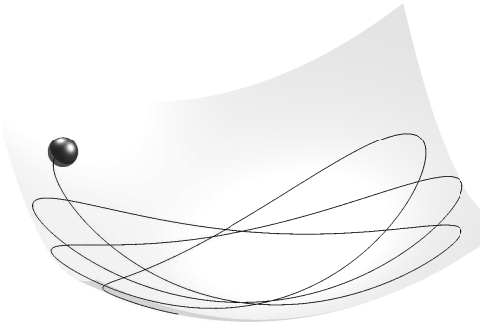
Modelling And Simulation

Additional Lecture notes for the NTNU course TTK4130

Sebastien Gros

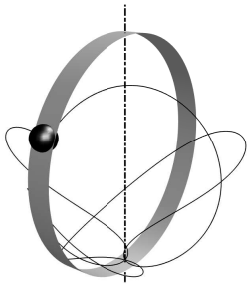
Edition: January 6, 2020



$$\mathscr{L}(q,\dot{q}) = T(q,\dot{q}) - V(q)$$

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = Q$$

$$\mathbb{p}[\boldsymbol{\theta}\,|\,y] = \frac{\mathbb{p}[y\,|\,\boldsymbol{\theta}]\mathbb{p}[\boldsymbol{\theta}]}{\int \mathbb{p}[y\,|\,\boldsymbol{\theta}]\mathbb{p}[\boldsymbol{\theta}]d\boldsymbol{\theta}}$$



$$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}}\ \mathbb{p}[\boldsymbol{\theta}\,|\,y] = \text{argmax}_{\boldsymbol{\theta}}\ \mathbb{p}[y\,|\,\boldsymbol{\theta}]$$

CONTENTS

1 NOTATION & BACKGROUND MATERIAL

Most often, we will note vectors in the bold math roman format, i.e. $\mathrm{a}\mathbb{R}^n$. Scalars will be noted in the standard math format, i.e. $a \in \mathbb{R}$. Matrices will be noted using capital letters, i.e. $A \in \mathbb{R}^{n\times m}$. There will be a few exceptions to these rules.

We will use subscript to denote the elements of vectors and matrices, i.e. $\mathrm{a}_i \in \mathbb{R}$ will be the i:th element of vector a, and A_{ij} the element in the i:th row and j:th column of matrix A. Generic functions will obey the same notation rules. We will reserve the capital letter I for the identity matrix.

1.1 NORMS AND SCALAR PRODUCTS

In this course we will often use the notion of norm, noted as $\|.\|$. A norm associates to a vector a positive scalar. It is often use to simply "measure the length" of a vector, but it has a deeper meaning that this. An example of norm is:

$$\|\mathrm{x}\| = \sqrt{\mathrm{x}^\top \mathrm{x}} \tag{1.1}$$

But any operator ρ taking an element of the vector space under consideration into $\mathbb{R}_+$ and sastisfying:

1. $\rho(\mathrm{x}+\mathrm{y}) \leq \rho(\mathrm{x}) + \rho(\mathrm{y})$
2. $\rho(a\mathrm{x}) = |a|\rho(\mathrm{x})$ for any $a \in \mathbb{R}$
3. $\rho(\mathrm{x}) = 0$ entails that $\mathrm{x} = 0$

There are several norms that differ from (1.1), e.g. the 1-norm

$$\|\mathrm{x}\|_1 = \sum_{k=1}^{n} |\mathrm{x}_k| \tag{1.2}$$

and the infinity norm

$$\|\mathrm{x}\|_\infty = \max_{k=1,\ldots,n} |\mathrm{x}_k| \tag{1.3}$$

Note that since all norms are equivalent (i.e. boundedness in one norm implies boundedness in the others), we often omit (when irrelevant) which type of norm we use.

We will use the notation

$$\langle ., . \rangle \tag{1.4}$$

to denote scalar products on any vector space. On $\mathbb{R}^n$, the scalar product between two vectors $\mathrm{x}, \mathrm{y} \in \mathbb{R}^n$ reads as:

$$\langle \mathrm{x}, \mathrm{y} \rangle = \mathrm{x}^\top \mathrm{y} \tag{1.5}$$

1.2 PROBABILITY

We will use probabilities in one of the sections of this course. Let us review some basic principles and notations here. We will note

$$\mathbb{P}[A] \tag{1.6}$$

the probability that statement A is true. We will abuse this notation and use it for both probabilities and probability distributions. When $\mathbb{P}$ is used for a probability distribution, then $\mathbb{P}[x]$ ought to be understood as "probability that variables x takes a specific value", and $\mathbb{P}[x]$ will be a function of x. E.g.

$$\mathbb{P}[x] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}x^2} \tag{1.7}$$

is a normal, centred probability distribution describing the probability that a variable takes a specific value x. We will additionally need the notion of conditional probability (or conditional probability distribution), that we will note as:

$$\mathbb{P}[A \,|\, B] \tag{1.8}$$

meaning "probability that A is true knowing that B is true". We will also use the same notation conditional probability distributions, i.e. $\mathbb{P}[x \,|\, y]$ will be the "probability that variable x takes a specific value knowing that variable y has taken the specific value". $\mathbb{P}[x \,|\, y]$ will then be a function of x and y, e.g. it could be:

$$\mathbb{P}[x \,|\, y] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-y)^2} \tag{1.9}$$

We will need the notion of expected value, noted and defined as:

$$\mathbb{E}[x] = \int x\mathbb{P}[x] \tag{1.10}$$

for continuous distributions. Note that $\mathbb{E}[x]$ is a number or vector of the same size as x. In practice, one can construe the expected value as "the average of variable x over many experiments", i.e. if one were to draw the random variable x many times and average the result, one would get something close to the expected value. Similarly, we can define the conditional expected value as:

$$\mathbb{E}[x \,|\, y] = \int x\mathbb{P}[x \,|\, y] \tag{1.11}$$

In practice, one can construe the conditional expected value as "the average of variable x over many experiments provided that y takes a specific value". Note that $\mathbb{E}[x \,|\, y]$ is a function of y alone, and returns a scalar if x is scalar or a vector of the size of x.

1.3 Some basics from Linear Algebra

Let us review here some basic principles of linear algebra that can be useful in this course.

- The eigenvalues λ and eigenvectors v satisfy the equation

$$A\mathrm{v} = \lambda \mathrm{v} \tag{1.12}$$

  i.e. the eigenvectors are transformed into scaled versions of themselves by the application of matrix A. The eigenvalues can be computed by solving the polynomial equation

$$\det(A - \lambda I) = 0 \tag{1.13}$$

  where det is the matrix determinant.

- A matrix A is invertible (i.e. A^{-1} exists) iff
  - $\det(A) \neq 0$ or
  - All the eigenvalues of A are nonzero

  An invertible matrix is said “full rank".

- A matrix A is said symmetric if $A = A^\top$
- A matrix A is said positive-definite if all its eigenvalues are real positive.
- A quadratic form is the operation:

$$\mathrm{x}^\top A\mathrm{x} \in \mathbb{R} \tag{1.14}$$

  where A is a square matrix, and x is a vector of adequate dimension. If A is a positive-definite matrix, then

$$\mathrm{x}^\top A\mathrm{x} \geq 0 \tag{1.15}$$

  for any x.

1.4 Some basics from Calculus

We will use a lot of calculus in this course, and you need to be comfortable with it. We recall some basic calculus here.

- The Jacobian of a multi-variate function $\mathrm{f} : \mathbb{R}^m \to \mathbb{R}^n$

$$\mathrm{f}(\mathrm{x}) = \begin{bmatrix} \mathrm{f}_1(\mathrm{x}) \\ \vdots \\ \mathrm{f}_n(\mathrm{x}) \end{bmatrix}, \quad \mathrm{x} \in \mathbb{R}^m \tag{1.16}$$

is a $\mathbb{R}^{n\times m}$ matrix given by:

$$\frac{\partial \mathrm{f}}{\partial \mathrm{x}} = \begin{bmatrix} \frac{\partial \mathrm{f}_1}{\partial \mathrm{x}_1} & \cdots & \frac{\partial \mathrm{f}_1}{\partial \mathrm{x}_m} \\ \vdots & & \vdots \\ \frac{\partial \mathrm{f}_n}{\partial \mathrm{x}_1} & \cdots & \frac{\partial \mathrm{f}_n}{\partial \mathrm{x}_m} \end{bmatrix} \tag{1.17}$$

- It will be convenient certain times to use the gradient operator ∇, which is essentially the transpose of the Jacobian operator, i.e.

$$\nabla_{\mathrm{x}} \mathrm{f} = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}^{\top} \tag{1.18}$$

  The gradient is most often used for scalar functions $\mathrm{f} \in \mathbb{R}$, but the notion of gradient can be readily applied to all functions.

- Chain ruling will be used a lot in this course. For a composition of functions $\mathrm{f}\big(\mathrm{g}(\mathrm{x})\big)$, chain ruling reads as:

$$\frac{\partial \mathrm{f}\big(\mathrm{g}(\mathrm{x})\big)}{\partial \mathrm{x}} = \frac{\partial \mathrm{f}(\mathrm{y})}{\partial \mathrm{y}}\bigg|_{\mathrm{y}=\mathrm{g}(\mathrm{x})} \frac{\partial \mathrm{g}(\mathrm{x})}{\partial \mathrm{x}} \tag{1.19}$$

  Note that the order in the right-hand-side of the equation matters for multi-variate functions, as the Jacobians are matrices. This order is reversed under the gradient operator, i.e.

$$\nabla_{\mathrm{x}} \mathrm{f}\big(\mathrm{g}(\mathrm{x})\big) = \nabla_{\mathrm{x}} \mathrm{g}(\mathrm{x})\ \nabla_{\mathrm{y}} \mathrm{f}\big(\mathrm{y}\big)\big|_{\mathrm{y}=\mathrm{g}(\mathrm{x})} \tag{1.20}$$

- We will need a few times to distinguish between total and partial derivatives. The notion can be tricky and even ambiguous sometimes, but let us recall here the basic principle. Consider a function:

$$\mathrm{f}\big(\mathrm{x},\mathrm{y}\big) \tag{1.21}$$

  where $\mathrm{y} = \mathrm{g}(\mathrm{x})$. Then the partial derivative ignores that y is intrinsically a function of x, i.e.

$$\frac{\partial \mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\partial \mathrm{x}} \tag{1.22}$$

  disregards this dependency, while the total derivatives takes it into account and does:

$$\frac{\mathrm{d}\mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\mathrm{d}\mathrm{x}} = \frac{\partial \mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\partial \mathrm{x}} + \frac{\partial \mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\partial \mathrm{y}} \frac{\partial \mathrm{y}}{\partial \mathrm{x}} = \frac{\partial \mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\partial \mathrm{x}} + \frac{\partial \mathrm{f}\big(\mathrm{x},\mathrm{y}\big)}{\partial \mathrm{y}} \frac{\partial \mathrm{g}}{\partial \mathrm{x}} \tag{1.23}$$

We will briefly review next some basic notions that will be useful in this course, and introduce some new material that you may have not seen before.

1.5 Ordinary Differential Equations (ODE)

Ordinary Differential Equations (ODEs) will play a crucial role in this course. ODEs are a way to relate the outputs y of a system to the inputs u acting on the system. Generally speaking, an ODE reads as

$$\varphi\left(\mathrm{y}^{(m)}, \mathrm{y}^{(m-1)}, \ldots, \mathrm{y}, \mathrm{u}^{(m-1)}, \mathrm{u}^{(m-2)}, \ldots, \mathrm{u}\right) = 0 \tag{1.24}$$

for some function φ and where we use the notation

$$\mathrm{y}^{(k)} = \frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathrm{y} \tag{1.25}$$

a trivial example of such an equation is the motion of a mass m attached to a spring of constant K, and subject to an external force u and viscous friction of parameter ξ. The ODE describing such a system reads as:

$$\varphi = m\ddot{y} + \xi\dot{y} + Ky - u = 0 \tag{1.26}$$

Most often we prefer to work with ODEs in their state-space form rather than in the form (1.24). An ODE in the state-space is most often written as:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) \tag{1.27}$$

for some state $\mathrm{x} \in \mathbb{R}^n$. E.g. the spring-mass example (1.26) in its state-space form reads as:

$$\dot{\mathrm{x}} = \begin{bmatrix} \mathrm{x}_2 \\ \frac{1}{m}\left(u - \xi\dot{y} - Ky\right) \end{bmatrix}, \qquad \mathrm{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \tag{1.28}$$

A complete state-space model is then made of the dynamics (1.27) and an output function $\mathrm{y} = \mathrm{h}(\mathrm{x}, \mathrm{u})$ telling us what we can “measure" or alternatively what we “care about" in the system. The full state-space model then reads as:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) \tag{1.29a}$$

$$\mathrm{y} = \mathrm{h}(\mathrm{x}, \mathrm{u}) \tag{1.29b}$$

We can make a few remarks here concerning (1.29).

- For $\mathrm{x} \in \mathbb{R}^n$, n provides the order of the system (state-space dimension)
- Functions f, h are generally nonlinear, often smooth
- The model is said *time-invariant* if the functions f, h do not *explicitly* depend on time

We will see later in the course that it can be sometimes useful to treat the dynamics of a system via an implicit ODE, which has the generic form:

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{x}, \mathrm{u}) = 0 \tag{1.30}$$

i.e. it delivers the derivatives of the state space not via a function f that can be evaluated directly, but implicitly via the equations (1.30).

1.6 LINEAR ODE

A very important class of models is the class of *linear* models, where functions f, h are linear. We can then rewrite (1.29) in the form:

$$\dot{\mathrm{x}} = A\mathrm{x} + B\mathrm{u} \tag{1.31a}$$

$$\mathrm{y} = C\mathrm{x} + D\mathrm{u} \tag{1.31b}$$

for some matrices $A \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$, $B \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}}$, $C \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{x}}$, $D \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{u}}$. Linear systems have the superposition property, i.e. if $\mathrm{x}_1(t)$ and $\mathrm{x}_2(t)$ are solution of (1.31) for the input profiles $\mathrm{u}_1(t)$, $\mathrm{u}_2(t)$ respectively, then $\alpha_1\mathrm{x}_1(t)+\alpha_2\mathrm{x}_2(t)$ is solution corresponding to the input profile $\alpha_1\mathrm{u}_1(t)+\alpha_2\mathrm{u}_2(t)$. Linear ODEs are "nice" to work with because one can treat them exploiting the powerful theorems provided by functional analysis, which - among other things - give us the Fourier and Laplace transforms.

When matrices A, B, C, D are fixed, then system (1.31) is said Linear Time Invariant (LTI). If (any of) the matrices $A(t), B(t), C(t), D(t)$ is functions of time, then system (1.31) is said Linear Time Varying (LTV).

1.7 LINEARIZATION

Linearization consists in forming *locally valid* linear approximations of nonlinear ODEs (1.29). The linearization of an ODE is nothing more than an application of the Taylor expansion, to a first order. I.e. the functions f, h are replaced by their first-order approximation with respect to all arguments. In that context, we will consider deviations in the states and inputs from a given reference, and establish how this deviation will evolve in time, to a first-order approximation. Let us build this step-by-step. Consider first the first-order expansion of function f. Consider a *reference* trajectory $\mathrm{x}(t)$ solution of (1.29a) for a given *reference* input profile u and *reference* initial conditions x_0. Consider then a deviation $\Delta\mathrm{u}(t)$ and $\Delta\mathrm{x}_0$ in the input profile and/or initial conditions, and consider $\Delta\mathrm{x}(t)$ the resulting deviation in the ODE trajectories. We observe that:

$$\dot{\mathrm{x}}(t) + \dot{\Delta\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t) + \Delta\mathrm{x}(t), \mathrm{u}(t) + \Delta\mathrm{u}(t)), \qquad \Delta\mathrm{x}(0) = \Delta\mathrm{x}_0 \tag{1.32a}$$

$$\mathrm{y}(t) + \Delta\mathrm{y}(t) = \mathrm{h}(\mathrm{x}(t) + \Delta\mathrm{x}(t), \mathrm{u}(t) + \Delta\mathrm{u}(t)) \tag{1.32b}$$

must hold. For small deviations $\Delta\mathrm{u}(t)$, $\Delta\mathrm{x}(t)$, we can form the first-order Taylor expansion of the right-hand-side of (1.32), and obtaing:

$$\dot{\mathrm{x}}(t) + \dot{\Delta\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t)) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \Delta\mathrm{x}(t) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \Delta\mathrm{u}(t) + \mathcal{O}\left(\|\Delta\mathrm{x}\|^2, \|\Delta\mathrm{u}\|^2\right) \tag{1.33a}$$

$$\mathrm{y}(t) + \Delta\mathrm{y}(t) = \mathrm{h}(\mathrm{x}(t), \mathrm{u}(t)) + \left.\frac{\partial \mathrm{h}}{\partial \mathrm{x}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \Delta\mathrm{x}(t) + \left.\frac{\partial \mathrm{h}}{\partial \mathrm{u}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \Delta\mathrm{u}(t) + \mathcal{O}\left(\|\Delta\mathrm{x}\|^2, \|\Delta\mathrm{u}\|^2\right) \tag{1.33b}$$

Since $\dot{\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t))$, we equivalently get:

$$\dot{\Delta\mathrm{x}}(t) \approx A(t)\Delta\mathrm{x}(t) + B(t)\Delta\mathrm{u}(t) \tag{1.34a}$$

$$\Delta\mathrm{y}(t) \approx C(t)\Delta\mathrm{x}(t) + D(t)\Delta\mathrm{u}(t) \tag{1.34b}$$

for the time-varying matrices:

$$A(t) = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}(t),\mathrm{u}(t)}, \quad B(t) = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \tag{1.35a}$$

$$C(t) = \left.\frac{\partial \mathrm{h}}{\partial \mathrm{x}}\right|_{\mathrm{x}(t),\mathrm{u}(t)}, \quad D(t) = \left.\frac{\partial \mathrm{h}}{\partial \mathrm{u}}\right|_{\mathrm{x}(t),\mathrm{u}(t)} \tag{1.35b}$$

A very commonly used special case of (1.34)-(1.35) is when the reference inputs and trajectories $\mathrm{u}(t)$, $\mathrm{x}(t)$ are constant, in which case matrices $A(t)$, $B(t)$, $C(t)$, $D(t)$ are also constant (because they come from Jacobians evaluated on fixed arguments). Such trajectories are in fact a stationary point of the dynamics, and can be found by solving:

$$\mathrm{f}(\mathrm{x}_0, \mathrm{u}_0) = 0 \tag{1.36}$$

and using $\mathrm{x} = \mathrm{x}_0$, $\mathrm{u}(t) = \mathrm{u}_0$ such that $\dot{\mathrm{x}} = 0$. We can then use these reference trajectory in (1.35).

Example Consider the nonlinear dynamics

$$\dot{\mathrm{x}}_1 = (1 - \mathrm{x}_2^2)\mathrm{x}_1 - \mathrm{x}_2 + u, \tag{1.37a}$$

$$\dot{\mathrm{x}}_2 = \mathrm{x}_1, \tag{1.37b}$$

with the output function:

$$h(\mathrm{x}) = \mathrm{x}_1 + \mathrm{x}_2^3 \tag{1.38}$$

One can compute:

$$A = \begin{bmatrix} 1 - \mathrm{x}_2(t)^2 & -2x_1x_2 - 1 \\ 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 3\mathrm{x}_2(t)^2 \end{bmatrix}, \qquad D = 0 \tag{1.39}$$

where $\mathrm{x}(t)$ is a trajectory of the system. A steady state $\bar{\mathrm{x}}$ of the system is obtained by solving:

$$\dot{\mathrm{x}}_1 = (1 - \mathrm{x}_2^2)\mathrm{x}_1 - \mathrm{x}_2 + u = 0, \tag{1.40a}$$

$$\dot{\mathrm{x}}_2 = \mathrm{x}_1 = 0, \tag{1.40b}$$

yielding

$$\bar{\mathrm{x}} = \begin{bmatrix} 0 \\ \bar{u} \end{bmatrix} \tag{1.41}$$

for a given $\bar{u}$ constant. A linearization at steady-state then reads as:

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 3\bar{u}^2 \end{bmatrix}, \qquad D = 0 \tag{1.42}$$

It is important to observe here that the linear system approximating the nonlinear dynamics depends in the input $\bar{u}$ selected.

□

1.8 SOLUTION OF ODES

Let us now turn to a crucial question in this course. Consider an ODE in its state-space form

$$\dot{x} = f(x, u) \tag{1.43}$$

An important question to ask here is “provided the initial conditions $x(0) = x_0$, is there a trajectory $x(t)$ solution of (1.43), and is it unique?".

Before treating that question, let us consider two simple examples showing that it is not trivial.

Example

- Consider the ODE:

$$\dot{x}(t) = x(t)^2, \qquad x(0) = 1 \tag{1.44}$$

One can verify that (1.44) admits

$$x(t) = \frac{1}{1-t} \tag{1.45}$$

as solution. We illustrate the trajectory $x(t)$ in the next Figure. One can easily observe from (1.45) that the state $x(t)$ becomes arbitrarily large as t approaches 1. The trajectories, in fact, do not exist at and beyond $t = 1$.

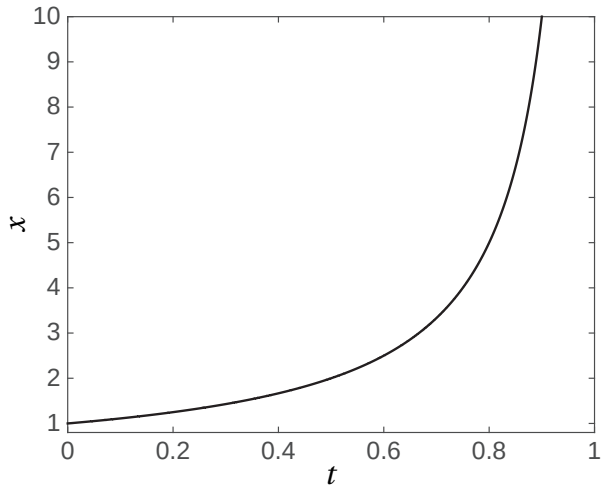


Figure 1.1: Illustration of the solution (1.45). The trajectories reach ∞ in finite time (when getting close to $t = 1$.

This simple examples illustrates that the solution of an ODE can “explode" in finite time, and cease to exist beyond a limited time interval.

- Consider the ODE

$$\dot{x} = \sqrt{|x|}, \quad x(0) = 0 \tag{1.46}$$

One can verify that:

$$x(t) = \begin{cases} \frac{(t-t_0)^2}{4} & t \geq t_0 \\ 0 & t < t_0 \end{cases} \tag{1.47}$$

is an admissible solution for (1.46) for any $t_0 \geq 0$. The issue here is that it is an admissible solution regardless of t_0. In other words, there are infinitely many trajectories $x(t)$ (for infinitely many choice of t_0) that are solution of (1.46). We illustrate these solutions in the next Figure.

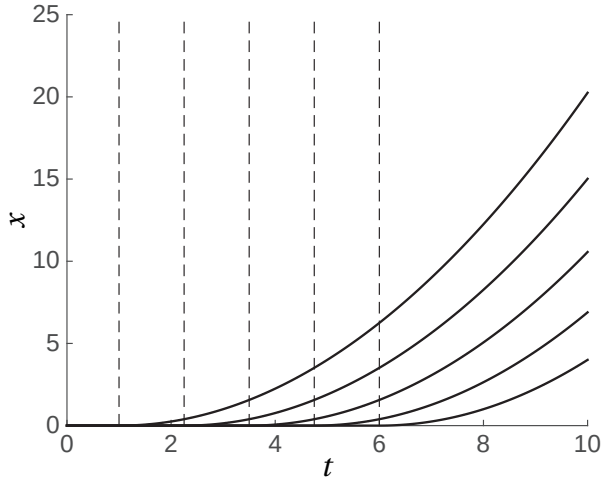


Figure 1.2: Illustration of the solutions (1.47) for different values of t_0 (dashed lines).

Let us introduce two theorems that unpack the two issues (existence and unicity) raised in the examples above. The first theorem deals with the existence of the ODE solution.

Theorem 1. *Consider the ODE*

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}) \tag{1.48}$$

where f *is continuous. Then if*[1]

$$\left\| \mathrm{f}(\mathrm{x}) - \mathrm{f}(\mathrm{y}) \right\| \leq c \cdot \left\| \mathrm{x} - \mathrm{y} \right\|, \quad \forall \mathrm{x}, \mathrm{y} \tag{1.49}$$

for some finite constant c, then the solution of (1.48) *exists and is unique for all t.*

[1] the following property is labelled “Lipschitz continuity"

Let us return to the first example above. One can observe that

$$\mathrm{f}(x) = x^2 \tag{1.50}$$

does not satisfy condition (1.49). Indeed, in this case,

$$\left\|\mathrm{f}(x) - \mathrm{f}\big(y\big)\right\| = x^2 - y^2 = (x+y)(x-y) \tag{1.51}$$

we can observe that the term $x+y$ can be arbitrarily large, such that we cannot find a constant c for which (1.49) holds.

Unfortunately, property (1.49) can be difficult to verify for non-trivial ODEs. Let us introduce a second theorem that makes our life easier.

Theorem 2. *Consider the ODE*

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}) \tag{1.52}$$

Then if f *is continuously differentiable (i.e. the Jacobian* $\frac{\partial \mathrm{f}}{\partial \mathrm{x}}$ *exists and is continuous), then the solution to the ODE exists and is unique on some time interval.*

Let us return to the second example above. One can observe that

$$\frac{\partial \mathrm{f}}{\partial \mathrm{x}} = \frac{1}{2\sqrt{x}} \tag{1.53}$$

such that f is not differentiable at $x = 0$. This results in the non-unicity of the solution of (1.46). Note that the first theorem does not apply to the ODE (1.46) as function f has “infinite slopes" at $x = 0$, such that condition (1.49) does not hold around $x = 0$. Note that the second theorem is a “weaker" version of the first one, and requires weaker conditions.

Before closing, let us consider the case of linear ODEs. We observe that the theorems above readily apply to any linear ODE as linear functions readily satisfy (1.49), and are continuously differentiable. That is, linear ODEs always have a unique solution over the time interval $[0, \infty[$. However, the solution may be unbounded, i.e. it can grow forever if the ODE is unstable, but there is not specific time where the solution becomes infinite (unlike (1.44)).

1.9 Solution to LTI ODEs

Consider the LTI ODE:

$$\dot{\mathrm{x}} = A\mathrm{x} + B\mathrm{u}, \qquad \mathrm{x}(t) = \mathrm{x}_0 \tag{1.54}$$

One can verify that the solution to (1.54) reads as:

$$\mathrm{x}(t) = e^{At}\mathrm{x}_0 + \int_0^t e^{A(t-\tau)} B\mathrm{u}(\tau)\,\mathrm{d}\tau \tag{1.55}$$

We ought to recall here the definition of a “matrix exponential" like e^{At}. Here we simply extend the series corresponding to the exponential function to matrices. More specifically, similarly to:

$$e^t = 1 + t + \frac{t^2}{2!} + \ldots = \sum_{k=0}^{\infty} \frac{t^k}{k!} \tag{1.56}$$

we define:

$$e^{At} = I + At + \frac{(At)^2}{2!} + \ldots = \sum_{k=0}^{\infty} \frac{(At)^k}{k!} \tag{1.57}$$

Note that here the power of a matrix A^k does not correspond to taking the power of the matrix entries, but rather to multiplying the matrix by itself k times. We additionally observe that the differentiation rule:

$$\frac{\mathrm{d}}{\mathrm{d}t} e^{at} = a e^{at} \tag{1.58}$$

becomes for the matrix exponential function:

$$\frac{\mathrm{d}}{\mathrm{d}t} e^{At} = A e^{At} \tag{1.59}$$

Note that the matrix exponential function is “expm.m" in Matlab, and does not (necessarily) correspond to the classic exponential function “exp.m".

We can conclude here that LTI ODEs do not need to be “simulated" as one can compute their solution explicitly from (1.55). For nonlinear ODEs (and some LTV ODEs), however, the best way of building their trajectories is by using computer-based simulations.

1.10 Z TRANSFORM

In one part of this course we will briefly use the Z-transform. We briefly recall here its principle. The Z-transform applies to LTI **discrete**-time systems, i.e. systems of e.g. the form:

$$\mathrm{y}_k = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i \mathrm{y}_{k-i} \tag{1.60}$$

We observe that the system output y is here described as a discrete sequence $\mathrm{y}_0, \ldots, \mathrm{y}_\infty$. Similarly to the Laplace transform the Z-transform allows one to treat dynamics in the form (1.60) in the “polynomial world".

The Z-transform of a signal y_k is given by:

$$\mathrm{Y}(z) = Z(\mathrm{y}) = \sum_{k=0}^{\infty} \mathrm{y}_k z^{-k} \tag{1.61}$$

The Z-transform has a number of useful properties (such as e.g. linearity), among which the time-shift property:

$$Z(\mathrm{y}_{k-i}) = z^{-i} Z(\mathrm{y}) \tag{1.62}$$

allows to write the Z-transform of the dynamics (1.60) as:

$$\mathrm{Y}(z) = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i z^{-i} \mathrm{U}(z) + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i z^{-i} \mathrm{Y}(z) \tag{1.63}$$

i.e.

$$\mathrm{Y}(z) = \frac{\sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i z^{-i}}{1 - \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i z^{-i}} \mathrm{U}(z) \tag{1.64}$$

2 LAGRANGE MECHANICS

In this Chapter we will study the basics of Lagrange mechanics. Lagrange mechanics is a powerful tool to build mathematical models for complex mechanical systems, and is used in many applications. Lagrange mechanics

- Allows to describe arbitrarily complex mechanical system, including relative moving parts, accelerated frames, gyroscopic and centrifugal effects without hassle.
- Often allows to build simple model (in terms of complexity of the resulting model equations) even for mechanical systems that are normally described by very complex models.

We define the generalized coordinates of a mechanical system as a vector of time-varying "coordinates" $\mathrm{q}(t) \in \mathbb{R}^{n_c}$ that must be able to describe the "configuration" of the system at a given time t. One can construe these coordinates as forming a "snapshot" of the system. The snapshot does not tell us how the system will evolve in time, but it can tell us in what configuration the system is at a given time. The generalized coordinates typically gather positions and angles, but can also include more abstract representations of the system. For the sake of simplicity, in the following we will omit the explicit dependence of the generalized coordinates and other time-varying objects in the notation, i.e. we will e.g. note the time-varying generalized coordinates $\mathrm{q}(t)$ simply as q.

Lagrange mechanics is based on a description of the mechanical system in terms of energy. In order to build the Lagrange framework and models using the Lagrange equations, we need to compute the kinetic and potential energy functions of the system, conventionally labelled T and V, respectively. Building these objects is not necessarily straightforward, but it can be done fairly systematically, and with a very limited risk of error. Let us discuss the kinetic and potential energy functions next.

2.1 KINETIC ENERGY

We recall here that the kinetic energy of a "punctual mass" (i.e. of a dimensional particle of mass m) whose position with respect to a fixed (inertial) reference frame is given by $\mathrm{p} \in \mathbb{R}^D$ (where the number of dimensions can be $D = 1, 2, 3$) reads as:

$$T = \frac{1}{2} m \left\| \dot{\mathrm{p}} \right\|^2 = \frac{1}{2} m \dot{\mathrm{p}}^\top \dot{\mathrm{p}} \tag{2.1}$$

A few useful observation ought to be made here:

- The kinetic energy function is a *quadratic form* of the mass velocity $\dot{\mathrm{p}}$. It is always positive (as $m > 0$) and takes the minimum value $T = 0$. This observation is generally true for any mechanical system
- The generalized coordinates q describe the configuration of the entire mechanical system we consider. I.e. if our punctual mass is part of a mechanical system having

the generalized coordinates q, then its position p is a function of q (in the sense that p can be calculated from q). More formally, one could write $\mathrm{p}(\mathrm{q})$. The mass velocity $\dot{\mathrm{p}}$ is then a direct result of the chain rule:

$$\dot{\mathrm{p}} = \frac{\partial \mathrm{p}}{\partial \mathrm{q}} \dot{\mathrm{q}} \tag{2.2}$$

- Imagine that our mechanical system is made of N punctual masses of position p_i, fully determined by the generalized coordinates q. The kinetic energy of the system is then an addition of all the individual kinetic energies, i.e.

$$T = \frac{1}{2} \sum_{i=1}^{N} m_i \left\| \dot{\mathrm{p}}_i \right\|^2 = \frac{1}{2} \sum_{i=1}^{N} m_i \dot{\mathrm{p}}_i^\top \dot{\mathrm{p}}_i \tag{2.3}$$

  The generalized coordinates describe the configuration of the whole system, i.e. it describes the position p_i of each mass. Each velocity is given by the chain rule:

$$\dot{\mathrm{p}}_i = \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}} \dot{\mathrm{q}} \tag{2.4}$$

  The kinetic energy of the N masses is then given by:

$$T = \frac{1}{2} \sum_{i=1}^{N} m_i \dot{\mathrm{p}}_i^\top \dot{\mathrm{p}}_i = \frac{1}{2} \sum_{i=1}^{N} m_i \dot{\mathrm{q}}^\top \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}}^\top \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}} \dot{\mathrm{q}} = \frac{1}{2} \dot{\mathrm{q}}^\top \left(\sum_{i=1}^{N} m_i \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}}^\top \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}} \right) \dot{\mathrm{q}} \tag{2.5}$$

  Here we can define:

$$W = \sum_{i=1}^{N} m_i \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}}^\top \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}} \tag{2.6}$$

  and we observe that W is a square, symmetric, (semi-)positive-definite matrix of size $\mathbb{R}^{n_\mathrm{q} \times n_\mathrm{q}}$. This matrix is in general a function of q, but can also be constant for “smart" choices of generalized coordinates.

- The observation above holds for systems having infinitely many “particles" (e.g. distributed mass systems). Generally, the kinetic energy function will take the form:

$$T(\mathrm{q}, \dot{\mathrm{q}}) = \frac{1}{2} \dot{\mathrm{q}}^\top W(\mathrm{q}) \dot{\mathrm{q}} \tag{2.7}$$

  where each of these variables (i.e. q and $\dot{\mathrm{q}}$) are time-varying, and where W is a square, symmetric, positive-definite matrix of size $\mathbb{R}^{n_\mathrm{q} \times n_\mathrm{q}}$, possibly function of q.

- Note that since the kinetic energy function T is given by (2.7), it is essentially defined via matrix $W(\mathrm{q})$, hence in our constructions in the following, matrix $W(\mathrm{q})$ will play a central role. In particular, we will see that $W(\mathrm{q})$ actually being a function of q or being constant plays a central role in the complexity of the model describing the corresponding mechanical system.

Example - mass on a rigid rod: imagine a punctual mass m attached to the origin by a rigid, massless rod of length L, and moving in the plane $x-y$. We can describe the system by the angle θ between the x axis and the rod. This angle alone describes entirely this simple system, as knowing specifies in what "configuration" (or position) the system is. Our (unique) generalized coordinate here is $\mathrm{q} = \theta \in \mathbb{R}$. The position of the mass in the $x-y$ plane is then given by:

$$\mathrm{p} = L \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{2.8}$$

The chain rule then tells us that:

$$\dot{\mathrm{p}} = L \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \dot{\theta} \tag{2.9}$$

Our kinetic energy is then:

$$T = \frac{1}{2} m \dot{\mathrm{p}}^\top \dot{\mathrm{p}} = \frac{1}{2} m L^2 \dot{\theta}^2 \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}^\top \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = \frac{1}{2} m L^2 \dot{\theta}^2 \left(\sin^2\theta + \cos^2\theta\right) = \frac{1}{2} m L^2 \dot{\theta}^2 \tag{2.10}$$

Here "matrix" W is simply

$$W = mL^2 \tag{2.11}$$

and is constant. **Example - mass on an elastic rod**: imagine a punctual mass m attached to the origin by an elastic, massless rod of varying length L, and moving in the plane $x-y$. To describe the system we now need the angle θ between the x axis and the rod, and the length L (as the latter is also changing and therefore not a simple constant). Our generalized coordinates are then:

$$\mathrm{q} = \begin{bmatrix} \theta \\ L \end{bmatrix} \tag{2.12}$$

as both are needed at a given time t to describe the "configuration" (or position) the system. The position of the mass in the $x-y$ plane is still given by:

$$\mathrm{p} = L \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{2.13}$$

but now L is no longer a constant and is part of our generalized coordinates. The chain rule then tells us that:

$$\dot{\mathrm{p}} = L \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \dot{\theta} + \dot{L} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{2.14}$$

Our kinetic energy is then:

$$\begin{aligned} T &= \frac{1}{2} m \dot{\mathrm{p}}^\top \dot{\mathrm{p}} \\ &= \frac{1}{2} m L^2 \dot{\theta}^2 \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}^\top \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} + \frac{1}{2} m \dot{L}^2 \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}^\top \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ &\quad + \frac{1}{2} m L \dot{L} \dot{\theta} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}^\top \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ &= \frac{1}{2} m L^2 \dot{\theta}^2 + \frac{1}{2} m \dot{L}^2 = \frac{1}{2} \dot{\mathrm{q}}^\top \begin{bmatrix} mL^2 & 0 \\ 0 & m \end{bmatrix} \dot{\mathrm{q}} \end{aligned} \tag{2.15}$$

Here matrix W is diagonal:

$$W = \begin{bmatrix} mL^2 & 0 \\ 0 & m \end{bmatrix} \tag{2.16}$$

and not constant as L is part of the state.

Example - rigid rod with a mass: imagine a uniform rod of length L and of negligible thickness attached at the origin, having a mass m (uniformly distributed). The rod position is again described by the angle θ alone (we consider the length L fixed as the rod is rigid), hence $\mathrm{q} = \theta$. We can consider our rod as made of infinitely many punctual masses distributed along the rod axis. If we locate these masses via their position $v \in [0, L]$ along the rod axis, we can describe their position in the plane $x - y$ by:

$$\mathrm{p}(v) = v \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{2.17}$$

and their velocity is given by the chain rule:

$$\dot{\mathrm{p}}(v) = v \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \dot{\theta} \tag{2.18}$$

The overall kinetic energy of the rod is given by the summation of the kinetic energy of all the infinitesimal masses (each having a mass $\frac{m}{L}\mathrm{d}v$). It can be computed by the integral:

$$\begin{aligned} T &= \frac{m}{L} \int_0^L \left(v\dot{\theta} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \right)^\top \left(v\dot{\theta} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \right) \mathrm{d}v \\ &= \frac{m}{L} \int_0^L v^2 \dot{\theta}^2 \, \mathrm{d}v = \frac{m}{3L} \dot{\theta}^2 v^3 \Big|_{v=0}^{v=L} = \frac{m}{3L} \dot{\theta}^2 L^3 = \frac{1}{3} m L^2 \dot{\theta}^2 \end{aligned} \tag{2.19}$$

Our "matrix" W is now reduced to 2/3 of the one we had for the mass concentrated at the end of the rod, i.e.

$$W = \frac{2}{3} m L^2 \tag{2.20}$$

(one can compare (2.19) and (2.10) to see that).

□

2.2 POTENTIAL ENERGY

Let us provide a few examples of potential energy, some of which will be extensively used in this course.

- The potential energy due to gravity in most “standard" mechanical applications derives from:

$$V = mgz \tag{2.21}$$

  which gives the potential energy of a punctual mass m, where “z" is the “height" of the mass in the field of gravity. Consider e.g. in the examples above that the mass m is concentrated at the end of the rigid rod. The position of the mass is given by (2.17), such that its vertical position is given by:

$$\mathrm{p}_z = L\sin\theta \tag{2.22}$$

  Its potential energy is then given by:

$$V = mgL\sin\theta \tag{2.23}$$

  In contrast, consider that the mass is distributed throughout the rod. Computing the potential energy then follows similar lines as (2.19), i.e. we need to “sum up" the potential energy of every “particle of the rod". A “particle" is described here as a infinitesimal piece of the rod, of length $\mathrm{d}v$, having a mass $\frac{m}{L}\mathrm{d}v$ and a vertical position $v\sin\theta$. We can then do:

$$V = \frac{mg}{L}\int_0^L v\sin\theta\mathrm{d}v = \frac{mg}{L}\frac{1}{2}v^2\sin\theta\Big|_{v=0}^{v=L} = \frac{1}{2}mgL\sin\theta \tag{2.24}$$

  hence the potential energy then corresponds to considering that the whole mass of the rod is concentrated at the half-length (this is to put in contrast with the kinetic energy computation, where one can consider that the whole mass of the rod is concentrated at the $2/3$ of the length of the rod!).

- The considerations above are valid for mechanical systems evolving “at a small scale" in the field of gravity, for which one can consider the field as “straight" and uniform. For mechanical systems evolving at a “large scale" such as e.g. a satellite orbiting a planet, one needs to apply (at least) the genuine formula for classic gravity, i.e.:

$$V = -G\frac{m}{r} \tag{2.25}$$

  where r is the distance to the center of the planet.

Note that the Lagrange formalism can handle potential energy from other sources (magnetic field, electric field, etc.), but we will not consider these cases in this course. Another source of potential energy that is commonly present in mechanical systems is potential energy stored in flexible components. We will often simply consider springs in this course, but

any part of the system that undergoes elastic deformations does in principle store kinetic energy. Consider the spring illustrated below.

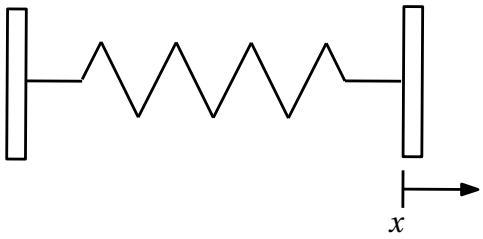


Figure 2.1: Illustration of a spring. The position $x = 0$ is set at the spring rest length. The rigidity constant of the spring is k

The force delivered by the spring is given by

$$F = -kx \tag{2.26}$$

where x is the position of the spring end, relative to its rest position (position at which the spring yields no force). The potential energy stored in the spring is given by:

$$V = \int_x^0 -kv\mathrm{d}v = -\frac{1}{2}kv^2\Big|_x^0 = \frac{1}{2}kx^2 \tag{2.27}$$

Note that if the reference frame is chosen such that the rest length of the spring is not at $x = 0$ but rather at an arbitrary position $x = x_0$, then the potential energy is given by:

$$V = \frac{1}{2}k(x - x_0)^2 \tag{2.28}$$

Let us consider now a slightly more complex example illustrated in the figure below.

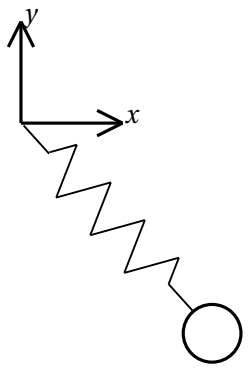


Figure 2.2: Illustration of a spring moving in the plane. The rigidity constant of the spring is k. We suppose that the rest length of the spring is L_0.

We consider that the position of the end of the spring is given by

$$\mathrm{p} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.29}$$

such that the elongation of the spring with respect to its rest length is given by:

$$\Delta = \left(\mathrm{p}^\top \mathrm{p}\right)^{\frac{1}{2}} - L_0 \tag{2.30}$$

The potential energy stored in the spring is then given by:

$$V = \frac{1}{2} k \Delta^2 = \frac{1}{2} k \left(\left(\mathrm{p}^\top \mathrm{p}\right)^{\frac{1}{2}} - L_0 \right)^2 \tag{2.31}$$

The latter expression is fairly complex, as the $.^{\frac{1}{2}}$ does not simplify with the $.^2$ unless $L_0 = 0$. If $L_0 = 0$, then the potential energy simplifies to:

$$V = \frac{1}{2} k \mathrm{p}^\top \mathrm{p} \tag{2.32}$$

In many exercises in this course, will use the simplifying assumption that $L_0 = 0$ in order to get simpler expressions.

