necessary.

We construct a dependency table to identify which constraints to differentiate:

	$v_{x,\text{dot}}$	$v_{z,\text{dot}}$	$v_{d,\text{dot}}$	$v_{r,\text{dot}}$	u_1	u_2	τ	θ_{dot}
(Constraint 1)	1	0	0	0	0	0	1	0
(Constraint 2)	0	1	0	0	0	0	1	0
(Constraint 3)	0	0	1	0	1	0	1	0
(Constraint 4)	0	0	0	1	0	1	1	0
(Constraint 5)	0	0	0	0	0	0	0	0
(Constraint 6)	0	0	0	0	0	0	0	0
(Constraint 7)	0	0	0	0	0	0	0	0
(Constraint 8)	0	0	0	0	0	0	0	0

a 0 means the the function do not depend on the variable on the column. A 1 means dependence. The last four constraints are candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus the last constraints and substitute known variable in differential form using the ODE part

$$\begin{aligned}
&\text{(Constraint 5)}' \quad 0 = v_r \sin\theta + r(\cos\theta)\theta_{\text{dot}} + v_d - v_x \\
&\text{(Constraint 6)}' \quad 0 = v_r \cos\theta - r(\sin\theta)\theta_{\text{dot}} - v_z \\
&\text{(Constraint 7)}' \quad 0 = p_1'(t) - v_x \\
&\text{(Constraint 8)}' \quad 0 = p_2'(t) - v_z
\end{aligned}$$

so that algebraic parte of the DAE become:

$$
\begin{cases}
0 = M_2 v_{x,\mathrm{dot}} + \tau \sin\theta & \text{(Constraint 1)} \\
0 = M_2 v_{z,\mathrm{dot}} + \tau \cos\theta - mg & \text{(Constraint 2)} \\
0 = M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau \sin\theta & \text{(Constraint 3)} \\
0 = J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2 \tau & \text{(Constraint 4)} \\
0 = v_r \sin\theta + r(\cos\theta)\theta_{\mathrm{dot}} + v_d - v_x & \text{(Constraint 5)}' \\
0 = v_r \cos\theta - r(\sin\theta)\theta_{\mathrm{dot}} - v_z & \text{(Constraint 6)}' \\
0 = p_1'(t) - v_x & \text{(Constraint 7)}' \\
0 = p_2'(t) - v_z & \text{(Constraint 8)}'
\end{cases}
$$

$$
\begin{cases}
0 = r\sin\theta + d - x & \text{(Constraint 5)} \\
0 = r\cos\theta - z & \text{(Constraint 6)} \\
0 = p_1(t) - x & \text{(Constraint 7)} \\
0 = p_2(t) - z & \text{(Constraint 8)}
\end{cases}
$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine the algebraic variable and must be differentiated again.

Step 2: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	$v_{x,\mathrm{dot}}$	$v_{z,\mathrm{dot}}$	$v_{d,\mathrm{dot}}$	$v_{r,\mathrm{dot}}$	u_1	u_2	τ	θ_{dot}
(Constraint 1)	1	0	0	0	0	0	1	0
(Constraint 2)	0	1	0	0	0	0	1	0
(Constraint 3)	0	0	1	0	1	0	1	0
(Constraint 4)	0	0	0	1	0	1	1	0
(Constraint 5)′	0	0	0	0	0	0	0	1
(Constraint 6)′	0	0	0	0	0	0	0	1
(Constraint 7)′	0	0	0	0	0	0	0	0
(Constraint 8)′	0	0	0	0	0	0	0	0

The last two constraints are candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus the last constraints and substitute known variable in differential form using the ODE part

$$
\begin{aligned}
&\text{(Constraint 7)}'' \quad 0 = p_1''(t) - v_{x,\mathrm{dot}} \\
&\text{(Constraint 8)}'' \quad 0 = p_2''(t) - v_{z,\mathrm{dot}}
\end{aligned}
$$

and constraint part of the DAE become

$$
\begin{cases}
0 = M_2 v_{x,\mathrm{dot}} + \tau \sin\theta & \text{(Constraint 1)} \\
0 = M_2 v_{z,\mathrm{dot}} + \tau \cos\theta - mg & \text{(Constraint 2)} \\
0 = M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau \sin\theta & \text{(Constraint 3)} \\
0 = J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2 \tau & \text{(Constraint 4)} \\
0 = v_r \sin\theta + r(\cos\theta)\theta_{\mathrm{dot}} + v_d - v_x & \text{(Constraint 5)}' \\
0 = v_r \cos\theta - r(\sin\theta)\theta_{\mathrm{dot}} - v_z & \text{(Constraint 6)}' \\
0 = p_1''(t) - v_{x,\mathrm{dot}} & \text{(Constraint 7)}'' \\
0 = p_2''(t) - v_{z,\mathrm{dot}} & \text{(Constraint 8)}''
\end{cases}
$$

$$
\begin{cases}
0 = r\sin\theta + d - x & \text{(Constraint 5)} \\
0 = r\cos\theta - z & \text{(Constraint 6)} \\
0 = p_1(t) - x & \text{(Constraint 7)} \\
0 = p_2(t) - z & \text{(Constraint 8)} \\
0 = p_1'(t) - v_x & \text{(Constraint 7)}' \\
0 = p_2'(t) - v_z & \text{(Constraint 8)}'
\end{cases}
$$

Step 3: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	$v_{x,\mathrm{dot}}$	$v_{z,\mathrm{dot}}$	$v_{d,\mathrm{dot}}$	$v_{r,\mathrm{dot}}$	u_1	u_2	τ	θ_{dot}
(Constraint 1)	1	0	0	0	0	0	1	0
(Constraint 2)	0	1	0	0	0	0	1	0
(Constraint 3)	0	0	1	0	1	0	1	0
(Constraint 4)	0	0	0	1	0	1	1	0
(Constraint 5)′	0	0	0	0	0	0	0	1
(Constraint 6)′	0	0	0	0	0	0	0	1
(Constraint 7)″	1	0	0	0	0	0	0	0
(Constraint 8)″	0	1	0	0	0	0	0	0

There are no constraints candidate for derivation. We can check if the constraints are enough to determine algebraic variable.