2.3 Lagrange Equation

Before detailing the Euler-Lagrange equation, we need to define the Lagrange function

$$\mathscr{L}\left(\mathrm{q}, \dot{\mathrm{q}}\right) = T\left(\mathrm{q}, \dot{\mathrm{q}}\right) - V\left(\mathrm{q}\right) \in \mathbb{R} \tag{2.33}$$

simply made from substracting the potential energy to the kinetic energy. In general, the Lagrange function takes the generalized coordinates q and their time derivative $\dot{\mathrm{q}}$ as arguments. Using the observations of the Sections above, we observe that the Lagrange function takes the form:

$$\mathscr{L}\left(\mathrm{q}, \dot{\mathrm{q}}\right) = \frac{1}{2} \dot{\mathrm{q}}^\top W\left(\mathrm{q}\right) \dot{\mathrm{q}} - V\left(\mathrm{q}\right) \tag{2.34}$$

The Euler-Lagrange equation then reads as:

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = 0 \tag{2.35}$$

and defines the model of the mechanical system. It is useful to observe here that (2.35) defines equations as a *lying vector*. Indeed, e.g. $\frac{\partial \mathscr{L}}{\partial \mathrm{q}} \in \mathbb{R}^{1 \times n_\mathrm{q}}$ is a line-vector. One can verify that the first term is (of course) also a line vector such that the substraction between the two terms is well defined. As we tend to prefer working with equations in a “standing-vector form", it can be useful when needed to rewrite (2.35) in its “transposed" version, i.e.:

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathrm{q}}} \mathscr{L} - \nabla_\mathrm{q} \mathscr{L} = 0 \tag{2.36}$$

where we use the “gradient" notation:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = \frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}}^\top, \qquad \nabla_\mathrm{q} \mathscr{L} = \frac{\partial \mathscr{L}}{\partial \mathrm{q}}^\top \tag{2.37}$$

This transformation is cosmetic and of secondary importance, but it is useful to point it out. Remember this observation in the following whenever we transpose the terms appearing in (2.35), as we will do it whenever it will help us write things in a neat and structured form.

Before investigating how (2.35) delivers the model of the system, it is worth here pausing to understand the mathematical meaning of the first term in (2.35). Indeed, the partial derivative of the Lagrange function $\mathscr{L}$ with respect to the time derivative of the generalized coordinates $\dot{\mathrm{q}}$, i.e. $\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}}$ may appear surprising or ambiguous. In fact, it is a very straightforward operation. In order to perform it correctly, one has to consider $\dot{\mathrm{q}}$ as a variable in itself, independent of all other variables, and take the classical differential operations accordingly. More specifically, considering (2.34), the operation $\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}}$ simply yields:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = \nabla_{\dot{\mathrm{q}}} \left(\frac{1}{2} \dot{\mathrm{q}}^\top W(\mathrm{q}) \dot{\mathrm{q}} - V(\mathrm{q}) \right) = \nabla_{\dot{\mathrm{q}}} \left(\frac{1}{2} \dot{\mathrm{q}}^\top W(\mathrm{q}) \dot{\mathrm{q}} \right) = W(\mathrm{q}) \dot{\mathrm{q}}. \tag{2.38}$$

Forming the term $\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}}$ in the Lagrange equation requires a *total differentiation* with respect to time (operator $\frac{\mathrm{d}}{\mathrm{d}t}$). Using the chain rule, this yields:

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) = \frac{\partial}{\partial \dot{\mathrm{q}}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} = W(\mathrm{q}) \ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} \tag{2.39}$$

We can then pack the developments above to observe that the Lagrange equation yields:

$$W(\mathrm{q}) \ddot{\mathrm{q}} \quad + \quad \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} \quad - \underbrace{\nabla_{\mathrm{q}} \mathscr{L}}_{= \nabla_{\mathrm{q}} T - \nabla_{\mathrm{q}} V} = 0 \tag{2.40}$$

In order to understand the "mechanics" behind using the Lagrange equations, it is useful to run through a handful of simple and classical examples. It is really useful to underline here that **the computations performed hereafter are best carried out in a Computer Algebra System (CAS) such as the Matlab Symbolic Toolbox**. Indeed, the computations required to deploy Lagrange mechanics are very systematic, but become quickly involved. It is best to perform them in the computer. In the exercises associated to this course, we recommend you to use this approach to generate your models.

We can make a few observations here.

1. The Lagrange equation delivers the differential equation in an *implicit form*, i.e. it is a set of equations relating the generalized coordinates and their first and second-order time derivatives, i.e. q, $\dot{\mathrm{q}}$, $\ddot{\mathrm{q}}$.

2. The Lagrange equation allows one to compute the system acceleration $\ddot{\mathrm{q}}$ for a given system configuration q and its time derivative $\dot{\mathrm{q}}$ if matrix W(q is invertible. They can be used for simulating the system, if the initial conditions q, $\dot{\mathrm{q}}$ are provided (i.e. we need to specify the "position" and "velocity" of the system in order to predict its future trajectory). For q, $\dot{\mathrm{q}}$ given, then the Lagrange equations can be solved for the acceleration $\ddot{\mathrm{q}}$, from which the time evolution of q, $\dot{\mathrm{q}}$ can be computed (i.e. q is obtained from integrating $\dot{\mathrm{q}}$ and $\dot{\mathrm{q}}$ is obtained by integrating $\ddot{\mathrm{q}}$).

3. The second term

$$\nabla_{\mathrm{q}}\mathscr{L} = \nabla_{\mathrm{q}} T - \nabla_{\mathrm{q}} V \tag{2.41}$$

in the Lagrange equation is generally speaking introducing forces that are intrinsic to the system (as opposed to forces applied "externally" to the system), such as e.g. forces deriving from potentials (coming from the "V" part, see remark following (2.51)) and centrifugal forces (coming from the "T" part).

4. The Lagrange equation is linear in the accelerations $\ddot{\mathrm{q}}$. This is true for Lagrange functions in the form (2.34), and stem from (2.39), which yields the first term in the Lagrange equation in the form $W(\mathrm{q})\ddot{\mathrm{q}}$, which is linear in $\ddot{\mathrm{q}}$.

5. Because the accelerations enter linearly in the Lagrange equation, we can solve it for $\ddot{q}$. More specifically, if one write the Lagrange equation in the form (2.40), the accelerations $\ddot{\mathrm{q}}$ can be explicitly expressed as:

$$\ddot{\mathrm{q}} = W\left(\mathrm{q}\right)^{-1}\left[\nabla_{\mathrm{q}}\mathscr{L} - \frac{\partial}{\partial \mathrm{q}}\left(W\left(\mathrm{q}\right)\dot{\mathrm{q}}\right)\dot{\mathrm{q}}\right] \tag{2.42}$$

Unfortunately, writing the model in this explicit form is not always a very good move. Indeed, for systems that do not have a vector of generalized coordinates q of very low dimension (e.g. > 2), the inverse $W\left(\mathrm{q}\right)^{-1}$ can be very complex, even if matrix $W(\mathrm{q})$ is fairly simple. An exception to this observation is the case where matrix $W(\mathrm{q})$ is constant (i.e. not actually function of q). In this case

$$\nabla_{\mathrm{q}} T = 0 \quad \Rightarrow \quad \nabla_{\mathrm{q}}\mathscr{L} = -\nabla_{\mathrm{q}} V \qquad \text{and} \qquad \frac{\partial}{\partial \mathrm{q}}\left(W\left(\mathrm{q}\right)\dot{\mathrm{q}}\right)\dot{\mathrm{q}} = 0 \tag{2.43}$$

such that (2.42) becomes:

$$\ddot{\mathrm{q}} = -W^{-1}\nabla_{\mathrm{q}} V \tag{2.44}$$

where matrix W is purely "numerical" (i.e. it contains only numbers, not expressions), and is therefore easy to invert.

Let us try to nail down these remarks via a series of examples, starting simple and ramping up to complex systems.

Examples

1. Consider the spring-mass system depicted in Fig. 2.3 below.

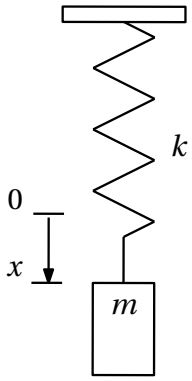


Figure 2.3: Vertical spring-mass system. The mass “0" position is set at the spring rest length. The rigidity constant of the spring is k, and the hanging mass is m.

Let us deploy the principles of Lagrange modelling on this simple system. Our goal is to unpack what the Lagrange equation does, and what the different terms “physically" correspond to. Let us start with setting up our generalized coordinates. We need to describe the mass position. Fig. 2.3 proposes to set the “0" position at the rest length of the spring, and consider that x increases when the mass goes down. This choice is not unique: we could set this “0" position anywhere we want and decide that x increases when the mass goes up.

The kinetic energy function is simple to calculate for this kind of system. We can use (2.1), where our position $\mathrm{p} \equiv x$ is a scalar (we work in 1D). We then simply get:

$$T = \frac{1}{2} m \dot{x}^2 \tag{2.45}$$

One can observe here that our kinetic energy function is in the form (2.7), with $W(x) = m$. The potential energy is composed of the sum of gravity and of the energy stored in the spring (note that energies always add). These two terms read as:

$$V_{\text{gravity}} = -mgx \qquad \text{and} \qquad V_{\text{spring}} = \frac{1}{2} k x^2 \tag{2.46}$$

Note the minus sign on V_{gravity}. One can easily guess it by observing that if x increases the mass goes down and therefore the potential energy decreases. This requires the minus sign. The potential energy of a spring is always given by the quadratic form used here, i.e. it is given by the elongation of the spring (from rest length) squared, multiplied by the rigidity and divided by two.

We can now assemble the Lagrange function:

$$\mathscr{L} = T - \left(V_{\text{gravity}} + V_{\text{spring}}\right) = \frac{1}{2} m \dot{x}^2 + mgx - \frac{1}{2} k x^2 \tag{2.47}$$

Once the Lagrange function is assembled, the modelling does not require “intelligence" anymore, as the rest is just calculus and algebraic manipulations in order to

extract a differential equation from (2.35). This procedure can actually be automated, and its deployment entrusted to computer tools. If we do the exercise of computing (2.35) here. Let us start with:

$$\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \nabla_{\dot{x}}\mathscr{L} = m\dot{x} \tag{2.48}$$

We observe that this term is a *momentum*. As a matter of fact, the term $\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}}$ will always describe the momentum present in the system. We can also compare this expression with (2.38).

Furthermore, we can compute:

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \frac{\mathrm{d}}{\mathrm{d}t}(m\dot{x}) = m\ddot{x} \tag{2.49}$$

We observe that this term is equivalent to the "$m \cdot \mathrm{a}$" of the $\mathrm{F} = m \cdot \mathrm{a}$ of Newton, i.e. it corresponds to a mass multiplied by accelerations. As in Newton, the product of a mass by an acceleration will equate forces. This observation will also hold true for all system, though "masses" and "forces" will take a somewhat more general meaning in Lagrange mechanics. We can also compare this expression to (2.39), and observe that

$$\frac{\partial}{\partial \mathrm{q}}\left(W\left(\mathrm{q}\right)\dot{\mathrm{q}}\right)\dot{\mathrm{q}} = \underbrace{\frac{\partial W}{\partial \mathrm{x}}}_{=0}\dot{x}^2 = 0 \tag{2.50}$$

because W is constant. We will see later on that having a matrix $W(\mathrm{q})$ that is constant simplifies significantly the resulting model equations.

We can turn to computing:

$$\nabla_{\mathrm{q}}\mathscr{L} = \nabla_{x}\mathscr{L} = mg - kx \tag{2.51}$$

We ought to observe that this term delivers the forces acting on the system resulting from the potentials present in the system. We can now assemble the Lagrange equations, i.e. (2.35) for this simple example reads as:

$$m\ddot{x} - mg + kx = 0 \tag{2.52}$$

Because the acceleration $\ddot{x}$ enters linearly in the Lagrange equation, we can solve it for $\ddot{x}$. Here we can trivially manipulate (2.52) to get:

$$m\ddot{x} = mg - kx. \tag{2.53}$$

This equation is essentially the Newton equation "$\mathrm{F} = m \cdot \mathrm{a}$" for the spring-mass system. Indeed, we observe that:

$$\underbrace{m\ddot{x}}_{\equiv m\cdot \mathrm{a}} = \underbrace{mg - kx}_{\equiv \mathrm{F}} \tag{2.54}$$

2. We will now consider a linear crane as depicted in Fig. 2.4. This kind of system is e.g. important for loading/unloading cargo ships in large harbors. The proposed generalized coordinates are visible in Fig. 2.4, i.e.

$$\mathrm{q} = \begin{bmatrix} x \\ \theta \end{bmatrix} \in \mathbb{R}^2 \tag{2.55}$$

is made of the position of the cart on the rail, and the angle θ is the angle of the rod linking the cart to the hanging mass with respect to the vertical. We observe here that there is no problem mixing positions and angles (and any other physical quantity describing the "configuration" of a mechanical system) in the generalized coordinates.

We can now compute the kinetic and potential energy functions. As energies are additive, we can compute them separately for the two masses.

The kinetic energy of the cart is simply $T_{\text{cart}} = \frac{1}{2}M\dot{x}^2$. The energy of the hanging mass is a bit more complex to compute here, but can still be simply derived from formula (2.1). Indeed, we observe that the position of the hanging mass is given by:

$$\mathrm{p}_m = \begin{bmatrix} x + L\sin\theta \\ -L\cos\theta \end{bmatrix} \tag{2.56}$$

Note that the positive sign in the first line will specify in which direction a positive angle will "rotate" the hanging mass. We can then compute:

$$\dot{\mathrm{p}}_m = \frac{\partial \mathrm{p}_m}{\partial \mathrm{q}}\dot{\mathrm{q}} = \begin{bmatrix} \dot{x} + L\dot{\theta}\cos\theta \\ L\dot{\theta}\sin\theta \end{bmatrix} \tag{2.57}$$

and

$$\left\|\dot{\mathrm{p}}_m\right\|^2 = \dot{\mathrm{p}}_m^\top \dot{\mathrm{p}}_m = \ldots = L^2\dot{\theta}^2 + \dot{x}^2 + 2L\dot{\theta}\dot{x}\cos\theta \tag{2.58}$$

The kinetic energy of the hanging crane is therefore given by:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{\mathrm{p}}_m^\top\dot{\mathrm{p}}_m = \frac{1}{2}(m+M)\dot{x}^2 + \frac{1}{2}mL^2\dot{\theta}^2 + mL\dot{\theta}\dot{x}\cos\theta \tag{2.59}$$

We can deduce that the matrix $W(\mathrm{q})$ is now:

$$W(\mathrm{q}) = \begin{bmatrix} m+M & mL\cos\theta \\ mL\cos\theta & mL^2 \end{bmatrix} \tag{2.60}$$

and is not constant in this example (it is function of θ). It can be useful to observe here that the matrix $W(\mathrm{q})$ can be readily computed from the kinetic energy function by computing its Hessian with respect to $\dot{\mathrm{q}}$, i.e.

$$W(\mathrm{q}) = \frac{\partial^2 T\left(\mathrm{q},\dot{\mathrm{q}}\right)}{\partial \dot{\mathrm{q}}^2} \tag{2.61}$$

The potential energy of the hanging crane is just gravity, and is related to the "z" position of the hanging mass, i.e.:

$$V = mg \begin{bmatrix} 0 & 1 \end{bmatrix} \mathrm{p}_m = -mgL\cos\theta \tag{2.62}$$

Our Lagrange function can now be assembled, and reads as:

$$\mathscr{L} = \frac{1}{2}(m+M)\dot{x}^2 + \frac{1}{2}mL^2\dot{\theta}^2 + mL\dot{\theta}\dot{x}\cos\theta + mgL\cos\theta \tag{2.63}$$

We can now develop the terms of the Lagrange equation:

$$\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} (M+m)\dot{x} + mL\dot{\theta}\cos\theta \\ mL^2\dot{\theta} + mL\dot{x}\cos\theta \end{bmatrix} = W(\mathrm{q})\dot{\mathrm{q}} \tag{2.64}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} (M+m)\ddot{x} + mL\cos\theta\ddot{\theta} - mL\sin\theta\dot{\theta}^2 \\ mL\cos\theta\ddot{x} + mL^2\ddot{\theta} - mL\dot{\theta}\dot{x}\sin\theta \end{bmatrix} \tag{2.65}$$

Moreover,

$$\nabla_{\mathrm{q}}\mathscr{L} = -mL\sin\theta \begin{bmatrix} 0 \\ g + \dot{\theta}\dot{x} \end{bmatrix} \tag{2.66}$$

The Lagrange equation then reads as:

$$\begin{bmatrix} (M+m)\ddot{x} + mL\cos\theta\ddot{\theta} - mL\sin\theta\dot{\theta}^2 \\ mL\cos\theta\ddot{x} + mL^2\ddot{\theta} + mgL\sin\theta \end{bmatrix} = 0 \tag{2.67}$$

The acceleration $\ddot{\mathrm{q}}$ enters linearly in (2.67). We can actually observe that (2.67) can be written as:

$$W(\mathrm{q})\ddot{\mathrm{q}} + mL\sin\theta \begin{bmatrix} -\dot{\theta}^2 \\ g \end{bmatrix} = 0 \tag{2.68}$$

And inverting W(q, we can write:

$$\ddot{\mathrm{q}} = mL\sin\theta W(\mathrm{q})^{-1} \begin{bmatrix} \dot{\theta}^2 \\ -g \end{bmatrix} \tag{2.69}$$

Unfortunately, this construction yields fairly complex expressions. We can verify that:

$$\ddot{\mathrm{q}} = \frac{\sin\theta}{(M+m-m\cos\theta^2)} \begin{bmatrix} m(L\dot{\theta}^2 + g\cos\theta) \\ -\frac{1}{L}((M+m)g + mL\cos\theta\dot{\theta}^2) \end{bmatrix} \tag{2.70}$$

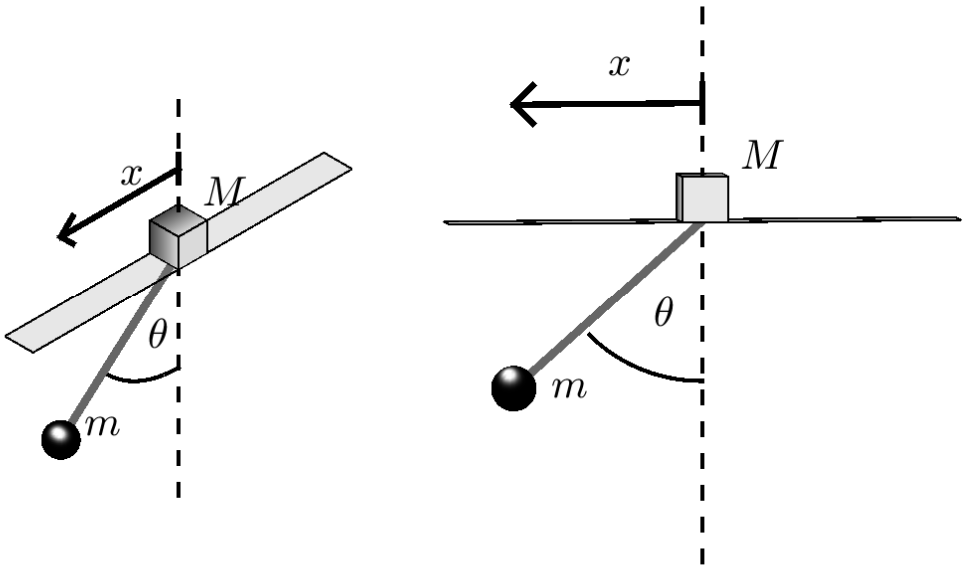


Figure 2.4: Simple hanging crane. The mass M moves on a rail (position x) and mass m is linked to it via a massless rod of length L. We describe the position of the hanging mass via the angle θ.

3. Let us consider an elastic crane, identical to the one depicted in Fig. 2.4, but where the rod is not rigid, hence the length L becomes variable, and the associated potential energy stored in the rod is then given by:

$$V_{\text{rod}} = \frac{1}{2}K(L-L_0)^2 \tag{2.71}$$

where L_0 is the rest-length of the rod. We will now adopt the generalized coordinates

$$\mathrm{q} = \begin{bmatrix} x \\ \theta \\ L \end{bmatrix} \tag{2.72}$$

The kinetic energy of the system now needs to account for L being time-varying, but the computations performed previously do not change. I.e. we can compute:

$$\dot{\mathrm{p}}_m = \frac{\partial \mathrm{p}_m}{\partial \mathrm{q}}\dot{\mathrm{q}} = \begin{bmatrix} \dot{x} + L\dot{\theta}\cos\theta + \dot{L}\sin\theta \\ L\dot{\theta}\sin\theta - \dot{L}\cos\theta \end{bmatrix} \tag{2.73}$$

and

$$\left\|\dot{\mathrm{p}}_m\right\|^2 = \dot{\mathrm{p}}_m^\top \dot{\mathrm{p}}_m = \ldots = L^2\dot{\theta}^2 + \dot{x}^2 + 2L\dot{\theta}\dot{x}\cos\theta + \dot{L}^2 + 2\dot{L}\dot{x}\sin\theta \tag{2.74}$$

The kinetic energy of the hanging crane is therefore given by:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{\mathrm{p}}_m^\top\dot{\mathrm{p}}_m = \frac{1}{2}(m+M)\dot{x}^2 + \frac{1}{2}mL^2\dot{\theta}^2 + mL\dot{\theta}\dot{x}\cos\theta + \frac{1}{2}m\dot{L}^2 + m\dot{L}\dot{x}\sin\theta \tag{2.75}$$

Matrix $W(\mathrm{q})$ is not quite complex in this example. It reads as:

$$W(\mathrm{q}) = \begin{bmatrix} m+M & mL\cos\theta & m\sin\theta \\ mL\cos\theta & mL^2 & 0 \\ m\sin\theta & 0 & m \end{bmatrix} \tag{2.76}$$

It is the same as (2.60), but with an extra row and an extra column stemming from the new coordinate L. Our potential energy now includes the new term (2.71) corresponding to the elastic rod. It reads as:

$$V = -mgL\cos\theta + \frac{1}{2}K(L-L_0)^2 \tag{2.77}$$

We can then assemble the Lagrange function and compute the different terms of the Lagrange equation. We have:

$$\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} (M+m)\dot{x} + mL\dot{\theta}\cos\theta + m\dot{L}\sin\theta \\ mL^2\dot{\theta} + mL\dot{x}\cos\theta \\ m\dot{L} + m\dot{x}\sin\theta \end{bmatrix} = W(\mathrm{q})\dot{\mathrm{q}} \tag{2.78a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} (M+m)\ddot{x} + mL\cos\theta\ddot{\theta} - mL\sin\theta\dot{\theta}^2 + 2m\dot{L}\cos\theta\dot{\theta} + m\ddot{L}\sin\theta \\ mL\cos\theta\ddot{x} + mL^2\ddot{\theta} - mL\dot{\theta}\dot{x}\sin\theta + m\dot{L}\dot{x}\cos\theta + 2mL\dot{L}\dot{\theta} \\ m\ddot{L} + m\ddot{x}\sin\theta + m\dot{\theta}\dot{x}\cos\theta \end{bmatrix} \tag{2.78b}$$

$$\nabla_{\mathrm{q}}\mathscr{L} = \begin{bmatrix} 0 \\ -m(Lg\sin\theta - \dot{L}\dot{x}\cos\theta + L\dot{\theta}\dot{x}\sin\theta) \\ mL\dot{\theta}^2 + m\dot{x}\cos\theta\dot{\theta} + K(L-L_0) + mg\cos\theta \end{bmatrix} \tag{2.78c}$$

The Lagrange equation then reads as:

$$\begin{bmatrix} (M+m)\ddot{x} + m\sin\theta\ddot{L} + mL\cos\theta\ddot{\theta} - mL\sin\theta\dot{\theta}^2 + 2m\dot{L}\cos\theta\dot{\theta} + \\ mL(L\ddot{\theta} + 2\dot{L}\dot{\theta} + \ddot{x}\cos\theta + g\sin\theta) \\ m\ddot{L} + m\sin\theta\ddot{x} + K(L-L_0) - mg\cos\theta - mL\dot{\theta}^2 \end{bmatrix} = 0 \tag{2.79}$$

As previously, the acceleration $\ddot{\mathrm{q}}$ enters linearly in (2.79). We can again observe that (2.79) can be written as:

$$W(\mathrm{q})\ddot{\mathrm{q}} + \begin{bmatrix} m\dot{\theta}(2\dot{L}\cos\theta - L\dot{\theta}\sin\theta) \\ mL(2\dot{L}\dot{\theta} + g\sin\theta) \\ -mL\dot{\theta}^2 + K(L-L_0) - mg\cos\theta \end{bmatrix} = 0 \tag{2.80}$$

and the acceleration can be computed explicitly using:

$$\ddot{\mathrm{q}} = -W(\mathrm{q})^{-1} \begin{bmatrix} m\dot{\theta}(2\dot{L}\cos\theta - L\dot{\theta}\sin\theta) \\ mL(2\dot{L}\dot{\theta} + g\sin\theta) \\ -mL\dot{\theta}^2 + K(L-L_0) - mg\cos\theta \end{bmatrix} \tag{2.81}$$

Unfortunately, (2.81) yields very long expressions and we skip writing it here.

4. Let us revisit the elastic crane example with a different choice of generalized coordinates. Instead of specifying the position of the hanging mass using (θ, L) (these are polar coordinates), it is equally valid to use cartesian coordinates. Let us describe the hanging mass using

$$\mathrm{p} = \begin{bmatrix} \mathrm{p}_1 \\ \mathrm{p}_2 \end{bmatrix} \in \mathbb{R}^2 \tag{2.82}$$

as illustrated in Fig. 2.5, and we chose to order our generalized coordinates q as:

$$\mathrm{q} = \begin{bmatrix} x \\ \mathrm{p} \end{bmatrix} \in \mathbb{R}^3 \tag{2.83}$$

The kinetic energy then reads as:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{\mathrm{p}}^\top\dot{\mathrm{p}} \tag{2.84}$$

and the $W(\mathrm{q})$ matrix reads as:

$$W(\mathrm{q}) = \begin{bmatrix} M & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{2.85}$$

and is constant. The potential energy in the rod reads as:

$$V_{\text{rod}} = \frac{1}{2}K(L-L_0)^2 \quad \text{where} \quad L = \left\| \mathrm{p} - \begin{bmatrix} x \\ 0 \end{bmatrix} \right\| \tag{2.86}$$

and the potential energy of the hanging mass is:

$$V_{\text{gravity}} = -mg\mathrm{p}_2 \tag{2.87}$$

(note that the minus sign here is due to the reference frame chosen for the hanging mass, with the vertical basis vector oriented down, see Fig. 2.5). Hence the Lagrange function reads as:

$$\mathscr{L} = T - V = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\dot{\mathrm{p}}^\top\dot{\mathrm{p}} - \frac{1}{2}K(L-L_0)^2 - mg\mathrm{p}_2 \tag{2.88}$$

We can then compute:

$$\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} M\dot{x} \\ m\dot{\mathrm{p}} \end{bmatrix} \tag{2.89a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathrm{q}}}\mathscr{L} = \begin{bmatrix} M\ddot{x} \\ m\ddot{\mathrm{p}} \end{bmatrix} \tag{2.89b}$$

$$\nabla_{\mathrm{q}}\mathscr{L} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \frac{K(L-L_0)}{L}\begin{bmatrix} \mathrm{p}_1 - x \\ x - \mathrm{p}_1 \\ -\mathrm{p}_2 \end{bmatrix} \tag{2.89c}$$

And write the model as:

$$\begin{bmatrix} \ddot{x} \\ \ddot{\mathrm{p}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{K(L-L_0)}{L}\begin{bmatrix} M^{-1}(\mathrm{p}_1 - x) \\ m^{-1}(x - \mathrm{p}_1) \\ -m^{-1}\mathrm{p}_2 \end{bmatrix} \tag{2.90}$$

5. Consider an elastic hanging chain as depicted in Fig. 2.7. The chain is made of N punctual masses each of mass m that can move in a 3-dimensional space, linked together by elastic rods of rigidity K, and linked to two fixed points by elastic rods as well. The "configuration" of the hanging chain can be described by listing the positions $\mathrm{p}_{1,\dots,N}$ of each mass in the vector $\mathrm{q} \in \mathbb{R}^{3N}$. We will label the fixed end-points as p_0 and p_{N+1}, and these two points will not be part of the generalized coordinates q. For the sake of simplicity, we will consider that the elastic links have a rest-length $L_0 = 0$. The kinetic energy will read as:

$$T = \frac{1}{2}m\dot{\mathrm{q}}^\top\dot{\mathrm{q}} \tag{2.91}$$

hence the matrix $W(\mathrm{q})$ reads as:

$$W(\mathrm{q}) = mI \tag{2.92}$$

where $I \in \mathbb{R}^{3N\times 3N}$ is the identity matrix, and is therefore constant. The potential energy function reads as:

$$V = \underbrace{mg\sum_{k=1}^{N}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\mathrm{p}_k}_{\text{gravity}} + \underbrace{\frac{1}{2}K\sum_{k=0}^{N}\left\|\mathrm{p}_{k+1} - \mathrm{p}_k\right\|^2}_{\text{spring}} \tag{2.93}$$

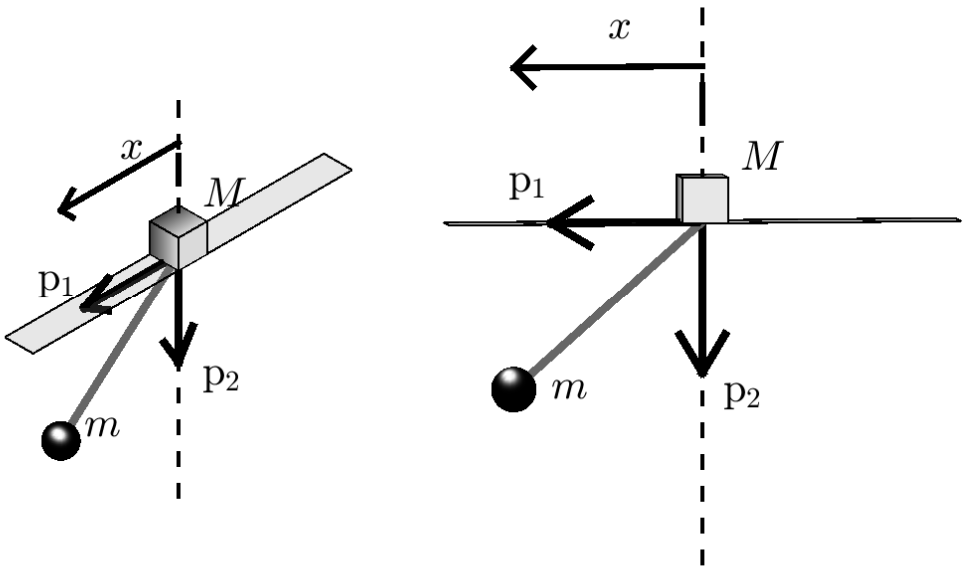


Figure 2.5: Simple hanging crane similar to Fig. 2.4. Here we use a cartesian coordinate system.

where p_0 and p_{N+1} are the fixed positions of the end points. The Lagrange function reads as:

$$\mathscr{L} = \frac{1}{2} m \dot{\mathrm{q}}^\top \dot{\mathrm{q}} - mg \sum_{k=1}^{N} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathrm{p}_k - \frac{1}{2} K \sum_{k=0}^{N} \left\| \mathrm{p}_{k+1} - \mathrm{p}_k \right\|^2 \tag{2.94}$$

We can then compute:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = m \dot{\mathrm{q}} \tag{2.95a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathrm{q}}} \mathscr{L} = m \ddot{\mathrm{q}} \tag{2.95b}$$

$$\nabla_{\mathrm{q}} \mathscr{L} = K \begin{bmatrix} \mathrm{p}_0 - \mathrm{p}_1 \\ \mathrm{p}_1 - \mathrm{p}_2 \\ \vdots \\ \mathrm{p}_{N-1} - \mathrm{p}_N \end{bmatrix} + K \begin{bmatrix} \mathrm{p}_2 - \mathrm{p}_1 \\ \mathrm{p}_3 - \mathrm{p}_2 \\ \vdots \\ \mathrm{p}_{N+1} - \mathrm{p}_N \end{bmatrix} + m \begin{bmatrix} \mathrm{g} \\ \mathrm{g} \\ \vdots \\ \mathrm{g} \end{bmatrix}, \qquad \mathrm{g} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{2.95c}$$

The Lagrange equation then delivers:

$$m \ddot{\mathrm{q}} + K \begin{bmatrix} \mathrm{p}_0 - \mathrm{p}_1 \\ \vdots \\ \mathrm{p}_{N-1} - \mathrm{p}_N \end{bmatrix} + K \begin{bmatrix} \mathrm{p}_2 - \mathrm{p}_1 \\ \vdots \\ \mathrm{p}_{N+1} - \mathrm{p}_N \end{bmatrix} + m \begin{bmatrix} \mathrm{g} \\ \vdots \\ \mathrm{g} \end{bmatrix} = 0 \tag{2.96}$$

which is trivial to put in an explicit form:

$$\ddot{\mathrm{q}} = -\frac{K}{m} \begin{bmatrix} \mathrm{p}_0 - \mathrm{p}_1 \\ \vdots \\ \mathrm{p}_{N-1} - \mathrm{p}_N \end{bmatrix} - \frac{K}{m} \begin{bmatrix} \mathrm{p}_2 - \mathrm{p}_1 \\ \vdots \\ \mathrm{p}_{N+1} - \mathrm{p}_N \end{bmatrix} - \begin{bmatrix} \mathrm{g} \\ \vdots \\ \mathrm{g} \end{bmatrix} \tag{2.97}$$

6. Let us consider now a 2-pendulum crane as illustrated in Fig. 2.6. We proceed with building the kinetic and potential energy functions for this system. To that end, it is best to start with describing the position of the two hanging masses (in a cartesian reference frame), as functions of the generalized coordinates

$$\mathrm{q} = \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \end{bmatrix} \tag{2.98}$$

The cart is at a position:

$$\mathrm{p}_0 = \begin{bmatrix} x \\ 0 \end{bmatrix} \tag{2.99}$$

The first hanging mass is at a cartesian position:

$$\mathrm{p}_1 = \mathrm{p}_0 + R(\theta_1) \begin{bmatrix} 0 \\ -L \end{bmatrix} \tag{2.100}$$

where

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \tag{2.101}$$

and the second hanging mass is at a position: The first hanging mass is at a cartesian position:

$$\mathrm{p}_2 = \mathrm{p}_1 + R(\theta_2) \begin{bmatrix} 0 \\ -L \end{bmatrix} \tag{2.102}$$

We can then readily apply the chain rule (2.2), i.e. to obtain $\dot{\mathrm{p}}_{0,\ldots 2}$. The kinetic energy function then reads as (the cart and the two haging masses are all of mass m):

$$T = \frac{1}{2} m \sum_{k=0}^{2} \dot{\mathrm{p}}_k^\top \dot{\mathrm{p}}_k \tag{2.103}$$

but it is a fairly complex expression given by:

$$T = \frac{1}{2} \dot{\mathrm{q}}^\top W(\mathrm{q}) \dot{\mathrm{q}} \tag{2.104}$$

where matrix $W(\mathrm{q})$ reads as:

$$W(\mathrm{q}) = m \begin{bmatrix} 3 & 2L\cos\theta_1 & L\cos\theta_2 \\ 2L\cos\theta_2 & 2L^2 & L\cos(\theta_1 - \theta_2) \\ L\cos\theta_2 & L\cos(\theta_1 - \theta_2) & L^2 \end{bmatrix} \tag{2.105}$$

The potential energy function is given by:

$$V = mg \sum_{k=1}^{2} \mathrm{p}_{k,2} = -2mgL\cos\theta_1 - mgL\cos\theta_2 \tag{2.106}$$

We can then compute:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = W(\mathrm{q}) \dot{\mathrm{q}} \tag{2.107a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathrm{q}}} \mathscr{L} = W(\mathrm{q}) \ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} \tag{2.107b}$$

$$\nabla_{\mathrm{q}} \mathscr{L} = -mL \begin{bmatrix} 0 \\ g\sin(\theta_1) + g\sin(\theta_1) + \dot{\theta}_1 \dot{x} sin\theta_1 + \dot{x}\dot{\theta}_1 \sin\theta_1 + L\dot{\theta}_1\dot{\theta}_2 \sin(\theta_1 - \theta_2) \\ g\sin\theta_2 + \dot{\theta}_2 \dot{x}\sin\theta_2 - L\dot{\theta}_1\dot{\theta}_2 \sin(\theta_1 - \theta_2) \end{bmatrix} \tag{2.107c}$$

where

$$\frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} = mL \begin{bmatrix} 2\sin\theta_1 \dot{\theta}_1^2 + \sin\theta_2 \dot{\theta}_2^2 \\ L\dot{\theta}_1\dot{\theta}_2 \sin(\theta_1 - \theta_2) + 2\dot{x}\dot{\theta}_1 \sin\theta_1 - L\dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \\ -L\dot{\theta}_1\dot{\theta}_2 \sin(\theta_1 - \theta_2) + \dot{x}\dot{\theta}_2 \sin\theta_2 + L\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) \end{bmatrix} \tag{2.108}$$

If one tries to assemble the resulting model in e.g. its explicit form, by computing (identically to (2.42)):

$$\ddot{\mathrm{q}} = W\left(\mathrm{q}\right)^{-1}\left[\nabla_{\mathrm{q}}\mathscr{L} - \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\dot{\mathrm{q}}\right)\dot{\mathrm{q}}\right] \tag{2.109}$$

However, the symbolic complexity resulting from computing (2.109) is high (and not reported here for this reason).

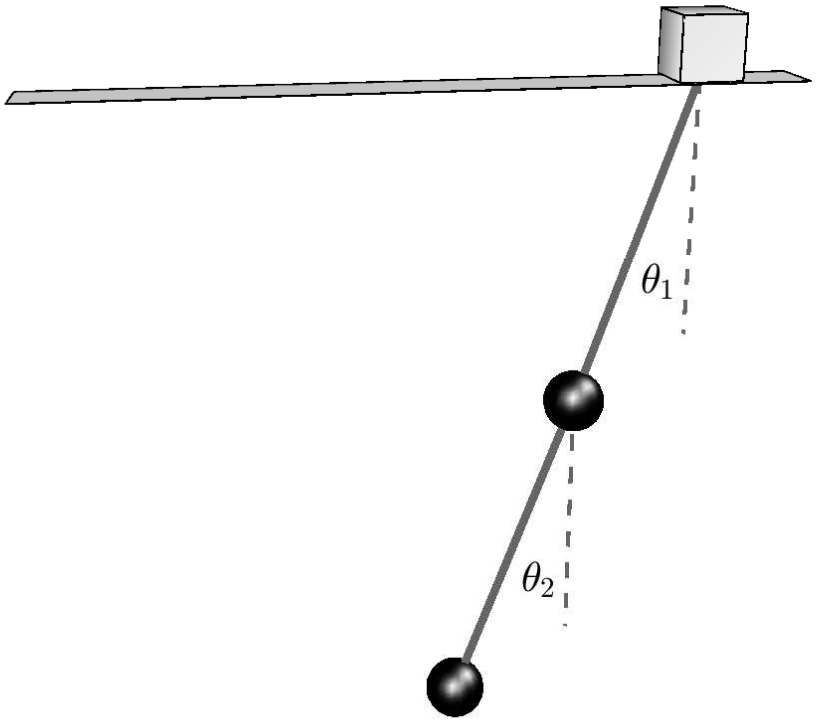


Figure 2.6: Simple hanging crane. The cart of mass m moves on a rail (position x, not shown here) and the two hanging masses also of mass m are linked to it via massless rods of length L. We describe the position of the hanging masses via the angle $\theta_{1,2}$ (relative to the verticals).

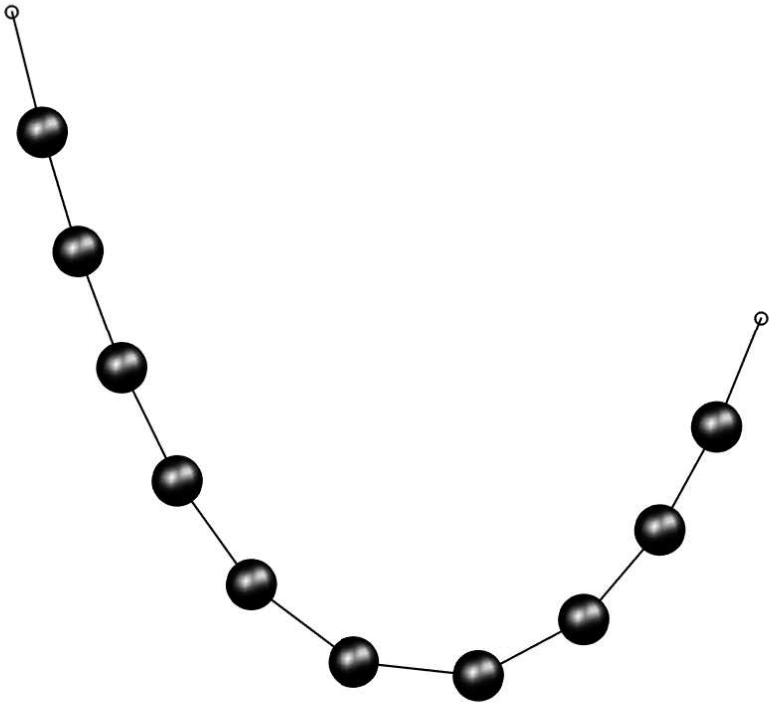


Figure 2.7: Illustration of the elastic hanging chain

□

2.4 EXTERNAL FORCES

The approach we have looked at so far does not include the possibility of having external forces and moments. We will close this gap now. One ought to observe that the Lagrange modelling approach is intrinsically an energy-based point of view of the mechanical system. As a result in the Lagrange approach, we will have to look at the external forces and moments in terms of the energy they deliver to or remove from the system. In other words, we will have to consider the work produced on the system by the external forces and moments. Work is related to motions occurring under forces and moments, and motions are described in the Lagrange approach as changes in the generalized coordinates q. The way external forces and moments are included in the Lagrange formulation in terms of **generalized forces** that we will label Q, which describe the amount of work produced on the system when moving the generalized coordinates. More formally, the generalized forces should satisfy:

$$\nabla_{\mathrm{q}} E = \mathrm{Q} \tag{2.110}$$

i.e. they describe the change of energy in the system when the generalized coordinates q are “moved a little bit".

This concept is often presented in the literature via the following equation:

$$\delta W = \langle \mathrm{Q}, \delta \mathrm{q} \rangle \tag{2.111}$$

which is essentially saying the same as (2.110), i.e. a “small" motion δq, combined with the external forces and moments produces a “small" amount of work δW (a change of energy in the system), and the generalized forces Q relate the small motion to small amount of work produced. Once the generalized forces Q are known, they can be readily included in the Lagrange formalism using:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = F_{\mathrm{q}} \tag{2.112}$$

A fairly systematic procedure can be derived from these concepts whenever a force of a torque is applied punctually somewhere in the system. Suppose that a force given by vector $\mathrm{F} \in \mathbb{R}^n$ (where $n = 1, 2$ or 3 is the number of dimensions in which we are working) in a given fixed reference frame $\mathscr{R}$ is applied at a specific point of the system, having a position $\mathrm{p} \in \mathbb{R}^n$ in the same reference frame $\mathscr{R}$. We then observe that a small change in the generalized coordinates yields a small displacement of the position p given by the Jacobian $\frac{\partial \mathrm{p}}{\partial \mathrm{q}}$, and a small work:

$$\delta W = \mathrm{F}^\top \frac{\partial \mathrm{p}}{\partial \mathrm{q}} \delta \mathrm{q} \tag{2.113}$$

It follows that in this case the generalized force corresponding to F is given by:

$$\mathrm{Q} = \frac{\partial \mathrm{p}}{\partial \mathrm{q}}^\top \mathrm{F} = \nabla_{\mathrm{q}} \mathrm{p} \mathrm{F} \tag{2.114}$$

This principle can be easily extended to several forces. When a set of forces $\mathrm{F}_{1,\dots,m}$ is applied to a set of points $\mathrm{p}_{1,\dots,m}$, then the generalized force is given by:

$$\mathrm{Q} = \sum_{i=1}^{m} \frac{\partial \mathrm{p}_i}{\partial \mathrm{q}}^\top \mathrm{F}_i \equiv \sum_{i=1}^{m} \nabla_{\mathrm{q}} \mathrm{p}_i \mathrm{F}_i \tag{2.115}$$

In order to understand how to use these concepts in practice, let us see a few examples.

Example

- Consider the simple case of a point whose position is described in an orthonormal references frame, such that the generalized coordinates are $\mathrm{q} \in \mathbb{R}^3$, and subject to a

force $\mathrm{F} \in \mathbb{R}^3$ described in the same reference frame. A small displacement $\mathrm{q} \to \mathrm{q} + \delta\mathrm{q}$ combined with the force F yields a small amount of work:

$$\delta W = \langle \mathrm{F}, \delta\mathrm{q} \rangle = \mathrm{F}^\top \delta\mathrm{q} \tag{2.116}$$

such that in this specific case $\mathrm{Q} = \mathrm{F}$, i.e. the generalized force in this system is the force F itself. One can verify that this is also given by (2.115). Indeed, since the position of the point is readily given by the generalized coordinates, i.e. $\mathrm{p} = \mathrm{q}$, it follows that an application of (2.115) yields (where $m = 1$):

$$\mathrm{Q} = \underbrace{\frac{\partial \mathrm{p}}{\partial \mathrm{q}}^\top}_{=I} \mathrm{F} = \mathrm{F} \tag{2.117}$$

- The very same principle applies to moments and rotations. Consider a axis of rotation whose position is described via the angle $\theta \in \mathbb{R}$, and subject to a moment $T \in \mathbb{R}$. The amount of work produces for a small motion $\delta\theta$ due to the moment T is then given by:

$$\delta W = \langle T, \delta\theta \rangle = T\delta\theta \tag{2.118}$$

  such that the generalized force for this case is $\mathrm{Q} = T$. These two examples are fairly trivial. Let us consider next more complex ones.

- Consider the crane example illustrated in Fig. 2.6. Recall that the position of the two masses in a fixed reference frame is given by (see (2.100) and (2.102)):

$$\mathrm{p}_1 = \begin{bmatrix} x \\ 0 \end{bmatrix} + R(\theta_1) \begin{bmatrix} 0 \\ -L \end{bmatrix} \tag{2.119a}$$

$$\mathrm{p}_2 = \mathrm{p}_1 + R(\theta_2) \begin{bmatrix} 0 \\ -L \end{bmatrix} \tag{2.119b}$$

  with the generalized coordinates

$$\mathrm{q} = \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \end{bmatrix} \tag{2.120}$$

  We then observe that:

$$\frac{\partial \mathrm{p}_1}{\partial \mathrm{q}}^\top = \begin{bmatrix} 1 & 0 \\ L\cos\theta_1 & L\sin\theta_1 \\ 0 & 0 \end{bmatrix}, \qquad \frac{\partial \mathrm{p}_2}{\partial \mathrm{q}}^\top = \begin{bmatrix} 1 & 0 \\ L\cos\theta_1 & L\sin\theta_1 \\ L\cos\theta_2 & L\sin\theta_2 \end{bmatrix} \tag{2.121}$$

  If forces $\mathrm{F}_{1,2} \in \mathbb{R}^2$ are applied to masses m_1, m_2 respectively, then the generalized force then reads as:

$$\mathrm{Q} = \begin{bmatrix} 1 & 0 \\ L\cos\theta_1 & L\sin\theta_1 \\ 0 & 0 \end{bmatrix} \mathrm{F}_1 + \begin{bmatrix} 1 & 0 \\ L\cos\theta_1 & L\sin\theta_1 \\ L\cos\theta_2 & L\sin\theta_2 \end{bmatrix} \mathrm{F}_2 \tag{2.122}$$

2.5 CONSTRAINED LAGRANGE MECHANICS

We have been discussing so far cases where the generalized coordinates can “move independently" from each other, in the sense that any element of the set $\mathbb{R}^{n_q}$ is admissible for the vector of generalized coordinates q. E.g.

- In the spring-mass system, any position x is “acceptable" for the system (even if for practical reasons the position of the mass may be restricted in a real system)
- In the hanging crane example, any value of θ and x are valid (even if in practice the rail may have a limited length and the hanging mass may not be physically allowed to get above the rail)

To clarify where we are going here, let us consider a simple example where the generalized coordinates are not free to “move independently". Let us consider a “bowl" in 3D, described by the scalar equation $c(p) = 0$ (see Fig. 2.8 for an illustration), where $p \in \mathbb{R}^3$ are cartesian coordinates and

$$c(p) = p_3 - \frac{1}{4}\left(p_2^2 + p_1^2\right) \in \mathbb{R} \tag{2.123}$$

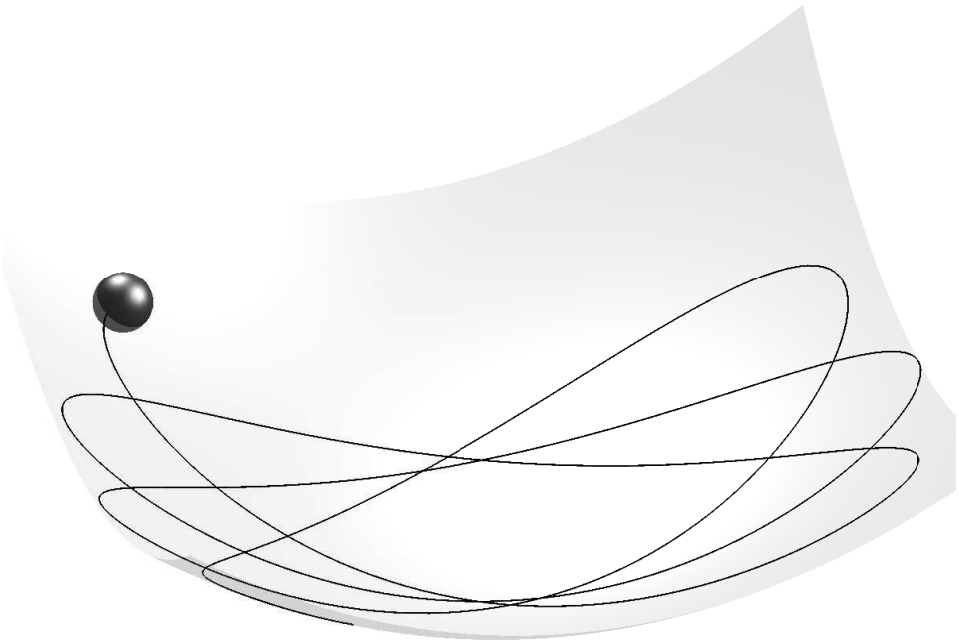


Figure 2.8: Illustration of the bowl example for (2.123).

Let us consider a mass m moving in 3D, but forced to slide on the surface of the bowl. The position of the mass can be described by the cartesian coordinates $p \in \mathbb{R}^3$, but since the mass moves at the surface of the bowl, the position p is not “free" to move everywhere, but

is “constrained" to move in the 2D “space" described by (2.123). Indeed, only positions that satisfy $\mathrm{c}(\mathrm{p}) = 0$ are admissible. Put differently, the generalized coordinates are not “free" to move independently.

More formally, we will consider mechanical systems of generalized coordinates $\mathrm{q} \in \mathbb{R}^{n_\mathrm{q}}$ where the generalized coordinates are constrained by $\mathrm{c}(\mathrm{q}) = 0$ with $\mathrm{c} : \mathbb{R}^{n_\mathrm{q}} \mapsto \mathbb{R}^{n_\mathrm{c}}$ and $n_\mathrm{c} < n_\mathrm{q}$. Mathematically, $\mathrm{c}(\mathrm{q}) = 0$ describes a *manifold* in $\mathbb{R}^{n_\mathrm{q}}$, i.e. a (not necessarily flat) surface of dimension $n_\mathrm{q} - n_\mathrm{c}$. We ought to observe here that the number of dimension of the manifold in which the system moves, i.e. the difference $n_\mathrm{q} - n_\mathrm{c}$ is actually the number of degrees of freedom (DoF) that the system physically has. Being able to treat this type of problem systematically is extremely useful in a number of complex mechanical applications.

Fortunately, the Lagrange formalism we have discussed so far can be readily extended to constrained problems, with minimal changes. The construction of the kinetic and potential energy functions for the system can be done without any regard to the constraint function $\mathrm{c}(\mathrm{q}) = 0$. The constraints appear in the Lagrange function, which is then assembled as:

$$\mathscr{L}(\mathrm{q}, \dot{\mathrm{q}}, \mathrm{z}) = T(\mathrm{q}, \dot{\mathrm{q}}) - V(\mathrm{q}) - \langle \mathrm{z}, \mathrm{c}(\mathrm{q}) \rangle \tag{2.124}$$

where z is a new set of variables, and usually labelled the *Lagrange multipliers* associated to the constraint function c. When function $\mathrm{c}(\mathrm{q})$ has as image space the vector space $\mathbb{R}^{n_\mathrm{c}}$ (i.e. when it returns a “standard" vector), then $\mathrm{z} \in \mathbb{R}^{n_\mathrm{c}}$ and we can rewrite (2.124) simply as[2]:

$$\mathscr{L}(\mathrm{q}, \dot{\mathrm{q}}, \mathrm{z}) = T(\mathrm{q}, \dot{\mathrm{q}}) - V(\mathrm{q}) - \mathrm{z}^\top \mathrm{c}(\mathrm{q}) \tag{2.125}$$

Beyond this point, the Lagrange formalism applies without any modification, i.e. the dynamics associated to a system of kinetic and potential energy T and V and constrained to evolve on the manifold $\mathrm{c}(\mathrm{q})$ are described by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = \mathrm{Q} \tag{2.126a}$$

$$\mathrm{c}(\mathrm{q}) = 0 \tag{2.126b}$$

The mathematical justification for this construction are beyond the scope of this course, but we will try to build some intuitions via examples. Before deploying examples of application of the constrained Lagrange equation, we can do the same work as in (2.38) and (2.39). I.e. one can verify that the equalities:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = W(\mathrm{q})\,\dot{\mathrm{q}}. \tag{2.127}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathrm{q}}} \mathscr{L} = \frac{\mathrm{d}}{\mathrm{d}t}\left(W(\mathrm{q})\,\dot{\mathrm{q}}\right) = W(\mathrm{q})\,\ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\,\dot{\mathrm{q}}\right)\dot{\mathrm{q}} \tag{2.128}$$

[2] the generic form (2.124) allows one to consider constraint function in “exotic" vector spaces such as e.g. matrix spaces or Hilbert spaces. We will discuss only the simple case of “standard" vector spaces here.

still hold, and are actually not impacted by the presence of the constraints $\mathrm{c}(\mathrm{q})$. The constraints modify the Lagrange equation via the term:

$$\nabla_{\mathrm{q}}\mathscr{L} = \nabla_{\mathrm{q}} T - \nabla_{\mathrm{q}} V - \nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z} \tag{2.129}$$

We ought to recall here remark 3. of Section 2.3: the term $\nabla_{\mathrm{q}}\mathscr{L}$ in the Lagrange equation holds forces that are "intrinsic" to the system (potential, centrifugal). This remark still holds here, and we will see in the illustrative examples below that the term $\nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z}$ is in fact akin to a force "keeping the system on the manifold c".

As observed previously in Section 2.3, we can assemble the Lagrange equation (2.126) in the form:

$$W(\mathrm{q})\,\ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\,\dot{\mathrm{q}}\right)\dot{\mathrm{q}} - \nabla_{\mathrm{q}} T + \nabla_{\mathrm{q}} V + \nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z} = \mathrm{Q} \tag{2.130a}$$

$$\mathrm{c}(\mathrm{q}) = 0 \tag{2.130b}$$

And the accelerations $\ddot{\mathrm{q}}$ can be explicitly expressed from (2.130a) as:

$$\ddot{\mathrm{q}} = W(\mathrm{q})^{-1}\left[\mathrm{Q} + \nabla_{\mathrm{q}} T - \nabla_{\mathrm{q}} V - \nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z} - \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\,\dot{\mathrm{q}}\right)\dot{\mathrm{q}}\right] \tag{2.131}$$

and as previously, for $W(\mathrm{q})$ constant, equation (2.44) reduces to:

$$\ddot{\mathrm{q}} = W^{-1}\left[\mathrm{Q} - \nabla_{\mathrm{q}} V - \nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z}\right] \tag{2.132}$$

We can observe again here that the Lagrange equation (and their explicit forms (2.131) and (2.132)) deliver the acceleration $\ddot{\mathrm{q}}$ as a function of q, $\dot{\mathrm{q}}$, Q and of the Lagrange multipliers z. Hence, a simulation of the model could be produced for given initial conditions q(0), $\dot{\mathrm{q}}(0)$, given external forces $\mathrm{Q}(t)$ (given at all time) and the Lagrange multipliers $\mathrm{z}(t)$. Hence, in order to compute a simulation of the model equations, we need to calculate somehow the (time-varying) Lagrange multipliers z at every time instant of the simulation. We will investigate in the next section how to do that.

Let us now consider a list of examples in order to digest these theoretical concepts.

Examples

Let us go through a few examples of constrained Lagrange modelling and try to gather some intuitions on the meaning of the constrained Lagrange equation (2.126).

1. Let us start with our bowl example of Fig. 2.8 to detail the procedure behind the constrained Lagrange equation (2.126). The mass position is described in the cartesian coordinates $\mathrm{q} \equiv \mathrm{p} \in \mathbb{R}^3$, such that the kinetic and potential energy function read as:

$$T = \frac{1}{2} m \dot{\mathrm{p}}^{\top}\dot{\mathrm{p}}, \qquad V = mg\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\mathrm{p} \tag{2.133}$$

Note that we can assemble these functions by completely disregarding the existence of the constraint $c(q) = 0$. We then assemble the constrained Lagrange function as per (2.125):

$$\mathscr{L} = \frac{1}{2} m \dot{p}^\top \dot{p} - mg \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} p - z^\top \left(p_3 - \frac{1}{4}\left(p_2^2 + p_1^2\right) \right) \tag{2.134}$$

Note that here the constraint is scalar, such that $z \in \mathbb{R}$, and the scalar product $z^\top c(q)$ is a simple product here, i.e. we can rewrite the Lagrange function:

$$\mathscr{L} = \frac{1}{2} m \dot{p}^\top \dot{p} - mg \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} p - z \left(p_3 - \frac{1}{4}\left(p_2^2 + p_1^2\right) \right) \tag{2.135}$$

and deploy the Lagrange equation as usual:

$$\nabla_{\dot{q}} \mathscr{L} = m \dot{p} \tag{2.136a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{q}} \mathscr{L} = m \ddot{p} \tag{2.136b}$$

$$\nabla_q \mathscr{L} = -mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \nabla_q c \cdot z, \qquad \text{where} \qquad \nabla_q c = \begin{bmatrix} -\frac{1}{2} p_1 \\ -\frac{1}{2} p_2 \\ 1 \end{bmatrix} \tag{2.136c}$$

The dynamics of the mass in the bowl are then given by:

$$m \ddot{p} = -mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - z \begin{bmatrix} -\frac{1}{2} p_1 \\ -\frac{1}{2} p_2 \\ 1 \end{bmatrix} \tag{2.137a}$$

$$0 = p_3 - \frac{1}{4}\left(p_2^2 + p_1^2\right) \tag{2.137b}$$

As detailed in the theory above, it is worth here briefly underlining that:

- equation (2.137a) delivers the system acceleration $\ddot{p}$ for p and z known
- equation (2.137b) is scalar, and describes the condition that the position p ought to satisfy in order to "be on" the manifold (i.e. the bowl here). However, one ought to observe that (2.137b) does not deliver the unknown z. As a matter of fact, z does not even appear in (2.137b). Hence, while (2.137) provides 4 equations for the 4 unknown variables $\ddot{q}$, z, it does not provide them as such, as it fails to provide z.

2. Let us revisit the crane of Fig. 2.5 in cartesian coordinates, assuming that the link between the cart and the hanging mass is (infinitely) rigid. The kinetic energy function is built in the same way as in the elastic example above, i.e. we use the coordinates

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \in \mathbb{R}^2 \tag{2.138}$$

for position of the hanging mass, and x for the position of the cart. We chose to order our generalized coordinates q as:

$$\mathrm{q} = \begin{bmatrix} x \\ \mathrm{p} \end{bmatrix} \tag{2.139}$$

The kinetic energy reads as (2.84), i.e.:

$$T = \frac{1}{2} M \dot{x}^2 + \frac{1}{2} m \dot{\mathrm{p}}^\top \dot{\mathrm{p}} \tag{2.140}$$

and the W matrix reads as (2.85), i.e.:

$$W = \begin{bmatrix} M & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{2.141}$$

and is constant. The potential energy now involves only the energy from gravity, i.e.

$$V = -mg\mathrm{p}_2 \tag{2.142}$$

The constraint is then the distance between the cart and the hanging mass. The vector describing the link is given by:

$$\boldsymbol{\delta} = \mathrm{p} - \begin{bmatrix} x \\ 0 \end{bmatrix} \tag{2.143}$$

And the constraint imposed by the link in the system can e.g. be written as:

$$\left(\boldsymbol{\delta}^\top \boldsymbol{\delta}\right)^{\frac{1}{2}} - L = 0 \tag{2.144}$$

For computational reasons, it is useful to “get rid" of the square root function in (2.144) and to scale the constraint by $\frac{1}{2}$, i.e. we write:

$$\mathrm{c}\left(\mathrm{q}\right) = \frac{1}{2}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) = 0 \tag{2.145}$$

which is equivalent to (2.144). The Lagrange function then reads as:

$$\mathscr{L} = \frac{1}{2} M \dot{x}^2 + \frac{1}{2} m \dot{\mathrm{p}}^\top \dot{\mathrm{p}} + mg\mathrm{p}_2 - \frac{1}{2}\mathrm{z}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) \tag{2.146}$$

We can then deploy the Lagrange equation as usual:

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = W \dot{\mathrm{q}} \tag{2.147a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathrm{q}}} \mathscr{L} = W \ddot{\mathrm{q}} \tag{2.147b}$$

$$\nabla_{\mathrm{q}} \mathscr{L} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \mathrm{z} \begin{bmatrix} x - \mathrm{p}_1 \\ \mathrm{p}_1 - x \\ \mathrm{p}_2 \end{bmatrix} \tag{2.147c}$$

The model then reads as:

$$W\ddot{\mathrm{q}} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathrm{z} \begin{bmatrix} x - \mathrm{p}_1 \\ \mathrm{p}_1 - x \\ \mathrm{p}_2 \end{bmatrix} = 0 \tag{2.148a}$$

$$\frac{1}{2}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) = 0 \tag{2.148b}$$

It a more explicit form, it reads as:

$$\ddot{\mathrm{q}} = W^{-1}\left(\begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \mathrm{z} \begin{bmatrix} x - \mathrm{p}_1 \\ \mathrm{p}_1 - x \\ \mathrm{p}_2 \end{bmatrix}\right) \tag{2.149a}$$

$$0 = \frac{1}{2}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) \tag{2.149b}$$

The observations already made above concerning solving (2.149) for z still hold here. It is interesting at this stage to compare (2.149) to its equivalent (2.70) developed in polar coordinates (i.e. using the angle θ to describe the position of the hanging mass). While (2.149) holds more equations than (2.70) (4 vs. 2), its symbolic complexity is lower. Indeed, while (2.70) comprises several trigonometric terms (sin and cos), and a division, (2.149) includes only bilinear terms (i.e. products of the variables).

3. We will revisit now the 2-pendulum crane illustrated in Fig. 2.6. Modelling this system using polar coordinates (i.e. the angles of the pendulums) was resulting in a model of very high symbolic complexity (see (2.107)-(2.108) and the following remarks). Let us see the outcome of approaching the modelling of the same system using cartesian coordinates and constrained Lagrange. We select the generalized coordinates:

$$\mathrm{q} = \begin{bmatrix} x \\ \mathrm{p}_1 \\ \mathrm{p}_2 \end{bmatrix}, \qquad \mathrm{p}_{1,2} \in \mathbb{R}^2 \tag{2.150}$$

The kinetic energy function reads as:

$$T = \frac{1}{2} m \dot{\mathrm{q}}^\top \dot{\mathrm{q}} \tag{2.151}$$

hence $W = mI$ (where I is the 3×3 identity matrix). The potential energy function reads as:

$$V = -mg \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \end{bmatrix} \mathrm{q} \tag{2.152}$$

The constraints then read as:

$$\mathrm{c}(\mathrm{q}) = \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta}_1^\top \boldsymbol{\delta}_1 - L^2 \\ \boldsymbol{\delta}_2^\top \boldsymbol{\delta}_2 - L^2 \end{bmatrix} = 0 \tag{2.153}$$

where

$$\boldsymbol{\delta}_1 = \begin{bmatrix} x \\ 0 \end{bmatrix} - \mathrm{p}_1, \qquad \boldsymbol{\delta}_2 = \mathrm{p}_2 - \mathrm{p}_1 \tag{2.154}$$

The Lagrange function reads as:

$$\mathscr{L} = \frac{1}{2} m \dot{\mathrm{q}}^\top \dot{\mathrm{q}} + mg \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \end{bmatrix} \mathrm{q} - \frac{1}{2} \mathrm{z}^\top \begin{bmatrix} \boldsymbol{\delta}_1^\top \boldsymbol{\delta}_1 - L^2 \\ \boldsymbol{\delta}_2^\top \boldsymbol{\delta}_2 - L^2 \end{bmatrix} \tag{2.155}$$

where $\mathrm{z} \in \mathbb{R}^2$. We can then compute

$$\nabla_{\dot{\mathrm{q}}} \mathscr{L} = m \dot{\mathrm{q}} \tag{2.156a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathrm{q}}} \mathscr{L} = m \ddot{\mathrm{q}} \tag{2.156b}$$

$$\nabla_{\mathrm{q}} \mathscr{L} = \begin{bmatrix} 0 \\ 0 \\ mg \\ 0 \\ mg \end{bmatrix} - \begin{bmatrix} x - \begin{bmatrix} 1 & 0 \end{bmatrix} \mathrm{p}_1 & 0 \\ -\boldsymbol{\delta}_1 & -\boldsymbol{\delta}_2 \\ 0 & \boldsymbol{\delta}_2 \end{bmatrix} \mathrm{z} \tag{2.156c}$$

The final model in a explicit form then reads as:

$$\ddot{\mathrm{q}} = \begin{bmatrix} 0 \\ 0 \\ g \\ 0 \\ g \end{bmatrix} - \frac{1}{m} \begin{bmatrix} x - \begin{bmatrix} 1 & 0 \end{bmatrix} \mathrm{p}_1 & 0 \\ -\boldsymbol{\delta}_1 & -\boldsymbol{\delta}_2 \\ 0 & \boldsymbol{\delta}_2 \end{bmatrix} \mathrm{z} \tag{2.157}$$

$$0 = \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta}_1^\top \boldsymbol{\delta}_1 - L^2 \\ \boldsymbol{\delta}_2^\top \boldsymbol{\delta}_2 - L^2 \end{bmatrix} \tag{2.158}$$

This ought to be compared to (2.107)-(2.108), where the final model was not even provided because of its very high symbolic complexity. The reduction of symbolic complexity is resulting from the choice of cartesian coordinates, which yields a constant and diagonal W matrix. As a result, the complex terms $\frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q}) \dot{\mathrm{q}} \right) \dot{\mathrm{q}}$ in

$$\ddot{\mathrm{q}} = W\left(\mathrm{q}\right)^{-1} \left[Q + \nabla_{\mathrm{q}} T - \nabla_{\mathrm{q}} V - \nabla_{\mathrm{q}} \mathrm{c} \cdot \mathrm{z} - \frac{\partial}{\partial \mathrm{q}} \left(W\left(\mathrm{q}\right) \dot{\mathrm{q}} \right) \dot{\mathrm{q}} \right] \tag{2.159}$$

disappears, and the inverse $W\left(\mathrm{q}\right)^{-1}$ is trivial.

2.5.1 Handling Models from Constrained Lagrange

In the previous section we have seen that we can write the accelerations $\ddot{\mathrm{q}}$

$$\ddot{\mathrm{q}} = W^{-1} \left[\mathrm{Q} - \nabla_{\mathrm{q}} V - \nabla_{\mathrm{q}} \mathrm{c} \cdot \mathrm{z} \right] \tag{2.160}$$

As observed before, in order to compute a simulation from (2.160) we need the (time-varying) Lagrange multipliers z at every time instant of the simulation.

Unfortunately, a problem arises here. Indeed, since we use equation (2.126a) (or its equivalent form (2.130a)) to compute $\ddot{q}$, we are left with equation (2.166b) to determine the unknown z. However, equation (2.166b) *does not* deliver z, indeed it is not even a function of z, and thus cannot inform us - as such - about z. We will see in the following how to circumvent this problem, and we will put some further formalism around this issue in Section 5.
Running simulations for mechanical system requires one to be able to compute the system accelerations $\ddot{q}$ as a function of the positions and velocities q, $\dot{q}$ and of the external forces Q. In the unconstrained case, this is readily feasible as long as matrix $W(q)$ is full rank.

In the constrained case, we have raised the issue before that a model arising from constrained Lagrange and taking the form:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = Q \tag{2.161a}$$

$$c(q) = 0 \tag{2.161b}$$

does not deliver as such the accelerations of the system. Indeed, while equation (2.161a) delivers the acceleration $\ddot{q}$ via its explicit version:

$$\ddot{q} = W(q)^{-1}\left[Q + \nabla_q T - \nabla_q V - \nabla_q c \cdot z - \frac{\partial}{\partial q}\left(W(q)\dot{q}\right)\dot{q}\right] \tag{2.162}$$

the accelerations $\ddot{q}$ are function of the unknown z, which is not delivered by (2.161b), as $c(q)$ is not even a function of z. In this section we will see how to tackle this problem.

If one sets z = 0 in the dynamic equation (2.161a), then one can verify that the equation would be describing a “free" motion, i.e. what the system would do if the constraints $c(q) = 0$ were not present at all. E.g. in the “bowl" example above, the ball would free fall, and in the 2-pendulums crane example, the hanging masses would free fall and the cart would move along its rail as if not connected to anything. One can therefore construe $\nabla_q c \cdot z$ in (2.162) as a term that will enforce the constraints $c(q) = 0$ by manipulating $\ddot{q}$ resulting from (2.162) via adequate selections of the variables z. As a matter of fact, the combination of terms

$$W(q)^{-1}\nabla_q c \cdot z \tag{2.163}$$

arising in (2.162) yield an acceleration that is added to the other “sources" of accelerations (e.g. stemming from forces, gravity, centrifugal effects, etc.). One can observe that $W(q)^{-1}\nabla_q c$ is a matrix of dimension $n_q \times n_c$, such that the term (2.163) yields accelerations in the subspace spanned by the columns of $W(q)^{-1}\nabla_q c$.

Let us then build some intuition behind what will happen here. We know that z ought to be chosen such that the accelerations $\ddot{\mathrm{q}}$ enforce the constraints $\mathrm{c}(\mathrm{q}) = 0$. However, the constraints specify conditions on the system position, not on its accelerations. We ought to understand, though, that the positions q are obviously influenced by the accelerations $\ddot{\mathrm{q}}$ (via 2 integrations), such that the accelerations influence $\mathrm{c}(\mathrm{q})$ (via 2 integrations). In order to chose the z adequately, we need to make the impact of the accelerations on $\mathrm{c}(\mathrm{q})$ appear explicitly.

Fortunately, this influence is fairly simple to unpack. Indeed, let us take two time derivatives of the constraints $\mathrm{c}(\mathrm{q})$ (in order to "rewind" the two integrations from $\ddot{\mathrm{q}}$ to q). We should observe that if $\mathrm{c}(\mathrm{q}) = 0$ is enforced throughout the trajectory of the system (i.e. at every time instant), then:

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathrm{c}(\mathrm{q}) = 0 \tag{2.164}$$

also hold at all time for any $k \geq 0$. We also observe that $\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathrm{c}(\mathrm{q}) = 0$ is a condition where the accelerations appear explicitly. In order to see that, we simply need to apply some chain rules:

$$\dot{\mathrm{c}}(\mathrm{q}, \dot{\mathrm{q}}) = \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{c}(\mathrm{q}) = \frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}} \tag{2.165a}$$

$$\ddot{\mathrm{c}}(\mathrm{q}, \dot{\mathrm{q}}, \ddot{\mathrm{q}}) = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathrm{c}(\mathrm{q}) = \frac{\partial \mathrm{c}}{\partial \mathrm{q}}\ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}}\left(\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}}\right)\dot{\mathrm{q}} \tag{2.165b}$$

We can then assemble the Lagrange equation (2.126a) and equation (2.165b) in the new model:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = Q \tag{2.166a}$$

$$\ddot{\mathrm{c}}(\mathrm{q}, \dot{\mathrm{q}}, \ddot{\mathrm{q}}) = 0 \tag{2.166b}$$

Or in the explicit form similar to (2.130):

$$W(\mathrm{q})\ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\dot{\mathrm{q}}\right)\dot{\mathrm{q}} - \nabla_{\mathrm{q}}T + \nabla_{\mathrm{q}}V + \nabla_{\mathrm{q}}\mathrm{c}\cdot\mathrm{z} = Q \tag{2.167a}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\ddot{\mathrm{q}} + \frac{\partial}{\partial \mathrm{q}}\left(\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}}\right)\dot{\mathrm{q}} = 0 \tag{2.167b}$$

Recall that our original problem is that the constrained Lagrange equations (2.161) do not deliver the variables z, which are needed in order to compute the accelerations $\ddot{\mathrm{q}}$. In order to solve the problem we have time-differentiated the constraints $\mathrm{c}(\mathrm{q}) = 0$, and obtained (2.167). However, It is not yet obvious that the modified constraints (2.167b) deliver z. As a matter of fact, they are still not a function of z. Let us investigate whether (2.167) is now capable of delivering the pair $\ddot{\mathrm{q}}$, z. Here we can proceed via algebraic manipulations over

(2.167). However, it is much easier to address the question by observing that (2.167) is linear in $\ddot{\mathrm{q}}$, z. Indeed on can rewrite (2.167) as:

$$\underbrace{\begin{bmatrix} W(\mathrm{q}) & \nabla_{\mathrm{q}}\mathrm{c} \\ \nabla_{\mathrm{q}}\mathrm{c}^{\top} & 0 \end{bmatrix}}_{:=M(\mathrm{q})} \begin{bmatrix} \ddot{\mathrm{q}} \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} Q - \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\dot{\mathrm{q}}\right)\dot{\mathrm{q}} + \nabla_{\mathrm{q}}T - \nabla_{\mathrm{q}}V \\ -\frac{\partial}{\partial \mathrm{q}}\left(\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}}\right)\dot{\mathrm{q}} \end{bmatrix} \tag{2.168}$$

Here we can readily see that (2.168) delivers $\ddot{\mathrm{q}}$, z jointly if matrix $M(\mathrm{q})$ is full rank. One can also write the model in an explicit form:

$$\begin{bmatrix} \ddot{\mathrm{q}} \\ \mathrm{z} \end{bmatrix} = M(\mathrm{q})^{-1} \begin{bmatrix} Q - \frac{\partial}{\partial \mathrm{q}}\left(W(\mathrm{q})\dot{\mathrm{q}}\right)\dot{\mathrm{q}} + \nabla_{\mathrm{q}}T - \nabla_{\mathrm{q}}V \\ -\frac{\partial}{\partial \mathrm{q}}\left(\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}}\right)\dot{\mathrm{q}} \end{bmatrix} \tag{2.169}$$

However, as observed previously for matrix $W(\mathrm{q})$ alone, inverting the symbolic matrix $M(\mathrm{q})$ can yield an extremely complex matrix $M(\mathrm{q})^{-1}$, even if $M(\mathrm{q})$ is fairly "simple". For this reason, it is often preferable to work with the model (2.168) in its implicit form, or to treat the matrix inversion $M(\mathrm{q})^{-1}$ numerically when deploying model (2.169).

Examples

Let us revisit some of the examples of Section 2.5 in the light of the model transformation described above.

1. The bowl example was ending with the model equations

$$m\ddot{\mathrm{p}} = -mg\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \mathrm{z}\begin{bmatrix} -\frac{1}{2}\mathrm{p}_1 \\ -\frac{1}{2}\mathrm{p}_2 \\ 1 \end{bmatrix} \tag{2.170a}$$

$$0 = \mathrm{p}_3 - \frac{1}{4}\left(\mathrm{p}_2^2 + \mathrm{p}_1^2\right) \tag{2.170b}$$

   where (2.170b) is $\mathrm{c}(\mathrm{q}) = 0$ written explicitly. In order to perform the model transformation detailed above on this model, we need to time-differentiate (2.170b) twice:

$$\mathrm{c}(\mathrm{q}) = \mathrm{p}_3 - \frac{1}{4}\left(\mathrm{p}_2^2 + \mathrm{p}_1^2\right) \tag{2.171a}$$

$$\dot{\mathrm{c}}(\mathrm{q}) = \dot{\mathrm{p}}_3 - \frac{1}{2}\left(\mathrm{p}_2\dot{\mathrm{p}}_2 + \mathrm{p}_1\dot{\mathrm{p}}_1\right) \tag{2.171b}$$

$$\ddot{\mathrm{c}}(\mathrm{q}) = \ddot{\mathrm{p}}_3 - \frac{1}{2}\left(\mathrm{p}_2\ddot{\mathrm{p}}_2 + \mathrm{p}_1\ddot{\mathrm{p}}_1 + \dot{\mathrm{p}}_2^2 + \dot{\mathrm{p}}_1^2\right) \tag{2.171c}$$

   These expressions can be obtained directly as above, or via using the construction (2.165). The transformed model then reads as:

$$m\ddot{\mathrm{p}} = -mg\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \mathrm{z}\begin{bmatrix} -\frac{1}{2}\mathrm{p}_1 \\ -\frac{1}{2}\mathrm{p}_2 \\ 1 \end{bmatrix} \tag{2.172a}$$

$$0 = \ddot{\mathrm{p}}_3 - \frac{1}{2}\left(\mathrm{p}_2\ddot{\mathrm{p}}_2 + \mathrm{p}_1\ddot{\mathrm{p}}_1 + \dot{\mathrm{p}}_2^2 + \dot{\mathrm{p}}_1^2\right) \tag{2.172b}$$

It can be put in the form (2.168):

$$\underbrace{\begin{bmatrix} m & 0 & 0 & -\frac{1}{2}\mathrm{p}_1 \\ 0 & m & 0 & -\frac{1}{2}\mathrm{p}_2 \\ 0 & 0 & m & 1 \\ -\frac{1}{2}\mathrm{p}_1 & -\frac{1}{2}\mathrm{p}_2 & 1 & 0 \end{bmatrix}}_{:=M(\mathrm{q})} \begin{bmatrix} \ddot{\mathrm{p}}_1 \\ \ddot{\mathrm{p}}_2 \\ \ddot{\mathrm{p}}_3 \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \\ \dot{\mathrm{p}}_2^2 + \dot{\mathrm{p}}_1^2 \end{bmatrix} \tag{2.173}$$

The determinant of matrix $M(\mathrm{q})$ reads as:

$$\det\left(M(\mathrm{q})\right) = -\frac{m^2}{4}\left(\mathrm{p}_1^2 + \mathrm{p}_2^2 + 4\right) < 0 \tag{2.174}$$

and is non-zero for any position p, such that (2.173) is always well defined. The inverse of matrix $M(\mathrm{q})$ is fairly complex, such that writing (2.173) explicitly is tedious.

2. Let us look at the crane example, which had the original model

$$W\ddot{\mathrm{q}} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \mathrm{z}\begin{bmatrix} x - \mathrm{p}_1 \\ \mathrm{p}_1 - x \\ \mathrm{p}_2 \end{bmatrix} \tag{2.175a}$$

$$0 = \frac{1}{2}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) \tag{2.175b}$$

where $\boldsymbol{\delta} = \mathrm{p} - \begin{bmatrix} x \\ 0 \end{bmatrix}$ and

$$W = \begin{bmatrix} M & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{2.176}$$

We can then proceed with time-differentiating the constraints:

$$\mathrm{c}(\mathrm{q}) = \frac{1}{2}\left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - L^2\right) \tag{2.177a}$$

$$\dot{\mathrm{c}}(\mathrm{q}) = \boldsymbol{\delta}^\top \dot{\boldsymbol{\delta}} \tag{2.177b}$$

$$\ddot{\mathrm{c}}(\mathrm{q}) = \boldsymbol{\delta}^\top \ddot{\boldsymbol{\delta}} + \dot{\boldsymbol{\delta}}^\top \dot{\boldsymbol{\delta}} \tag{2.177c}$$

The model put in the form (2.168) then reads as:

$$\underbrace{\begin{bmatrix} & & & x - \mathrm{p}_1 \\ & W & & \mathrm{p}_1 - x \\ & & & \mathrm{p}_2 \\ x - \mathrm{p}_1 & \mathrm{p}_1 - x & \mathrm{p}_2 & 0 \end{bmatrix}}_{:=M(\mathrm{q})} \begin{bmatrix} \ddot{x} \\ \ddot{\mathrm{p}} \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \\ \dot{x}^2 + \dot{\mathrm{p}}_1^2 + \dot{\mathrm{p}}_2^2 - 2\dot{x}\dot{\mathrm{p}}_1 \end{bmatrix} \tag{2.178}$$

We observe that the determinant of matrix $M(\mathrm{q})$ is non-zero as long as $\begin{bmatrix} x \\ 0 \end{bmatrix} \neq \mathrm{p}$, otherwise the last row and last column of $M(\mathrm{q})$ are zero, and the matrix becomes rank-deficient. This situation corresponds, physically, to the hanging mass coinciding with the cart (i.e. the length of the link is zero). The inverse of matrix $M(\mathrm{q})$ is very complex, such that writing (2.178) explicitly is not recommended.

□

2.6 CONSISTENCY CONDITIONS

Before finishing this Section, we need to approach an important consequence of the model transformation detailed above in terms of model simulation. The original model that is describing the physical system reads as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = Q \tag{2.179a}$$

$$\mathrm{c}(\mathrm{q}) = 0 \tag{2.179b}$$

while the transformed model reads as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = Q \tag{2.180a}$$

$$\ddot{\mathrm{c}}(\mathrm{q}, \dot{\mathrm{q}}, \ddot{\mathrm{q}}) = 0 \tag{2.180b}$$

An important question to ask here is whether the transformed model (2.180) is equivalent to the original model (2.179). Since the trajectories arising from the original model (2.179) satisfy $\mathrm{c} = 0$ at all time, they also satisfies (2.180b) at all time. Hence, the trajectories of the original model are also trajectories of the transformed model (2.180). Unfortunately, the converse is not always true. Indeed, the trajectories of the transformed model (2.179) satisfy $\ddot{\mathrm{c}} = 0$ at all time, which does not entail that they satisfy $\mathrm{c} = 0$ at all time. Actually, if the trajectories satisfy $\ddot{\mathrm{c}} = 0$ at all time, then one can verify that the constraints c must obey the time evolution:

$$\mathrm{c}(\mathrm{q}(t)) = \mathrm{c}(\mathrm{q}(0)) + t \cdot \dot{\mathrm{c}}(\mathrm{q}(0), \dot{\mathrm{q}}(0)) \tag{2.181}$$

such that $\mathrm{c}(\mathrm{q}(t)) = 0$ does not necessarily hold. Fortunately, (2.181) delivers the conditions that are required in order for the trajectories of the transformed model (2.180) to be trajectories of the original model (2.179). Indeed, (2.181) readily tells us that $\mathrm{c}(\mathrm{q}) = 0$ holds at all time if:

$$\mathrm{C}(\mathrm{q}(0), \dot{\mathrm{q}}(0)) := \begin{bmatrix} \mathrm{c}(\mathrm{q}(0)) \\ \dot{\mathrm{c}}(\mathrm{q}(0), \dot{\mathrm{q}}(0)) \end{bmatrix} = 0 \tag{2.182a}$$

hence, the transformed model delivers meaningful trajectories **if** the initial conditions $\mathrm{q}(0), \dot{\mathrm{q}}(0)$ satisfy $\mathrm{C}(\mathrm{q}(0), \dot{\mathrm{q}}(0)) = 0$. Condition (2.182a) is referred to as the **consistency conditions**

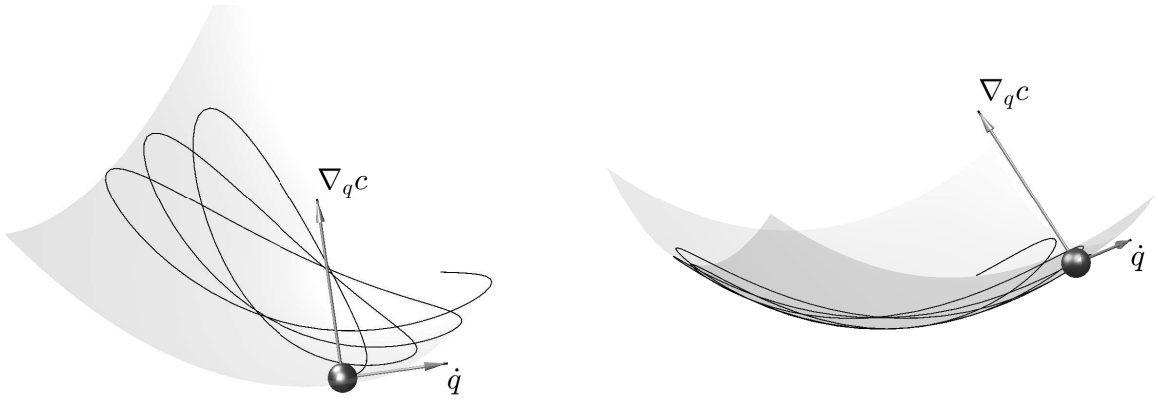


Figure 2.9: Consistency conditions for the bowl example. The condition $\nabla_q c^\top \dot{q} = 0$ simply requires that $\dot{q}(0)$ is tangent to the bowl.

associated to the model transformation. They are required in order to impose initial conditions in the model that are physically meaningful.

In order to build some intuition behind these observation using the bowl example above. The consistency conditions for the bowl read as:

$$C\left(q(0), \dot{q}(0)\right) := \begin{bmatrix} p_3 - \frac{1}{4}p_1^2 - \frac{1}{4}p_2^2 \\ \dot{p}_3 - \frac{1}{2}\dot{p}_1 p_1 - \frac{1}{2}\dot{p}_2 p_2 \end{bmatrix} \tag{2.183}$$

While these expressions do not carry much intuition, they have a very strong geometric interpretation. The first condition in C simply states that the initial condition $q(0)$ must satisfy the constraints $c\left(q\right)$, and is a quite natural requirement. The second condition states that:

$$\dot{c}\left(q(0), \dot{q}(0)\right) = \frac{\partial c}{\partial q}\dot{q} = \nabla_q c^\top \dot{q} = 0 \tag{2.184}$$

i.e. it requires that the scalar product between the gradient $\nabla_q c$ and the velocities $\dot{q}$ is zero. In other words, it requires the velocities $\dot{q}$ to be orthogonal to the gradient $\nabla_q c$. This requirement has a very intuitive interpretation in the bowl example, see Fig. 2.9. Indeed, the gradient $\nabla_q c$ is describing a normal to the surface described by the equation $c\left(q(0)\right) = 0$. In our bowl example, this surface is the bowl itself (more complex systems may not deliver such simple picture), and the condition $\nabla_q c\left(q(0)\right)^\top \dot{q}(0) = 0$ essentially requires that the initial velocities $\dot{q}(0)$ are tangent to the bowl surface. This requirement is physically needed in order for the ball to slide on the surface of the bowl.

2.7 Constraints drift

We have seen in the previous section that the initial conditions of the transformed model must satisfy some consistency conditions in order for the transformed model to deliver

meaningful trajectories. The consistency conditions can be derived from the observation that the constraints $\mathrm{c}(\mathrm{q})$ obey the time evolution (2.181), i.e.:

$$\mathrm{c}(\mathrm{q}(t)) = \mathrm{c}(\mathrm{q}(0)) + t \cdot \dot{\mathrm{c}}(\mathrm{q}(0), \dot{\mathrm{q}}(0)) \tag{2.185}$$

on the trajectories $\mathrm{q}(t)$ computed from the transformed model. However, this time evolution holds if and only if the condition $\ddot{\mathrm{c}} = 0$ is imposed. While this condition is imposed by the transformed model (2.166) (and all the variants we have looked at), when simulating the system, the condition is imperfectly imposed for numerical reasons. As a result, the time evolution (2.185) is "noisy", and it tends to accumulate the numerical errors over time. This effect is illustrated in Fig. 2.10, which is the outcome of simulating the transformed model (2.172) with consistent initial conditions, but a low integration accuracy. As a result, the condition $\ddot{\mathrm{c}} = 0$ is not enforced very accurately, and tends to create a drift in the constraints.