We use sprank of MATLAB to check if the matrix can be non singular.

```
sparsity_pattern = [...
1, 0, 0, 0, 0, 0, 1, 0; ...
0, 1, 0, 0, 0, 0, 1, 0; ...
0, 0, 1, 0, 1, 0, 1, 0; ...
0, 0, 0, 1, 0, 1, 1, 0; ...
0, 0, 0, 0, 0, 0, 0, 1; ...
0, 0, 0, 0, 0, 0, 0, 1; ...
1, 0, 0, 0, 0, 0, 0, 0; ...
0, 1, 0, 0, 0, 0, 0, 0; ...
];
```

```
fprintf('Estimated rank = %d\n', sprank(sparsity_pattern) );
```

```
> Estimated rank = 6
```

Matrix is not full rank. To continue the algorithm we can apply elimination to remove redundancy. First of all rewrite the constraint as a linear system

$$\underbrace{\begin{bmatrix} M_2 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & 0 \\ 0 & M_2 & 0 & 0 & 0 & 0 & \cos(\theta) & 0 \\ 0 & 0 & M_1 & 0 & -1 & 0 & -\sin(\theta) & 0 \\ 0 & 0 & 0 & J & 0 & C_3 & -C_3^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r\cos(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r\sin(\theta) \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{A}} \begin{pmatrix} v_{x,\mathrm{dot}} \\ v_{z,\mathrm{dot}} \\ v_{d,\mathrm{dot}} \\ v_{r,\mathrm{dot}} \\ u_1 \\ u_2 \\ \tau \\ \theta_{\mathrm{dot}} \end{pmatrix} = \underbrace{\begin{bmatrix} 0 \\ mg \\ -C_1 v_d \\ -C_2 v_r \\ v_x - v_d - v_r \sin(\theta) \\ v_z - v_r \cos(\theta) \\ -p_1''(t) \\ -p_2''(t) \end{bmatrix}}_{\boldsymbol{b}}$$

given $\boldsymbol{L}_1$ defined forward we have

$$\boldsymbol{L}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ M_2^{-1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & M_2^{-1} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{L}_1\boldsymbol{A} = \begin{bmatrix} M_2 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & 0 \\ 0 & M_2 & 0 & 0 & 0 & 0 & \cos(\theta) & 0 \\ 0 & 0 & M_1 & 0 & -1 & 0 & -\sin(\theta) & 0 \\ 0 & 0 & 0 & J & 0 & C_3 & -C_3^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r\cos(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r\sin(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & M_2^{-1}\sin(\theta) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & M_2^{-1}\cos(\theta) & 0 \end{bmatrix}$$

given $\boldsymbol{R}_1$ defined forward we have

$$\boldsymbol{R}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos(\theta) & -\sin(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos(\theta) & -\sin(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\boldsymbol{R}_1\boldsymbol{L}_1\boldsymbol{A} = \begin{bmatrix} M_2 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & 0 \\ 0 & M_2 & 0 & 0 & 0 & 0 & \cos(\theta) & 0 \\ 0 & 0 & M_1 & 0 & -1 & 0 & -\sin(\theta) & 0 \\ 0 & 0 & 0 & J & 0 & C_3 & -C_3^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & M_2^{-1} & 0 \end{bmatrix}$$

$\boldsymbol{R}_1\boldsymbol{L}_1\boldsymbol{A}$ have two rows zero, thus multiplying by $\boldsymbol{R}_1\boldsymbol{L}_1$ the constraints permits to find two algebraic constraints that can be derived.

$$\boldsymbol{R}_1\boldsymbol{L}_1 \begin{bmatrix} M_2 v_{x,\mathrm{dot}} + \tau \sin\theta \\ M_2 v_{z,\mathrm{dot}} + \tau\cos\theta - mg \\ M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau\sin\theta \\ J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2\tau \\ v_r \sin\theta + r(\cos\theta)\theta_{\mathrm{dot}} + v_d - v_x \\ v_r\cos\theta - r(\sin\theta)\theta_{\mathrm{dot}} - v_z \\ p_1''(t) - v_{x,\mathrm{dot}} \\ p_2''(t) - v_{z,\mathrm{dot}} \end{bmatrix} = \begin{bmatrix} M_2 v_{x,\mathrm{dot}} + \tau \sin(\theta) \\ M_2 v_{z,\mathrm{dot}} + \tau\cos(\theta) - mg \\ M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau\sin(\theta) \\ J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2\tau \\ v_d - v_x + \cos(\theta) + r\theta_{\mathrm{dot}} + \sin(\theta) v_z \\ (v_d - v_x)\sin(\theta) - \cos(\theta) v_z + v_r \\ \cos(\theta) p_1''(t) - p_2''(t)\sin(\theta) + M_2^{-1}\sin(\theta) mg \\ \cos(\theta) p_2''(t) + p_1''(t)\sin(\theta) + M_2^{-1}(\tau - \cos(\theta)) mg \end{bmatrix}$$

Collecting the recombined constraints

$$\begin{cases} 0 = M_2 v_{x,\mathrm{dot}} + \tau \sin\theta & \text{(C1)} \\ 0 = M_2 v_{z,\mathrm{dot}} + \tau\cos(\theta) - mg & \text{(C2)} \\ 0 = M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau\sin(\theta) & \text{(C3)} \\ 0 = J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2\tau & \text{(C4)} \\ 0 = v_d - v_x + \cos(\theta) + r\theta_{\mathrm{dot}} + \sin(\theta) v_z & \text{(C5)} \\ 0 = (v_d - v_x)\sin(\theta) - \cos(\theta) v_z + v_r & \text{(C6)} \\ 0 = \cos(\theta) p_1''(t) - p_2''(t)\sin(\theta) + M_2^{-1}\sin(\theta) mg & \text{(C7)} \\ 0 = \cos(\theta) p_2''(t) + p_1''(t)\sin(\theta) + M_2^{-1}(\tau - \cos(\theta)) mg & \text{(C8)} \end{cases}$$

where we have renamed constraints because now are mixed by the multiplication of $\boldsymbol{L}^{-1}$. The constraints (C6) and (C7) can be derived.