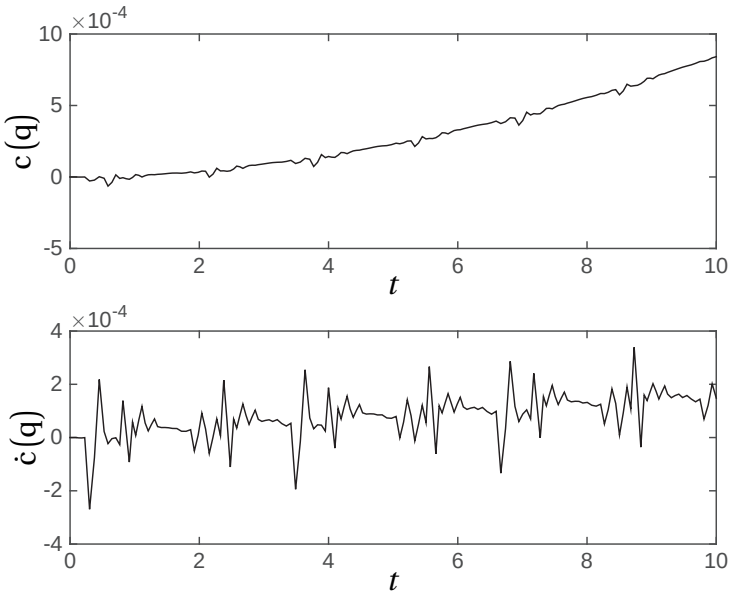


Figure 2.10: Illustration of the constraints drift for the bowl example.

This issue can be treated via **Baumgarte stabilization**, whereby the transformed model does not impose $\ddot{\mathrm{c}} = 0$, but rather dynamics on c that stabilizes them to zero. This can be e.g. done by imposing:

$$\ddot{\mathrm{c}} + 2\alpha\dot{\mathrm{c}} + \alpha^2\mathrm{c} = 0 \tag{2.186}$$

for some $\alpha > 0$ (such that the dynamics (2.186) are stable, with two real poles at $-\alpha$). The transformed model then reads as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathscr{L}}{\partial \dot{\mathrm{q}}} - \frac{\partial \mathscr{L}}{\partial \mathrm{q}} = Q \tag{2.187a}$$

$$\ddot{\mathrm{c}} + 2\alpha\dot{\mathrm{c}} + \alpha^2\mathrm{c} = 0 \tag{2.187b}$$

3 NEWTON METHOD

The Newton method is a central tool for simulation, optimization, and estimation. It is therefore no surprise that we will encounter it here. Generally speaking, the Newton method aims at solving a set of equations, that we can e.g. write as:

$$\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)=0 \tag{3.1}$$

in $\mathrm{x}\in\mathbb{R}^{n_{\mathrm{x}}}$ for a given $\mathrm{y}\in\mathbb{R}^{n_{\mathrm{x}}}$. Clearly, in order for the problem of finding x to be well-posed, one needs (3.1) to hold "as many equations as unknown variables", i.e. function $\boldsymbol{\varphi}$ is

$$\boldsymbol{\varphi}\ :\ \mathbb{R}^{n_{\mathrm{x}}}\times\mathbb{R}^{n_{\mathrm{y}}}\ \mapsto\ \mathbb{R}^{n_{\mathrm{x}}} \tag{3.2}$$

We will see later that this condition is not sufficient. When the system of equations described by (3.1) is nonlinear, finding a solution x can typically not be done explicitly (i.e. we typically cannot provide a set of symbolic expressions that describe x as a function of y). This does not mean that x is not perfectly well described as a function of y by (3.1). Formally, one say that x is an **implicit** function of y defined by (3.1).

The Newton method then allows one to compute x as a function of y numerically. In this chapter we will introduce the basics of the Newton method.

3.1 BASIC IDEA OF NEWTON METHOD

The Newton method replaces the problem of solving the nonlinear set of equations (3.1) by solving a succession of surrogate linear approximations. These linear approximations are built be successive linearizations of (3.1), i.e. instead of solving (3.1), we solve:

$$\boldsymbol{\varphi}\left(\mathrm{x}_{+},\mathrm{y}\right)\approx\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)+\frac{\partial\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)}{\partial\mathrm{x}}\left(\mathrm{x}_{+}-\mathrm{x}\right)=0 \tag{3.3}$$

for x_{+}, based on a given guess x. Note that (3.3) is a first-order Taylor approximation of $\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)$, i.e.

$$\boldsymbol{\varphi}\left(\mathrm{x}_{+},\mathrm{y}\right)=\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)+\frac{\partial\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)}{\partial\mathrm{x}}\left(\mathrm{x}_{+}-\mathrm{x}\right)+\mathscr{O}\left(\left\|\mathrm{x}_{+}-\mathrm{x}\right\|^{2}\right) \tag{3.4}$$

and is an affine form in x_{+}, i.e. x_{+} can be obtained from (3.3) explicitly from:

$$\mathrm{x}_{+}=\mathrm{x}-\frac{\partial\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)}{\partial\mathrm{x}}^{-1}\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right) \tag{3.5}$$

It is common to label

$$\Delta\mathrm{x}=-\frac{\partial\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right)}{\partial\mathrm{x}}^{-1}\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right) \tag{3.6}$$

as a (full) *Newton step* that corrects the guess x to the update x_+. See Fig 3.1 for an illustration. Using this construction, the Newton method is then *iterative*, in the sense that the (3.5) is repeated until the solution to $\boldsymbol{\varphi}(x,y) = 0$ is reached (or close enough). More specifically, the Newton method uses the algorithm:

Algorithm: Full-step Newton method

Input: Variable y, initial guess x, and tolerance tol

while $\|\boldsymbol{\varphi}(x,y)\|_\infty \geq$ tol **do**

Compute

$$\boldsymbol{\varphi}(x,y) \quad \text{and} \quad \frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x} \tag{3.7}$$

Compute the Newton step

$$\frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x}\Delta x + \boldsymbol{\varphi}(x,y) = 0 \tag{3.8}$$

Take the Newton step

$$x \leftarrow x + \Delta x \tag{3.9}$$

return x

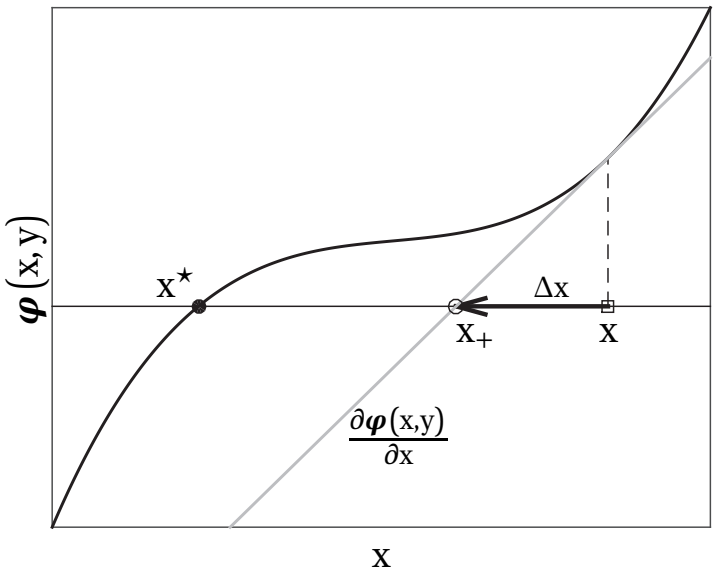


Figure 3.1: Illustration of the Newton principle on a scalar equation (i.e. $n_x = 1$ and y is not used here). The update x_+ is obtained by finding the root of the linear approximation (grey line) of $\boldsymbol{\varphi}(x,y)$ built at a candidate point x. The update x_+ is closer to the solution $x^\star$ of $\boldsymbol{\varphi}(x,y) = 0$ than the candidate x.

A few crucial remarks ought to be made here:

- If it converges, the Newton method converges to $x^\star$ a solution of $\boldsymbol{\varphi}(x,y) = 0$
- Each step of the Newton method requires evaluating the function $\boldsymbol{\varphi}(x,y)$ and its Jacobian $\frac{\partial \boldsymbol{\varphi}(x)}{\partial x}$, and solving the linear system (3.8).
- The Newton method requires that the square Jacobian matrix $\frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x}$ is full rank (i.e. invertible), in order for the linear system (3.8) to be well-posed.
- If function $\boldsymbol{\varphi}(x,y)$ is linear in x (and well posed), then the Newton method finds the solution $x^\star$ in one step. It is then fully equivalent to solving the linear system.

3.2 Convergence of the Newton method

A crucial question for the Newton method is whether the Newton algorithm converges or not. If it converges, then it has to converge to a solution[3] $x^\star$ of $\boldsymbol{\varphi}(x,y) = 0$. Indeed, as long as $\boldsymbol{\varphi}(x,y) \neq 0$, the Newton algorithm performs corrections $\Delta x \neq 0$, and only stops correcting the initial guess when $\boldsymbol{\varphi}(x,y) \approx 0$ (depending on the tolerance). Unfortunately, the convergence of the Newton method is not trivial. We will discuss some basic results in this section.

3.2.1 Convergence rate

Let us first assume that the Newton iteration presented above converges, and let us investigate its convergence rate, i.e how quickly $\|x - x^\star\|$ decreases. This question is addressed as follows:

Theorem 3. *if the full step Newton iteration converges, then it converges at a quadraticrate, i.e.*

$$\left\|x_+ - x^\star\right\| \leq C \cdot \left\|x - x^\star\right\|^2 \tag{3.10}$$

for some constant $C > 0$.

Before delivering a formal proof of the quadratic convergence rate, one ought to understand that this result if fairly intuitive. Indeed, one can observe that at every iteration, the error that the Newton step is making is of order 2. Hence, every iteration “removes" the first-order inaccuracy between the solution candidate and the solution to $\boldsymbol{\varphi}(x,y) = 0$. It follows that at every step, the Newton method retains an error of order $\left\|x - x^\star\right\|^2$. This observation alone give an intuition of why (3.10) ought to be true. We provide next a proof of Theorem 3. This proof is not part of the examination though.

[3] Note that there may be more than one solution! Hence the Newton method could converge to different points, depending on where it is started.

Proof. for the sake of simplicity, let us dismiss the fixed argument y, and let us write the Newton step as

$$\Delta \mathrm{x} = -M^{-1}\boldsymbol{\varphi}(\mathrm{x}) \tag{3.11}$$

We first observe that since $\boldsymbol{\varphi}(\mathrm{x}^\star) = 0$, the equality

$$\mathrm{x}_+ - \mathrm{x}^\star = \mathrm{x} - \mathrm{x}^\star - M^{-1}\left(\boldsymbol{\varphi}(\mathrm{x}) - \boldsymbol{\varphi}(\mathrm{x}^\star)\right) \tag{3.12}$$

holds. We then use classical results from analysis to observe that:

$$\boldsymbol{\varphi}(\mathrm{x}) - \boldsymbol{\varphi}(\mathrm{x}^\star) = \int_0^1 \frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}}\left(\mathrm{x} - \mathrm{x}^\star\right)\mathrm{d}\tau \tag{3.13}$$

Using (3.12) and (3.13), we can write:

$$\mathrm{x}_+ - \mathrm{x}^\star = \mathrm{x} - \mathrm{x}^\star - M^{-1}\left(\int_0^1 \frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}}\left(\mathrm{x} - \mathrm{x}^\star\right)\mathrm{d}\tau\right) \tag{3.14}$$

Or equivalently

$$\mathrm{x}_+ - \mathrm{x}^\star = M^{-1}\left(M - \int_0^1 \frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}}\mathrm{d}\tau\right)\left(\mathrm{x} - \mathrm{x}^\star\right) \tag{3.15}$$

We can modify (3.15) as:

$$\mathrm{x}_+ - \mathrm{x}^\star = M^{-1}\left(M - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}} - \int_0^1 \left[\frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right]\mathrm{d}\tau\right)\left(\mathrm{x} - \mathrm{x}^\star\right) \tag{3.16}$$

We can then bound (using triangular inequalities):

$$\begin{aligned}\left\|\mathrm{x}_+ - \mathrm{x}^\star\right\| \leq{}& \left\|M^{-1}\left(M - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right)\right\|\left\|\mathrm{x} - \mathrm{x}^\star\right\| \\ &+ \left\|M^{-1}\int_0^1 \left[\frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right]\mathrm{d}\tau\right\|\left\|\mathrm{x} - \mathrm{x}^\star\right\|\end{aligned} \tag{3.17}$$

Since $M = \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}$, the first term is zero. Moreover, the term under the integral sign in the second term can be bounded (on any closed set around $\mathrm{x}^\star$) by

$$\left\|M^{-1}\left(\frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}\left(\mathrm{x}^\star\right)}{\partial \mathrm{x}}\right)\right\| \leq c\left\|\mathrm{x} - \mathrm{x}^\star\right\| \tag{3.18}$$

for some constant $c > 0$ for $\boldsymbol{\varphi}$ smooth. It follows that

$$\begin{aligned}&\left\|M^{-1}\int_0^1 \left[\frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right]\mathrm{d}\tau\right\| \leq \int_0^1 \left\|M^{-1}\left[\frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right]\right\|\mathrm{d}\tau \\ &\leq \int_0^1 c\left\|\mathrm{x} - \mathrm{x}^\star\right\|\tau\,\mathrm{d}\tau = \frac{c}{2}\left\|\mathrm{x} - \mathrm{x}^\star\right\|\end{aligned} \tag{3.19}$$

such that

$$\left\|\mathrm{x}_+ - \mathrm{x}^\star\right\| \leq \frac{c}{2}\left\|\mathrm{x} - \mathrm{x}^\star\right\|^2 \tag{3.20}$$

□

Let us illustrate this convergence rate for the example provided in Fig. 3.3. A few remarks can be useful here:

- A quadratic convergence rate is a very strong convergence. It means that at every iteration the number of accurate digits is doubled, i.e. over the iterations, the error $\left\|\mathrm{x}-\mathrm{x}^\star\right\|$ follows a decay of e.g. the form

$$\begin{aligned}\left\|\mathrm{x}-\mathrm{x}^\star\right\| &= 10^{-1}\\ \left\|\mathrm{x}-\mathrm{x}^\star\right\| &= 10^{-2}\\ \left\|\mathrm{x}-\mathrm{x}^\star\right\| &= 10^{-4}\\ \left\|\mathrm{x}-\mathrm{x}^\star\right\| &= 10^{-8}\\ \left\|\mathrm{x}-\mathrm{x}^\star\right\| &= 10^{-16}\end{aligned}$$

  Hence a few iterations are typically sufficient to reach machine precision ($\epsilon = 1e^{-16}$)

- While (3.18) appears very technical, it can be understood as a "measure" of how nonlinear the function $\boldsymbol{\varphi}(\mathrm{x})$ is. Indeed, if $\boldsymbol{\varphi}(\mathrm{x})$ is linear in x, i.e.

$$\boldsymbol{\varphi}(\mathrm{x}) = A\mathrm{x}+\mathrm{b} \tag{3.21}$$

  (we dismiss the argument y here) for a matrix A and a vector b, then

$$\frac{\partial\boldsymbol{\varphi}(\mathrm{x})}{\partial\mathrm{x}} = A, \qquad \forall\,\mathrm{x} \tag{3.22}$$

  and the left-hand side of (3.18) is zero, such that $c = 0$. Conversely, if $\boldsymbol{\varphi}(\mathrm{x})$ is strongly nonlinear, then its Jacobian $\frac{\partial\boldsymbol{\varphi}(\mathrm{x})}{\partial\mathrm{x}}$ varies a lot and the difference

$$\frac{\partial\boldsymbol{\varphi}\left(\mathrm{x}+\tau\left(\mathrm{x}^\star-\mathrm{x}\right)\right)}{\partial\mathrm{x}} - \frac{\partial\boldsymbol{\varphi}(\mathrm{x})}{\partial\mathrm{x}} \tag{3.23}$$

  can be very large, yielding a large constant c.

- The proof above also informs us on the condition required for the Newton iteration to converge. Indeed, in order for the bound (3.10) to guarantee the decay of $\left\|\mathrm{x}-\mathrm{x}^\star\right\|$, one ought to require that the initial guess x provided to the algorithm is such that:

$$\left\|\mathrm{x}-\mathrm{x}^\star\right\| \leq 2c^{-1} \tag{3.24}$$

  It is interesting here to understand (3.24) properly. Indeed, it tells us that

  1. in order for the full-step Newton iteration to converge, it ought to be provided with an initial guess that is **close enough** to a solution $\mathrm{x}^\star$. This observation is illustrated in Fig. 3.2.
  2. "how far" the initial guess provided to the full-step Newton iteration can be from a solution $\mathrm{x}^\star$ is larger if c is small, i.e. if $\boldsymbol{\varphi}(\mathrm{x})$ is close to linear. Conversely, if $\boldsymbol{\varphi}(\mathrm{x})$ is very nonlinear, then c is large and the initial guess provided to the full-step Newton iteration must be close to a solution $\mathrm{x}^\star$ in order to guarantee the convergence of the iteration

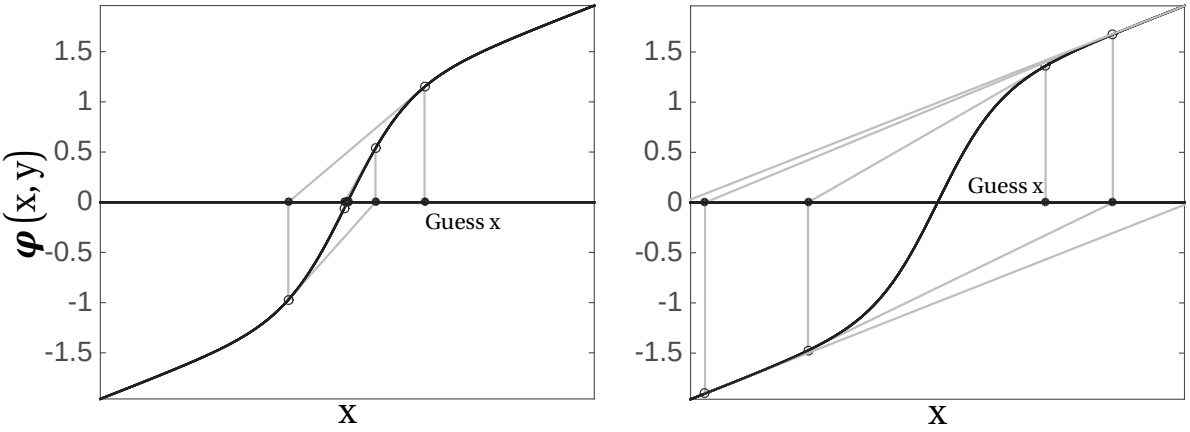


Figure 3.2: Newton iteration on a nonlinear, scalar function $\boldsymbol{\varphi}(x,y)$ (five steps are displayed here). On the left graph the iteration converges to the solution $x^\star$ of $\boldsymbol{\varphi}(x^\star,y) = 0$, while on the right graph the iteration diverges. Convergence is obtained for the initial guess provided to the full-step Newton iteration being close enough to a solution $x^\star$, while an initial guess further away (possibly) results from an unstable iteration.

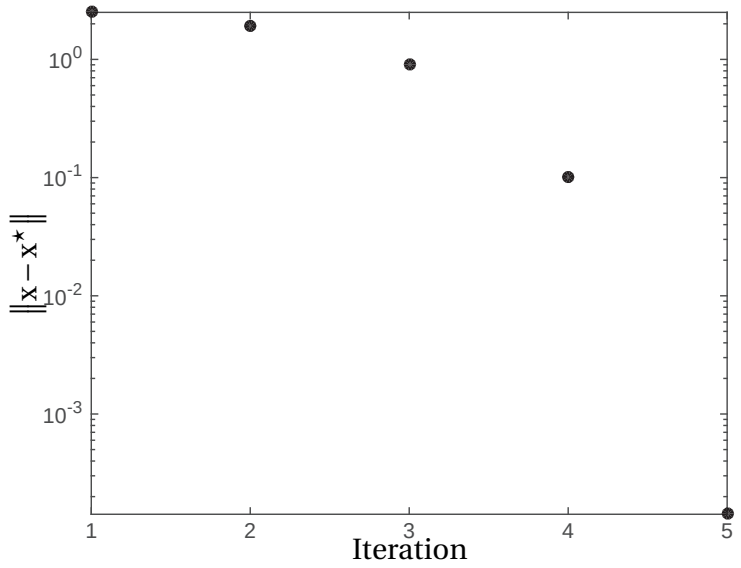


Figure 3.3: Converging full-step Newton iteration. The quadratic convergence rate yields a decrease of the solution error $\|x - x^\star\|$ that “collapses" downward in a semi-log plot.

3.2.2 REDUCED NEWTON STEPS

The full-step Newton algorithm as presented above can diverge if the initial guess provided to the Newton iteration is "too far" from a solution $x^\star$ of $\boldsymbol{\varphi}(x,y) = 0$. Fortunately, this shortcoming can be addressed using reduced Newton steps. In this section we present the justification and deployment of the approach,

The motivation behind reduced Newton steps can be stated as follows.

Theorem 4. *if it exists, the Newton step* Δx *solution of the linear system*

$$\frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x}\Delta x + \boldsymbol{\varphi}(x,y) = 0 \tag{3.25}$$

is a descent direction for function $\boldsymbol{\varphi}(x,y)$*, i.e.*

$$\left\|\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right\| < \left\|\boldsymbol{\varphi}(x,y)\right\| \tag{3.26}$$

for some $t \in]0,1]$.

Note that the result above is valid for any choice of norm $\|.\|$ (this follows directly from the equivalence between norms), but it is easiest to prove using the 2-norm.

Proof. inequality (3.26) is equivalent to[4]

$$\left\|\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right\|^2 - \left\|\boldsymbol{\varphi}(x,y)\right\|^2 < 0 \tag{3.27}$$

The first-order Taylor expension of (3.27) yields:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\|\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right\|^2\Big|_{t=0}\cdot t + \mathcal{O}\left(t^2\right) < 0 \tag{3.28}$$

such that (3.27) holds for $t > 0$ small enough if

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\|\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right\|^2\Big|_{t=0} < 0 \tag{3.29}$$

We observe that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left\|\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right\|^2\Big|_{t=0} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\varphi}(x + t\cdot\Delta x, y)^\top \boldsymbol{\varphi}(x + t\cdot\Delta x, y)\right)\Big|_{t=0} \\
&= \boldsymbol{\varphi}(x,y)^\top \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\varphi}(x + t\cdot\Delta x, y)\Big|_{t=0} && (3.30)\\
&= \boldsymbol{\varphi}(x,y)^\top \frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x}\Delta x && (3.31)\\
&= -\boldsymbol{\varphi}(x,y)^\top \boldsymbol{\varphi}(x,y) < 0 && (3.32)
\end{aligned}$$

if $\|\boldsymbol{\varphi}(x,y)\| > 0$. □

[4] squaring the 2-norm is helpful here, as it makes it smooth

An important consequence of this result is that while the Newton method presented above may diverge, a careful selection of *reduced Newton steps*, i.e. modifications of x in the direction of the Newton step Δx but possibly scaled down must necessarily converge, as long as the Newton steps Δx exist. This motivates the modification of the full-step Newton algorithm into:

Algorithm: Newton method with reduced steps

Input: Variable y, initial guess x, and tolerance tol

while $\|\boldsymbol{\varphi}(x,y)\|_\infty \geq$ tol **do**

Compute

$$\boldsymbol{\varphi}(x,y) \quad \text{and} \quad \frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x} \tag{3.33}$$

Compute the Newton step

$$\frac{\partial \boldsymbol{\varphi}(x,y)}{\partial x}\Delta x + \boldsymbol{\varphi}(x,y) = 0 \tag{3.34}$$

Select step size $t \in]0,1]$

Take Newton step

$$x \leftarrow x + t\Delta x \tag{3.35}$$

return $x \approx x^\star$

Finding the adequate step size $t \in]0,1]$ is typically performed via a *line-search* strategy, based on testing first a full step ($t = 1$), and then reducing it if the condition (3.26) is not met. Clever methods exist in order to decide whether a step-size t is "good enough", or too short or too long. Other methods also exist in order to adjust the Newton step Δx and guarantee the convergence (see e.g. *trust-region techniques*), but they are more difficult to describe and implement.

A Newton iteration equipped with the possibility of taking reduced steps is guaranteed to converge to a solution $x^\star$ of $\boldsymbol{\varphi}(x,y) = 0$ as long as the Newton steps Δx exist throughout the iterations. It shall be underlined here that when reduced steps (i.e. using $t < 1$) are necessary, then the quadratic convergence rate detailed in Theorem 3 is lost. The Newton iteration with reduced steps has the following behavior:

- when x is "far" from a solution $x^\star$, then reduced steps ($t < 1$) are necessary, and the algorithm converges "slowly". The resulting convergence rate can be very poor, even though it is often close to linear in practice
- after a certain amount of iteration, x becomes close enough to $x^\star$ and full steps ($t = 1$) are acceptable. The convergence then becomes quadratic. Once x is close to $x^\star$, full

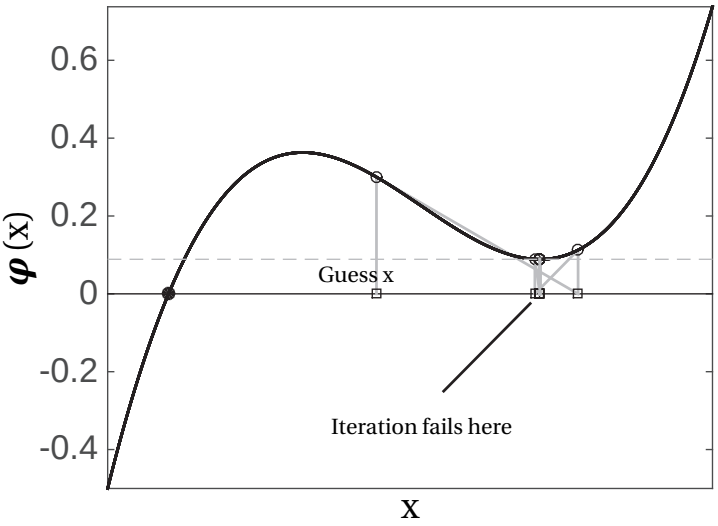


Figure 3.4: Newton iteration with reduced steps on a nonlinear, scalar function $\boldsymbol{\varphi}(\mathrm{x})$ (five steps are displayed here). Here the iteration does not diverge, but is fails at a point where $\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}} = 0$. At this point, the linear system (3.8) does not have a well-defined solution and the Newton step $\Delta \mathrm{x}$ ceases to exist.

steps can be taken all the way to $\mathrm{x}^\star$ and the convergence is very fast.

The Newton algorithm with reduced steps converges globally to a solution $\mathrm{x}^\star$ unless the linear system (3.34) becomes ill-posed. In order to develop some intuition on this potential (and frequent) issue, let us consider the example illustrated in Fig. 3.4. A simple inspection reveals that the Newton iteration may converge to a point at which $\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}} = 0$, where the Newton step ceases to exist. In such a case, Theorem 4 ceases to apply, and the Newton iteration fails. One can also readily see that if the Newton iteration was started closer to the solution $\mathrm{x}^\star$ (black dot in the graph), then it would converge to $\mathrm{x}^\star$.

3.3 Implicit Function Theorem

We have presented the Newton method as a numerical approach to solve the equations

$$\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right) = 0 \tag{3.36}$$

in terms of x. In many cases, the equations are dependent on some other argument, labelled y here, which in practice can gather e.g. parameters or data appearing in the equations. If one changes the y, it is natural to expect that the solution x to $\boldsymbol{\varphi}\left(\mathrm{x},\mathrm{y}\right) = 0$ is affected. Formally, we ought to think of x as being implicitly a function of y, i.e. we should consider x as a function $\mathrm{x}\left(\mathrm{y}\right)$ that we usually cannot write explicitly, but which is implicitly defined by

$$\boldsymbol{\varphi}\left(\mathrm{x}\left(\mathrm{y}\right),\mathrm{y}\right) = 0 \tag{3.37}$$

Some questions naturally arise then, such as e.g. is $\mathrm{x}\left(\mathrm{y}\right)$ well-defined, is it differentiable with respect to y and what is its derivative. These questions are object of the Implicit Func-

tion Theorem (IFT), which is one of the cornerstones in the field of numerical analysis.

Theorem 5. *(IFT, simplified version): let function* $\boldsymbol{\varphi}(\mathrm{x},\mathrm{y})$ *be smooth, and consider a point* $\bar{\mathrm{x}},\bar{\mathrm{y}}$ *such that* $\boldsymbol{\varphi}(\bar{\mathrm{x}},\bar{\mathrm{y}}) = 0$. *Suppose that the Jacobian*

$$\left.\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\,\mathrm{y}=\bar{\mathrm{y}}} \tag{3.38}$$

is full rank. Then there exists an open set Y *around the point* $\bar{\mathrm{y}}$ *in which there exist a unique, smooth function* $\mathrm{x}(\mathrm{y})$ *satisfying:*

$$\boldsymbol{\varphi}(\mathrm{x}(\mathrm{y}),\mathrm{y}) = 0, \qquad \forall\, \mathrm{y} \in Y \tag{3.39}$$

Moreover, the Jacobian of function $\mathrm{x}(\mathrm{y})$ *is given by*

$$\frac{\partial \mathrm{x}(\mathrm{y})}{\partial \mathrm{y}} = -\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}}^{-1} \left.\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{y}}\right|_{\mathrm{x}=\mathrm{x}(\mathrm{y}),\mathrm{y}} \tag{3.40}$$

The proof of the IFT is fairly involved and we will not do it here. However, equality (3.40) is trivial to prove.

Proof. (of (3.40)) Because of (3.39), the equality:

$$\frac{\mathrm{d}}{\mathrm{dy}}\boldsymbol{\varphi}(\mathrm{x}(\mathrm{y}),\mathrm{y}) = 0, \quad \text{holds} \quad \forall\, \mathrm{y} \in Y \tag{3.41}$$

Using the chain rule, we observe that:

$$\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{x}}\frac{\partial \mathrm{x}(\mathrm{y})}{\partial \mathrm{y}} + \left.\frac{\partial \boldsymbol{\varphi}(\mathrm{x},\mathrm{y})}{\partial \mathrm{y}}\right|_{\mathrm{x}=\mathrm{x}(\mathrm{y}),\mathrm{y}} = 0 \tag{3.42}$$

which yields (3.40). □

The importance of the IFT in numerical analysis can hardly be understated, but let us briefly review what it provides us:

- even though function $\mathrm{x}(\mathrm{y})$ cannot be written explicitly and can only be evaluated numerically (using a Newton iteration), its derivative is trivial to compute, using (3.40) at the point x found by the Newton iteration for given point y.
- the IFT guarantees that if $\boldsymbol{\varphi}(\mathrm{x},\mathrm{y}) = 0$ is well posed (i.e. the Jacobian (3.38) is full rank) at a point y, then it has a neighborhood where it is also well posed, i.e. we can “move" y (locally), and x “moves" accordingly, and also locally.

- One ought to observe that the assumption of the Jacobian $\frac{\partial \boldsymbol{\varphi}(\tilde{\mathrm{x}},\tilde{\mathrm{y}})}{\partial \mathrm{x}}$ being full rank is also required for the Newton iteration to be well-behaved (i.e. for the linear system (3.34) to be well-posed). In other words, if the Newton iteration is well-behaved, then the IFT holds. Conversely, the assumption of the IFT must hold in order for a system of equation $\boldsymbol{\varphi}(\mathrm{x},\mathrm{y}) = 0$ to be solvable for x.

We will use the IFT in several places in the following.

3.4 Jaccobian Approximation

In some applications of the Newton iteration, the evaluation of the Jacobian $\frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}$ is very expensive. It can then be useful to consider using an approximation that is less expensive to evaluate. Let us label this approximation:

$$M \approx \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}} \tag{3.43}$$

The resulting *Newton-type* step reads as:

$$\Delta \mathrm{x} = -M^{-1}\boldsymbol{\varphi}(\mathrm{x}) \tag{3.44}$$

The use of an approximate Jacobian has an impact on the convergence of the Newton method. Let us briefly review how it is changed. In order to do that, we will revisit Theorem 3 and 4 with the Jacobian approximation.

Theorem 6. *the convergence of the full-step Newton method with an approximate Jacobian follows:*

$$\left\|\mathrm{x}_+ - \mathrm{x}^\star\right\| \leq \left(\kappa + \frac{c}{2}\left\|\mathrm{x} - \mathrm{x}^\star\right\|\right)\left\|\mathrm{x} - \mathrm{x}^\star\right\| \tag{3.45}$$

for some constants $c, \kappa > 0$.

Proof. We can re-use (3.17), repeated here for simplicity:

$$\begin{aligned}\left\|\mathrm{x}_+ - \mathrm{x}^\star\right\| \leq & \left\|M^{-1}\left(M - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right)\right\|\left\|\mathrm{x} - \mathrm{x}^\star\right\| \\ & + \left\|M^{-1}\int_0^1 \left[\frac{\partial \boldsymbol{\varphi}\left(\mathrm{x} + \tau\left(\mathrm{x}^\star - \mathrm{x}\right)\right)}{\partial \mathrm{x}} - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right] \mathrm{d}\tau\right\|\left\|\mathrm{x} - \mathrm{x}^\star\right\|\end{aligned} \tag{3.46}$$

and we observe that if $M \neq \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}$ the first term in the right-hand side of the inequality does not disappear. We can then bound (on any closed set around $\mathrm{x}^\star$):

$$\left\|M^{-1}\left(M - \frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right)\right\| = \left\|I - M^{-1}\frac{\partial \boldsymbol{\varphi}(\mathrm{x})}{\partial \mathrm{x}}\right\| \leq \kappa, \quad \forall\, \mathrm{x} \tag{3.47}$$

while the second term in the right-hand side of (3.46) can be bounded as in Theorem 3. Bound (3.45) follows. □

It is useful here to make some remarks concerning Theorem 6.

- The bound (3.45) predicts a linear convergence rate, due to the term $\kappa \left\| x - x^\star \right\|$ in (3.45). A linear convergence rate is significantly slower than a quadratic convergence rate, as it does not go through a “collapse" to very small values. See Fig. 3.5 for an illustration.

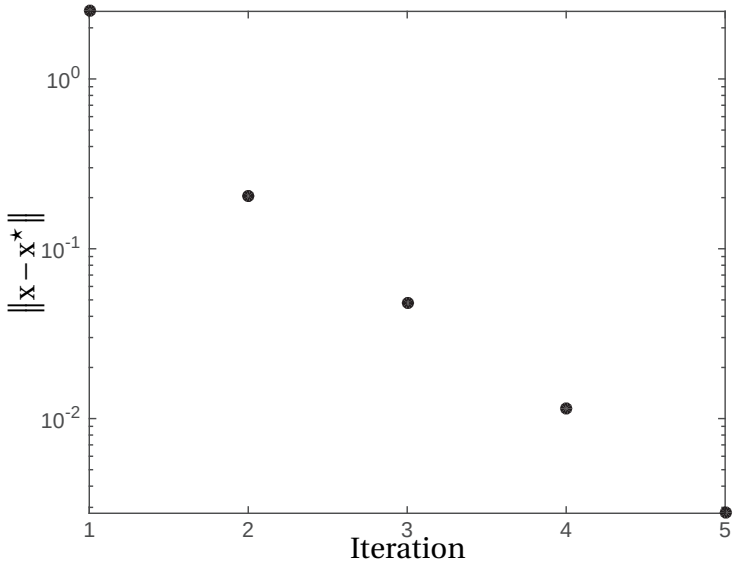


Figure 3.5: Stable full-step Newton iteration with approximate Jacobian. The linear convergence rate yields a decrease of the solution error $\left\| x - x^\star \right\|$ that follows a “line" downward in a semi-log plot. This is to be contrasted with Fig. 3.3.

- Bound (3.47) informs us on the impact of the error between M and the true Jacobian $\frac{\partial \boldsymbol{\varphi}(x)}{\partial x}$. Indeed, for $M = \frac{\partial \boldsymbol{\varphi}(x)}{\partial x}$, $\kappa = 0$ can be selected, and one recovers a quadratic convergence (see Th. 3). If M and $M = \frac{\partial \boldsymbol{\varphi}(x)}{\partial x}$ are “very different", then κ has to be large. One can observe from (3.45) that $\kappa < 1$ must hold in order for the convergence of the full-step quasi-Newton iteration to be guaranteed (even for x being arbitrarily close to $x^\star$).

3.5 Newton Methods for Unconstrained Optimization

An important application of the Newton method is for solving minimization problems. In this chapter, we will consider the case of unconstrained optimization, i.e. problems of the form:

$$\min_x \ \Phi(x, y) \tag{3.48}$$

for a scalar function $\Phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \mapsto \mathbb{R}$. If function Φ is smooth with respect to x, the solution $x^\star$ to (3.48) is given by solving:

$$\nabla_x \Phi(x, y) = 0 \tag{3.49}$$

for x. The Newton methods detailed above then apply directly to solving (3.49). More specifically, a Newton iteration for solving (3.49) reads as:

Algorithm: Newton method for optimization

Input: Variable y, initial guess x, and tolerance tol

while $\|\nabla_x \Phi(x,y)\|_\infty \geq \text{tol}$ **do**

Compute

$$\nabla_x \Phi(x,y) \quad \text{and} \quad \nabla_x^2 \Phi(x,y) \tag{3.50}$$

Compute the Newton step

$$\nabla_x^2 \Phi(x,y)\,\Delta x + \nabla_x \Phi(x,y) = 0 \tag{3.51}$$

Select step size $t \in]0,1]$

Take Newton step

$$x \leftarrow x + t\Delta x \tag{3.52}$$

return $x \approx x^\star$

Let us review the results presented in the previous sections in the specific context of optimization.

- The convergence of Newton for Optimization is quadratic in a neighborhood of the solution $x^\star$ and "slow" otherwise. If the Hessian $\nabla_x^2 \Phi(x,y)$ is approximated by M, then the method has a linear convergence rate (i.e. Theorem 6 applies).
- The iteration can fail at a point where (3.51) is ill-defined, i.e. at a point where the Hessian of Φ is rank-deficient.

In the context, of optimization problems, an additional remark can be made concerning the choice of Hessian approximation, summarized hereafter.

Theorem 7. *if $M > 0$ (positive-definite), then the quasi-Newton step*

$$\Delta x = -M^{-1} \nabla_x \Phi(x,y) \tag{3.53}$$

is a descent direction for the cost function $\Phi(x,y)$, i.e.

$$\Phi(x + t\Delta x, y) < \Phi(x,y) \tag{3.54}$$

for some $t > 0$.

Proof. for the sake of simplicity, let us write the proof dismissing the argument y. We observe that

$$\Phi(x + t\Delta x) - \Phi(x) = \frac{\partial \Phi(x)}{\partial x} \Delta x t + \mathcal{O}(t^2) \tag{3.55}$$

$$= -t \nabla_x \Phi(x,y)^\top M^{-1} \nabla_x \Phi(x,y) + \mathcal{O}(t^2) < 0 \tag{3.56}$$

if $M > 0$ and for $t > 0$ sufficiently small. □

Hence it is sufficient to choose a positive-definite Hessian approximation to guarantee that the Newton iteration converges. However, the results of Theorem 6 still apply, hence a poor Hessian approximation results in a strong degradation of the convergence rate.

3.5.1 Gauss-Newton Hessian approximation

Let us consider optimization problems of the form

$$\min_{\mathrm{x}} \ \frac{1}{2} \left\| \boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y} \right\|^2 \tag{3.57}$$

for some function $\boldsymbol{\phi} : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_{\boldsymbol{\phi}}}$.

Optimization problems of the form (3.57) are very common in system identification, see Section 4. The gradient and Hessian of the cost function $\Phi\left(\mathrm{x},\mathrm{y}\right) = \frac{1}{2}\left\| \boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y} \right\|^2$, then read as:

$$\nabla_\mathrm{x} \Phi\left(\mathrm{x},\mathrm{y}\right) = \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right) \left(\boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y}\right) \tag{3.58}$$

$$\nabla^2_\mathrm{x} \Phi\left(\mathrm{x},\mathrm{y}\right) = \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right) \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right)^\top + \left[\nabla_{\mathrm{x}_i,\mathrm{x}_j} \boldsymbol{\phi}\left(\mathrm{x}\right) \left(\boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y}\right) \right]_{i,j} \tag{3.59}$$

where $[.]_{i,j}$ denotes a matrix where the elements i, j are detailed between the brackets.

For many fitting problems, the evaluation of the second term in the Hessian $\nabla^2_\mathrm{x} \Phi\left(\mathrm{x},\mathrm{y}\right)$ is expensive, such that the Gauss-Newton Hessian approximation:

$$\nabla^2_\mathrm{x} \Phi\left(\mathrm{x},\mathrm{y}\right) \approx M_\mathrm{GN} = \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right) \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right)^\top \tag{3.60}$$

is often preferred. The Gauss-Newton Hessian approximation (3.60) is valid if:

- the function $\boldsymbol{\phi}\left(\mathrm{x}\right)$ is not very nonlinear, such that its second-order derivatives $\nabla_{\mathrm{x}_i,\mathrm{x}_j} \boldsymbol{\phi}\left(\mathrm{x}\right)$ are small, or
- the optimal solution to the fitting problem (3.57) yields $\boldsymbol{\phi}\left(\mathrm{x}^\star\right) \approx \mathrm{y}$, such that $\boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y}$ is small.

Both cases justify the dismissal of the second term $\left[\nabla_{\mathrm{x}_i,\mathrm{x}_j} \boldsymbol{\phi}\left(\mathrm{x}\right) \left(\boldsymbol{\phi}\left(\mathrm{x}\right) - \mathrm{y}\right) \right]_{i,j}$ in $\nabla^2_\mathrm{x} \Phi\left(\mathrm{x},\mathrm{y}\right)$. It is useful to observe that the Gauss-Newton Hessian approximation is by construction semi-positive definite, as $M_\mathrm{GN} \geq 0$. In order to ensure its strict positiveness, it is common to add some *regularization*, i.e. to use a Hessian approximation of the form:

$$M = \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right) \nabla_\mathrm{x} \boldsymbol{\phi}\left(\mathrm{x}\right)^\top + \alpha I \tag{3.61}$$

for some $\alpha > 0$ reasonably small.

3.5.2 CONVEX OPTIMIZATION

Before concluding this section, we ought to briefly talk about convexity in optimization problems. For a (sufficiently) smooth cost function $\Phi\left(\mathrm{x},\mathrm{y}\right)$, the solution $\mathrm{x}^\star$ must satisfy the condition

$$\nabla_{\mathrm{x}}\Phi\left(\mathrm{x}^\star,\mathrm{y}\right)=0 \tag{3.62}$$

However, a point satisfying condition (3.62) is not necessarily the solution to the underlying optimization problem (3.48). This problem is illustrated in Fig. 3.6.

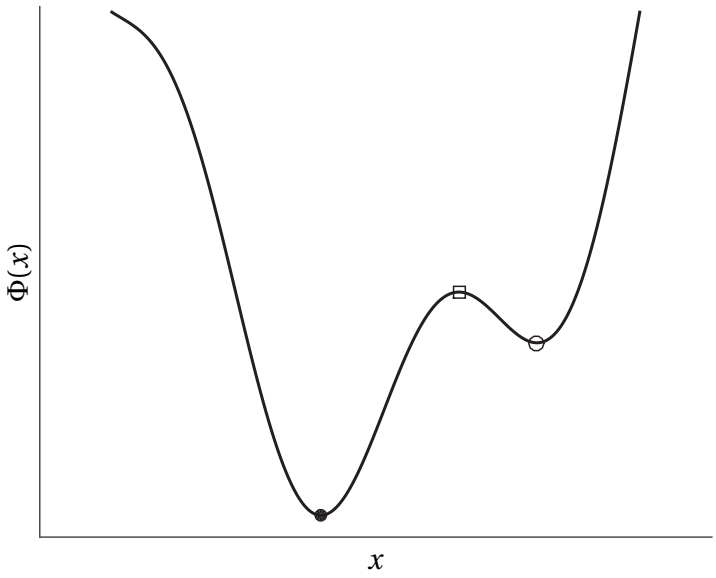


Figure 3.6: Illustration of the difficulty of finding the minimum of a function $\Phi(x)$ via solving (3.62). Several points satisfy (3.62) here. The black dot is the true minimum, but (3.62) also yields a so-called local minimum (circle), which is lower than its neighbours but not lower than the black dot. Condition (3.62) also returns (possibly local) maxima (square).

Mathematically speaking we would say that condition (3.62) is necessary but not sufficient to deliver the solution to (3.48). An exception to this lack of equivalence between condition (3.62) and solving (3.48) is when the cost function $\Phi\left(\mathrm{x},\mathrm{y}\right)$ is **convex** in x, i.e. if

$$\nabla_{\mathrm{x}}^2\Phi\left(\mathrm{x},\mathrm{y}\right)>0, \qquad \forall\ \mathrm{x} \tag{3.63}$$

In this case, the solution to (3.62) is unique and delivers the solution to (3.48). Moreover, the Newton iteration with reduced steps is guaranteed to converge to the solution of (3.48).

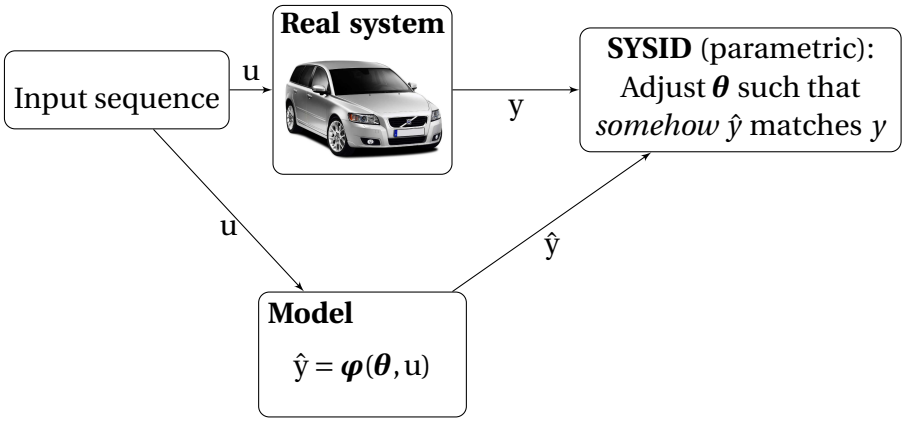
3.6 SUMMARY

The results concerning the Newton method are a bit convoluted. We provide hereafter a brief summary condensing the main points:

- Exact reduced Newton steps Δx improves $\boldsymbol{\varphi}$ for sufficiently small step sizes $t \in]0, 1]$
- Inexact reduced Newton steps Δx improve $\boldsymbol{\varphi}$ for a sufficiently small step size $t \in]0, 1]$ if M is sufficiently close to $\frac{\partial \boldsymbol{\varphi}}{\partial x}$. In the context of optimization, $M > 0$ and sufficiently small steps $t \in]0, 1]$ reduce the cost function Φ.
- Exact full ($t = 1$) Newton steps converge quadratically if close enough to the solution
- Inexact full ($t = 1$) Newton steps converge linearly if close enough to the solution and if the Jacobian approximation is "sufficiently good"
- The Newton iteration fails if $\frac{\partial \boldsymbol{\varphi}}{\partial x}$ becomes singular
- Newton methods with reduced steps converge in two phases: damped (slow) phase when reduced steps ($t < 1$) are needed, quadratic/ linear when full steps are possible.

4 System Identification (SYSID)

In this Chapter, we will consider the problem of estimating the parameters of a model of a system based on data collected on the system. The principle is illustrated below. An input sequence u is applied to the real system, generating an output sequence y. A model (that can take various forms) holds a set of parameters $\boldsymbol{\theta}$, and generates an output sequence $\hat{y}$ from the input sequence u. The goal of SYSID is to adjust the parameters $\boldsymbol{\theta}$ such that the model output $\hat{y}$ matches in some sense the output of the real system y.



Note that we will work in discrete time for SYSID. Indeed, since the measurements are always collected in a discrete fashion on the real system, it makes sense to build a theory where the model output is also discrete.

The system identification procedure we will consider here will then hold the following objects:

- A mathematical model of the system, aimed at explaining an “input-output" relationship:

$$\hat{y} = \boldsymbol{\varphi}(\boldsymbol{\theta}, u) \tag{4.1}$$

  where u is our input and $\hat{y}$ our model output, and $\boldsymbol{\theta}$ a set of model parameters.

- A set of data (or measurements), y physically taken from the system itself.

- A noise realization e explaining the mismatch between the model and the measurements, which is often (but not always!) considered as additive, i.e.:

$$y = \hat{y} + e = \boldsymbol{\varphi}(\boldsymbol{\theta}, u) + e \tag{4.2}$$

The question that will be at the heart of this chapter is: for a given model $\boldsymbol{\varphi}$, a given input u and a given data set y, what is the best model parameter vector $\boldsymbol{\theta}$?

A classic approach to answer that question is to use a least-squares minimization, i.e. to compute the parameters by solving:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) \right\|^2 \tag{4.3}$$

Note that the least-squares approach tipically tries to minimize the noise e explaining the mismatch between the model and the real system. Indeed, one can trivially see that the choice of additive noise (4.2):

$$\mathrm{e} = \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) \tag{4.4}$$

While the least-squares approach is very commonly used in order to solve the system identification problem, it is very important to understand what it means, why and when it works, and what alternatives can be used when it fails.

Note that in the context of dynamic systems, the input-output of the system are typically input and output sequences, i.e.

$$\mathrm{y} = \begin{bmatrix} \hat{\mathrm{y}}_0 \\ \vdots \\ \hat{\mathrm{y}}_{N-1} \end{bmatrix}, \qquad \mathrm{u} = \begin{bmatrix} \mathrm{u}_0 \\ \vdots \\ \mathrm{u}_{N-1} \end{bmatrix} \tag{4.5}$$

and a (causal) model is then in the form:

$$\hat{\mathrm{y}} = \begin{bmatrix} \hat{\mathrm{y}}_0 \\ \vdots \\ \hat{\mathrm{y}}_{N-1} \end{bmatrix} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) = \begin{bmatrix} \varphi_0(\boldsymbol{\theta}, \mathrm{u}_0) \\ \varphi_1(\boldsymbol{\theta}, \mathrm{u}_0, \mathrm{u}_1) \\ \vdots \\ \varphi_{N-1}(\boldsymbol{\theta}, \mathrm{u}_0, \ldots, \mathrm{u}_{N-1}) \end{bmatrix} \tag{4.6}$$

Before building some theory, let us consider a simple example of the least-squares approach applied to a basic “FIR" (Finite-Impulse Response) model.

Examples

Let us consider a first example of parameter estimation for dynamic systems. Consider a model of the form:

$$\hat{y}_k = \sum_{i=0}^{n} \boldsymbol{\theta}_i u_{k-i} \tag{4.7}$$

having $n+1$ parameters. We can also rewrite it in the compact form:

$$\hat{\mathrm{y}} = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} u_0 & \ldots & u_{-n} \\ u_1 & \ldots & u_{1-n} \\ \vdots & & \vdots \\ u_{N-1} & \ldots & u_{N-1-n} \end{bmatrix}}_{:=U} \boldsymbol{\theta} = U\boldsymbol{\theta} \tag{4.8}$$

Hence the model is linear in the parameters $\boldsymbol{\theta}$, and reads as:

$$\boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) = U\boldsymbol{\theta} \tag{4.9}$$

The least-square estimation problem then reads as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) \right\|^2 = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \left\| \mathrm{y} - U\boldsymbol{\theta} \right\|^2 \tag{4.10}$$

We will see in a bit that the solution to (4.10) is:

$$\hat{\boldsymbol{\theta}} = \left(U^\top U \right)^{-1} U^\top \mathrm{y} \tag{4.11}$$

We will discuss in detail the meaning of this construction later in this Chapter. Let us illustrate next this example for an FIR model with $n+1 = 20$. The next Figure shows the outcome $\hat{\boldsymbol{\theta}}$ resulting from (4.11) for different inputs (left-hand graphs), yielding different output of the real system and model (centred graphs), and set of parameters $\hat{\boldsymbol{\theta}}$ (right-hand side graphs)

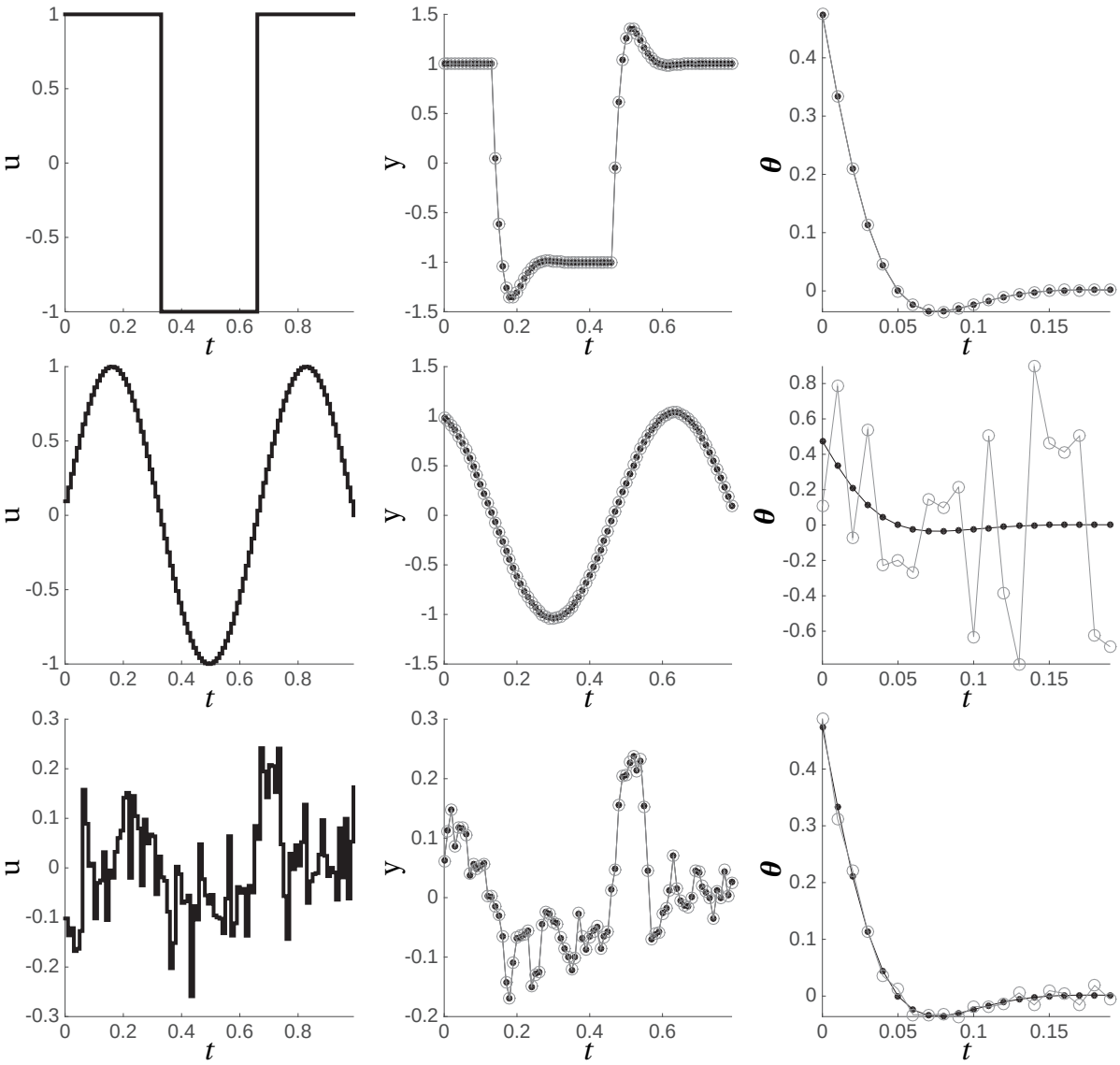


Figure 4.1: Illustration of the least-squares approach for a model of the form (4.7) for $n+1=20$. The real system here is also of the form (4.7) with $n+1=20$, but a slight measurement noise is introduced. The input sequences u are displayed on the left-hand side graphs. The black signals represent the outputs y (centred graphs) and parameters (right-hand side graphs) of the real system, while the grey signals represent the outputs $\hat{y}$ (centred graphs) and estimated parameters (right-hand side graphs) of the model, where the parameters are estimated using (4.11).

One can observe that the choice of input has a strong impact on the quality of the parameter estimation $\hat{\boldsymbol{\theta}}$, and that a poor parameter estimation can be obtained while the model output fits the system output fairly well. We will discuss these observations further in these notes.

□

Next we will build a theory behind the system identification problem. In order to do so, we need to start from the Bayesian principle.

4.1 BAYESIAN REASONING & BAYESIAN ESTIMATION

In order to make sense of the theory coming below, let us consider the following game. Consider we have an opaque bag containing N balls, possibly black (B) and white (W). We draw a ball, note its color, *put it back in the bag*, and repeat. Suppose we made n draws and saw only white balls. What can we say about the hypothesis (H) that "all balls in the bag are white"?

We can start with making a few intuitive observations:

- H (the hypothesis that all balls are white) is true or false, but with limited observations we cannot conclude with certainty that it is true or false
- Regardless of n (even with $n >> N$), we can still not be certain about H. We may just be "unlucky" and keep drawing only white balls even though there are some black balls in the bag.
- Nonetheless, it should be clear that as n becomes larger and larger (lots of only white balls drawn), we should feel more and more confident that all balls in the bag are white.

How to approach these observations in a formal way, and build a framework where we can discuss the validity of a given hypothesis using limited information? This problem is addressed via Bayesian reasoning. Bayesian reasoning is the mathematical tool that allows medical doctors to conclude that a new drug is effective using trials. It is the tool that allows biologists to assess the state of our environment with limited observations. It is the tool that allows physicists to use experiments to validate theories. It is a tool that allows the detection of e.g. fraudulent trading activities from unusual patterns in stock exchanges. In this section we will have a brief look at the formalism used in Bayesian reasoning.

Bayesian reasoning replaces the true/false, or equivalently "0/1", status of a hypothesis by a scale that can take all values in $[0,1]$. One classic interpretation of this scale is to look at it as the *probability that the hypothesis is true*, where 1 means that the hypothesis is certainly true, and 0 means that it is certainly wrong. Note that these probabilities will always be conditional, conditioned on the existing observations that ought to inform us on the validity of the hypothesis. In the mathematical language, we are talking about a *conditional probability* here, which is usually noted as:

$$\mathbb{P}[\,\mathrm{H}\,|\,\mathrm{O}\,] \tag{4.12}$$

and means “Probability that the hypothesis H is true provided that we gathered the observation O". More specifically, in our B/W balls game, we would have:

$$\mathbb{P}[\quad \underbrace{\text{All balls in the bag are white}}_{\text{H}} \quad | \quad \underbrace{n \text{ white balls drawn}}_{\text{O}} \quad] \tag{4.13}$$

Experimental sciences consider a hypothesis as validated if $\mathbb{P}[\,\text{H}\,|\,\text{O}\,]$ is sufficiently high (typically at least 0.95, but this value is purely arbitrary!). In the context of estimation, we use Bayesian reasoning in a slightly different way. We will rather consider making a choice between different competing hypothesis (say $H_1, ..., H_N$) and make that choice by simply favoring the hypothesis that has the highest probability of being true. In the game described above, the hypothesis H_k would be:

$$\text{H}_k: \text{ there are } k \text{ white balls in the bag} \tag{4.14}$$

and one would assess the probability of the different H_k to be true provided the observations, and select the one having the highest probability.

Let us now apply Bayesian reasoning in the context of parameter estimation. Consider the model

$$\hat{\text{y}} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \text{u}) \tag{4.15}$$

mapping the inputs u into the model outputs $\hat{\text{y}}$. We want to find the parameter $\boldsymbol{\theta}$ that best explains the measurements (observations) y collected on the real system.
We can then consider y as the observations, and the different possible values for $\boldsymbol{\theta}$ as competing hypothesis, that we assess via the conditional probability:

$$\mathbb{P}[\,\boldsymbol{\theta}\,|\,\text{y}\,] \tag{4.16}$$

meaning “probability that a *given* parameter $\boldsymbol{\theta}$ is true *provided* the measurements y". One ought to observe that $\mathbb{P}[\,\boldsymbol{\theta}\,|\,\text{y}\,]$ is in fact a function of $\boldsymbol{\theta}$ and y, which returns a scalar value in the interval $[0,1]$. Selecting the parameters $\boldsymbol{\theta}$ that explain best the observations y then boils down to solving:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathbb{P}\left[\,\boldsymbol{\theta}\,|\,\text{y}\,\right] \tag{4.17}$$

i.e. we ought to select the parameters $\hat{\boldsymbol{\theta}}$ as the parameters *most likely to be correct (true)*.

Unfortunately, evaluating (4.16) is impossible for most cases of interest. Fortunately, Bayes theorem saves the day. Indeed, as we will soon see, computing $\mathbb{P}[\,\text{y}\,|\,\boldsymbol{\theta}\,]$ is in most cases a lot easier than (4.16), and the Bayes theorem allows us to perform this “reversal" of the hypothesis and observations. An application of the Bayes theorem to (4.16) yields:

$$\mathbb{P}[\,\boldsymbol{\theta}\,|\,\text{y}\,] = \frac{\mathbb{P}[\,\text{y}\,|\,\boldsymbol{\theta}\,]\mathbb{P}[\,\boldsymbol{\theta}\,]}{\int \mathbb{P}[\,\text{y}\,|\,\boldsymbol{\theta}\,]\mathbb{P}[\,\boldsymbol{\theta}\,]\text{d}\boldsymbol{\theta}} \tag{4.18}$$

A slight difficulty with (4.18) is to provide the *prior* distribution $\mathbb{P}[\,\boldsymbol{\theta}\,]$, which is basically "the probability that a parameter $\boldsymbol{\theta}$ is correct *before any observation has been made*". This distribution is referred to as the "prior" in the Bayesian context as in "prior knowledge". In the framework we are developing here, though, we ought to adopt an *indifferent* view of the competing values of the parameter $\boldsymbol{\theta}$, i.e. we ought to consider that since we have no idea what that parameter is a priori (before we have collected observations), all values have equal probability. More formally, we ought to consider that:

$$\mathbb{P}[\,\boldsymbol{\theta}\,] = c \tag{4.19}$$

for some constant[5] c.

It follows that (4.18) simplifies to:

$$\mathbb{P}[\,\boldsymbol{\theta} \mid \mathrm{y}\,] = \frac{\mathbb{P}[\,\mathrm{y} \mid \boldsymbol{\theta}\,]}{\int \mathbb{P}[\,\mathrm{y} \mid \boldsymbol{\theta}\,]\mathrm{d}\boldsymbol{\theta}} \quad \propto \quad \mathbb{P}[\,\mathrm{y} \mid \boldsymbol{\theta}\,] \tag{4.20}$$

(here we use $\propto$ for "proportional to"). We then observe that

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathbb{P}\left[\,\boldsymbol{\theta} \mid \mathrm{y}\,\right] = \arg\max_{\boldsymbol{\theta}} \mathbb{P}\left[\,\mathrm{y} \mid \boldsymbol{\theta}\,\right] \tag{4.21}$$

i.e. *selecting the parameters most likely to be true* as done in (4.17) is equivalent to *selecting the parameters most likely to yield the measurements* y as done in (4.21), right-hand side. The latter is referred to as Maximum-Likelihood Estimation (MLE). We discuss this approach further next, for the specific problem of estimation model parameters.

4.2 Maximum Likelihood Estimation

Let us start with looking at the likelihood function. In words, the concept described by this function is: what is the probability of observing a set of measurements y for a given set of parameters $\boldsymbol{\theta}$? As detailed in (4.2), we often assume (but not always) that the mismatch between the measurements y and the model $\hat{\mathrm{y}} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$ can be explained by some additive noise e. Asking what is the probability of observing the given measurements y for a given parameter value $\boldsymbol{\theta}$ then boils down to asking what is the probability that the additive noise takes the value:

$$\mathrm{e} = \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) \tag{4.22}$$

for a given $\boldsymbol{\theta}$ i.e. the Likelihood function read as:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\,\mathrm{y} = \mathrm{e} + \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})\,\right] \tag{4.23}$$

and can be understood as follows:

[5] an attentive reader may notice here that (4.19) is actually not a distribution. Indeed, the choice (4.19) yields $\int \mathbb{P}[\,\boldsymbol{\theta}\,]\mathrm{d}\boldsymbol{\theta} \neq 1$. In fact, the integral is unbounded. Fortunately, this is not a problem in the context of the Bayes theorem, and (4.19) is sometimes referred to as a pseudo-distribution.

- The hypothesis used in (4.23) is $\mathrm{y} = \mathrm{e} + \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$. It describes the mismatch between the measurements y and the model output $\boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$ "explained" by the noise e. It is important here to understand that since y, u are known and $\boldsymbol{\theta}$ is provided (as an argument in the Likelihood function), the only random variable here is e, hence the notion of probability applies to e.
- We can easily make the latter observation explicit by rewriting (4.23) as:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})\ \right] \tag{4.24}$$

  Here we form the hypothesis that the noise e took a specified value $\mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$ (specified since y, u are known, and $\boldsymbol{\theta}$ is provided).

An important observation ought to be made here. In order to be able to specify the likelihood function (4.24) further, we need to decide on a "noise model", i.e. we need to specify what is the probability distribution of the random variable e, such that we can translate (4.23) into specific expressions.

In order to digest these concepts, let us make some simple examples.

Examples

1. Let us start with a trivial, but hopefully helpful case. Consider a single measurement, $\hat{y} \in \mathbb{R}$ used to estimate a single parameter $\theta \in \mathbb{R}$, based on the underlying model:

$$\hat{y} = \theta \cdot u \tag{4.25}$$

   where $u \in \mathbb{R}$ is a given (single) input. Estimating the parameter θ in this case is fairly obvious, as one ought to compute $\hat{\theta} = \frac{y}{u}$. Let us nevertheless use this example in order to unpack the notion of likelihood function (and the concepts that we will introduce later on). Since we expect the measurement y to be corrupted by measurement error, we ought to view our estimation problem as:

$$y = \theta \cdot u + e \tag{4.26}$$

   where e explains the difference between the measurement and the model. The likelihood function then reads as:

$$L(\theta) = \mathbb{P}\left[e = y - \theta \cdot u\right] \tag{4.27}$$

   In order to further specify the likelihood function, we need to pin a distribution to the random variable e. If we e.g. use a normal centred distribution, i.e. $e \sim \mathcal{N}(0, \sigma)$, then we can write:

$$\mathbb{P}[e = x] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}x^2} \tag{4.28}$$

   and our likelihood function reads as:

$$L(\theta) = \mathbb{P}\left[e = y - \theta \cdot u\right] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\theta\cdot u)^2} \tag{4.29}$$

2. Consider the basic problem of taking N measurements of a scalar quantity to estimate (e.g. the temperature in a room) in order to reduce the risk of having an erroneous estimation. We label θ the quantity to estimate via direct measurements, and we write the trivial model

$$\hat{\mathrm{y}} = \underbrace{\begin{bmatrix} \theta \\ \vdots \\ \theta \end{bmatrix}}_{:=\boldsymbol{\theta}} \tag{4.30}$$

representing the fact that we take direct measurements of that quantity. Note that we do not have “inputs" in this example. The error vector e is then given by:

$$\mathrm{y} = \hat{\mathrm{y}} + \mathrm{e} \tag{4.31}$$

The likelihood function then reads as:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta} \,\right] \tag{4.32}$$

In order to further specify the likelihood function, we need a probability distribution for the random variable e. If we select a noise model (i.e. a probability distribution for the random variable e) using e.g. the very common centred *multi-variate* normal distribution:

$$\mathrm{e} \sim \mathcal{N}(0, \Sigma) \tag{4.33}$$

for which the probability distribution reads as:

$$\mathbb{P}[\, \mathrm{e} = \mathrm{x} \,] = \frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}\mathrm{x}^\top \Sigma^{-1}\mathrm{x}} \tag{4.34}$$

Note that if the random variable e is uncorrelated, i.e. e_i is independent of e_j for $i \neq j$, then matrix Σ is diagonal, and (4.34) becomes a product of scalar normal distributions. It then reads as:

$$\mathbb{P}[\, \mathrm{e} = \mathrm{x} \,] = \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi\Sigma_k}} e^{-\frac{1}{2\Sigma_k}\mathrm{x}_k^2} \tag{4.35}$$

where Σ_k are the elements in the diagonal of matrix Σ, and correspond to the variance of the individual elements e_k, i.e. $\Sigma_k = \sigma_k^2$.

Using these information, we can write the likelihood function (4.32) more explicitly, i.e.

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta} \,\right] = \frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathrm{y}-\boldsymbol{\theta})^\top \Sigma^{-1}(\mathrm{y}-\boldsymbol{\theta})} \tag{4.36}$$

and if we adopt an uncorrelated noise model of variance σ_k^2, (4.36) translates to

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta} \,\right] = \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(\mathrm{y}_k-\theta)^2} \tag{4.37}$$

3. Consider the model:

$$\boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) = \boldsymbol{\theta}_1 \mathrm{u} + \boldsymbol{\theta}_0 \tag{4.38}$$

which is *affine in input*, and suppose we obtain the data y. The likelihood function then reads as:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}_1 \mathrm{u} - \boldsymbol{\theta}_0 \,\right] \tag{4.39}$$

If we select a noise model using e.g. the centred normal distribution again:

$$\mathrm{e} \sim \mathcal{N}(0, \Sigma) \tag{4.40}$$

we can write the likelihood function (4.39) as:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}_1 \mathrm{u} - \boldsymbol{\theta}_0 \,\right] = \frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathrm{y}-\boldsymbol{\theta}_1\mathrm{u}-\boldsymbol{\theta}_0)^\top \Sigma^{-1}(\mathrm{y}-\boldsymbol{\theta}_1\mathrm{u}-\boldsymbol{\theta}_0)} \tag{4.41}$$

and if we adopt a normally distributed uncorrelated noise model of variance σ_k^2, (4.41) translates to:

$$\mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}_1 \mathrm{u} - \boldsymbol{\theta}_0 \,\right] = \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(\mathrm{y}_k - \boldsymbol{\theta}_1 \mathrm{u}_k + \boldsymbol{\theta}_0)^2} \tag{4.42}$$

4. In order to conclude this list of simple examples, let us revisit the first example in this list, but assuming now that the random variable e does not follow a normal centred distribution, but e.g. a uniform distribution in the interval $[-1, 1]$. The distribution of e then reads as:

$$\mathbb{P}[e = x] = \begin{cases} \frac{1}{2} & \text{if} \quad x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases} \tag{4.43}$$

Our likelihood function then reads as:

$$L(\theta) = \mathbb{P}\left[e = y - \theta \cdot u\right] = \begin{cases} \frac{1}{2} & \text{if} \quad y - \theta \cdot u \in [-1, 1] \\ 0 & \text{otherwise} \end{cases} \tag{4.44}$$

which we could rewrite as:

$$\mathbb{P}\left[e = y - \theta \cdot u\right] = \begin{cases} \frac{1}{2} & \text{if} \quad -1 \leq y - \theta \cdot u \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.45}$$

□

Equipped with the concept of likelihood function, one can provide a very meaningful solution to the problem of estimating model parameters from data: the most sensible parameter to select is the one that maximizes the likelihood function. In other words, the maximum likelihood estimation approach selects the parameter as the one for which the model

has the highest probability of delivering the observed data. Mathematically, we express the maximum likelihood estimator (MLE) as:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \ \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})\ \right] \tag{4.46}$$

For a given model $\boldsymbol{\varphi}$, measurements y, and input u, the MLE optimization problem (4.46) returns as estimation the value of $\boldsymbol{\theta}$ that achieves the maximum of the likelihood function. We can now revisit our examples above, applying the MLE approach.

Examples

1. Our first problem had the likelihood function:

$$L(\theta) = \mathbb{P}\left[e = y - \theta \cdot u\right] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\theta\cdot u)^2} \tag{4.47}$$

   which is essentially a Gaussian function. It takes its maximum when the argument of the exponential is zero, i.e. when $y - \theta \cdot u = 0$. It follows that, unsurprisingly enough, the MLE of θ in this case is simply $\hat{\theta} = \frac{y}{u}$.

2. Our second example had the likelihood function:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}\ \right] = \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(\mathrm{y}_k - \theta)^2} \tag{4.48}$$

   In order to apply the MLE, we need to maximize (4.48) over θ. This can be achieved with a bit of calculus. Indeed, one can verify that:

$$\nabla_\theta \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}\ \right] = \sum_{i=0}^{N-1} \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(\mathrm{y}_k - \theta)^2} \frac{1}{\sigma_i^2}\left(\mathrm{y}_i - \theta\right) \tag{4.49}$$

$$= \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}\ \right] \cdot \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2}\left(\mathrm{y}_i - \theta\right) \tag{4.50}$$

   and $\nabla_\theta \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}\ \right] = 0$ is achieved by:

$$\sum_{i=0}^{N-1} \frac{1}{\sigma_i^2}\left(\mathrm{y}_i - \theta\right) = 0 \tag{4.51}$$

   and yields:

$$\hat{\theta} = \frac{1}{\sum_{i=0}^{N-1} \frac{1}{\sigma_i^2}} \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \mathrm{y}_i \tag{4.52}$$

   One can observe that (4.52) is a weighted average of the measurements $\hat{y}_i$, where the measurements having a low variance σ_i^2 receive a proportionally higher weight. If all measurements have the same variance, then (4.52) boils down to a simple average:

$$\hat{\theta} = \frac{1}{N} \sum_{i=0}^{N-1} \mathrm{y}_i \tag{4.53}$$

3. In this example with an affine model, we obtained the likelihood function:

$$L(\boldsymbol{\theta}) = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}_1 \mathrm{u} - \boldsymbol{\theta}_0 \,\right] = \prod_{k=0}^{N-1} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}\left(\mathrm{y}_k - \boldsymbol{\theta}_1 \mathrm{u}_k - \boldsymbol{\theta}_0\right)^2} \tag{4.54}$$

We can apply the same principles as in the previous example, and get:

$$\nabla_{\boldsymbol{\theta}} \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta} \,\right] = \mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - \boldsymbol{\theta} \,\right] \cdot \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left(\mathrm{y}_i - \boldsymbol{\theta}_1 \mathrm{u}_i - \boldsymbol{\theta}_0\right) \begin{bmatrix} 1 \\ u_i \end{bmatrix} \tag{4.55}$$

The MLE $\boldsymbol{\theta}$ is then given by:

$$\sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left(\mathrm{y}_i - \boldsymbol{\theta}_1 \mathrm{u}_i - \boldsymbol{\theta}_0\right) \begin{bmatrix} 1 \\ u_i \end{bmatrix} = 0 \tag{4.56}$$

While this equation is linear in $\boldsymbol{\theta}$ and can therefore be solved fairly easily, we will leave the final solution aside, and make more sense of what we are doing here in the next section.

4. Our last example had the likelihood function:

$$L(\theta) = \mathbb{P}\left[e = y - \theta \cdot u\right] = \begin{cases} \frac{1}{2} & \text{if} \quad -1 \leq y - \theta \cdot u \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.57}$$

Unfortunately, this likelihood function is discontinuous, such that trying to find its maximum by solving $\nabla_\theta \mathbb{P}\left[e = y - \theta \cdot u\right] = 0$ is of no help. However, function (4.57) takes only two possible values, the largest of which is $\frac{1}{2}$. Hence, any choice of parameter θ that achieves the value $\frac{1}{2}$ is a valid MLE. More specifically, we can describe the MLE of θ as:

$$\hat{\theta} = \left\{ \theta \quad | \quad -1 \leq y - \theta \cdot u \leq 1 \right\} \tag{4.58}$$

if the set is not empty. If the set is empty, then (4.57) takes only 0 as a value, and any θ is a valid MLE (though not a very meaningful one).

□

4.3 Least-Squares Estimation

We will now develop the least-squares estimation (and its possible variants) from the MLE principle. The big picture is that the MLE can be converted into a *fitting problem* for many types of distributions of the error e. In particular, for an additive noise having a normal centred distribution, the MLE is equivalent to a least-square estimation (LSE). Let us start with this specific transformation.

Consider the MLE (4.46) for a given model $\boldsymbol{\varphi}$, a given input u and measurements y:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \ \mathbb{P}\left[\ \mathrm{e} = \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})\ \right] \tag{4.59}$$

and for a normal centred distribution of e, i.e.

$$\mathbb{P}\left[\ \mathrm{e} = \mathrm{x}\ \right] = \frac{1}{\det\left(2\pi\Sigma\right)^{\frac{1}{2}}} e^{-\frac{1}{2}\mathrm{x}^{\top}\Sigma^{-1}\mathrm{x}} \tag{4.60}$$

Then the MLE reads as[6]:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \ e^{-\frac{1}{2}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)^{\top}\Sigma^{-1}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)} \tag{4.61}$$

We will now proceed with a sequence of transformation on (4.61). First consider taking the log on the likelihood function in (4.61). Because the log function is monotonous, a function and its log achieve their maximum at the same point, hence:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \ \log\left[e^{-\frac{1}{2}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)^{\top}\Sigma^{-1}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)}\right] \tag{4.62}$$

holds, or equivalently:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \ -\frac{1}{2}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)^{\top}\Sigma^{-1}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right) \tag{4.63}$$

Then one can observe that the maximum of a function f is achieved at the same point as the minimum of $-f$, i.e.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)^{\top}\Sigma^{-1}\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right) \tag{4.64}$$

holds. We then observe that (4.64) is a least-squares problem that is often written as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2}\left\|\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right\|^2_{\Sigma^{-1}} \tag{4.65}$$

i.e. the MLE (4.61) delivers the exact same parameter estimation $\hat{\boldsymbol{\theta}}$ as the LSE (4.65). Let us state this observation as clearly as possible hereafter.

The MLE in the presence of an additive error term e that is centred normal distributed is equivalent to a Least-Squares Estimation.

[6] Note that we can dismiss the constant $\frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}}$ here, as it does not change the value of $\boldsymbol{\theta}$ that achieves the maximum of the likelihood function

Example

- Let us consider a first example of parameter estimation for dynamic systems. Consider a model of the form:

$$\hat{y}_k = \sum_{i=0}^{n} \boldsymbol{\theta}_i u_{k-i} \tag{4.66}$$

having $n+1$ parameters. We can also rewrite it in the compact form:

$$\hat{\mathrm{y}} = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} u_0 & \dots & u_{-n} \\ u_1 & \dots & u_{1-n} \\ \vdots & & \vdots \\ u_{N-1} & \dots & u_{N-1-n} \end{bmatrix}}_{:=U} \boldsymbol{\theta} = U\boldsymbol{\theta} \tag{4.67}$$

The measurements y are then given by:

$$\mathrm{y} = U\boldsymbol{\theta} + \mathrm{e} \tag{4.68}$$

Consider that the error term e is normally distributed and centred, such that the MLE associated to estimating the model parameters $\boldsymbol{\theta}$ reads as:

$$\mathbb{P}\left[\, \mathrm{e} = \mathrm{y} - U\boldsymbol{\theta} \,\right] = \frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathrm{y}-U\boldsymbol{\theta})^{\top}\Sigma^{-1}(\mathrm{y}-U\boldsymbol{\theta})} \tag{4.69}$$

and the equivalent LSE problem reads as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2} \left\| \mathrm{y} - U\boldsymbol{\theta} \right\|_{\Sigma^{-1}}^2 \tag{4.70}$$

- The transformation (**??**) to (4.65) can be generalized to many distributions. Let us illustrate this principle on e.g. an uncorrelated "*fair*" normal centered distribution, i.e.[7]

$$\mathbb{P}\left[\mathrm{e} = \mathrm{x}\right] \sim \prod_{k=0}^{N-1} e^{-\frac{\mathrm{x}_k^2}{2\sigma^2\left(1+\alpha\left|\mathrm{x}_k\right|\right)}} \tag{4.71}$$

[7] Note that σ^2 is not the variance anymore. This distribution describes a random variable that "looks like" it is normal for e small, but that makes the occurance of large e more likely than under a normal distribution. This type of distribution is often referred to as a "fat-tail" distribution, because it looks like a Gaussian distribution but with "fatter" tails.

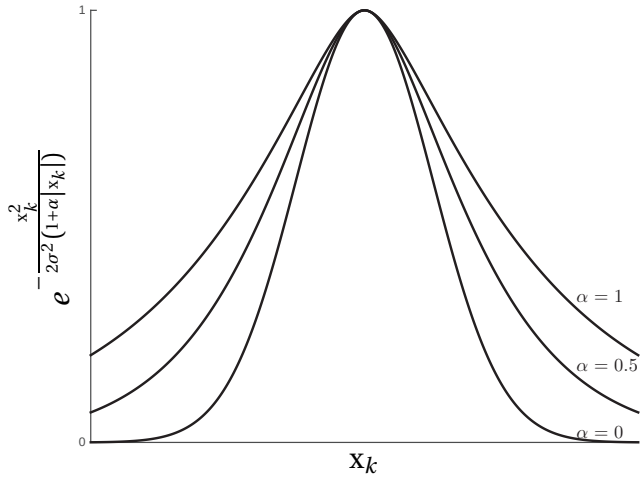


Figure 4.2: Illustration of the fair distribution for a few values of the constant α.

The MLE problem for model (4.68) then reads as:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \prod_{k=0}^{N-1} e^{-\frac{\left(y_k - U_k\boldsymbol{\theta}\right)^2}{2\sigma^2\left(1+\alpha\left|y_k - U_k\boldsymbol{\theta}\right|\right)}} \tag{4.72}$$

and its transformation via the log function reads as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{2\sigma^2} \sum_{k=0}^{N-1} \frac{\left(y_k - U_k\boldsymbol{\theta}\right)^2}{1+\alpha\left|y_k - U_k\boldsymbol{\theta}\right|} \tag{4.73}$$

We can further rewrite it as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{2\sigma^2} \left\|y - U\boldsymbol{\theta}\right\|_{\text{Fair}} \tag{4.74}$$

where we use the notation

$$\|x\|_{\text{Fair}} = \sum_{k=0}^{N-1} \frac{x_k^2}{1+\alpha|x_k|} \tag{4.75}$$

Note that we are somewhat abusing the standard notation $\|.\|$ here, as (4.75) does not define a norm (see definition of a mathematical norm in Sec. 1.3).

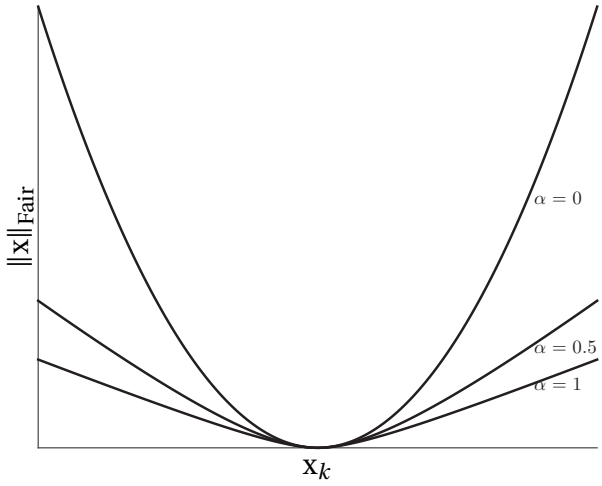


Figure 4.3: Illustration of the cost function (4.75) associated to the fair distribution for a few values of the constant α.

4.4 Solution to the least-squares problem

In this section, we will review the solution of least-squares problem (4.65), i.e. problems of the general form:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \frac{1}{2} \left\| \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}) \right\|_W^2 \tag{4.76}$$

We need to distinguish here two cases: linear least-squares problems (LLS) where the model $\boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$ is linear in $\boldsymbol{\theta}$ and nonlinear least-squares problems (NLS) where the model $\boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u})$ is nonlinear in $\boldsymbol{\theta}$. Note that the model function can be nonlinear in u and be linear in $\boldsymbol{\theta}$ nonetheless. While both LLS and NLS use the same mathematical principles, the solution approaches to obtain the solution are radically different. Indeed, LLS problems have an explicit solution, while NLS problems usually require the deployment of a Newton-type method (see Sec. 3).

Linear least-squares matching e.g. the form obtained in (4.70) can be generally written as:

$$\hat{\boldsymbol{\theta}} = \min_{\theta} \frac{1}{2} \left\| \mathrm{y} - A\boldsymbol{\theta} \right\|_W^2 \tag{4.77}$$

which equivalently read as:

$$\hat{\boldsymbol{\theta}} = \min_{\theta} \frac{1}{2} \left(\mathrm{y} - A\boldsymbol{\theta} \right)^\top W \left(\mathrm{y} - A\boldsymbol{\theta} \right) \tag{4.78}$$

The solution approach to solving (4.77) relies on finding the roots of the gradient of the cost function underlying (4.77), i.e. labelling $J\left(\boldsymbol{\theta}, \mathrm{y}\right) = \frac{1}{2} \left\| \mathrm{y} - A\boldsymbol{\theta} \right\|_W^2$, the parameter estimation $\hat{\boldsymbol{\theta}}$ is given by solving:

$$\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}, \mathrm{y}\right) = \nabla_{\boldsymbol{\theta}} \frac{1}{2} \left\| \mathrm{y} - A\boldsymbol{\theta} \right\|_W^2 = 0 \tag{4.79}$$

The equation stemming from (4.79) then read as:

$$A^\top W\left(A\boldsymbol{\theta}-\mathrm{y}\right)=0 \tag{4.80}$$

and delivers the standard solution to LLS problems, i.e.:

$$\hat{\boldsymbol{\theta}}=\left(A^\top WA\right)^{-1}A^\top W\mathrm{y} \tag{4.81}$$

There is a handful of crucial observations to be made here:

- The parameter estimation $\hat{\boldsymbol{\theta}}$, if it exists, is a linear function of the data/measurements y.
- The parameter estimation $\hat{\boldsymbol{\theta}}$, if it exists, is unique (this is to be contrasted with NLS problems)
- The parameter estimation $\hat{\boldsymbol{\theta}}$ exists if and only if the square matrix $A^\top WA$ is full rank, i.e. invertible.

When the model $\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})$ is nonlinear in $\boldsymbol{\theta}$, then the parameter estimation is still obtained by finding the root of the gradient of the underlying cost function, i.e.

$$\nabla_{\boldsymbol{\theta}}J\left(\boldsymbol{\theta},\mathrm{y}\right)=\nabla_{\boldsymbol{\theta}}\frac{1}{2}\left\|\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right\|_W^2=0 \tag{4.82}$$

but the developments above do not apply further. Equation (4.82) also reads as:

$$\nabla_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})W\left(\mathrm{y}-\boldsymbol{\varphi}(\boldsymbol{\theta},\mathrm{u})\right)=0 \tag{4.83}$$

and is nonlinear in $\boldsymbol{\theta}$. It is therefore generally impossible to solve explicitly, and one typically deploys a numerical method in order to find θ. The methods detailed on Section 3.5 can be used.

4.5 Covariance, Bias & Consistency

There are some crucial properties underlying parameter estimation that we ought to discuss here. These properties essentially discuss the extent to which a chosen estimation technique delivers a meaningful estimation $\hat{\boldsymbol{\theta}}$. Discussing these properties is the object of this section. First, let us consider a generic parameter estimation technique:

$$\hat{\boldsymbol{\theta}}=\Psi\left(\mathrm{u},\mathrm{y},N\right) \tag{4.84}$$

taking the inputs u and the measurements y, and turning them into a parameter estimation. The integer $N\in\mathbb{N}$ represents the number of data point used in building $\hat{\boldsymbol{\theta}}$ (e.g. N in (4.67)). We observe that e.g. a Least-Squares estimation approach such as (4.65) can be written in the form (4.84), but also all the examples presented above.

It is important to observe here that the data y ought to be considered as the realization of a random variable, e.g. of the noise e if we consider that $y = \boldsymbol{\varphi}(\boldsymbol{\theta}, u) + e$. It follows that, $\hat{\boldsymbol{\theta}}$ becomes also a realization of a random variable, as it is a function of y through the estimator Ψ in (4.84). These observations, in practice, imply that if one runs the estimation procedure a number of times, each time using a set of newly collected data y, the outcome of that process is that $\hat{\boldsymbol{\theta}}$ will be different each time, and therefore build over many experiments a distribution itself. It is then useful to discuss what the distribution of $\hat{\boldsymbol{\theta}}$ looks like. Is it wide or narrow? Does it average to the true parameter $\hat{\boldsymbol{\theta}}$? How does the number of data point N impact the distribution of $\hat{\boldsymbol{\theta}}$? These are the questions we discuss next. We need to define first a basic terminology on estimators. Let us label $\boldsymbol{\theta}_\star$ the "true" model parameter[8] that we are trying to estimate through $\hat{\boldsymbol{\theta}}$.

1. An estimator Ψ is labelled **unbiased** if

$$\mathbb{E}\left[\hat{\boldsymbol{\theta}}\right] = \boldsymbol{\theta}_\star \tag{4.85}$$

   where $\mathbb{E}[.]$ labels the expected value operator (i.e. roughly speaking the average over a very large number of experiments giving different $\hat{\boldsymbol{\theta}}$). In words, if one uses an unbiased estimator, running the estimation process a large number of times to collect many $\hat{\boldsymbol{\theta}}$ and averaging them yields the correct parameter $\boldsymbol{\theta}_\star$.