$$\text{(C6)}' = ((v_d - v_x)\theta_{\mathrm{dot}} - v_{z,\mathrm{dot}})\cos(\theta) + (\theta_{\mathrm{dot}} v_z + v_{d,\mathrm{dot}} - v_{x,\mathrm{dot}})\sin(\theta) + v_{r,\mathrm{dot}}$$

$$\text{(C7)}' = p_1'''(t)\cos(\theta) - p_2'''(t)\sin(\theta) - \left(p_2''(t)\cos(\theta) + p_1''(t)\sin(\theta)\right)\theta_{\mathrm{dot}} + M_2^{-1}\theta_{\mathrm{dot}}\cos(\theta) mg$$

and the constraints and hidden constraints become:

$$
\begin{cases}
0 = M_2 v_{x,\mathrm{dot}} + \tau \sin\theta & \text{(C1)} \\
0 = M_2 v_{z,\mathrm{dot}} + \tau \cos(\theta) - mg & \text{(C2)} \\
0 = M_1 v_{d,\mathrm{dot}} + C_1 v_d - u_1 - \tau \sin(\theta) & \text{(C3)} \\
0 = J v_{r,\mathrm{dot}} + C_2 v_r + C_3 u_2 - C_3^2 \tau & \text{(C4)} \\
0 = v_d - v_x + \cos(\theta) + r\theta_{\mathrm{dot}} + \sin(\theta) v_z & \text{(C5)} \\
0 = ((v_d - v_x)\theta_{\mathrm{dot}} - v_{z,\mathrm{dot}})\cos(\theta) + (\theta_{\mathrm{dot}} v_z + v_{d,\mathrm{dot}} - v_{x,\mathrm{dot}})\sin(\theta) + v_{r,\mathrm{dot}} & \text{(C6)}' \\
0 = p_1'''(t)\cos(\theta) - p_2'''(t)\sin(\theta) - \left(p_2''(t)\cos(\theta) + p_1''(t)\sin(\theta)\right)\theta_{\mathrm{dot}} + M_2^{-1}\theta_{\mathrm{dot}}\cos(\theta) mg & \text{(C7)}' \\
0 = \cos(\theta) p_2''(t) + p_1''(t)\sin(\theta) + M_2^{-1}(\tau - \cos(\theta) mg & \text{(C8)}
\end{cases}
$$

$$
\begin{cases}
0 = r\sin\theta + d - x & \text{(Constraint 5)} \\
0 = r\cos\theta - z & \text{(Constraint 6)} \\
0 = p_1(t) - x & \text{(Constraint 7)} \\
0 = p_2(t) - z & \text{(Constraint 8)} \\
0 = p_1'(t) - v_x & \text{(Constraint 7)}' \\
0 = p_2'(t) - v_z & \text{(Constraint 8)}' \\
0 = (v_d - v_x)\sin(\theta) - \cos(\theta) v_z + v_r & \text{(C6)} \\
0 = \cos(\theta) p_1''(t) - p_2''(t)\sin(\theta) + M_2^{-1}\sin(\theta) mg & \text{(C7)}
\end{cases}
$$

Step 4: Differentiate the algebraic constraints?

To find the right candidate(s) here we write a table of the dependencies

	$v_{x,\mathrm{dot}}$	$v_{z,\mathrm{dot}}$	$v_{d,\mathrm{dot}}$	$v_{r,\mathrm{dot}}$	u_1	u_2	τ	θ_{dot}
(C1)	1	0	0	0	0	0	1	0
(C2)	0	1	0	0	0	0	1	0
(C3)	0	0	1	0	1	0	1	0
(C4)	0	0	0	1	0	1	1	0
(C5)	0	0	0	0	0	0	0	1
(C6)′	1	1	1	1	0	0	0	1
(C7)′	0	0	0	0	0	0	0	1
(C8)	0	0	0	0	0	0	1	0

There are no constraints candidate for derivation. We can check if the constraints are enough to determine algebraic variable.

We use sprank of MATLAB to check if the matrix can be non singular.

```
sparsity_pattern = [...
1, 0, 0, 0, 0, 0, 1, 0;...
0, 1, 0, 0, 0, 0, 1, 0;...
0, 0, 1, 0, 1, 0, 1, 0;...
0, 0, 0, 1, 0, 1, 1, 0;...
0, 0, 0, 0, 0, 0, 0, 1;...
1, 1, 1, 1, 0, 0, 0, 1;...
```

```
0, 0, 0, 0, 0, 0, 0, 1;...
0, 0, 0, 0, 0, 0, 1, 0;...
];

fprintf('Estimated rank = %d\n', sprank(sparsity_pattern) );

> Estimated rank = 7
```

the matrix is singular. Instead of apply LU try to combine (C5) and (C7)$'$ to eliminate θ_{dot}

$$\begin{aligned}\widetilde{\text{C7}} &= \text{(C5)}\left(M_2^{-1}mg\cos(\theta) - p_2''(t)\cos(\theta) - p_1''(t)\sin(\theta)\right) + \text{(C7)}'r \\ &= \left(\cos(\theta)p_1'(t) - \cos(\theta)v_d - \sin(\theta)p_2'(t)\right)\left(\cos(\theta)p_2''(t) + \sin(\theta)p_1''(t)\right) \\ &+ \left(\sin(\theta)p_2'''(t) - \cos(\theta)p_1'''(t)\right)r \\ &+ M_2^{-1}\left(\cos(\theta)v_d - \cos(\theta)p_1'(t) + \sin(\theta)p_2'(t)\right)\cos(\theta)mg\end{aligned}$$