2. An estimator is **consistent** if[9]

$$\lim_{N\to\infty} \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_\star \tag{4.86}$$

   In words, it one collects a very large number of data points y (i.e. $N \to \infty$), the parameter estimation $\hat{\boldsymbol{\theta}}$ will match the correct value.

3. For an **efficient** estimator, the estimates $\hat{\boldsymbol{\theta}}$ become normally distributed for N large enough, and the variance meets the *Cramer-Rao lower bound*. The latter bound comes from information theory and is essentially "the best one can do for a given set of data". An estimator is therefore efficient if it makes the "best possible use" of the data. It also implies that for $N \to \infty$ (and some conditions on the problem and data), the normal distribution of the estimates becomes "arbitrarily narrow" (i.e. it tends to a Dirac distribution), such that $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_\star$ almost surely.

Two remarks ought to be useful here.

[8] it is debatable that such a true parameter actually exists. Most models are indeed not capturing the reality perfectly such that there is no such thing as a "true" parameter. It is, however, very useful to reason based on the assumption that there is a true parameter when discussing estimation techniques. Indeed, if such a true parameter could be found, it would arguably be desirable from an estimation technique that it does find this parameter in one way or another.

[9] to be more formal, we ought to say that $\hat{\boldsymbol{\theta}} \to \boldsymbol{\theta}_\star$ *almost surely* as $N \to \infty$, where "almost surely" means "with probability 1". The distinction between this statement and the limit (4.86) is subtle and of secondary importance here.

- Efficiency implies consistency. This observation is explained at the end of point 3. Indeed if the estimates distribution tends to a Dirac function, then $\hat{\boldsymbol{\theta}}$ will be arbitrarily close to $\boldsymbol{\theta}_\star$ with probability 1.
- Efficiency or consistency do not necessarily imply unbiasedness, i.e. an estimator can be efficient (and) or consistent and still biased. Biasedness has to do with the behavior of estimators for limited (possibly very few) data point, and whether the estimates average to the correct parameter value for *a large amount of experiments*, as opposed to a large amount of data.

We are now ready to discuss the properties of Maximum-Likelihood Estimation (MLE).

- An MLE estimator is **in general biased**. It is unbiased in some special cases, though
- An MLE estimator is always **efficient** and therefore **consistent**

Let us now turn to more practical considerations. Consider the solution

$$\hat{\boldsymbol{\theta}} = \left(A^\top W A\right)^{-1} A^\top W \mathrm{y} \tag{4.87}$$

given in (4.81) to the least-squares problem (4.77), i.e.

$$\hat{\boldsymbol{\theta}} = \min_\theta \frac{1}{2} \left\| A\boldsymbol{\theta} - \mathrm{y} \right\|_W^2 \tag{4.88}$$

Here we can readily observe how $\hat{\boldsymbol{\theta}}$ is a (linear) function of the data y. Let us assume that there is a true parameter $\boldsymbol{\theta}_\star$ and that the measurements are given by the model disturbed by normally distributed, centered, additive noise, i.e.

$$\mathrm{y} = A\boldsymbol{\theta}_\star + \mathrm{e} \tag{4.89}$$

It follows that

$$\hat{\boldsymbol{\theta}} = \left(A^\top W A\right)^{-1} A^\top W \left(A\boldsymbol{\theta}_\star + \mathrm{e}\right) = \boldsymbol{\theta}_\star + \left(A^\top W A\right)^{-1} A^\top W \mathrm{e} \tag{4.90}$$

We can observe that if the noise e is centred (i.e. $\mathbb{E}[\mathrm{e}] = 0$), then **regardless of the distribution**:

$$\mathbb{E}\left[\hat{\boldsymbol{\theta}}\right] = \boldsymbol{\theta}_\star \tag{4.91}$$

holds, i.e. the linear least-squares estimation is unbiased. Hence if the noise is normal distributed, then LSE is a special case of MLE that achieves unbiasedness. We also observe that since $\hat{\boldsymbol{\theta}}$ is an affine transformation of e, then if e is normally distributed then so is $\hat{\boldsymbol{\theta}}$. We can assess the covariance of the estimates, i.e.

$$\mathbb{E}\left[\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right)\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right)^\top\right] = \left(A^\top W A\right)^{-1} A^\top W \Sigma_\mathrm{e} W A \left(A^\top W A\right)^{-1} \tag{4.92}$$

where we use $\Sigma_e = \mathbb{E}\left[ee^\top\right]$. Finally, in order for our least squares problem to be an MLE (see (4.70)), we need e to be normal distributed, and we need to choose $W = \Sigma_e^{-1}$ (see (4.70)). We then get:

$$\Sigma_{\hat{\boldsymbol{\theta}}} = \mathbb{E}\left[\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right)\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right)^\top\right] = \left(A^\top \Sigma_e^{-1} A\right)^{-1} \tag{4.93}$$

We therefore observe that for Σ_e large, $\Sigma_{\hat{\boldsymbol{\theta}}}$ is also large, i.e. in words, a lot of measurement noise yield a large co-variance of the parameters. Moreover, we observe that if matrix $A^\top \Sigma_e^{-1} A$ is close to being rank deficient (i.e. ill-conditioned or "close to being not invertible"), then $\Sigma_{\hat{\boldsymbol{\theta}}}$ becomes large (at least in some directions). This explains the observations made in Figure 4.1. Indeed, in the Least-Squares problem (4.11), the matrix $U = A$ and $W = I$. Matrix $A^\top \Sigma_e^{-1} A = U^\top U$ is then built based on the input sequence u, such that the co-variance of the parameter estimation depends highly on the choice of input sequence. The sinusoidal input (middle row) in Figure 4.1 yields a matrix $U^\top U$ that is close to singular and yields a very poor parameter estimation. The other signals yield better conditioned matrices and yield better estimations.

Let us wrap up our observations.

- A Linear Least Square (LLS) problem is equivalent to a MLE if the noise is normally distributed, centred, and for the adequate choice of weights (i.e. $W = \Sigma_e^{-1}$).
- A LLS is unbiased if the noise is additive and centered (regardless of its distribution)
- The covariance of the estimates $\hat{\boldsymbol{\theta}}$, i.e. $\Sigma_{\hat{\boldsymbol{\theta}}}$, depends on matrix A and the noise covariance Σ_e (regardless of its distribution)

4.6 Estimation for Linear Dynamic Systems

Let us now apply what we have established so far in the context of estimation for dynamic systems. We will work here with discrete-time linear dynamic systems. System identification for continuous systems is a bit more intricate and we will leave it out of these course notes. In general, linear dynamic systems can be represented via the equation:

$$\hat{\mathrm{y}}_k = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i \hat{\mathrm{y}}_{k-i} \tag{4.94}$$

describing the model output y at time k as a function of the past (and possibly present) inputs $\mathrm{u}_k, \ldots \mathrm{u}_{k-N_\mathrm{b}}$ and past outputs $\hat{\mathrm{y}}_{k-1}, \ldots \hat{\mathrm{y}}_{k-N_\mathrm{a}}$. The model parameters here are the vectors a and b, i.e. we can e.g. write:

$$\boldsymbol{\theta} = \begin{bmatrix} \mathrm{a} \\ \mathrm{b} \end{bmatrix} \tag{4.95}$$

We can then investigate how to deploy the estimation techniques investigated above to this specific type of model.

4.6.1 Estimation of Finite Impulse Response models

To start simple, let us first consider a simple special case of (4.94), namely Finite Impulse Response models, which have a = 0, i.e.

$$\hat{\mathrm{y}}_k = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} \tag{4.96}$$

We have in fact already considered this case in an example above (see (4.66)) for the case $\hat{\mathrm{y}}_k \in \mathbb{R}$. Let us expand on this case. Recall that a sequence of model output can be put in a matrix form:

$$\hat{\mathrm{y}} = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} u_0 & \dots & u_{-N_\mathrm{b}} \\ u_1 & \dots & u_{1-n} \\ \vdots & & \vdots \\ u_{N-1} & \dots & u_{N-1-N_\mathrm{b}} \end{bmatrix}}_{:=U} \boldsymbol{\theta} = U\boldsymbol{\theta} \tag{4.97}$$

where we label $\mathrm{b} = \boldsymbol{\theta}$. If the mismatch between the measurements y and the model output $\hat{\mathrm{y}}$ are to be explained by additive, centered, normally distributed noise e, i.e.

$$\mathrm{y} = \hat{\mathrm{y}} + \mathrm{e} \tag{4.98}$$

then MLE of the model parameters $\hat{\boldsymbol{\theta}}$ is a linear least squares problem (see (4.70)):

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2} \left\| \mathrm{y} - U\boldsymbol{\theta} \right\|^2_{\Sigma^{-1}} \tag{4.99}$$

The estimate $\hat{\boldsymbol{\theta}}$ is then efficient and consistent (delivers true parameter for N large enough), and is also unbiased because it is given by a linear least squares problem. The covariance of $\hat{\boldsymbol{\theta}}$ is given by the formula (4.93) and reads as

$$\Sigma_{\hat{\boldsymbol{\theta}}} = \mathbb{E}\left[\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right) \left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\star\right)^\top \right] = \left(U^\top \Sigma_\mathrm{e}^{-1} U \right)^{-1} \tag{4.100}$$

In the specific case that the noise e is white (i.e. noise samples e_i and $\mathrm{e}_{j\neq i}$ are uncorrelated), and has a uniform covariance (i.e. all noise sample e_i have the same variance σ^2), then the estimate covariance is given by:

$$\Sigma_{\hat{\boldsymbol{\theta}}} = \sigma^2 \left(U^\top U \right)^{-1} \tag{4.101}$$

We observe that if we have a "large" amount of data, i.e. if $N >> N_\mathrm{b}$, then matrix $U^\top U$ looks as follows:

$$\boxed{U^\top} \ \boxed{\begin{matrix} \\ U \\ \\ \\ \end{matrix}} = \boxed{U^\top U}$$

i.e. it is made of the product of a “long" lying matrix by a “tall" matrix, and has dimensions $N_\mathrm{b} \times N_\mathrm{b}$, which creates some “richness" in matrix $U^\top U$. The opposite, i.e. $N < N_\mathrm{b}$ (insufficient data) makes matrix $U^\top U$ rank deficient, and the estimates $\hat{\boldsymbol{\theta}}$ are undefined (infinite covariance!).

4.6.2 Auto-regressive models - Simulation error

Let us now turn to more general models in the form (4.94), which we label as auto-regressive as the past model outputs participate in building the next model output. Let us consider a series of model outputs $\hat{\mathrm{y}}_{0,\dots,N}$ and investigate how they relate to the model parameters. We have seen that in the FIR case, this relationship can be put in the linear form $\hat{\mathrm{y}} = U\boldsymbol{\theta}$. Is it still the case for more generic models of the form (4.94)? It is in fact easy to verify that the model output of generic models are in general not a linear function of the parameters $\boldsymbol{\theta}$. Indeed consider the simple example:

$$\hat{\mathrm{y}}_k = \mathrm{b}_0 \mathrm{u}_k + \mathrm{a}_1 \hat{\mathrm{y}}_{k-1} \tag{4.102}$$

We can then compute:

$$\hat{\mathrm{y}}_1 = \mathrm{b}_0 \mathrm{u}_1 + \mathrm{a}_1 \hat{\mathrm{y}}_0 \tag{4.103a}$$
$$\hat{\mathrm{y}}_2 = \mathrm{b}_0 \mathrm{u}_2 + \mathrm{a}_1 \left(\mathrm{b}_0 \mathrm{u}_1 + \mathrm{a}_1 \hat{\mathrm{y}}_0\right) = \mathrm{b}_0 \mathrm{u}_2 + \mathrm{a}_1 \mathrm{b}_0 \mathrm{u}_1 + \mathrm{a}_1^2 \hat{\mathrm{y}}_0 \tag{4.103b}$$
$$\vdots \tag{4.103c}$$

We can deduce from these developments that more generally:

$$\hat{\mathrm{y}}_k = \mathrm{a}_1^k \hat{\mathrm{y}}_0 + \mathrm{b}_0 \sum_{i=1}^{k} \mathrm{a}_1^{k-i} \mathrm{u}_i \tag{4.104}$$

We observe that the model output is a linear function of the model inputs u and initial conditions $\hat{\mathrm{y}}_0$ (this is expected as the model is linear), but it is a nonlinear function of the model parameters b_0, a_1 as they are are combined nonlinearly (powers and multiplications). We can conclude that it is impossible to write the model output using a linear expression of the form (4.97).

Nonetheless, we observe that (4.104) can be put in the generic model form (4.15) i.e.:

$$\hat{\mathrm{y}} = \begin{bmatrix} \hat{\mathrm{y}}_1 \\ \vdots \\ \hat{\mathrm{y}}_N \end{bmatrix} = \begin{bmatrix} \mathrm{a}_1 \hat{\mathrm{y}}_0 + \mathrm{b}_0 \mathrm{u}_1 \\ \vdots \\ \mathrm{a}_1^N \hat{\mathrm{y}}_0 + \mathrm{b}_0 \sum_{i=1}^{N} \mathrm{a}_1^{N-i} \mathrm{u}_i \end{bmatrix} = \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}, \hat{\mathrm{y}}_0) \tag{4.105}$$

such that the developments of section 4.3 hold, and in particular the MLE-based parameter estimation for (4.105) is given by (4.65), i.e. it is a **Nonlinear Least-Squares problem** of the form:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2} \left\| \mathrm{y} - \boldsymbol{\varphi}(\boldsymbol{\theta}, \mathrm{u}, \hat{\mathrm{y}}_0) \right\|_{\Sigma^{-1}}^2 \tag{4.106}$$

The estimates $\hat{\boldsymbol{\theta}}$ must then be obtained (in most cases) numerically via a Newton method solving (4.83). It may be useful here to underline that **even though the model is linear, the resulting least-squares problem is nonlinear**. In this case, the resulting estimates $\hat{\boldsymbol{\theta}}$ hold the properties of MLE, namely they are efficient and consistent, but **they are generally biased**.

4.6.3 Auto-regressive models - Prediction error

It is time now to reflect on what we are trying to do when estimating the parameters of the AR model (4.94) using collected data y. The model outputs generated by the AR model (4.94) stem from the input sequence u and the initial conditions $\hat{y}_0$, such as is e.g. explicitly shown in example (4.105). The model parameters are then adjusted so that the sequence of model outputs follows as closely as possible the data y. This approach, put in other words, seeks at finding model parameters a, b such that the model provides a good *simulation* of the real system. While this approach makes sense in a number of cases, it is not the only view one can adopt to adjusting the model parameters.

Another view consists in finding the model parameters that deliver a small *prediction error*. In that context, one would want to build a model that - fed with the most recent measurements y - delivers a good estimation of what the real system will do next. More specifically, one then considers the model:

$$\hat{y}_k = \sum_{i=0}^{N_b} b_i u_{k-i} + \sum_{i=1}^{N_a} a_i y_{k-i} \tag{4.107}$$

instead of (4.94). The difference between (4.107) and (4.94) is that (4.107) builds $\hat{y}_k$ based on past *measurements* y_{k-i} as opposed to past *model outputs* $\hat{y}_{k-i}$ as in (4.94). The Prediction Error model (4.107) will then focus on building model outputs $\hat{y}_k$ that are good prediction of the future system output based on the data measured so far y_{k-i}. The mismatch:

$$\hat{y}_k - y_k \tag{4.108}$$

between the model output $\hat{y}_k$ and the real system output or measurement y_k is called a prediction error, and adjusting the model parameters a, b according to

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2} \left\| \hat{y} - y \right\|^2_{\Sigma^{-1}} \tag{4.109}$$

where

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} \tag{4.110}$$

is given by (4.107). This approach is called a **Prediction Error Method** (PEM).

We ought to come back to the question whether a linear or nonlinear least-squares problem results from this alternative approach. To that end, we need to understand if $\hat{\mathrm{y}}$ in (4.107) can be written as a linear (or affine) function of the model parameters a, b. Answering that question is fairly straightforward. Indeed, we observe that:

$$\hat{\mathrm{y}} = \begin{bmatrix} \hat{\mathrm{y}}_0 \\ \vdots \\ \hat{\mathrm{y}}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \mathrm{y}_{-1} & \cdots & \mathrm{y}_{-N_\mathrm{a}} & \mathrm{u}_0 & \cdots & \mathrm{u}_{-N_\mathrm{b}} \\ \mathrm{y}_0 & \cdots & \mathrm{y}_{1-N_\mathrm{a}} & \mathrm{u}_1 & \cdots & \mathrm{u}_{1-N_\mathrm{b}} \\ \vdots & & & & & \\ \mathrm{y}_{N-1} & \cdots & \mathrm{y}_{N-N_\mathrm{a}} & \mathrm{u}_N & \cdots & \mathrm{u}_{N-N_\mathrm{b}} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \mathrm{a}_1 \\ \vdots \\ \mathrm{a}_{N_\mathrm{a}} \\ \mathrm{b}_0 \\ \vdots \\ \mathrm{b}_{N_\mathrm{b}} \end{bmatrix}}_{\boldsymbol{\theta}} = M\boldsymbol{\theta} \tag{4.111}$$

hence the model output sequence $\hat{\mathrm{y}}$ can indeed be written as a linear function of the model parameters $\boldsymbol{\theta}$. The Least-Squares problem (4.109) underlying the PEM method is therefore linear and reads as:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \ \frac{1}{2} \left\| M\boldsymbol{\theta} - \mathrm{y} \right\|^2_{\Sigma^{-1}} \tag{4.112}$$

where matrix M is in fact also a function of the measurements y.

4.6.4 Prediction error & Noise model

We now need to turn to a not really trivial question, which is to understand how we assume the noise enters into the problem when using the PEM approach. This question appears certainly unclear at this stage, so let us try to unpack it step by step. Let us get back to the FIR model first, where (see (4.96))

$$\hat{\mathrm{y}}_k = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} \tag{4.113}$$

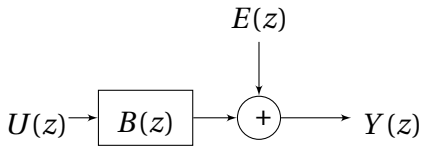
and

$$\mathrm{y}_k = \mathrm{e}_k + \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} \tag{4.114}$$

We observe first that in the FIR case the “simulation" approach of section 4.6.2 and the PEM of section 4.6.3 are in fact identical. Indeed, there is not “past data" that can be fed into the model here since the model output depends purely on the input u. If we take the Z-transform of (4.114), we obtain:

$$Y(z) = E(z) + \underbrace{\sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i z^{-i}}_{B(z)} U(z) \tag{4.115}$$

such that the FIR setup assumes that the signals are treated via the block-diagram



This picture ought appear fairly unsurprising. The noise e is additive in equations (4.114) and (4.115) and so is it in our block diagram. However, we will apply the same process to the AR model (4.107) used in the PEM now, and a different picture will emerge.

In the PEM context, recall that the model reads as (see (4.107)):

$$\hat{\mathrm{y}}_k = \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i \mathrm{y}_{k-i} \tag{4.116}$$

such that

$$\mathrm{y}_k = \mathrm{e}_k + \sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i \mathrm{u}_{k-i} + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i \mathrm{y}_{k-i} \tag{4.117}$$

The Z-transform of (4.117) reads as:

$$Y(z) = E(z) + \underbrace{\sum_{i=0}^{N_\mathrm{b}} \mathrm{b}_i z^{-i}}_{=B(z)} U(z) + \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i z^{-i} Y(z) \tag{4.118}$$

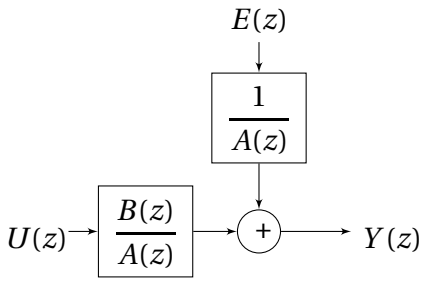
i.e.

$$A(z)Y(z) = E(z) + B(z)U(z) \tag{4.119}$$

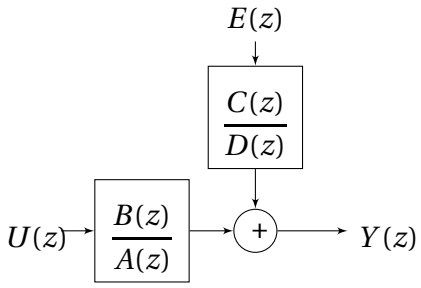
where $A(z) = 1 - \sum_{i=1}^{N_\mathrm{a}} \mathrm{a}_i z^{-i}$. Hence we finally have that:

$$Y(z) = \frac{B(z)}{A(z)} U(z) + \frac{1}{A(z)} E(z) \tag{4.120}$$

In a block-diagram representation, this relationship would look like:



And we see that the noise is no longer simply additive in the model dynamics, but is in fact "filtered" via the model denominator $A(z)$. An intuitive way of construing this observation is to simply observe that the noise in the PEM dynamics (4.117) does not in fact enter simply added to the model output, but also enters via the past measurements $\mathrm{y}_{k-1},\dots\mathrm{y}_{k-N_\mathrm{a}}$ which are subject to that noise, and are filtered by the "A" part of the model. The block-diagram above shows that we are in fact assuming in the PEM context that the noise is filtered by $1/A(z)$. It begs then the question: can we build more generic models of the "filter" affecting the noise in the PEM? More specifically, can we build a model for PEM that underlies the generic block-diagram



i.e. we would like to find a PEM-like formulation that corresponds to the Z-transform:

$$Y(z) = \frac{B(z)}{A(z)}U(z) + \frac{C(z)}{D(z)}E(z) \tag{4.121}$$

Unfortunately, the following developments are a bit tricky (and understanding them is not required for the exam), but we present them next for the interested readers. We first rewrite (4.121) as

$$Y(z) = G(z)U(z) + \underbrace{H(z)E(z)}_{:=V(z)} \tag{4.122}$$

In the time domain, we can write $V(z)$ as:

$$\mathrm{v}_k = \sum_{j=0}^{\infty} \mathrm{h}_j \mathrm{e}_{k-j} = \mathrm{e}_k + \sum_{j=1}^{\infty} \mathrm{h}_j \mathrm{e}_{k-j} \tag{4.123}$$

where h is the impulse response corresponding to $H(z)$ and where we assume $\mathrm{h}_0 = 1$ (without loss of generality). At this stage, in order to deliver a prediction $\hat{\mathrm{y}}_k$, we would need a prediction $\hat{\mathrm{v}}_k$ of v_k. It makes sense to write $\hat{\mathrm{v}}_k$ as:

$$\hat{\mathrm{v}}_k = \mathbb{E}\left[\mathrm{v}_k\right] = \mathbb{E}\left[\mathrm{e}_k\right] + \sum_{j=1}^{\infty} \mathrm{h}_j \mathrm{e}_{k-j} = \sum_{j=1}^{\infty} \mathrm{h}_j \mathrm{e}_{k-j} \tag{4.124}$$

hence

$$\hat{\mathrm{v}}_k = \left(\sum_{j=0}^{\infty} \mathrm{h}_j \mathrm{e}_{k-j}\right) - \mathrm{e}_k \tag{4.125}$$

If we convert the latter expression in the Z-domain, we see that

$$\hat{V}(z) = (H(z) - 1)\, E(z) \tag{4.126}$$

As the noise realisations $e_{k-1}, \ldots, e_{-\infty}$ are indirectly known through the past measurements $y_{k-1}, \ldots, y_{-\infty}$ using:

$$E(z) = \frac{1}{H(z)} V(z) \tag{4.127}$$

Hence

$$\hat{V}(z) = \frac{H(z) - 1}{H(z)} V(z) = \left(1 - H(z)^{-1}\right) V(z) \tag{4.128}$$

holds. From there on, we just have some algebraic manipulations to do in the Z domain. Similarly to (4.122), we observe that

$$\begin{aligned} \hat{Y}(z) = G(z)U(z) + \hat{V}(z) &= G(z)U(z) + \left(1 - H(z)^{-1}\right) V(z) \\ &= G(z)U(z) + \left(1 - H(z)^{-1}\right) (Y(z) - G(z)U(z)) \end{aligned} \tag{4.129}$$

Hence we finally get:

$$\hat{Y}(z) = H(z)^{-1} G(z)U(z) + \left(1 - H(z)^{-1}\right) Y(z) \tag{4.130}$$

or equivalently:

$$H(z)\,\hat{Y}(z) = G(z)U(z) + (H(z) - 1)\, Y(z) \tag{4.131}$$

Using $H(z) = \frac{C(z)}{D(z)}$ and $G(z) = \frac{B(z)}{A(z)}$, we can also write:

$$\frac{C(z)}{D(z)} \underbrace{\hat{Y}(z)}_{\text{prediction}} = \frac{B(z)}{A(z)} U(z) + \left(\frac{C(z)}{D(z)} - 1\right) \underbrace{Y(z)}_{\text{measurements}} \tag{4.132}$$

Let us consider two simple examples of application of this formula:

- If we want noise model $\frac{C(z)}{D(z)} = 1$ (output noise), we need to use

$$\underbrace{\hat{Y}(z)}_{\text{prediction}} = \frac{B(z)}{A(z)} U(z) \tag{4.133}$$

  i.e.

$$A(z) \underbrace{\hat{Y}(z)}_{\text{prediction}} = B(z)U(z) \tag{4.134}$$

  This is a pure simulation model (not using past measurements)

- If we want noise model $\frac{C(z)}{D(z)} = \frac{B(z)}{A(z)}$ (input noise), we need to use

$$\frac{B(z)}{A(z)} \underbrace{\hat{Y}(z)}_{\text{prediction}} = \frac{B(z)}{A(z)} U(z) + \left(\frac{B(z)}{A(z)} - 1 \right) \underbrace{Y(z)}_{\text{measurements}}$$

  i.e.

$$\text{i.e.} \qquad B(z) \underbrace{\hat{Y}(z)}_{\text{prediction}} = B(z)U(z) + (B(z) - A(z)) \underbrace{Y(z)}_{\text{measurements}}$$

5 DIFFERENTIAL ALGEBRAIC EQUATIONS (DAES)

We have seen so far formally ordinary differential equations (ODEs), which describe the time evolution of a vector of variables via relationships of the form:

$$\dot{x}(t) = f(x(t), u(t), t) \tag{5.1}$$

where $x \in \mathbb{R}^{n_x}$ is a vector of *differential states* and $u \in \mathbb{R}^{n_u}$ a vector of "external variables", often referred to as inputs (any variable that is not defined by (5.1), but rather "externally" can be classified as an "input") . For the sake of simplicity, the time dependence (t) of these variables is usually omitted.

In this chapter, we will approach a new form of differential equations, labelled *Differential-Algebraic Equations* (DAEs), which are used a lot in the modelling of complex and large-scale systems. DAEs have features and properties that are unlike ODEs. We will investigate them here.

5.1 WHAT ARE DAES?

Simply put, DAEs are a set of equations that do not define directly the entire state. This happens e.g. if we have a mix of differential equations and purely algebraic equations (equations where the time-differentiated variables do not appear) or if the equations mix *differential states*, i.e. variables that appear as time-differentiated in the equations, and *algebraic states*, i.e. variables (excluding inputs and parameters) whose time derivative does not appear at all. Before introducing the formal definition of a DAE, let us make some simple examples.

Example

- Consider the differential equation

$$\dot{x}_1 = 2x_1 + x_2 \tag{5.2a}$$
$$0 = 3x_1 - x_2 \tag{5.2b}$$

  Here we clearly observe that the state variable x_2 does not appear in its time-differentiated form $\dot{x}_2$, such that the equations do not define $\dot{x}_2$, but rather x_2. Put differently, x_2 is not a differential state but an algebraic state. Hence (5.2) is a DAE.

- Consider the differential equation

$$\dot{x}_1 + 2x_1 + \dot{x}_2 + x_2 = 0 \tag{5.3a}$$
$$2\dot{x}_1 + x_1 + 2\dot{x}_2 + 2x_2 = 0 \tag{5.3b}$$

here both variables $\dot{x}_1$ and $\dot{x}_2$ appear time-differentiated. However, one can also observe that replacing (5.3b) by $-2 \cdot (5.3a) + (5.3b)$ yields:

$$\dot{x}_1 + 2x_1 + \dot{x}_2 + x_2 = 0 \tag{5.4a}$$

$$-3x_1 = 0 \tag{5.4b}$$

Here we observe that (5.4) does not define the differential states $\dot{x}_1$ and $\dot{x}_2$ (in the sense that one cannot solve (5.4) to obtain $\dot{x}_1, \dot{x}_2$). However, (5.4) still provides a well-defined trajectory for x_1 and x_2. Indeed, (5.4b) specifies that $x_1 = 0$, such that (5.4a) reads as the simple ODE:

$$\dot{x}_2 + x_2 = 0 \tag{5.5}$$

which can be solved for x_2.

□

In order for the notion of DAE to be rigorous, we need a more formal definition.

Definition 1. *consider a differential equation having the state vector* $s \in \mathbb{R}^{n_s}$*, and defined by the equation*

$$F(\dot{x}, x, u, t) = 0 \tag{5.6}$$

The differential equation (5.6) *is a DAE if*

$$\det\left(\frac{\partial F}{\partial \dot{x}}\right) = 0 \tag{5.7}$$

i.e. if the Jacobian $\frac{\partial F}{\partial \dot{x}}$ *is rank-deficient.*

To clarify this definition, let us test it on (5.2) and (5.3).

Example

- Differential equation (5.2) can be written as

$$F(\dot{x}, x) = \begin{bmatrix} \dot{x}_1 - 2x_1 - x_2 \\ 3x_1 - x_2 \end{bmatrix} = 0 \tag{5.8a}$$

such that

$$\frac{\partial F}{\partial \dot{x}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.9}$$

is rank-deficient because the second column (row) of the Jacobian of F is zero. One can generalize this observation: algebraic states yield columns of zeros in the Jacobian $\frac{\partial F}{\partial \dot{x}}$ and make it rank-deficient.

- Differential equation (5.3) can be written as

$$\mathrm{F}(\dot{\mathrm{x}},\mathrm{x}) = \begin{bmatrix} \dot{\mathrm{x}}_1 + 2\mathrm{x}_1 + \dot{\mathrm{x}}_2 + \mathrm{x}_2 \\ 2\dot{\mathrm{x}}_1 + \mathrm{x}_1 + 2\dot{\mathrm{x}}_2 + 2\mathrm{x}_2 \end{bmatrix} = 0 \tag{5.10}$$

  such that

$$\frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \tag{5.11}$$

  is rank-deficient because its determinant is zero. Here the rank-deficient determinant captures the fact that (5.3) does not define the differential state properly.

□

In practice, most often DAEs are arising because some of the states are purely algebraic (i.e. they do not appear time-differentiated), as in (5.2). In order to stress the difference between differential and algebraic states, it is common to use the notation x for the differential states, and z the algebraic states. E.g. (5.2) would be written as:

$$\dot{\mathrm{x}} = 2\mathrm{x} + \mathrm{z} \tag{5.12a}$$

$$0 = 3\mathrm{x} - \mathrm{z} \tag{5.12b}$$

Definition 1 allows one to assess whether a differential equation is a DAE or an ODE in ambiguous cases. However, one can trivially write differential equations that cannot be categorized so simply. Let us consider two simple examples.

Example Let us investigate two “freak" differential equations that can switch between being ODEs and DAEs

- Consider the simple scalar differential equation

$$u\dot{x} + x = 0 \tag{5.13}$$

  here we observe that $\mathrm{F} = u\dot{x} + x$ and

$$\frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} = u \tag{5.14}$$

  Hence for $u \neq 0$ (5.13) is an ODE and reads as:

$$\dot{x} = -\frac{x}{u} \tag{5.15}$$

  while for $u = 0$ (5.13) is an “DAE" (in fact it is purely algebraic) and reads as:

$$x = 0 \tag{5.16}$$

  Hence (5.13) can switch between being an ODE and a DAE depending on the input u.

- Consider the differential equation

$$\dot{x}_1 + x_1 - u = 0 \tag{5.17a}$$
$$(x_1 - x_2)\dot{x}_2 + x_1 - x_2 = 0 \tag{5.17b}$$

One can verify that

$$\det\left(\frac{\partial F}{\partial \dot{x}}\right) = x_1 - x_2 \tag{5.18}$$

Hence for initial conditions x(0) satisfying $x_1(0) = x_2(0)$, (5.17) starts as a DAE and obeys the dynamics:

$$\dot{x}_1 + x_1 - u = 0 \tag{5.19a}$$
$$x_1 - x_2 = 0 \tag{5.19b}$$

hence it enforces $x_1(t) = x_2(t)$ for all time, and (5.17) remains a DAE. However, if the initial conditions satisfy $x_1(0) \neq x_2(0)$, then (5.17) starts as an ODE and remains an ODE throughout its trajectories.

These examples are meant to draw the attention to the fact that the notion of DAE can be fairly convoluted. In most practical cases, however, DAEs arise because the differential equations hold variables that do not appear time-differentiated. In this context, the problem of DAEs having exotic behaviors as in the examples above does not arise.

□

5.2 Different forms of DAEs

DAEs come in different forms, which are useful to recognize, as these different forms can have different intrinsic properties. We will briefly look at these forms here.

- **Fully-implicit DAEs** are in the form (5.6) of definition 1, i.e. they read as

$$F(\dot{x}, x, u) = 0 \tag{5.20}$$

where

$$\det\left[\frac{\partial F}{\partial \dot{x}}\right] = 0 \tag{5.21}$$

If the rank-deficiency arises because some the states x do not appear as time-differentiated in F (hence creating columns of zeros in the Jacobian $\frac{\partial F}{\partial \dot{x}}$), then they are commonly labelled as “z" states, and (5.20) is rewritten as:

$$F(\dot{x}, x, z, u) = 0 \tag{5.22}$$

Note that here condition (5.21) becomes

$$\det\left(\begin{bmatrix} \frac{\partial F}{\partial \dot{x}} & \frac{\partial F}{\partial \dot{z}} \end{bmatrix}\right) = \det\left(\begin{bmatrix} \frac{\partial F}{\partial \dot{x}} & 0 \end{bmatrix}\right) = 0 \tag{5.23}$$

hence (5.22) is by construction a DAE as $\dot{z}$ is not an argument in F.

- **Semi-explicit DAEs** split explicitly the differential and algebraic equations, and can generally be written in the form:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.24a}$$
$$0 = \mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.24b}$$

A semi-explicit DAE can be trivially written as a fully-implicit DAE by defining:

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{x}, \mathrm{z}, \mathrm{u}) = \begin{bmatrix} \dot{\mathrm{x}} - \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \\ \mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \end{bmatrix} = 0 \tag{5.25}$$

Conversely, a fully implicit can be trivially written as a semi-explicit DAE by introducing some “helper variables" (labelled v here), i.e.

$$\dot{\mathrm{x}} = \mathrm{v} \tag{5.26a}$$
$$0 = \mathrm{F}(\mathrm{v}, \mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.26b}$$

We observe that (5.24a) is a DAE by construction. Indeed, one can apply definition 1 on (5.25) (using the same construction as in (5.23)) and observe that:

$$\det\left(\begin{bmatrix} \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} & \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{z}}} \end{bmatrix}\right) = \det\left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}\right) = 0 \tag{5.27}$$

- **Linear DAEs** are DAEs where the underlying functions are linear. They are often put in the form:

$$E\dot{\mathrm{x}} = A\mathrm{x} + B\mathrm{u} \tag{5.28}$$

where definition 1 requires that E is rank-deficient in order for (5.28) to be a DAE. In a semi-explicit form, and underlying the distinction between algebraic and differential states, a linear DAE would read as:

$$\dot{\mathrm{x}} = A\mathrm{x} + B\mathrm{u} + C\mathrm{z} \tag{5.29a}$$
$$0 = D\mathrm{x} + E\mathrm{u} + F\mathrm{z} \tag{5.29b}$$

For matrix F full rank, we observe that the algebraic variables z can be eliminated using (5.29b) and replaced into (5.29a) such that the DAE can be reduced to an ODE with the addition of some “output variables" z:

$$\dot{\mathrm{x}} = \left(A - CF^{-1}D\right)\mathrm{x} + \left(B - CF^{-1}E\right)\mathrm{u} \tag{5.30a}$$
$$\mathrm{z} = -F^{-1}\left(D\mathrm{x} + E\mathrm{u}\right) \tag{5.30b}$$

Similarly to ODEs, nonlinear DAEs can be linearized into linear DAEs. E.g. a fully-implicit DAE of the form (5.22) can be linearized to

$$\frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}}\Delta\dot{\mathrm{x}} + \frac{\partial \mathrm{F}}{\partial \mathrm{x}}\Delta\mathrm{x} + \frac{\partial \mathrm{F}}{\partial \mathrm{z}}\Delta\mathrm{z} + \frac{\partial \mathrm{F}}{\partial \mathrm{u}}\Delta\mathrm{u} = 0 \tag{5.31}$$

and a semi-explicit DAE of the form can be linearized to

$$\Delta\dot{x} = \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial u}\Delta u + \frac{\partial f}{\partial z}\Delta z \tag{5.32a}$$

$$0 = \frac{\partial g}{\partial x}\Delta x + \frac{\partial g}{\partial u}\Delta u + \frac{\partial g}{\partial z}\Delta z \tag{5.32b}$$

and therefore takes the form (5.29). It is interesting to note here that the elimination of the algebraic variables Δz in (5.32) is possible if the Jacobian $\frac{\partial g}{\partial z}$ is full rank throughout the trajectory. We will get back to this notion soon.

5.3 Tikhonov Theorem

Most physical systems comprise some dynamics that are very fast in comparison with the main dynamics of the system. During the modelling of the system, these very fast dynamics are often disregarded. E.g. when modelling mechanical systems, many components of the system are considered as infinitely rigid, and their dynamics dismissed. For electrical systems, the inductance and capacitance of the cables are often neglected. Experienced engineers typically know which dynamics matter and ought to be included in their model, and which ones can be safely neglected. While the elimination of very fast dynamics is most often performed based on the intuition and experience of the engineer, it has strong foundations in mathematics. The elimination of fast dynamics is one of the reasons why DAEs arise in modelling. In this section we will see how and why eliminating fast dynamics from a model is justified.

Theorem 8. *Consider the ordinary differential equation (ODE):*

$$\dot{x} = f(x, z) \tag{5.33a}$$

$$\epsilon\dot{z} = g(x, z) \tag{5.33b}$$

where $0 < \epsilon << 1$ *is very small. Let us label* $x_\epsilon(t), z_\epsilon(t)$ *the solution to* (5.33). *Suppose:*

- *the dynamics* $\dot{z} = g(x, z)$ *are stable* $\forall\, x$
- *matrix* $\frac{\partial g}{\partial z}$ *is full rank (i.e. invertible) everywhere*

then

$$\lim_{\epsilon \to 0} \quad x_\epsilon(t), z_\epsilon(t) \;=\; x_0(t), z_0(t) \tag{5.34}$$

where $x_0(t), z_0(t)$ *is the solution of*

$$\dot{x} = f(x, z) \tag{5.35a}$$

$$0 = g(x, z) \tag{5.35b}$$

The limit (5.34) is to be understood in the sense of “almost everywhere" formally defined in the context of measure theory. From an intuitive point of view, the trajectories match everywhere but on a union of time intervals that are "infinitely short". An illustration of the Tikhonov theorem is provided in the following example.

Examples consider the linear system

$$\dot{x}_1 = -x_1 + x_2 \tag{5.36a}$$
$$\epsilon \dot{x}_2 = x_1 - a x_2 + u \tag{5.36b}$$

The hypothesis of Theorem 8 are satisfied for (5.36) if $a > 0$, such that $\frac{\partial g}{\partial z} = -a$ is full rank and (5.36b) is stable. The limit case $\epsilon = 0$ reads as

$$\dot{x}_1 = -x_1 + x_2 \tag{5.37a}$$
$$0 = x_1 - a x_2 + u \tag{5.37b}$$

We then distinguish two cases

- if $a \neq 0$ then (5.37) has the solution

$$\dot{x}_1 = (a^{-1} - 1)x_1 + a^{-1} u \tag{5.38a}$$
$$x_2 = a^{-1}(x_1 + u) \tag{5.38b}$$

- if $a = 0$ then (5.37) has the solution

$$\dot{x}_1 = -x_1 + x_2 \tag{5.39a}$$
$$0 = x_1 + u \tag{5.39b}$$

  which equivalently reads:

$$x_2 = -(u + \dot{u}) \tag{5.40a}$$
$$x_1 = -u \tag{5.40b}$$

  DAE (5.39) requires that u is continuous as it appears time-differentiated. This is in contrast with (5.36) for which any integrable input profile is admissible. Additionally, we observe that (5.36) is unstable for $a = 0$ (with a pole at 0), and therefore does not satisfy the hypothesis of Theorem 8. The trajectories are illustrated in Fig. 5.1.

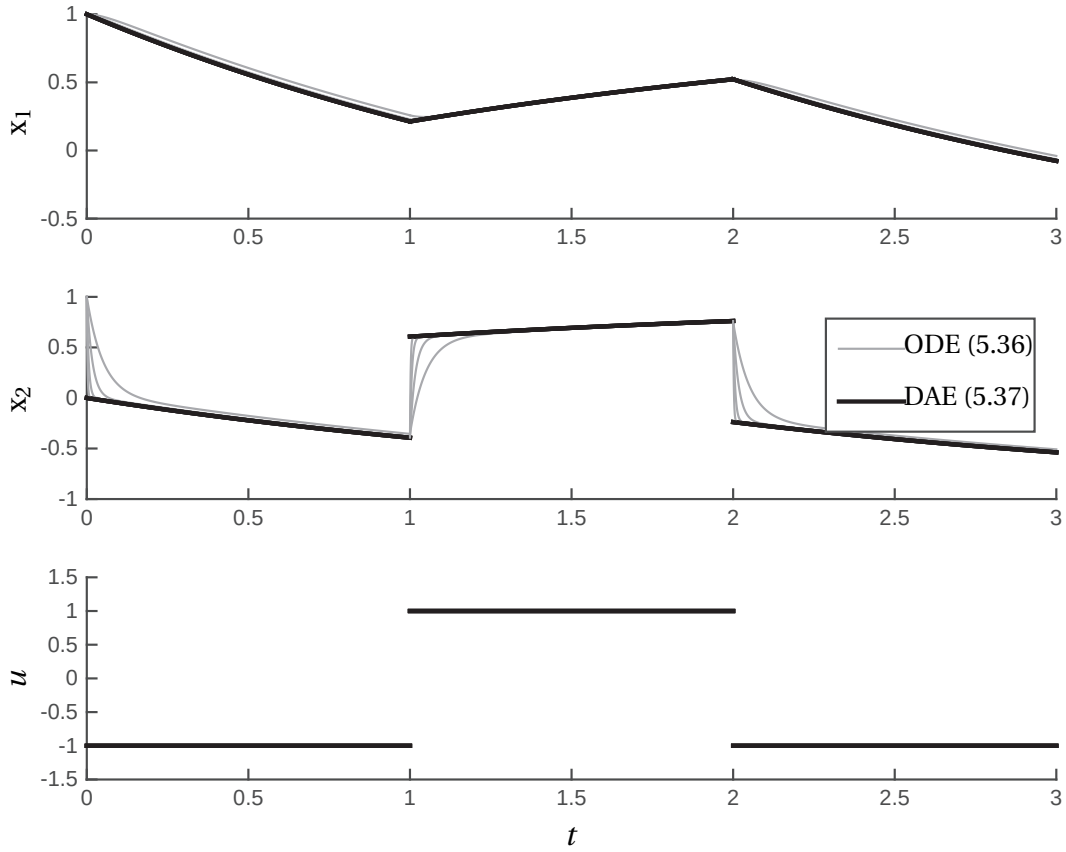


Figure 5.1: Illustration of the Tikhonov theorem for (5.36). The piecewise constant input illustrated in the third graph was selected. The grey curves represent the trajectories for ϵ ranging from 10^{-1} to $4\cdot 10^{-3}$ with the initial conditions $x_1(0) = x_2(0) = 1$ and for $a = 2$. The black curve represents the solution to the DAE (5.37). Here we see that the Tikhonov theorem applies.