and

$$\begin{aligned}\widetilde{\text{C7}}' &= \left(M_2^{-1}\left(2p_2'\theta_{\text{dot}} - p_1'' + v_{d,\text{dot}}\right)mg + \left(p_1' - v_d\right)p_2''' + p_1''p_2'' + 2\theta_{\text{dot}}(p_1' - v_d)p_1'' - \left(2\theta_{\text{dot}}p_2' + v_{d,\text{dot}}\right)p_2''\right)\cos(\theta)^2 \\ &+ \left(\left(M_2^{-1}\left(p_2''(t) + 2\theta_{\text{dot}}(p_1'(t) - v_d)\right)mg + (p_1' - v_d)p_1''' - p_2'''p_2' + p_1''p_1'' - p_2''p_2'' + \left(-2\theta_{\text{dot}}p_2' - v_{d,\text{dot}}\right)p_1'' + \left(-2\theta_{\text{d}}\right.\right.\right. \\ &+ \left(-p_2'p_1''' - p_2''p_1''\right)\sin(\theta)^2 + \\ &\left(r\theta_{\text{dot}}p_1''' + rp_2'''' + v_rp_2'''\right)\sin(\theta) - M_2^{-1}p_2'\theta_{\text{dot}}mg + \left(-\theta_{\text{dot}}p_1' + v_d\theta_{\text{dot}}\right)p_1'' + p_2''\theta_{\text{dot}}p_2'\end{aligned}$$

TO BE COMPLETED

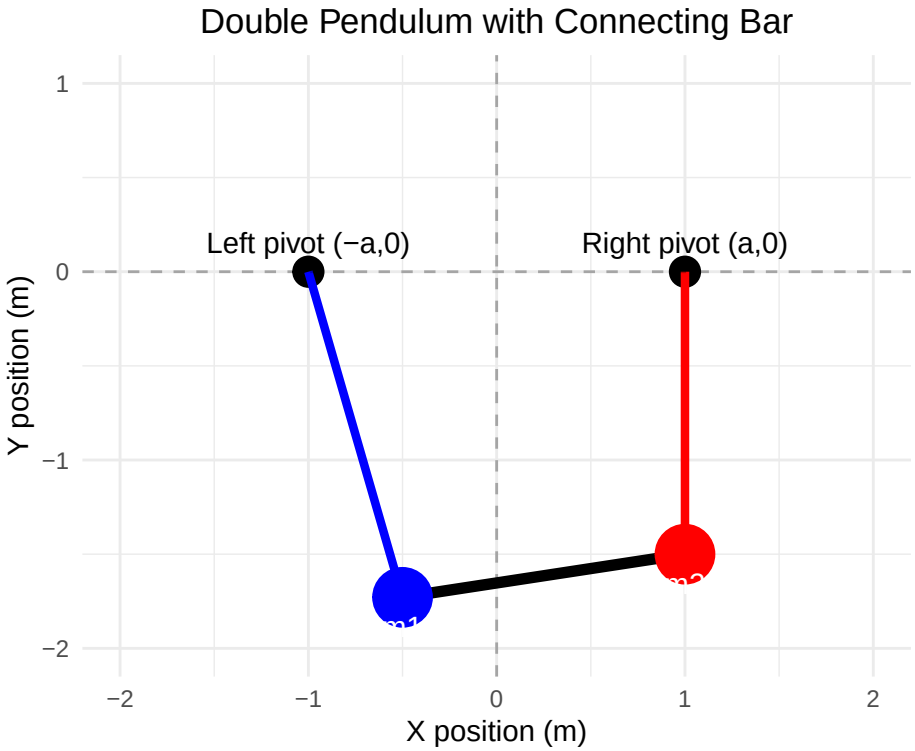


32. Example 9

A constrained mechanical system consisting of two rigid pendulums connected by a fixed-length bar. In details

1. Left Pendulum Assembly:
   - Pivot: Fixed at point $(-a, 0)$
   - Mass m_1 located at (x_1, y_1)
   - Constraint: Rigid rod maintains length $\ell_1^2 = (x_1 + a)^2 + y_1^2$
2. Right Pendulum Assembly:
   - Pivot: Fixed at point $(a, 0)$
   - Mass m_2 located at (x_2, y_2)
   - Length $\ell_2^2 = (x_2 - a)^2 + y_2^2$ fixed
3. Coupling Mechanism:
   - Connecting bar: Rigid link between m_1 and m_2
   - Constraint: Rigid rod maintains length $\ell_3^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$

32.1 Double Pendulum System with Connecting Bar



DAE System Derivation

Now the Differential-Algebraic Equations (DAE) for the double pendulum system with a connecting bar is derived.

We employ Cartesian coordinates with the following state variables:

- Position states:
  - (x_1, y_1): Coordinates of left mass m_1
  - (x_2, y_2): Coordinates of right mass m_2
- Velocity states:
  - (u_1, v_1): Velocity components of m_1
  - (u_2, v_2): Velocity components of m_2
- Constraint forces:
  - λ_1: Multiplier for left pendulum constraint
  - λ_2: Multiplier for right pendulum constraint
  - λ_3: Multiplier for connecting bar constraint

Applying Lagrange multiplier method with constraints yields:

Equations of Motion

For mass m_1

$$\begin{cases} m_1 u_1' = -\lambda_1 (x_1 + a) - \lambda_3 (x_1 - x_2) \\ m_1 v_1' = -\lambda_1 y_1 - \lambda_3 (y_1 - y_2) - m_1 g \end{cases}$$

For mass m_2

$$\begin{cases} m_2 u_2' = -\lambda_2 (x_2 - a) + \lambda_3 (x_1 - x_2) \\ m_2 v_2' = -\lambda_2 y_2 + \lambda_3 (y_1 - y_2) - m_2 g \end{cases}$$

Constraint Equations (Algebraic)

$$\begin{cases} (x_1+a)^2+y_1^2=\ell_1^2 \\ (x_2-a)^2+y_2^2=\ell_2^2 \\ (x_1-x_2)^2+(y_1-y_2)^2=\ell_3^2 \end{cases}$$

32.1.1 Complete DAE Structure

Introducing

$$\delta x = x_1 - x_2, \quad \delta y = y_1 - y_2, \quad \delta u = u_1 - u_2, \quad \delta v = v_1 - v_2,$$

the complete DAE system is:

Differential Part

$$\begin{cases} x_1' = u_1 \\ y_1' = v_1 \\ x_2' = u_2 \\ y_2' = v_2 \\ m_1 u_1' = -(x_1+a)\lambda_1 - \delta x \lambda_3 \\ m_1 v_1' = -y_1\lambda_1 - \delta y \lambda_3 - m_1 g \\ m_2 u_2' = -(x_2-a)\lambda_2 + \delta x \lambda_3 \\ m_2 v_2' = -y_2\lambda_2 + \delta y \lambda_3 - m_2 g \end{cases}$$

Algebraic Part

$$\begin{cases} (x_1+a)^2+y_1^2-\ell_1^2=0, & \text{(Constraint 1)} \\ (x_2-a)^2+y_2^2-\ell_2^2=0, & \text{(Constraint 2)} \\ \delta x^2+\delta y^2-\ell_3^2=0, & \text{(Constraint 3)} \end{cases}$$

32.1.2 Semi-Explicit Formulation

Decomposes previous DAE into differential (blue) and algebraic (red) components:

$$\begin{cases} x_1' = u_1 \\ y_1' = v_1 \\ x_2' = u_2 \\ y_2' = v_2 \\ u_1' = \dfrac{-(x_1+a)\lambda_1 - \delta x\lambda_3}{m_1} \\ v_1' = \dfrac{-y_1\lambda_1 - \delta y\lambda_3}{m_1} - g \\ u_2' = \dfrac{-(x_2-a)\lambda_2 + \delta x\lambda_3}{m_2} \\ v_2' = \dfrac{-y_2\lambda_2 + \delta y\lambda_3}{m_2} - g \end{cases} \oplus \begin{cases} (x_1+a)^2+y_1^2-\ell_1^2=0, & \text{(Constraint 1)} \\ (x_2-a)^2+y_2^2-\ell_2^2=0, & \text{(Constraint 2)} \\ \delta x^2+\delta y^2-\ell_3^2=0, & \text{(Constraint 3)} \end{cases}$$

For this DAE we have:
- **Differential variables**: $x_1, y_1, x_2, y_2, u_1, v_1, u_2, v_2$.
- **Algebraic variables**: $\lambda_1, \lambda_2, \lambda_3$.

no dummy variable are required.

32.1.3 Index Reduction Procedure

We attempt to solve for λ_1, λ_2 and λ_3 but the system is not yet solvable explicitly. We proceed by **differentiating**.

Step 1: Differentiate algebraic constraints

Now we must derivate the constraints in order to reduce the index. Here we want to differentiate **only** the algebraic equation that can reduce the index and do not introduce new differential equation. Here we want to avoid to introduce derivatove of the algebraic variable unless is strictly necessary.

We construct a dependency table to identify which constraints to differentiate:

	λ_1	λ_2	λ_3
(Constraint 1)	0	0	0
(Constraint 2)	0	0	0
(Constraint 3)	0	0	0

a 0 means the the function do not depend on the variable on the column. A 1 means dependence. All constraint are candidate for differentiation as it doesn't introduce new algebraic variables. Differentiate thus the constraints and substitute known variable in differential form using the ODE part

$$\begin{aligned}
\text{(Constraint 1)}' &= 2(x_1+a)x_1' + 2y_1y_1' = 2(x_1+a)u_1 + 2y_1v_1 \\
\text{(Constraint 2)}' &= 2(x_2-a)x_2' + 2y_2y_2' = 2(x_2-a)u_2 + 2y_2v_2 \\
\text{(Constraint 3)}' &= 2\delta x\delta x' + 2\delta y\delta y' = 2\delta x\delta u + 2\delta y\delta v
\end{aligned}$$

so that the DAE become:

$$\begin{cases}
x_1' = u_1 \\
y_1' = v_1 \\
x_2' = u_2 \\
y_2' = v_2 \\
u_1' = \dfrac{-(x_1+a)\lambda_1 - \delta x\lambda_3}{m_1} \\
v_1' = \dfrac{-y_1\lambda_1 - \delta y\lambda_3}{m_1} - g \\
u_2' = \dfrac{-(x_2-a)\lambda_2 + \delta x\lambda_3}{m_2} \\
v_2' = \dfrac{-y_2\lambda_2 + \delta y\lambda_3}{m_2} - g
\end{cases}
\oplus
\begin{cases}
2(x_1+a)u_1 + 2y_1v_1 = 0, & \text{(Constraint 1)}' \\
2(x_2-a)u_2 + 2y_2v_2 = 0, & \text{(Constraint 2)}' \\
2\delta x\delta u + 2\delta y\delta v = 0, & \text{(Constraint 3)}' \\
(x_1+a)^2 + y_1^2 - \ell_1^2 = 0, & \text{(Constraint 1)} \\
(x_2-a)^2 + y_2^2 - \ell_2^2 = 0, & \text{(Constraint 2)} \\
\delta x^2 + \delta y^2 - \ell_3^2 = 0, & \text{(Constraint 3)}
\end{cases}$$

The red algebraic equations are the new algebraic equation after the derivation. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

The red part cannot determine the algebraic variable and must be differentiated again.