- Let us consider two masses moving horizontally (positions x_1 and x_2), subject to a velocity-dependent friction with parameter ρ, and connected by elastic links of constant K. The dynamics of this two-mass system can be modelled via the linear ODE:

$$m_1\ddot{x}_1 = K(x_2 - x_1) - Kx_1 - \rho\dot{x}_1 + u \tag{5.41a}$$
$$m_2\ddot{x}_2 = K(x_1 - x_2) - \rho\dot{x}_2 \tag{5.41b}$$

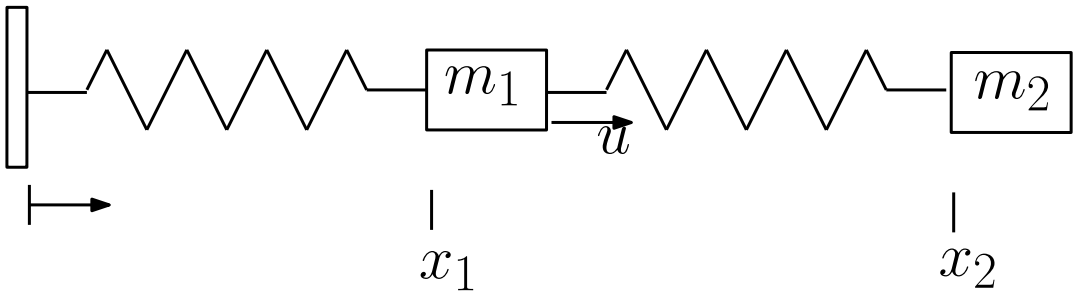


Figure 5.2: Illustration of the two mass system modelled by (5.41).

Let us assume that $m_1 << m_2$ and $\rho << K$, such that the oscillatory dynamics of mass 1 are very fast compared to the one of mass 2. Let us then consider the approximation:

$$0 = K(\mathrm{x}_2 - \mathrm{x}_1) - K\mathrm{x}_1 - \rho\dot{\mathrm{x}}_1 + u \tag{5.42a}$$
$$m_2\ddot{\mathrm{x}}_2 = K(\mathrm{x}_1 - \mathrm{x}_2) - \rho\dot{\mathrm{x}}_2 \tag{5.42b}$$

of (5.41) where m_1 being relatively small has been set to zero. ODE (5.42) is equivalent to:

$$\frac{\rho}{K}\dot{\mathrm{x}}_1 = (\mathrm{x}_2 - \mathrm{x}_1) - \mathrm{x}_1 + u \tag{5.43a}$$
$$m_2\ddot{\mathrm{x}}_2 = K(\mathrm{x}_1 - \mathrm{x}_2) - \rho\dot{\mathrm{x}}_2 \tag{5.43b}$$

Since $\frac{\rho}{K} << 1$, we can form the further approximation:

$$0 = (\mathrm{x}_2 - \mathrm{x}_1) - \mathrm{x}_1 + u \tag{5.44a}$$
$$m_2\ddot{\mathrm{x}}_2 = K(\mathrm{x}_1 - \mathrm{x}_2) - \rho\dot{\mathrm{x}}_2 \tag{5.44b}$$

which is a DAE since state x_2 does not appear time differentiated. We can easily observe that (5.44) boils down to eliminating $\dot{\mathrm{x}}_1, \ddot{\mathrm{x}}_1$ from equation (5.41a). I.e. while we keep considering x_1 as time-varying, we dismiss its dynamics.

Figure 5.3 compares the trajectories of ODE (5.41) and the ones of the DAE (5.44). One can verify that for both applications of the Tikhonov theorem (5.42) and (5.44), the hypothesis apply.

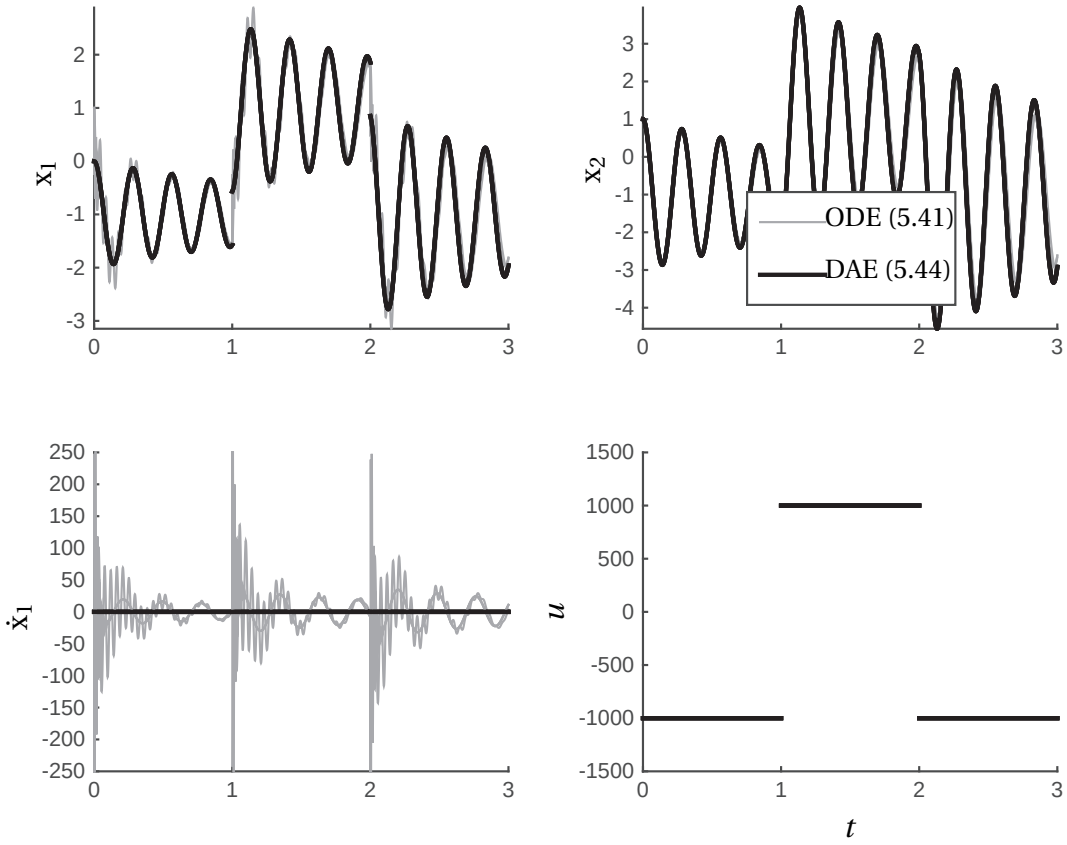


Figure 5.3: Illustration of the Tikhonov theorem for (5.41). The piecewise constant input illustrated in the lower-right graph was selected. The grey curves represent the trajectories for m_1 ranging from 10^{-1} to $4\cdot10^{-4}$ with the initial conditions $x_1(0) = x_2(0) = 1$ and $\dot{x}_{1,2}(0) = 0$. The parameters $\rho = 1$, $K = 10^3$ and $m_2 = 1$ were selected. The black curve represents the solution to the DAE (5.44). Here we see that the Tikhonov theorem applies.

□

To conclude these two examples, it is useful to provide an intuition behind the application of the Tikhonov theorem, which is the basis of the intuition of engineers in neglecting some of the dynamics of a system when developing a mathematical model. The intuition is easiest to build for the two-mass example. One can observe that the approximation at play here is essentially to replace the dynamics of mass 1 by its steady-state approximation. I.e. we consider that the motion of the mass decays instantaneously to its steady state provided by (5.44a). One should understand here that “steady-state" does not mean “not moving", as e.g. (5.44a) still allows x_1 to move. However, in the approximation (5.44) the motion of x_1 is entirely dictated by the motion of x_2 moving x_1 in different steady-state positions. This principle can very often be readily applied to models holding very fast dynamics. One ought to observe that the dynamics created by the DAE (5.44) do not hold very fast dynamics (as opposed to e.g. (5.42) which still hold a very fast dynamics for x_1). This observation

will connect to Sections 6.4.

5.4 Differential Index of DAEs

DAEs are unlike ODEs in various ways. One very important distinction is that DAEs hold the concept of differential index, which is crucial when it comes to solving them numerically. While the notion of differential index is not straightforward, the problem of high-index DAE, which is the main focus of this section, can be construed fairly intuitively.

One ought to understand that in order for a DAE e.g. in the semi-explicit form:

$$\dot{x} = f(x, z, u) \tag{5.45a}$$

$$0 = g(x, z, u) \tag{5.45b}$$

to be "solvable", one need to be able to compute the differential state derivative $\dot{x}$ and the the algebraic states z for a given differential state x and input u. Indeed, $\dot{x}$, z can be readily obtained (possibly numerically) from (5.45) for any given x, u, then one can process the system trajectories and build the trajectories corresponding to (5.45). In order to build this intuition, let us consider the following example.

Example

- Let us start with a simple linear DAE:

$$F = \begin{bmatrix} \dot{x}_1 - z \\ \dot{x}_2 - x_1 \\ \dot{x}_1 + x_2 - u \end{bmatrix} = 0 \tag{5.46}$$

  One can verify that, using simple algebraic manipulations, (5.46) can be rewritten as

$$z = u - x_2 \tag{5.47a}$$

$$\dot{x}_2 = x_1 \tag{5.47b}$$

$$\dot{x}_1 = u - x_2 \tag{5.47c}$$

  i.e. (5.46) delivers $\dot{x}$ and z as a function of x and u.

- Consider a linear DAE similar to but slightly different than (5.46)

$$F = \begin{bmatrix} \dot{x}_1 - z \\ \dot{x}_2 - x_1 \\ x_2 - u \end{bmatrix} = 0 \tag{5.48}$$

  One can verify that no algebraic manipulation can deliver $\dot{x}$ and z from (5.48). Indeed, (5.48) also reads as:

$$\underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{:=M} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ x_1 \\ x_2 - u \end{bmatrix} \tag{5.49}$$

and since matrix M is rank deficient (last row is zero), one cannot manipulate the equations defined by (5.48) in order to extract $\dot{x}$ and z. In contrast, we observe that (5.46) can be rewritten as

$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ x_2 - u \end{bmatrix} \tag{5.50}$$

and its matrix is full rank, allowing us to build (5.47). We ought to observe that the matrix M is obtained by taking the Jacobian

$$\begin{bmatrix} \frac{\partial F}{\partial \dot{x}} & \frac{\partial F}{\partial z} \end{bmatrix} \tag{5.51}$$

which need to be full rank in order for the DAEs to be solvable for $\dot{x}$ and z.

□

The principles deployed in the examples above can be generalized to nonlinear DAEs, as is stated next.

Theorem 9. *a fully implicit DAE*

$$F(\dot{x}, z, x, u) = 0 \tag{5.52}$$

with function F *smooth can be readily solved (i.e. solved for* $\dot{x}$, z*) if the Jacobian*

$$\begin{bmatrix} \frac{\partial F}{\partial \dot{x}} & \frac{\partial F}{\partial z} \end{bmatrix} \tag{5.53}$$

is full rank on all trajectories $\dot{x}, z, x, u$.

Proof. this theorem follows directly from the IFT (see Theorem 5). Indeed, for (5.53) full rank, there exist functions $\dot{x}(x,u)$ and $z(x,u)$ such that

$$F(\dot{x}(x,u), z(x,u), x, u) = 0 \tag{5.54}$$

hold locally for any x, u. It follows that (5.52) can be solved. Another way of construing this result is that if the Jacobian (5.53) is full rank, then a Newton iteration deployed on (5.52) in order to compute $\dot{x}, z$ converges locally (because the linear system (3.8) is well-posed). □

One can readily apply Theorem 9 to semi-explicit DAEs.

Corollary 1. *a semi-explicit DAE*

$$\dot{x} = f(x, z, u) \tag{5.55a}$$

$$0 = g(x, z, u) \tag{5.55b}$$

with function g *smooth can be readily solved (i.e. solved for* $\dot{x}$, z*) if the Jacobian*

$$\frac{\partial g}{\partial z} \tag{5.56}$$

is full rank on all trajectories z, x, u.

Proof. recall that a semi-explicit DAE can be transformed into a full implicit one via the transformation (5.25) recalled here:

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{x}, \mathrm{z}, \mathrm{u}) = \begin{bmatrix} \dot{\mathrm{x}} - \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \\ \mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \end{bmatrix} = 0 \tag{5.57}$$

A direct application of Theorem 9 then yields the Jacobian:

$$\begin{bmatrix} \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} & \frac{\partial \mathrm{F}}{\partial \mathrm{z}} \end{bmatrix} = \begin{bmatrix} I & \frac{\partial \mathrm{f}}{\partial \mathrm{z}} \\ 0 & \frac{\partial \mathrm{g}}{\partial \mathrm{z}} \end{bmatrix} \tag{5.58}$$

which is full rank if $\frac{\partial \mathrm{g}}{\partial \mathrm{z}}$ is full rank. □

An intuitive way of construing Corollary 1 is by observing that if the Jacobian (5.56) is full rank, then the algebraic equation (5.55b) can be solved for z at any point x, u (see IFT, Theorem 5). The solution z can then be injected in (5.55a) to obtain $\dot{\mathrm{x}}$.

Theorem 9 and its corollary 1 tell us that there are DAE that are “easy" to solve (those satisfying the full rankness of the Jacobians (5.53) and (5.56)) and DAEs that are “hard" to solve (those where the Jacobians (5.53) and (5.56) are rank deficient). We will see next that Theorem 9 and its corollary 1 have to do with the differential index of DAEs.

Definition 2. *the differential index of a DAE is the number of times the operator $\frac{\mathrm{d}}{\mathrm{d}t}$ must be applied to the equations (+ possibly an arbitrary amount of algebraic manipulations) in order to convert the DAE into an ODE.*

Definition 2 is non-trivial and can require some care when being applied. Let us make a few examples in order to gather some intuitions on how it works.

Example

- Consider the linear DAE (5.47) repeated here:

$$\mathrm{z} = u - \mathrm{x}_2 \tag{5.59a}$$
$$\dot{\mathrm{x}}_2 = \mathrm{x}_1 \tag{5.59b}$$
$$\dot{\mathrm{x}}_1 = u - \mathrm{x}_2 \tag{5.59c}$$

  Recall that (5.47) is an “easy" DAE since it satisfies the assumption of Theorem 9. We also observe that a single application of $\frac{\mathrm{d}}{\mathrm{d}t}$ on (5.59a) yields:

$$\dot{\mathrm{z}} = \dot{u} - \dot{\mathrm{x}}_2 \tag{5.60a}$$
$$\dot{\mathrm{x}}_2 = \mathrm{x}_1 \tag{5.60b}$$
$$\dot{\mathrm{x}}_1 = u - \mathrm{x}_2 \tag{5.60c}$$

or equivalently

$$\dot{z} = \dot{u} - x_1 \tag{5.61a}$$
$$\dot{x}_2 = x_1 \tag{5.61b}$$
$$\dot{x}_1 = u - x_2 \tag{5.61c}$$

The latter differential equation is an ODE, hence (5.47) is of index 1. It is interesting to note here that by performing the transformation $DAE \rightarrow ODE$ (from (5.47) to (5.61)), we have restricted the functional space for u. Indeed, u only needs to be integrable in (5.47), while it needs to be differentiable in (5.61).

- Consider the linear DAE (5.48), repeated here

$$\dot{x}_1 - z = 0 \tag{5.62a}$$
$$\dot{x}_2 - x_1 = 0 \tag{5.62b}$$
$$x_2 - u = 0 \tag{5.62c}$$

Recall that (5.48) is a “hard" DAE since it fails the assumption of Theorem 9. A time differentiation of (5.48) (last row) yields

$$\dot{x}_1 - z = 0 \tag{5.63a}$$
$$\dot{x}_2 - x_1 = 0 \tag{5.63b}$$
$$\dot{x}_2 - \dot{u} = 0 \tag{5.63c}$$

An algebraic manipulation yields

$$\dot{x}_1 - z = 0 \tag{5.64a}$$
$$\dot{u} - x_1 = 0 \tag{5.64b}$$
$$\dot{x}_2 - \dot{u} = 0 \tag{5.64c}$$

A second time differentiation applied on (5.64b) yields

$$\dot{x}_1 - z = 0 \tag{5.65a}$$
$$\ddot{u} - \dot{x}_1 = 0 \tag{5.65b}$$
$$\dot{x}_2 - \dot{u} = 0 \tag{5.65c}$$

and an algebraic manipulation yields:

$$\ddot{u} - z = 0 \tag{5.66a}$$
$$\ddot{u} - \dot{x}_1 = 0 \tag{5.66b}$$
$$\dot{x}_2 - \dot{u} = 0 \tag{5.66c}$$

A third time differentiation applied to (5.66a) yields the ODE

$$\dot{z} = \dddot{u} \tag{5.67a}$$
$$\dot{x}_1 = \ddot{u} \tag{5.67b}$$
$$\dot{x}_2 = \dot{u} \tag{5.67c}$$

A total count of 3 time differentiations was used to transform the DAE (5.48) to the ODE (5.67). It follows that DAE (5.48) is of index 3.

□

Note that the same principles can be applied to nonlinear DAEs (see e.g. section 5.5 below).

Let us make now the connection between the concept of differential index and Theorem 9 and its corollary 9.

Theorem 10. *a fully-implicit DAE*

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{z}, \mathrm{x}, \mathrm{u}) = 0 \tag{5.68}$$

fulfils the assumption of Theorem 9 if and only if it is of index 1.

Proof. for an index-1 fully-implicit DAE, a single time differentiation allows a transformation to an ODE, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{F} = \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}}\ddot{\mathrm{x}} + \frac{\partial \mathrm{F}}{\partial \mathrm{x}}\dot{\mathrm{x}} + \frac{\partial \mathrm{F}}{\partial \mathrm{z}}\dot{\mathrm{z}} + \frac{\partial \mathrm{F}}{\partial \mathrm{u}}\dot{\mathrm{u}} = 0 \tag{5.69}$$

is an ODE. For the sake of clarity, let us introduce the state extension $\mathrm{v} = \dot{\mathrm{x}}$, such that (5.69) reads as:

$$\frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}}\dot{\mathrm{v}} + \frac{\partial \mathrm{F}}{\partial \mathrm{x}}\mathrm{v} + \frac{\partial \mathrm{F}}{\partial \mathrm{z}}\dot{\mathrm{z}} + \frac{\partial \mathrm{F}}{\partial \mathrm{u}}\dot{\mathrm{u}} = 0 \tag{5.70}$$

or equivalently (5.69) reads as

$$\begin{bmatrix} \dot{\mathrm{v}} \\ \dot{\mathrm{z}} \end{bmatrix} = -\begin{bmatrix} \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} & \frac{\partial \mathrm{F}}{\partial \mathrm{z}} \end{bmatrix}^{-1} \left(\frac{\partial \mathrm{F}}{\partial \mathrm{x}}\mathrm{v} + \frac{\partial \mathrm{F}}{\partial \mathrm{u}}\dot{\mathrm{u}} \right) \tag{5.71}$$

and the Jacobian matrix $\begin{bmatrix} \frac{\partial \mathrm{F}}{\partial \dot{\mathrm{x}}} & \frac{\partial \mathrm{F}}{\partial \mathrm{z}} \end{bmatrix}$ must be full rank. □

The same result extends to semi-explicit DAEs, i.e.

Corollary 2. *a semi-explicit DAE*

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.72a}$$

$$0 = \mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.72b}$$

fulfils the assumption of Theorem 1 if and only if it is of index 1.

Proof. if (5.72) is of index 1, then a single time differentiation on the algebraic equation yields an ODE, i.e.

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.73a}$$

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.73b}$$

is an ODE, i.e. equivalently, using a chain rule

$$\dot{x} = f(x, z, u) \tag{5.74a}$$

$$0 = \frac{\partial g}{\partial x}\dot{x} + \frac{\partial g}{\partial z}\dot{z} + \frac{\partial g}{\partial u}\dot{u} \tag{5.74b}$$

is an ODE. It follows that

$$\dot{x} = f(x, z, u) \tag{5.75a}$$

$$\dot{z} = -\frac{\partial g}{\partial z}^{-1}\left(\frac{\partial g}{\partial x}\dot{x} + \frac{\partial g}{\partial u}\dot{u}\right) \tag{5.75b}$$

is an ODE, which requires the Jacobian $\frac{\partial g}{\partial z}$ to be full rank. □

The message to take home from Theorem 10 and corollary 2 is that index-1 DAEs are “easy" to solve in the sense that the equations readily deliver $\dot{x}$ and z. DAEs of index more than 1 are generally referred to as “high-index" DAEs, and are notoriously more difficult to handle. When approaching DAEs numerically, index-1 DAEs are clearly preferred. This does not mean than high-index DAEs cannot be treated, but they are often best treated via a so-called index-reduction procedure, which we will introduce in Section 5.5 and further discuss in Section 5.6.

5.5 Connection to Lagrange mechanics

An attentive reader will have noticed that we have already approached DAEs in Section 2.5, when discussing constrained Lagrange mechanics. Indeed equation (2.161) stemming from constrained Lagrange mechanics:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = Q \tag{5.76a}$$

$$c(q) = 0 \tag{5.76b}$$

is in fact a semi-explicit DAE, where (5.76b) is the algebraic equation and (5.76a) yields (after minor treatments) an explicit ODE.

We have seen that (5.76) does not readily deliver $\ddot{q}$, z, and that it ought to be modified by time-differentiating the constraint equation (5.76b) twice, delivering the model (2.166) recalled here:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = Q \tag{5.77a}$$

$$\ddot{c}(q, \dot{q}, \ddot{q}) = 0 \tag{5.77b}$$

It is interesting here to verify the differential index of (5.76) by applying definition 2. The differential state for a mechanical system reads as

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{5.78}$$

The transformed Lagrange equation (5.77) can be written explicitly as:

$$\begin{bmatrix} I & 0 & 0 \\ 0 & W(\mathrm{q}) & 0 \\ 0 & 0 & M(\mathrm{q}) \end{bmatrix} \begin{bmatrix} \dot{\mathrm{x}} \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} \dot{\mathrm{q}} \\ \mathrm{Q} - \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q})\,\dot{\mathrm{q}} \right) \dot{\mathrm{q}} + \nabla_{\mathrm{q}} \mathscr{L} \\ -\frac{\partial}{\partial \mathrm{q}} \left(\frac{\partial \mathrm{c}}{\partial \mathrm{q}} \dot{\mathrm{q}} \right) \dot{\mathrm{q}} \end{bmatrix} \tag{5.79}$$

One can observe that (5.79) readily delivers $\dot{\mathrm{x}}$, z for $W(\mathrm{q})\,M(\mathrm{q})$ full rank. It follows that (5.79) fulfils the assumption of Theorem 9, and is therefore of index 1. Since (5.79) has been obtained after two time differentiations of (5.76), it follows that (5.76) is an **index-3** DAE. We can summarize this observation as:

$$\underbrace{(5.76\mathrm{b})}_{\text{index 3}} \xrightarrow{\frac{\mathrm{d}^2}{\mathrm{d}t^2}} \underbrace{(5.77\mathrm{b})}_{\text{index 1}} \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}t}} \text{ODE}$$

This observation is generic, i.e. models arising from constrained Lagrange mechanics with position-dependent constraints of the form $\mathrm{c}(\mathrm{q}) = 0$ yield index-3 DAEs, and two time differentiations of the constraints $\mathrm{c}(\mathrm{q}) = 0$ yield an index-1 DAE.

The important message to take home here is that a high-index DAE such as (5.76) can be transformed into an index-1 DAE (i.e. (5.77)) via time differentiations and algebraic manipulations. The trick we have explored to transform the DAEs arising from constrained Lagrange mechanics into low-index DAEs is not limited to this special case, but it a generic approach to transform “hard" (i.e. high-index) DAEs into “easy" ones (i.e. low-index DAEs), commonly referred to as an **index-reduction** procedure. We further detail this in the next section.

5.6 Index reduction

The index reduction of DAEs is identical to assessing their index as per definition 2, with the minor difference that the procedure ought to be stopped one step before reaching an ODE (i.e. at the index-1 step). More specifically, one ought to perform time differentiations and algebraic manipulations until an index-1 DAE emerges. It is difficult to automate this procedure in general (although some software packages offer index-reduction capabilities), as the algebraic manipulations to be performed typically require some insights into the DAE. However, we can attempt a detailed description of the procedure.

Let us consider a semi-explicit DAE of the form

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.80a}$$

$$0 = \mathrm{g}(\mathrm{x}, \mathrm{z}, \mathrm{u}) \tag{5.80b}$$

The index-reduction procedure can be described as follows.

1. Check if the DAE system is of index 1. If yes, stop.

2. Identify a subset of algebraic equations in (5.80b) that can be solved for a subset of algebraic variables z.
3. Apply $\frac{\mathrm{d}}{\mathrm{d}t}$ on the remaining algebraic equations containing some differential states x_j, this leads to terms $\dot{\mathrm{x}}_j$ appearing in these differentiated equations.
4. Substitute the terms $\dot{\mathrm{x}}_j$ by the corresponding expressions $\mathrm{f}_j(\mathrm{x},\mathrm{z},\mathrm{u})$, this delivers new algebraic equations to replace those differentiated in (5.80b).
5. With this new DAE system, go to step 1.

This procedure is not always straightforward to deploy. Let us consider an example to understand it better.

Example

- consider the linear semi-explicit DAE

$$\dot{\mathrm{x}} = \begin{bmatrix} \mathrm{x}_2 + \mathrm{z}_2 \\ \mathrm{z}_1 + u \end{bmatrix} \equiv \mathrm{f} \tag{5.81a}$$

$$0 = \begin{bmatrix} \mathrm{x}_1 - \mathrm{x}_2 \\ \mathrm{z}_2 - \mathrm{x}_1 \end{bmatrix} \equiv \mathrm{g} \tag{5.81b}$$

We observe that

$$\frac{\partial \mathrm{g}}{\partial \mathrm{z}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.82}$$

is rank deficient, such that (5.81) is of index > 1 (see Corollary 2). We observe that the second equation of (5.81b) delivers z_2 as:

$$\mathrm{z}_2 = \mathrm{x}_1 \tag{5.83}$$

We then apply $\frac{\mathrm{d}}{\mathrm{d}t}$ on the first equation in (5.81b) to obtain:

$$\dot{\mathrm{x}}_1 - \dot{\mathrm{x}}_2 = 0 \tag{5.84}$$

We use (5.81a) to replace $\dot{\mathrm{x}}_{1,2}$ by their corresponding expressions, delivering a new algebraic equation:

$$\mathrm{x}_2 + \mathrm{z}_2 - \mathrm{z}_1 - u = 0 \tag{5.85}$$

We obtain the new DAE where the differentiated algebraic equation has been replaced by (5.85):

$$\dot{\mathrm{x}} = \begin{bmatrix} \mathrm{x}_2 + \mathrm{z}_2 \\ \mathrm{z}_1 + u \end{bmatrix} \tag{5.86a}$$

$$0 = \begin{bmatrix} \mathrm{x}_2 + \mathrm{z}_2 - \mathrm{z}_1 - u \\ \mathrm{z}_2 - \mathrm{x}_1 \end{bmatrix} := \tilde{\mathrm{g}} \tag{5.86b}$$

We observe that

$$\frac{\partial \tilde{\mathrm{g}}}{\partial \mathrm{z}} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \tag{5.87}$$

is full rank, such that (5.86) is of index 1. This concludes the procedure. We can additionally deduce from the index reduction that (5.81) is of index 2.

- Consider the semi-explicit DAE given by:

$$\dot{\mathrm{x}} = A\mathrm{x} - \mathrm{b}z \tag{5.88a}$$

$$0 = \frac{1}{2}\left(\mathrm{x}^\top \mathrm{x} - 1\right) \tag{5.88b}$$

for some matrices A and vector b, and where $z \in \mathbb{R}$. Here the algebraic part is:

$$\mathrm{g}(\mathrm{x}) = \frac{1}{2}\left(\mathrm{x}^\top \mathrm{x} - 1\right) \tag{5.89}$$

such that $\frac{\mathrm{g}(\mathrm{x})}{\partial \mathrm{z}} = 0$ is rank deficient by construction. We then take the time derivative of g(x):

$$\tilde{\mathrm{g}}(\mathrm{x}, \mathrm{z}) = \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{g}(\mathrm{x}) = \mathrm{x}^\top \dot{\mathrm{x}} = \mathrm{x}^\top \left(A\mathrm{x} - \mathrm{b}z\right) = 0 \tag{5.90}$$

We observe that

$$\frac{\partial \tilde{\mathrm{g}}(\mathrm{x}, \mathrm{z})}{\partial z} = -\mathrm{x}^\top \mathrm{b} \tag{5.91}$$

is full rank if $\mathrm{x}^\top \mathrm{b} \neq 0$. Note that z is here readily delivered by $\tilde{\mathrm{g}}(\mathrm{x}, \mathrm{z}) = 0$. Indeed:

$$z = \frac{\mathrm{x}^\top A\mathrm{x}}{\mathrm{x}^\top \mathrm{b}} \tag{5.92}$$

The index-reduced DAE then reads as:

$$\dot{\mathrm{x}} = A\mathrm{x} - \mathrm{b}z \tag{5.93a}$$

$$0 = \mathrm{x}^\top \left(A\mathrm{x} - \mathrm{b}z\right) \tag{5.93b}$$

□

Similarly to Lagrange mechanics, the reduction of the index of a DAE requires one to collect a set of consistency conditions that need to be enforced in order for the resulting index-1 DAE to match the original one. Generally speaking, when performing the index reduction, one ought to **collect all the algebraic equations on which a time differentiation is performed**, and add them to the list of consistency conditions. E.g. in the example above, the first equation in (5.81b) is time-differentiated and is therefore the (only) consistency condition.

Similarly to the Lagrange context, consistency conditions can drift numerically over long simulations, and ought to be corrected if needed. Describing the introduction of a Baumgarte stabilization in generic index-reduced in a general form can be difficult, and we will leave this question out of these notes. In simple cases (such as e.g. the Lagrange context), the principle is fairly simple.

6 EXPLICIT INTEGRATION METHODS - RUNGE-KUTTA

For the remainder of these notes we will discuss the numerical treatment of differential equations. Simulating a system in e.g. the state-space form:

$$\dot{x} = f(x, u), \qquad x(0) = x_0 \tag{6.1}$$

over a time interval $[0, T]$ consists essentially in computing a discrete sequence of state vectors

$$x_0, \ldots, x_N \tag{6.2}$$

that approximate the true and continuous trajectories $x(t),\ t \in [0, T]$ solution of (6.1) on a given time grid $t_0, \ldots, t_N$ sufficiently accurately that they are useful for whichever goal we have to tackle, i.e. we want e.g. the approximation:

$$\|x_k - x(t_k)\| \leq \epsilon, \qquad k = 0, \ldots, N \tag{6.3}$$

to hold. For most model equations f, the sequence (6.2) can only be built numerically in a computer. For the remainder of the course, we will focus on understanding some (non-exhaustive) modern methods to build a sequence (6.2) of simulated states that are “reasonably close" to the true model trajectories.

6.1 EXPLICIT EULER

In order to get started on integration methods, it is natural to start with the most intuitive, yet *least efficient* integration method, namely the explicit Euler method. Explicit Euler ought to be avoided whenever efficiency[10] matters. The explicit Euler method is, however, deceivingly simple to implement and therefore sometimes a good choice when efficiency ought to be sacrificed to minimize the coding effort.

The explicit Euler method uses the rule:

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, u_k), \qquad k = 0, \ldots, N \tag{6.4}$$

where $u_k = u(t_k)$, starting from the given initial conditions x_0. The Euler step is illustrated in Figure 6.1, and is essentially linearly extrapolating from the model dynamics $f(x_k, u_k)$ in order to build the next discrete state x_{k+1} from the current one x_k.

[10] i.e. the amount of computations needed to reach a given accuracy

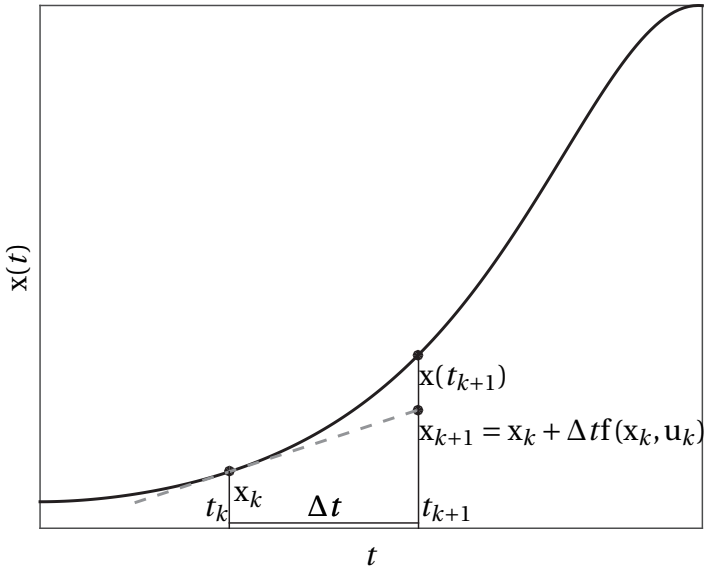


Figure 6.1: Illustration of the principle underlying the explicit Euler scheme (6.4)

It ought to be intuitively clear here that the larger the time step Δt is chosen, the larger the discrepancy between the true model trajectories $\mathrm{x}(t)$ and the simulated ones $\mathrm{x}_0, \ldots, \mathrm{x}_N$ are. Indeed, Euler is essentially ignoring the "curvature" of the model trajectories between the discrete sates and replacing it via a line. The longer the step, the "more curvature" is ignored.

6.1.1 Accuracy of the explicit Euler method

Is is useful to formally analyse the discrepancy between the simulated states and the true trajectories, and relate them to the choice and parameters (e.g. Δt) of the integration method. In order to do that, it is very useful to compute the **one-step error**, i.e. assuming that $\mathrm{x}_k = \mathrm{x}(t_k)$ (i.e. the simulation is exact at time t_k), then what is the error at time t_{k+1}, i.e. what is $\|\mathrm{x}_{k+1} - \mathrm{x}(t_{k+1})\|$? This can be done via Taylor arguments. Indeed, we observe that:

$$\mathrm{x}(t_{k+1}) = \underbrace{\mathrm{x}(t_k) + \Delta t \cdot \dot{\mathrm{x}}(t_k)}_{=\mathrm{x}_{k+1}} + \frac{1}{2}\Delta t^2 \ddot{\mathrm{x}}(\xi) \tag{6.5}$$

for some $\xi \in [t_k,\, t_{k+1}]$ (this is the Taylor expansion with an explicit remainder). It follows that:

$$\|\mathrm{x}_{k+1} - \mathrm{x}(t_{k+1})\| = \frac{\Delta t^2}{2}\|\ddot{\mathrm{x}}(\xi)\| \leq \max_{\xi \in [t_k\, t_{k+1}]} \frac{\Delta t^2}{2}\|\ddot{\mathrm{x}}(\xi)\| \tag{6.6}$$

Two remarks can be drawn from this expression:

- The one-step error is of the order Δt^2

- The one-step error is worse when $\ddot{\mathrm{x}}$ is large, i.e. if the model trajectory is more “curved".

Using this result we can then discuss the **global error** of the explicit Euler integration method, i.e. what is the final integration error $\|\mathrm{x}_N - \mathrm{x}(T)\|$? We can answer that question almost directly from (6.6). Indeed, we make the following observations:

- An integration up to time T with a time step Δt requires $N = \frac{T}{\Delta t}$ Euler steps
- Each step yields an error of order Δt^2
- After N step, the error will then be of the order $N\Delta t^2 = \frac{T}{\Delta t}\Delta t^2 = T\Delta t$
- After N step, the simulation error $\|\mathrm{x}_N - \mathrm{x}(T)\|$ will be of the order Δt. We will use the notation

$$\|\mathrm{x}_N - \mathrm{x}(T)\| = \mathscr{O}(\Delta t) \tag{6.7}$$

  throughout the notes, which formally means:

$$\|\mathrm{x}_N - \mathrm{x}(T)\| \leq c\Delta t \tag{6.8}$$

  holds for some constant $c > 0$ and for Δt small enough.

We illustrate next these principle on the integration of the following dynamics:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) = \begin{bmatrix} \sigma(\mathrm{x}_2 - \mathrm{x}_1) \\ \mathrm{x}_1(\rho - \mathrm{x}_3) - \mathrm{x}_2 \\ \mathrm{x}_1\mathrm{x}_2 - \beta\mathrm{x}_3 \end{bmatrix} \tag{6.9}$$

for $\sigma = 10$, $\beta = 3$, $\rho = 28$ and starting at the initial conditions $\mathrm{x}_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top$. Figures 6.2-6.4 illustrate the model trajectories obtained from the explicit Euler scheme (6.4) for different step sizes Δt, and figure 6.5 report the global error, i.e. the error observed at the end of the simulation.

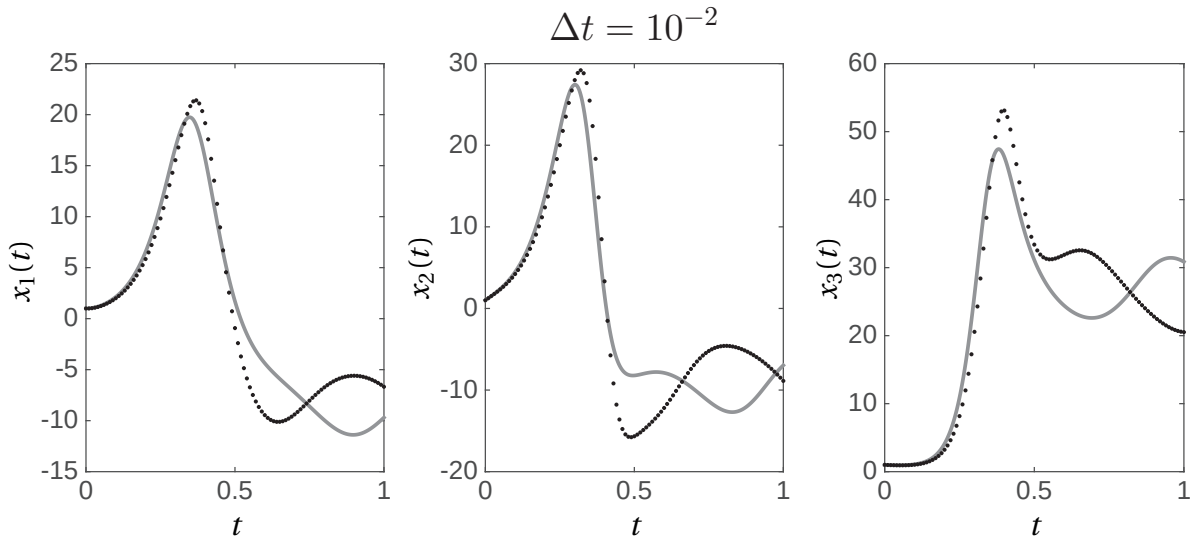


Figure 6.2: Numerical integration of the equation (6.9). In dark the trajectories from the explicit Euler scheme (6.4) using $\Delta t = 10^{-2}$, and in grey the trajectories obtained using a very high accuracy integration method.

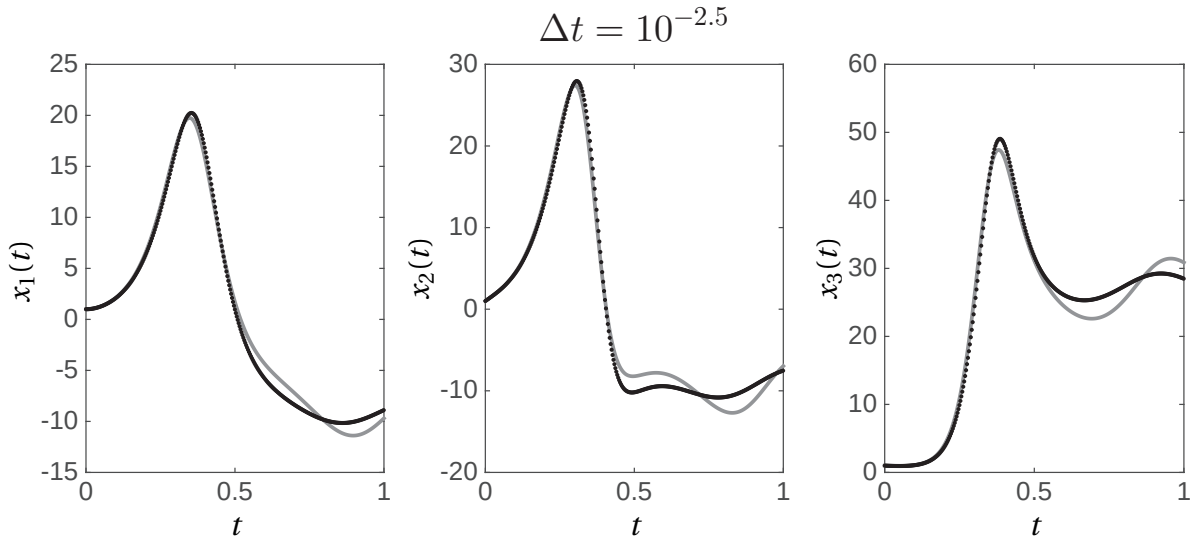


Figure 6.3: Numerical integration of the equation (6.9). In dark the trajectories from the explicit Euler scheme (6.4) using $\Delta t = 10^{-2.5}$, and in grey the trajectories obtained using a very high accuracy integration method.

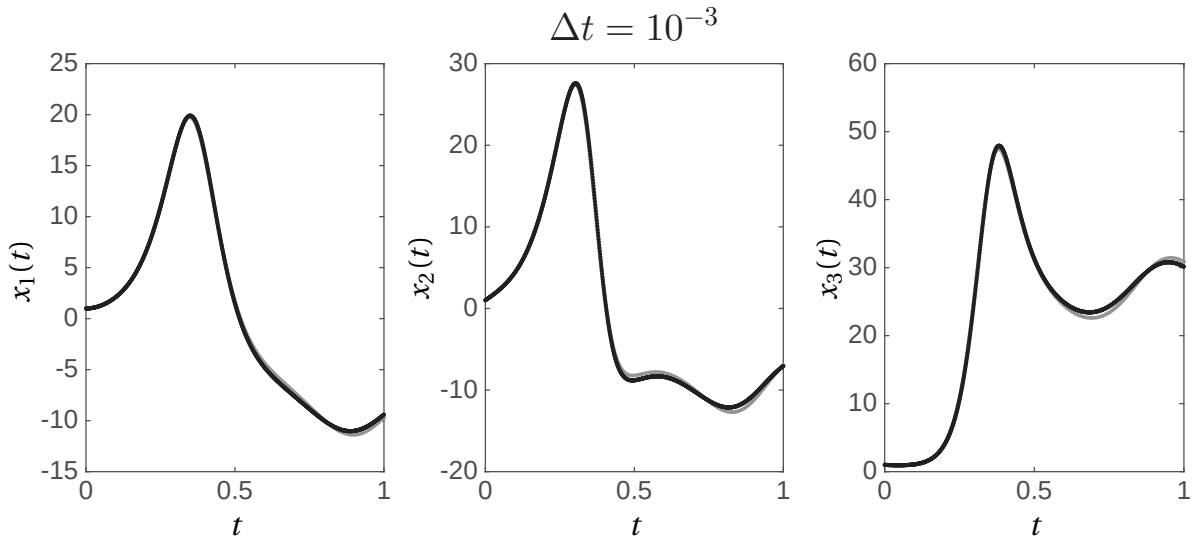


Figure 6.4: Numerical integration of the equation (6.9). In dark the trajectories from the explicit Euler scheme (6.4) using $\Delta t = 10^{-3}$, and in grey the trajectories obtained using a very high accuracy integration method.

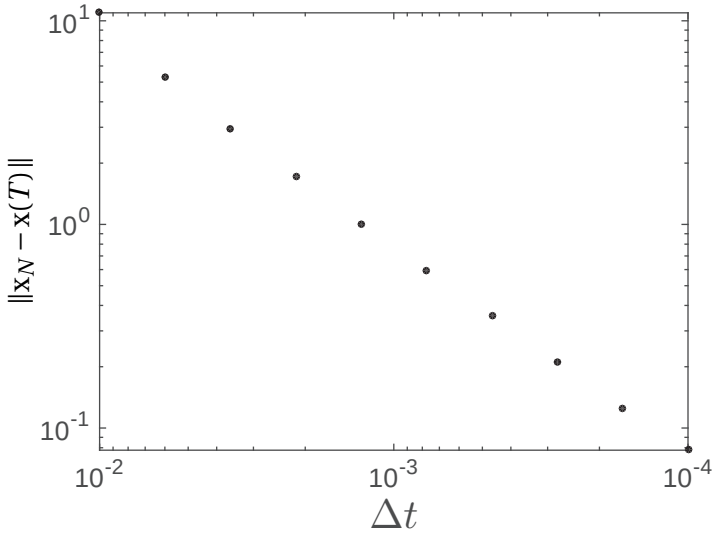


Figure 6.5: Global error vs. Δt using the explicit Euler scheme (6.4).