Step 2: Differentiate (again) the algebraic constraints

To find the right candidate(s) here we write a table of the dependencies

	λ_1	λ_2	λ_3
(Constraint 1)$'$	0	0	0
(Constraint 2)$'$	0	0	0
(Constraint 3)$'$	0	0	0

Again all the constraints can be differentiated

$$\begin{aligned}
\text{(Constraint 1)}'' &= 2x_1'u_1 + 2y_1'v_1 + 2(x_1+a)u_1' + 2y_1v_1' \\
&= 2(u_1^2+v_1^2) + 2(x_1+a)u_1' + 2y_1v_1' \\
\text{(Constraint 2)}'' &= 2x_2'u_2 + 2y_2'v_2 + 2(x_2-a)u_2' + 2y_2v_2' \\
&= 2(u_2^2+v_2^2) + 2(x_2-a)u_2' + 2y_2v_2' \\
\text{(Constraint 3)}'' &= 2\delta x'\delta u + 2\delta x\delta u' + 2\delta y'\delta v + 2\delta y\delta v' \\
&= 2(\delta u^2+\delta v^2) + 2\delta x\delta u' + 2\delta y\delta v'
\end{aligned}$$

the substitution of the derivative make the constraint complex. Thus, dummy variable are introduced to reduce complexity:

$$\begin{aligned}
\text{(Constraint 1)}'' &= 2(u_1^2+v_1^2) + 2(x_1+a)u_{1,\text{dot}} + 2y_1v_{1,\text{dot}} \\
\text{(Constraint 2)}'' &= 2(u_2^2+v_2^2) + 2(x_2-a)u_{2,\text{dot}} + 2y_2v_{2,\text{dot}} \\
\text{(Constraint 3)}'' &= 2(\delta u^2+\delta v^2) + 2\delta x(u_{1,\text{dot}}-u_{2,\text{dot}}) + 2\delta y(v_{1,\text{dot}}-v_{2,\text{dot}})
\end{aligned}$$

and the DAE, introducing

$$\delta u_{\text{dot}} = u_{1,\text{dot}} - u_{2,\text{dot}}, \qquad \delta v_{\text{dot}} = v_{1,\text{dot}} - v_{2,\text{dot}}$$

become:

$$
\begin{cases}
x_1' = u_1 \\
y_1' = v_1 \\
x_2' = u_2 \\
y_2' = v_2 \\
u_1' = u_{1,\text{dot}} \\
v_1' = v_{1,\text{dot}} \\
u_2' = u_{2,\text{dot}} \\
v_2' = v_{2,\text{dot}}
\end{cases}
\oplus
\begin{cases}
2(u_1^2+v_1^2)+2(x_1+a)u_{1,\text{dot}}+2y_1v_{1,\text{dot}}=0, & \text{(Constraint 1)}'' \\
2(u_2^2+v_2^2)+2(x_2-a)u_{2,\text{dot}}+2y_2v_{2,\text{dot}}=0, & \text{(Constraint 2)}'' \\
2(\delta u^2+\delta v^2)+2\delta x\delta u_{\text{dot}}+2\delta y\delta v_{\text{dot}}=0, & \text{(Constraint 3)}'' \\
m_1u_{1,\text{dot}}+(x_1+a)\lambda_1+\delta x\lambda_3=0, & \text{(Constraint 4)} \\
m_1v_{1,\text{dot}}+y_1\lambda_1+\delta y\lambda_3+m_1g=0, & \text{(Constraint 5)} \\
m_2u_{2,\text{dot}}+(x_2-a)\lambda_2-\delta x\lambda_3=0, & \text{(Constraint 6)} \\
m_2v_{2,\text{dot}}+y_2\lambda_2-\delta y\lambda_3+m_2g=0, & \text{(Constraint 7)} \\
\\
2(x_1+a)u_1+2y_1v_1=0, & \text{(Constraint 1)}' \\
2(x_2-a)u_2+2y_2v_2=0, & \text{(Constraint 2)}' \\
2\delta x\delta u+2\delta y\delta v=0, & \text{(Constraint 3)}' \\
\\
(x_1+a)^2+y_1^2-\ell_1^2=0, & \text{(Constraint 1)} \\
(x_2-a)^2+y_2^2-\ell_2^2=0, & \text{(Constraint 2)} \\
\delta x^2+\delta y^2-\ell_3^2=0, & \text{(Constraint 3)}
\end{cases}
$$

The red algebraic equations are the new algebraic equation after the derivation. Notice the four new algebraic constraints added from the introduction of dummy variables. The green algebraic equation is the hidden constraints that is removed from the DAE system but that is satisfied by choosing the right initial conditions.

Step 3: Differentiate the algebraic constraints?

To find the right candidate(s) here we write a table of the dependencies. Notice that now the table is larger due to the introduction of new dummy variables and four new constraints

	λ_1	λ_2	λ_3	$u_{1,\text{dot}}$	$v_{1,\text{dot}}$	$u_{2,\text{dot}}$	$v_{2,\text{dot}}$
(Constraint 1)$''$	0	0	0	1	1	0	0
(Constraint 2)$''$	0	0	0	0	0	1	1
(Constraint 3)$''$	0	0	0	1	1	1	1
(Constraint 4)	1	0	1	1	0	0	0
(Constraint 5)	1	0	1	0	1	0	0
(Constraint 6)	0	1	1	0	0	1	0
(Constraint 7)	0	1	1	0	0	0	1

Again all the constraints **cannot** be differentiated. At this point we must check if the jacobian of the constraints is **non-singular**. If is non-singular formally we can solve for λ_1, λ_2, λ_3, $u_{1,\text{dot}}$, $v_{1,\text{dot}}$,

$u_{2,\text{dot}}$ and $v_{2,\text{dot}}$ and the DAE is of index-1 and we can stop index reduction. To check if jacobian is singular before to try LU decomposion we analize the **nonzero** structure of the jacobian $\boldsymbol{S}$

$$\boldsymbol{S} = \begin{pmatrix} \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 \\ 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & 1 & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & 1 & \cdot & \cdot & \cdot & 1 \end{pmatrix}$$

In this form is difficult to understand if a matrix with nonzero at the 1 position can be non-singular.

We use sprank of MATLAB to check if the matrix can be non singular.

```
sparsity_pattern = [
  0 0 0 1 1 0 0;  % Constraint 1''
  0 0 0 0 0 1 1;  % Constraint 2''
  0 0 0 1 1 1 1;  % Constraint 3''
  1 0 1 1 0 0 0;  % Constraint 4
  1 0 1 0 1 0 0;  % Constraint 5
  0 1 1 0 0 1 0;  % Constraint 6
  0 1 1 0 0 0 1]; % Constraint 7

fprintf('Estimated rank = %d\n', sprank(sparsity_pattern) )
```

```
> Estimated rank = 7
```

The estimated rank is 7 so that matrix is **structurally non-simgular**.

To better understand how it work consider the Dulmage-Mendelsohn decomposition that find rows and columns permutation for maximum mathcing

```
[p,q,r,s,cc,rr] = dmperm(sparsity_pattern)

sparsity_pattern(p,q)

ans =

     1     0     1     1     0     0     0
     0     1     1     0     0     1     0
     1     0     1     0     1     0     0
     0     0     0     1     1     0     0
     0     0     0     1     1     1     1
     0     0     0     0     0     1     1
     0     1     1     0     0     0     1
```

you can see that the diagonal is all 1 thus a matrix with element non zero only on the diagonal is surely non singular. In practice

- do Dulmage-Mendelsohn decomposition
- Apply rows and columns permutation
- If the diadonal is non zeros (all 1) the matrix is structurally non singular.

Here, here we can do a step more because the system is linear. Let

$$\boldsymbol{X} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ u_1 \\ v_1 \\ u_2 \\ v_2 \end{pmatrix}, \quad \boldsymbol{Z} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ u_{1,\text{dot}} \\ v_{1,\text{dot}} \\ u_{2,\text{dot}} \\ v_{2,\text{dot}} \end{pmatrix}$$

then constraint become

$$\boldsymbol{A}(\boldsymbol{X},t)\boldsymbol{Z} = \boldsymbol{b}(\boldsymbol{X},t)$$

where

$$\boldsymbol{A}(\boldsymbol{X},t) = \begin{pmatrix} 0 & 0 & 0 & x_1+a & y_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_2-a & y_2 \\ 0 & 0 & 0 & \delta x & \delta y & -\delta x & -\delta y \\ x_1+a & 0 & \delta x & m_1 & 0 & 0 & 0 \\ y_1 & 0 & \delta y & 0 & m_1 & 0 & 0 \\ 0 & x_2-a & -\delta x & 0 & 0 & m_2 & 0 \\ 0 & y_2 & -\delta y & 0 & 0 & 0 & m_2 \end{pmatrix}$$

and

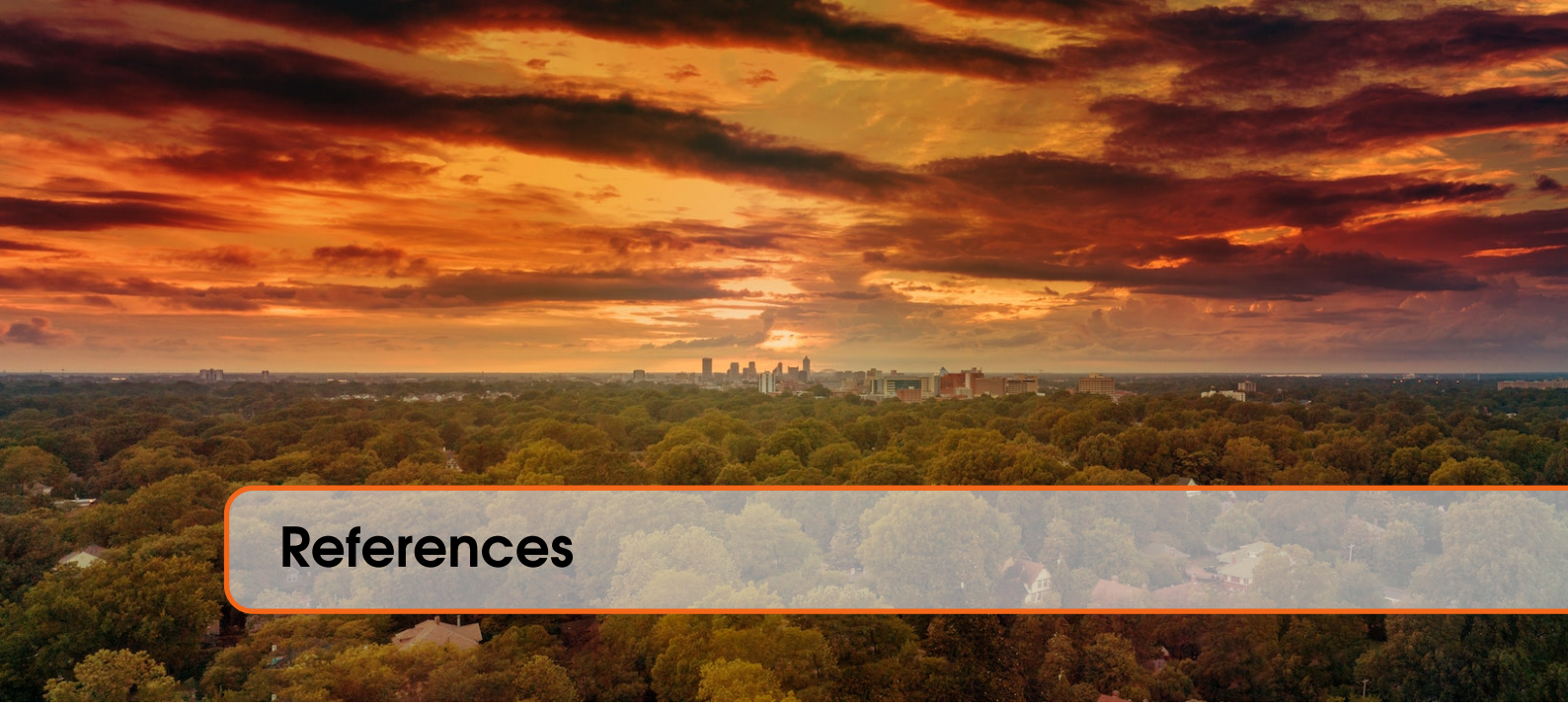
$$\boldsymbol{b}(\boldsymbol{X},t) = -\begin{pmatrix} u_1^2+v_1^2 \\ u_2^2+v_2^2 \\ \delta u^2+\delta v^2 \\ 0 \\ m_1 g \\ 0 \\ m_2 g \end{pmatrix}$$

and the DAE become

$$\begin{cases} x_1' = u_1 \\ y_1' = v_1 \\ x_2' = u_2 \\ y_2' = v_2 \\ u_1' = u_{1,\text{dot}} \\ v_1' = v_{1,\text{dot}} \\ u_2' = u_{2,\text{dot}} \\ v_2' = v_{2,\text{dot}} \\ \boldsymbol{A}(\boldsymbol{X},t)\boldsymbol{Z} = \boldsymbol{b}(\boldsymbol{X},t) \end{cases} \bigoplus \begin{cases} 2(x_1+a)u_1 + 2y_1v_1 = 0, & \text{(Constraint 1)}' \\ 2(x_2-a)u_2 + 2y_2v_2 = 0, & \text{(Constraint 2)}' \\ 2\delta x\delta u + 2\delta y\delta v = 0, & \text{(Constraint 3)}' \\ \\ (x_1+a)^2 + y_1^2 - \ell_1^2 = 0, & \text{(Constraint 1)} \\ (x_2-a)^2 + y_2^2 - \ell_2^2 = 0, & \text{(Constraint 2)} \\ \delta x^2 + \delta y^2 - \ell_3^2 = 0, & \text{(Constraint 3)} \end{cases}$$

where the green part are the hidden constraints that are satisfied by choosing the correct initial conditions.

Completion will follow the conclusion of the second part of the assignment.



References

[1] Vladimir Igorevich Arnold. 1989. *Mathematical methods of classical mechanics*. Springer Science & Business Media.

[2] Uri M Ascher, Hongsheng Chin, Linda R Petzold, and Sebastian Reich. 1995. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Structural Mechanics* 23, 2 (1995), 135–157.

[3] Joachim Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering* 1, 1 (1972), 1–16.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold. 1995. *Numerical solution of initial-value problems in differential-algebraic equations*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611971224

[5] J. C. Butcher. 2008. *Numerical methods for ordinary differential equations* (2nd ed.). John Wiley & Sons. https://doi.org/10.1002/9780470753767

[6] Earl A Coddington and Norman Levinson. 1955. *Theory of ordinary differential equations*. Tata McGraw-Hill Education.

[7] Feliks R Gantmacher. 1964. *The theory of matrices*. Taylor & Francis.

[8] Gene H. Golub and Charles F. Van Loan. 2013. *Matrix computations* (4th ed.). Johns Hopkins University Press.

[9] E. Hairer, S. P. Nørsett, and G. Wanner. 1993. *Solving ordinary differential equations i: Nonstiff problems* (2nd ed.). Springer. https://doi.org/10.1007/978-3-540-78862-1

[10] Joao P Hespanha. 2009. *Linear systems theory*. Princeton university press.

[11] Arieh Iserles. 2009. *A first course in the numerical analysis of differential equations* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511995569

[12] René Lamour and Dagmar Monett. 2016. A new algorithm for index determination in DAEs using algorithmic differentiation. *Numerical Algorithms* 72, 3 (2016), 667–682. https://doi.org/10.1007/s11075-015-0062-y

[13] Jerrold E Marsden and Tudor Ratiu. 2013. *Introduction to mechanics and symmetry: A basic exposition of classical mechanical systems*. Springer Science & Business Media.

[14] Sven Erik Mattsson and Gustaf Söderlind. 1993. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing* 14, 3 (1993), 677–692. https://doi.org/10.1137/0914043

[15] F. Mazzia and F. Iavernaro. 2003. *Test set for initial value problem solvers*. Department of Mathematics, University of Bari, Italy. Retrieved from http://pitagora.dm.uniba.it/testset/

[16] C. C. Pantelides. 1988. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing* 9, 2 (1988), 213–231. https://doi.org/10.1137/0909014

[17] Linda Petzold. 1982. Differential/algebraic equations are not ODE's. *SIAM Journal on Scientific and Statistical Computing* 3, 3 (1982), 367–384.

[18] Xiaolin Qin, Juan Tang, Yong Feng, Bernhard Bachmann, and Peter Fritzson. 2016. Efficient index reduction algorithm for large scale systems of differential algebraic equations. *Applied Mathematics and Computation* 277, (2016), 10–22. https://doi.org/10.1016/j.amc.2015.11.091

[19] Singiresu S Rao and Fook Fah Yap. 1995. *Mechanical vibrations*. Addison-Wesley Reading.

[20] Gunther Reissig, Wade Martinson, and Paul I. Barton. 2000. Differential-algebraic equations of index 1 may have an arbitrarily high structural index. *SIAM Journal on Scientific Computing* 21, 6 (2000), 1987–1990. https://doi.org/10.1137/S1064827599353853

[21] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. 2009. *Robotics: Modelling, planning and control*. Springer Science & Business Media.

[22] David S. Watkins. 2004. *Fundamentals of matrix computations* (2nd ed.). Wiley.