6.1.2 Stability of the explicit Euler method

Beyond the accuracy of integration methods, another crucial aspect to discuss is their **stability**. In order to introduce this concept, let us consider the trivial stable scalar linear system:

$$\dot{x} = -\lambda x, \qquad x(0) = x_0 \tag{6.10}$$

for $\lambda > 0$ and arbitrary initial conditions x_0. The dynamics (6.10) are linear and have therefore the explicit solution:

$$x(t) = x_0 e^{-\lambda t} \tag{6.11}$$

Let us nonetheless consider deploying the explicit Euler method on (6.10). The Euler steps then read as:

$$x_{k+1} = x_k - \lambda \Delta t x_k = (1 - \lambda \Delta t)\, x_k \tag{6.12}$$

We observe that (6.12) is a linear discrete dynamic system, and that it becomes **unstable** for $|1 - \lambda \Delta t| > 1$. Since λ, $\Delta t > 0$, this happens if $1 - \lambda \Delta t < -1$, i.e. if

$$\Delta t > \frac{2}{\lambda} \tag{6.13}$$

In other words, the **explicit Euler method delivers an unstable simulation** for the dynamics (6.10) **when the time step Δt is too large** compared to the pole of the dynamics λ. We need to stress here that the dynamics (6.10) are stable, only their numerical simulation is possibly unstable. We illustrate these observations in Fig. 6.6 below.

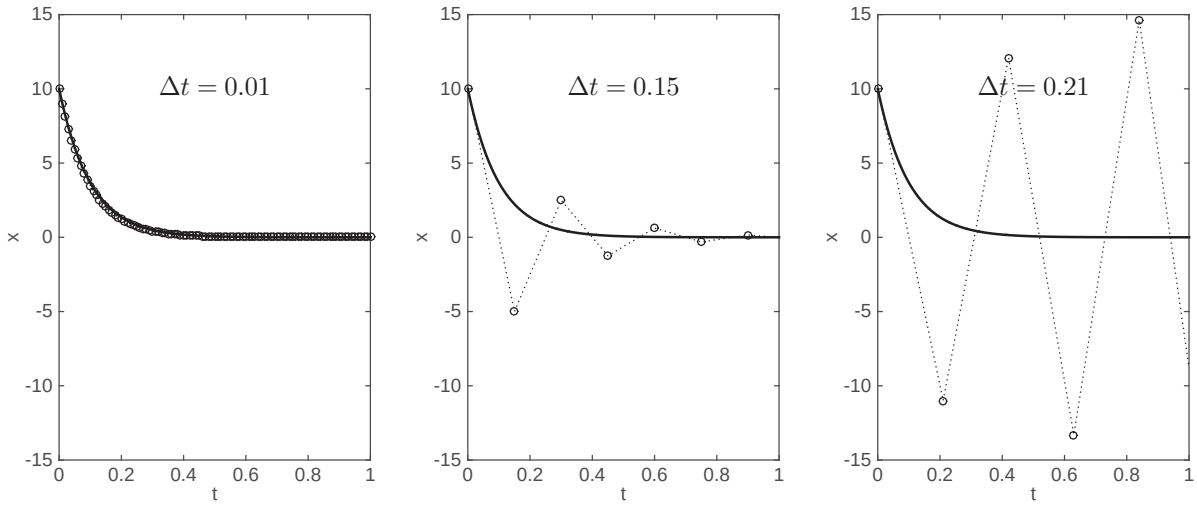


Figure 6.6: Illustration of the stability problem of the Euler method for $\lambda = 10$. Only a small enough step-size $\Delta t \leq 0.2$ allows the method to be stable (left and middle graph), while $\Delta t > 0.2$ yields an unstable simulation (right graph).

We will need to discuss the stability of each integration method we will study. This stability issue is in fact studied in a slightly more general way for various integration methods, namely the test system:

$$\dot{x} = \lambda x, \qquad x(0) = x_0 \tag{6.14}$$

is used with $\lambda \in \mathbb{C}$ and the integration method stability is described in terms of **region of stability** in the complex plane, for which a given $\lambda\Delta t$ yields a stable simulation. In the specific case of the explicit Euler method the region of stability is given by a circle of radius one centered at -1, i.e.

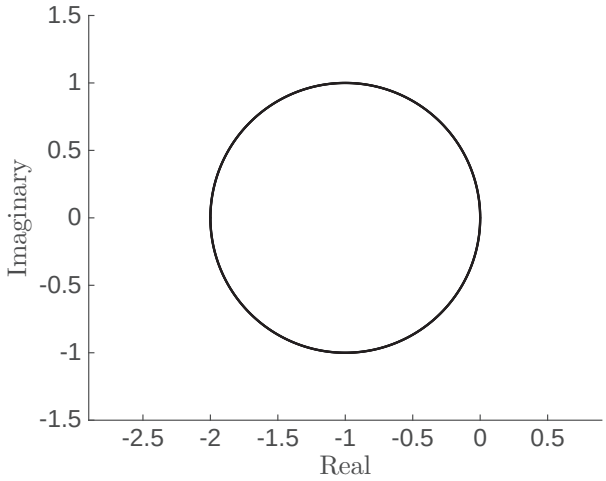


Figure 6.7: Stability region of the explicit Euler scheme (6.4) in the complex plane. All combinations of λ,Δt such that $\lambda\Delta t$ lies within the circle result in a stable (though not necessarily accurate) integration.

Obviously, the larger the region of stability is the more “stable" the method is considered.
Summary

- The explicit Euler method is a method of order 1, i.e. the global error is $\|\mathrm{x}_N - \mathrm{x}(T)\| = \mathcal{O}(\Delta t)$ this is the lowest order that one can accept from an integration method, as one expects from an method that the error gets smaller with the step size Δt.
- The explicit Euler method can become unstable for a too large step size. Its **region of stability** for the test system (6.14) is fairly small and given in Fig. 6.7

6.2 Explicit Runge-Kutta 2 methods

We will now consider an improvement of the explicit Euler method, while remaining in the family of explicit methods (the meaning of this label will become clear later). The most widely know family of explicit integration methods (beyond Euler) is the family of Runge-Kutta methods. Let us start with studying the Runge-Kutta 2 method (RK2) for which a simple interpretation of its construction can be developed.

The exact trajectories of the model dynamics (6.1) obey the following integral relationship:

$$\mathrm{x}(t_{k+1}) = \mathrm{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t))\, \mathrm{d}t \tag{6.15}$$

The key idea behind many integration methods is to try to provide good evaluations of the integral term in (6.15). In fact, one can interpret the explicit Euler method as a crude way of doing that. Indeed, explicit Euler approximates:

$$\int_{t_k}^{t_{k+1}} \mathrm{f}(\mathrm{x}(t),\mathrm{u}(t))\,\mathrm{d}t \approx (t_{k+1}-t_k)\,\mathrm{f}(\mathrm{x}(t_k),\mathrm{u}(t_k)) \approx \Delta t \mathrm{f}(\mathrm{x}_k,\mathrm{u}_k) \tag{6.16}$$

essentially using a one-step rectangular quadrature on the integral. This approximation illustrated in Figure 6.8.

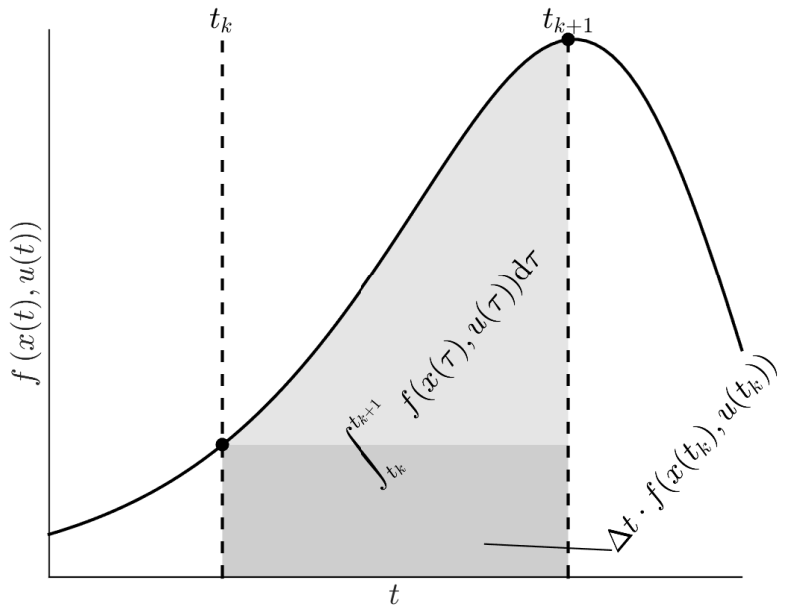


Figure 6.8: Illustration of the approximation (6.16). The light grey area is $\int_{t_k}^{t_{k+1}} \mathrm{f}(\mathrm{x}(t),\mathrm{u}(t))\,\mathrm{d}t$ (including the dark grey area), which the explicit Euler scheme is approximating with $\Delta t \mathrm{f}(\mathrm{x}_k,\mathrm{u}_k)$, i.e. the dark grey area.

A natural question then is how can we improve the approximation (6.16). To that end, one can e,g, use a mid-point rule, based on the approximation

$$\int_{t_k}^{t_{k+1}} \mathrm{f}(\mathrm{x}(t),\mathrm{u}(t))\,\mathrm{d}t \approx \Delta t \mathrm{f}\left(\mathrm{x}\left(t_k+\frac{\Delta t}{2}\right),\mathrm{u}\left(t_k+\frac{\Delta t}{2}\right)\right) \tag{6.17}$$

which tends to be better than (6.16) whenever the model trajectories have a “low curvature". This observation is illustrated in Figure 6.9.

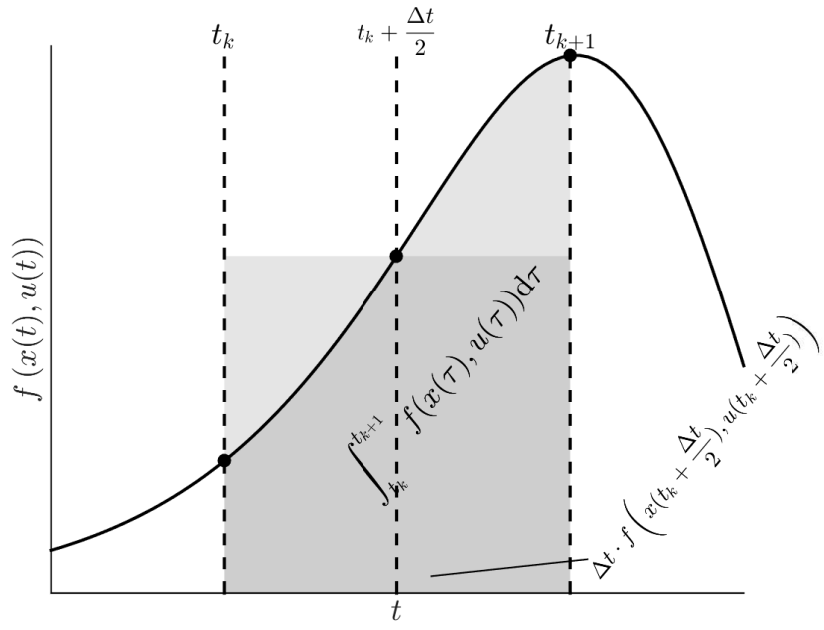


Figure 6.9: Illustration of the approximation (6.17). The light grey area is $\int_{t_k}^{t_{k+1}} \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t))\,\mathrm{d}t$ (including the dark grey area), which the explicit Euler scheme is approximating with $\Delta t \mathrm{f}\left(\mathrm{x}\left(t_k + \frac{\Delta t}{2}\right), \mathrm{u}\left(t_k + \frac{\Delta t}{2}\right)\right)$, i.e. the rectangular grey area.

Unfortunately, $\mathrm{x}\left(t_k + \frac{\Delta t}{2}\right)$ is unknown and needs to be itself replaced by an approximation built upon x_k. Here we rely again on an explicit Euler "half" step, i.e. we use:

$$\mathrm{x}\left(t_k + \frac{\Delta t}{2}\right) \approx \mathrm{x}_k + \frac{\Delta t}{2}\mathrm{f}(\mathrm{x}_k, \mathrm{u}_k) \tag{6.18}$$

Combining (6.18) and (6.17), we obtain the "mid-point" RK2 integration scheme:

$$\mathrm{K}_1 = \mathrm{f}(\mathrm{x}_k, \mathrm{u}(t_k)) \tag{6.19a}$$

$$\mathrm{K}_2 = \mathrm{f}\left(\mathrm{x}_k + \frac{\Delta t}{2}\mathrm{K}_1, \mathrm{u}\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{6.19b}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{K}_2 \tag{6.19c}$$

Despite the somewhat convoluted construction, we can fairly easily show that this RK2 scheme is of order 2 (hence the name). In order to compute the order, we will use a Taylor argument again. In order to reduce the complexity of the notation, let us do the calculation assuming that the model dynamics do not have an input u. Assuming again that $\mathrm{x}_k = \mathrm{x}(t_k)$ (i.e. the integration is exact at time t_k), we observe that:

$$\mathrm{x}(t_{k+1}) = \mathrm{x}(t_k) + \Delta t \mathrm{f}(\mathrm{x}(t_k)) + \frac{1}{2}\Delta t^2 \dot{\mathrm{f}}(\mathrm{x}(t_k)) + \mathscr{O}\left(\Delta t^3\right) \tag{6.20}$$

where $\dot{\mathrm{f}} = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\mathrm{f}$. We additionally observe that (6.19) in effect does:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}\left(\mathrm{x}_k + \frac{\Delta t}{2}\mathrm{f}(\mathrm{x}(t_k))\right) \tag{6.21}$$

and we finally observe that

$$\mathrm{f}\left(\mathrm{x}_k + \frac{\Delta t}{2}\mathrm{f}(\mathrm{x}(t_k))\right) = \mathrm{f}(\mathrm{x}(t_k)) + \frac{\Delta t}{2}\left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k} \mathrm{f}(\mathrm{x}(t_k)) + \mathscr{O}\left(\Delta t^2\right) \tag{6.22}$$

Hence

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}(\mathrm{x}(t_k)) + \frac{\Delta t^2}{2}\left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k} \mathrm{f}(\mathrm{x}(t_k)) + \mathscr{O}\left(\Delta t^3\right) \tag{6.23}$$

A comparison of (6.20) and (6.23) finally yields:

$$\|\mathrm{x}_{k+1} - \mathrm{x}(t_{k+1})\| = \mathscr{O}\left(\Delta t^3\right) \tag{6.24}$$

We can then conclude that the **one-step error of the RK2 scheme is of order 3**, and using a similar reasoning as for the explicit Euler scheme, the **global error of the RK2 scheme is of order 2**, i.e.

$$\|\mathrm{x}_N - \mathrm{x}(T)\| = \mathscr{O}\left(\Delta t^2\right) \tag{6.25}$$

(see Figure 6.10 for an illustration). We will discuss the stability of RK2 later, together with the other RK methods.

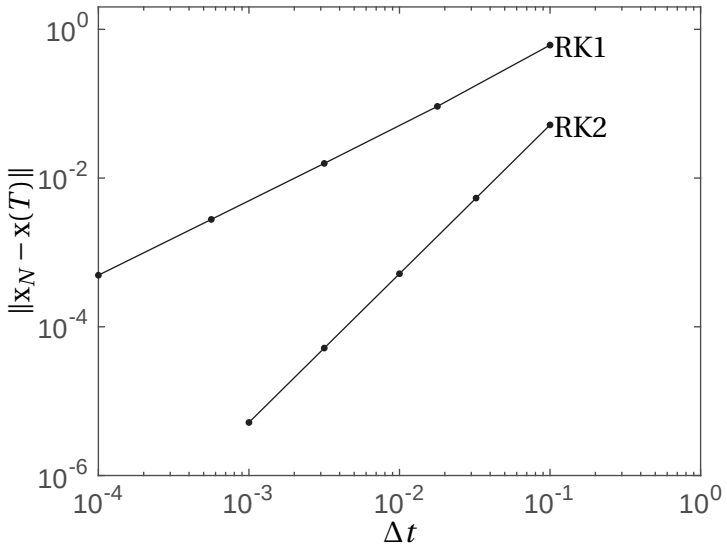


Figure 6.10: Global error of the RK2 method (6.19), and comparison with the explicit Euler scheme (6.4) (RK1), as obtained from numerical experiments. One can see the difference between having a first or second order method. The stronger slope of the RK2 method (in the log-log plot) indicates a higher power in the relationship Δt to $\|\mathrm{x}_N - \mathrm{x}(T)\|$.

6.3 General RK methods

The principles of the RK2 method can be generalized. We introduce here the general formula for RK methods, which is a generalization of (6.19).

$$\mathrm{K}_1 = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{u}(t_k + c_1\Delta t)\right) \tag{6.26a}$$

$$\vdots$$

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{u}(t_k + c_i\Delta t)\right) \tag{6.26b}$$

$$\vdots$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{u}(t_k + c_s\Delta t)\right) \tag{6.26c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} b_i\mathrm{K}_i \tag{6.26d}$$

where the coefficients a_{ij}, b_i, c_j define the RK method, and s is the number of stages of the method. We can easily observe that both the explicit Euler method and the RK2 methods introduced above can be written in this form. Indeed, it can be verified that:

- The explicit Euler method has $s = 1$ with

$$a_{11} = 0, \quad b_1 = 1, \quad c_1 = 0 \tag{6.27}$$

- The “mid-point" RK2 method has $s = 2$ with

$$a_{11} = 0, \quad a_{12} = 0, \quad c_1 = 0 \tag{6.28a}$$
$$a_{21} = \frac{1}{2}, \quad a_{22} = 0, \quad c_2 = \frac{1}{2} \tag{6.28b}$$
$$b_1 = 0, \quad b_2 = 1 \tag{6.28c}$$

6.3.1 Butcher tableau

The Butcher tableau is a convenient way of summarizing the coefficient of RK methods in a clean and compact way. The Butcher tableau reads as:

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}$$

E.g. explicit Euler has the Butcher tableau

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

and is therefore a RK1 method (because of order 1). The “mid-point" RK2 method has the Butcher tableau

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Other RK2 methods exist, with Butcher tableaus of the same size but with different coefficients, e.g.

Ralston's RK2

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

Henn's RK2

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

RK methods resulting from (6.26) ought to be divided in two categories **explicit** and **implicit**. In order to make these terms clear, let us consider all RK methods having two stages ($s = 2$). In this case (6.26) reads as:

$$\mathrm{K}_1 = \mathrm{f}(\,\mathrm{x}_k + \Delta t\,(a_{11}\mathrm{K}_1 + a_{12}\mathrm{K}_2)\,,\,\mathrm{u}(t_k + c_1\Delta t)\,) \tag{6.29a}$$

$$\mathrm{K}_2 = \mathrm{f}(\,\mathrm{x}_k + \Delta t\,(a_{21}\mathrm{K}_1 + a_{22}\mathrm{K}_2)\,,\,\mathrm{u}(t_k + c_2\Delta t)\,) \tag{6.29b}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t\,(\,b_1\mathrm{K}_1 + b_2\mathrm{K}_2\,) \tag{6.29c}$$

We observe that for $a_{11}, a_{12}, a_{22} = 0$ in (6.29) (such as in all the RK2 we have looked at above), K_1 can be computed explicitly from x_k, u(.), and then K_2 can be computed explicitly from x_k, u(.), K_1. However, for any of the a_{11}, a_{12}, a_{22} non-zero, this cannot be done as equations (6.29a)-(6.29b) become *implicit* (the unknown $\mathrm{K}_{1,2}$ appear on both sides of the equalities and are linked through the function f). We then say that the RK scheme is implicit. One can trivially see from the Butcher tableau of an RK method whether the method is explicit of implicit. We state this next.

A Butcher tableau defines an explicit integrator if and only if only the diagonal and upper-diagonal elements are zero. I.e. if and only if $a_{ij} = 0$ for any $j \geq i$. Otherwise it defines an implicit method.

The distinction is important in practice. Indeed, while explicit RK methods require simply

computing the K_i sequentially (K_1 then K_2, etc.), implicit RK methods require solving the equations together and numerically, typically using a Newton method. The latter is computationally more expensive because it requires solving linear systems a number of times for each integration steps. However, as we will see later, implicit RK schemes have powerful properties that make them attractive, even when computational time is important.

6.3.2 THE RK4 METHOD

It is worth discussing one of the most commonly used RK method, the explicit RK4 (order 4), which requires $s = 4$ stages. It has the Butcher tableau[11]

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

It is a popular approach because it has a very good trade-off between computational complexity (CPU time) and accuracy.

- The RK4 method has $s = 4$ stages
- The integration order of this scheme is 4, i.e.

$$\|\mathrm{x}_N - \mathrm{x}(T)\| \leq c\|\Delta t^4\| \tag{6.30}$$

  holds for some $c > 0$ and for Δt small enough. We should probably stress here the implication of having an order 4. The consequence of the order 4 is that if one divides the time step Δt by 2, then the amount of computation required to run a simulation on a time interval $[0, T]$ is doubled, but the numerical error of the simulation is divided by $2^4 = 16$. This Δt^4 effect of the order 4 makes it “cheap" to gain accuracy (cheap in terms of computational time).
- The method requires a few lines of computer code to implement

[11] Note that other Butcher tableaus can generate order 4 RK methods, but this one is arguably the most commonly used.

For the sake of completeness, let us write the pseudo-code corresponding to an RK4 method.

Algorithm: RK4 method for explicit ODEs

Input: Initial conditions x_0, input profile $\mathrm{u}(.)$, step size Δt

for $k = 0, \ldots, N-1$ **do**

Compute:

$$\mathrm{K}_1 = \mathrm{f}(\mathrm{x}_k, \mathrm{u}(t_k)) \tag{6.31a}$$

$$\mathrm{K}_2 = \mathrm{f}\left(\mathrm{x}_k + \frac{\Delta t}{2}\mathrm{K}_1,\ \mathrm{u}\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{6.31b}$$

$$\mathrm{K}_3 = \mathrm{f}\left(\mathrm{x}_k + \frac{\Delta t}{2}\mathrm{K}_2,\ \mathrm{u}\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{6.31c}$$

$$\mathrm{K}_4 = \mathrm{f}(\mathrm{x}_k + \Delta t \mathrm{K}_3,\ \mathrm{u}(t_k + \Delta t)) \tag{6.31d}$$

Assemble the RK step

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t\left(\frac{1}{6}\mathrm{K}_1 + \frac{1}{3}\mathrm{K}_2 + \frac{1}{3}\mathrm{K}_3 + \frac{1}{6}\mathrm{K}_4\right) \tag{6.32}$$

return $\mathrm{x}_{1,\ldots,N}$

6.3.3 Stages, order & efficiency of explicit RK methods

Let us discuss more the integration order of explicit RK methods. Before diving in, we should discuss more the implication of the number of stages s that the different RK methods have. It can be observed that the number of stages of an explicit RK method defines the computational complexity of evaluating one step $\mathrm{x}_k \to \mathrm{x}_{k+1}$ in the integrator. Indeed, each integration step (i.e. the execution of (6.26)) requires s evaluations of the model equations f and each of these evaluation translates directly into a (typically) fixed CPU time. We can deduce that the computational cost of one step of an explicit integrator is roughly proportional to s. In that sense, integrators with a low number of stages appear preferable

On the other hand, integrators with a larger number of stages s also achieve higher orders (let's label them o), and therefore achieve a higher accuracy for a given Δt. Indeed, remember that the simulation error is bounded by:

$$\|\mathrm{x}_N - \mathrm{x}(T)\| \leq c\Delta t^o \tag{6.33}$$

for some $c > 0$ and Δt small. Note that we should always consider Δt small, i.e. Δt^o becomes exponentially smaller as o increases. Hence by increasing the number of stages s we pay *proportionally* more in computations but we gain *exponentially* in accuracy. The gain tends to out-weight the loss because of the "power effect" of the order.

The RK methods we have discussed so far suggest that the number of stages s of the method equals its order o. Indeed, we have seen that:

- Explicit Euler has an order $o = 1$ and is an RK method with $s = 1$ stage. If Δt is divided by 2 then the error is divided by 2.
- The RK2 methods we have seen have an order $o = 2$ and have $s = 2$ stages. If Δt is divided by 2 then the error is divided by 4.
- The RK4 method we have seen has order $o = 4$ and has $s = 4$ stages. If Δt is divided by 2 then the error is divided by 16.

From these observations, it may appear that the higher s the better. Unfortunately, this pattern $o = s$ breaks beyond $o = 4$. Let us stress this fact in the following table (these numbers are not straightforward to establish, and we will leave that question out of these notes).

Order	Stages required	Δt divided by 2 $\rightarrow$ error divided by...
RK1	1	2
RK2	2	4
RK3	3	8
RK4	4	16
RK5	6	32
RK6	7	64
RK7	9	128
RK8	11	256

We observe from this table that for a low number of stages ($s \leq 4$), we have $o = s$, hence adding a stage adds an order (see Fig. 6.11 for an illustration). However, for a higher number of stages, the increase in the order of the method “stalls", and stops increasing as quickly as the number of stages.

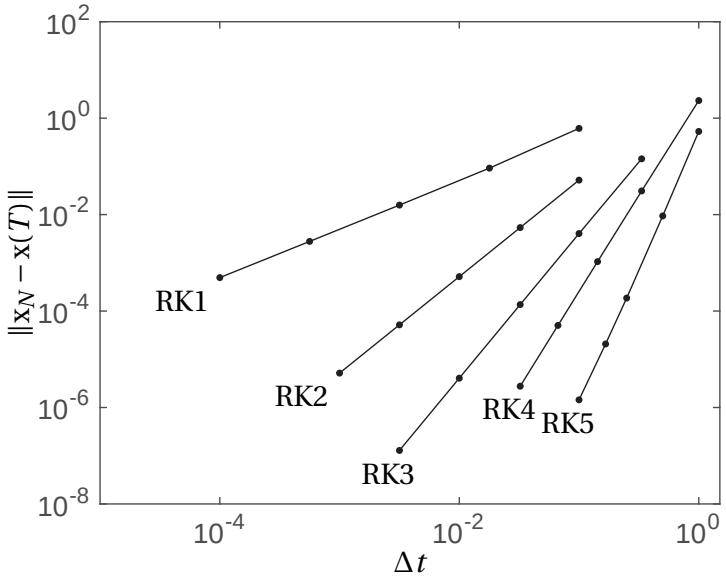


Figure 6.11: Global error of the different RK methods, as obtained from numerical experiments.

This effect has some important consequences in practice on the choice of an integration method. Indeed, the crux of any integration method is to achieve a given accuracy with a minimum computational budget. There are two ways of increasing the integrator accuracy: increasing the order o, i.e. increasing the number of stage s, or decreasing Δt. Having these two options makes the choice of integrator (choice of order and choice of time step) non-trivial.

However, using the table above, we can in fact somewhat formalize this choice. Let us assume that we want the global error to be limited to some given number Tol

- Then for an integrator of order o, we need:

$$\|x_N - x(T)\| \leq c\Delta t^o \leq \text{Tol} \tag{6.34}$$

  such that the step size Δt is limited to:

$$\Delta t \leq \left(\frac{\text{Tol}}{c}\right)^{\frac{1}{o}} \tag{6.35}$$

- In order to carry out a simulation on the time interval $[0,\ T]$, the number of integrator step required is:

$$N = \frac{T}{\Delta t} \tag{6.36}$$

  and for a number of stages s, the number of evaluation n of the system dynamics f is:

$$n = N \cdot s = \frac{sT}{\Delta t} \geq sT\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} \tag{6.37}$$

We deduce from this simple reasoning that the computational cost per unit of simulation time is at least:

$$\frac{n}{T} \geq s\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} \tag{6.38}$$

where o and s are related via the table above. It is then interesting to chart this relationship, we do that in the Figures below. Here it becomes clear that a very low or very high order is not optimum, and that the optimum is in the order 3-6.

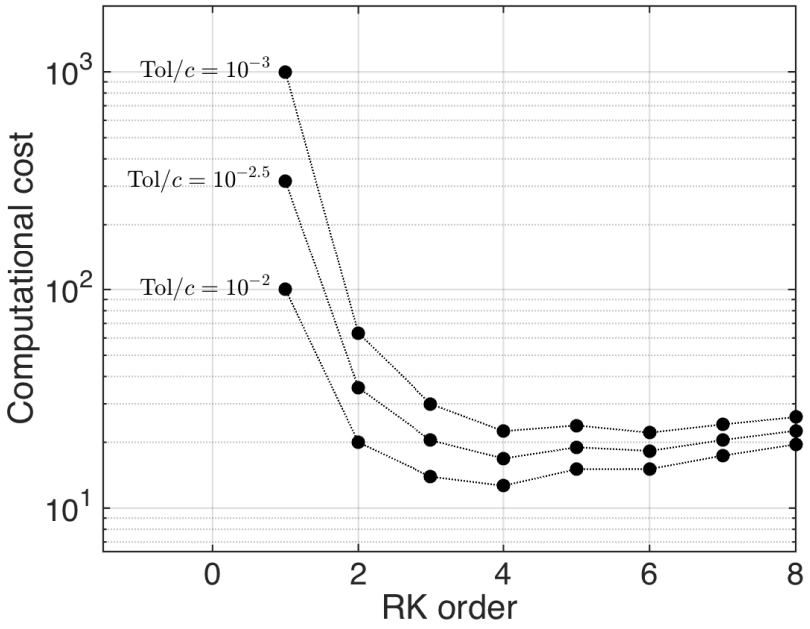


Figure 6.12: Illustration of formula (6.38) for different $\frac{\text{Tol}}{c}$. The horizontal axis represents the order o of the method, and the vertical the computational cost per simulation time T, i.e. $\frac{n}{T}$. The optimum is achieved in the middle range, and not for low nor high order methods.

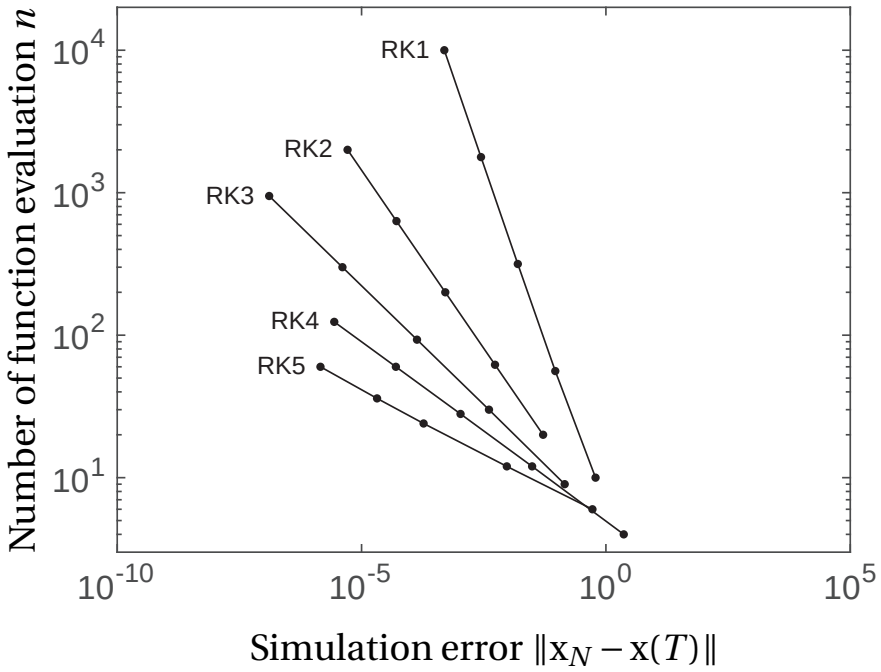


Figure 6.13: Illustration of the number of function evaluations required to reach different level of simulation accuracy for different explicit RK methods for the integration of a linear test example. Here the RK5 scheme beats the other schemes for high accuracies, and RK4 beats the other schemes for lower accuracies. This picture is model-dependent, and should just be understood as an example.

Before closing this section, a last point ought to be stressed regarding the order o achieved by different RK methods. Indeed, we have seen the relationship between the order o of different RK methods and the number of stages s required to achieve that order. We need to stress here that the order o of an RK method does not follow *simply* from using the adequate amount of stages. E.g. the RK2 schemes ($o = 2$) we have seen require $s = 2$ stages, but they also require the coefficients a, b, c to be chosen adequately. E.g. in the case of a $s = 2$ stages method, an order $o = 2$ is achieved if the coefficients satisfy:

$$b_1 + b_2 = 1, \qquad b_2 c_2 = \frac{1}{2}, \qquad a_{21} = c_2 \tag{6.39}$$

Similarly, for the other RK methods, the coefficients cannot take *arbitrary* value if one wants the method to achieve its highest possible order.

6.4 Stability of explicit RK schemes

Let us go back to the question of integration stability we investigated in section 6.1.2 in the more general context of RK methods of arbitrary order. The general formula defining the stability region of the test system

$$\dot{x} = \lambda x, \qquad x(0) = x_0 \tag{6.40}$$

for an RK method of order o is:

$$S = \left\{ \quad \lambda \Delta t \quad \text{s.t.} \quad \left| \sum_{k=0}^{o} \frac{(\lambda \Delta t)^k}{k!} \right| \leq 1 \quad \right\} \tag{6.41}$$

The regions of stability for different orders are depicted in Figure 6.14. One can observe that the regions increase with the order, but their size is fairly limited.

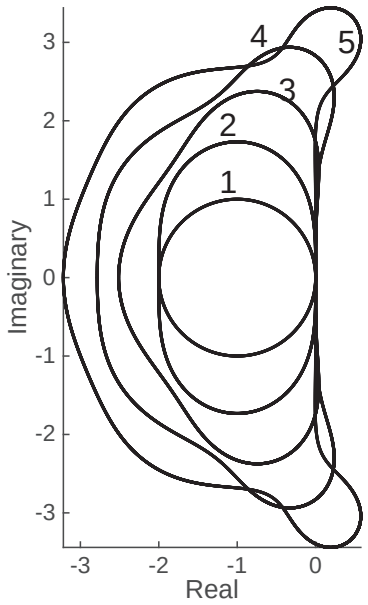


Figure 6.14: Illustration of the region os stability for the explicit RK methods 1 to 5. One can observe that the stability region grows with the order, but it nonetheless covers a limited range of admissible $\lambda \Delta t$.

6.4.1 Stiff systems

The numerical stability considerations discussed above have important practical consequences. Indeed, while the dynamics(6.40) are clearly of no interest, they operate as a trivial test system to discuss how integration methods react to **fast dynamics**, i.e. more specifically, time constants in the dynamics that are in the same ballpark as the integration step Δt. The question of whether an integration method is capable of handling such fast dynamics is crucial in the context of stiff systems, which appear very often in models for engineering application such as mechanical and electrical systems. It is important to “detect" the stiffness of a model when trying to simulate it.

We have in fact already encountered stiff systems in the DAE section. Indeed, (5.33) essentially describes a model having very fast dynamics (for ϵ small) on the states z, and slower dynamics on the states x. DAEs allow one to approximate these fast states via their “decayed" values (i.e. their trajectories after their fast, stable dynamics have decayed). However, one may want to process these dynamics without using DAE formulations, in which

case the fast dynamics have to be dealt with. As an example of a stiff system, consider the model equations

$$\dot{\mathrm{x}} = \begin{bmatrix} 0 & 5 & 1 & 0 \\ -5 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -10^{-1}\epsilon & \epsilon \\ 0 & \epsilon & -\epsilon & -10^{-1}\epsilon \end{bmatrix} \mathrm{x} \tag{6.42}$$

for $\epsilon = 5 \cdot 10^{-4}$. The corresponding trajectories are illustrated in Fig. 6.15 and 6.16. One can observe that states $\mathrm{x}_{1,2}$ evolve slowly (although they are influenced by the fast states and have some small oscillations in the beginning), while the states $\mathrm{x}_{3,4}$ oscillate fast and decay to slow trajectories. This kind of system can e.g. arise in mechanical systems or electrical circuits when some part have eigenfrequencies that are much higher than other parts. They are especially expensive to treat in numerical simulations because the fast dynamics require a small Δt in order for the numerical simulation to be stable, while long simulations (T large) are required in order to see the evolution of the “slow" states.

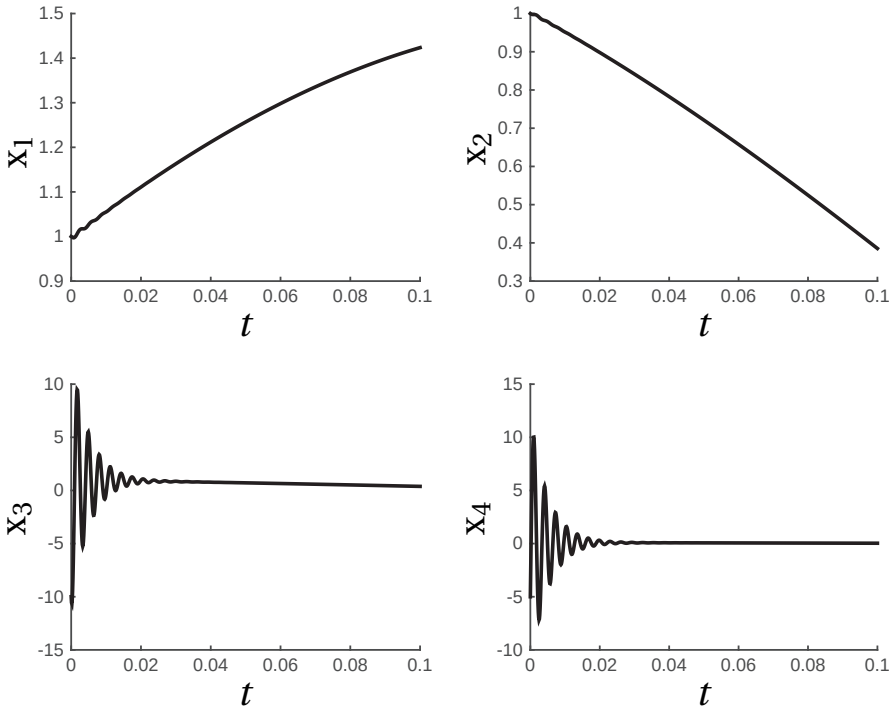


Figure 6.15: Illustration of the trajectories of the stiff system (6.42).

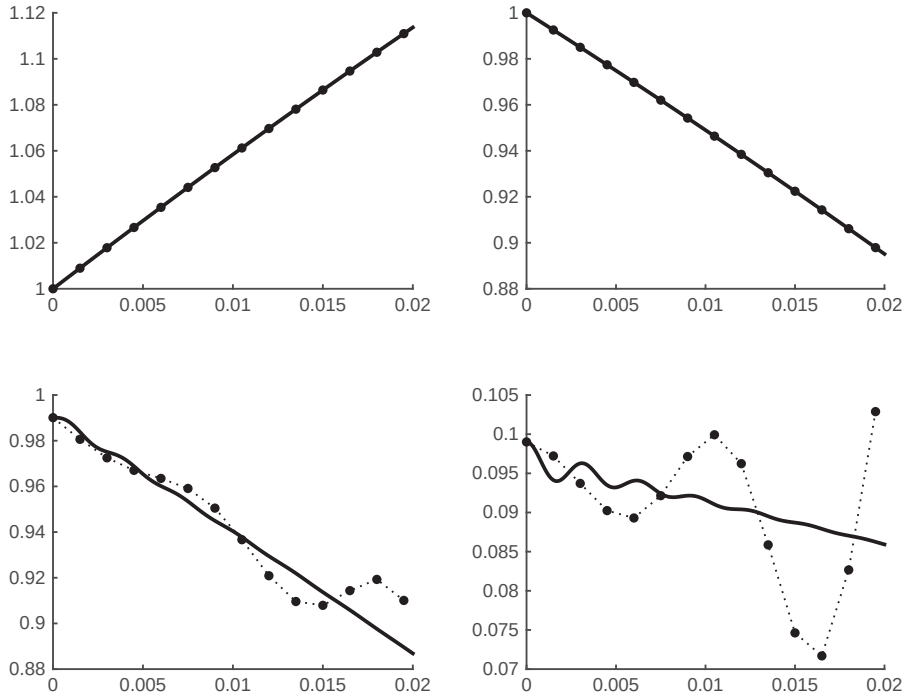


Figure 6.16: Illustration of the trajectories of the stiff system (6.42) vs. a simulation using RK4 with $\Delta t = 1.5 \cdot 10^{-3}$ (dotted curve). The initial conditions for the "fast" states are chosen such that $\dot{z} = 0$ at $t = 0$. The small step size is close to the limit for which the simulation becomes stable.

We will see later on that stiff systems are in fact one of the motivations for using implicit integration methods.

6.5 Error control & Adaptive integrators

We have seen above that the integrator accuracy and stability depend on the step size Δt. It can be tricky, however, to select the correct step size a priori. Moreover, as suggested in the order computation of the RK1 and 2 schemes, the accuracy depends on how "curved" the model trajectories are, and this "curvature" can vary significantly throughout the trajectories of the system. A careful approach would then be to simply use an overly small step size to ensure that the required accuracy is achieved. This conservative approach would, however, be overly expensive, as it would use very small step sizes Δt even when they are not needed. A classic example of an ODE triggering such difficulties is the Van Der Pol oscillator having the equations

$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2, \tag{6.43a}$$

$$\dot{x}_2 = x_1, \tag{6.43b}$$

and e.g. the trajectories displayed in Fig. 6.17.

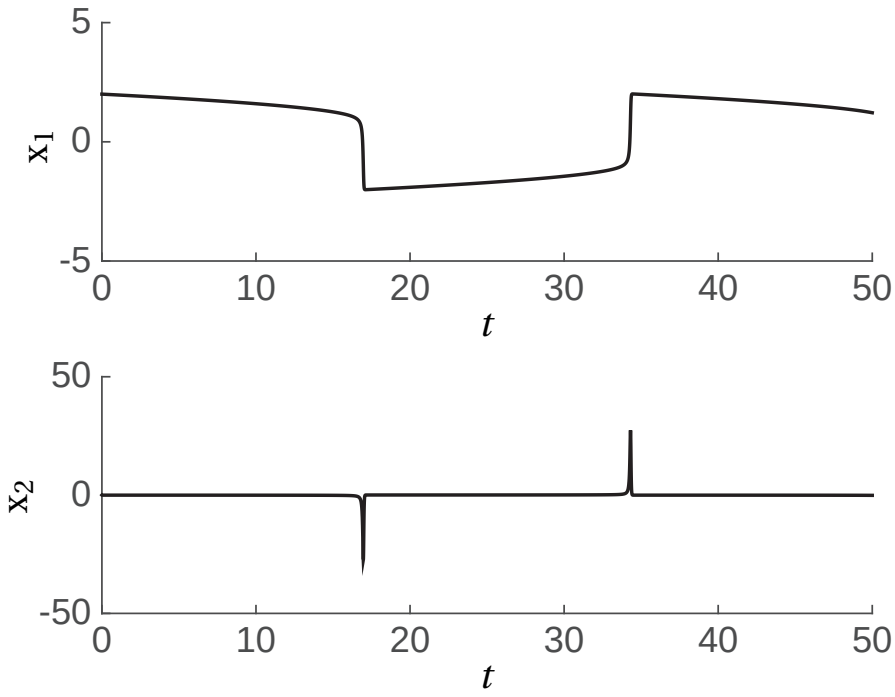


Figure 6.17: Illustration of the Van Der Pol trajectories. One can observe that the trajectories are fairly “benign" most of the time, but regularly go through very sharp changes. The Van Der Pol oscillator is a challenging ODE for numerical integration as very fine time steps Δt are needed to “survive" the sharp changes, while fairly large time steps can be used on the rest of the trajectories.

It then makes sense to use varying step sizes Δt throughout the trajectories. Adaptive integration schemes are a practical approach to do precisely that. The key idea here is to adjust the step size Δt during the simulations in order to meet the required accuracy. In practice, adaptive RK schemes perform an error control at every RK step, and when the error is larger than acceptable, the step size is reduces and a shorter step is attempted until the tolerance is met. Adaptive integrator require a baseline to assess the error at each step. A common practice is to compute the integration step using two different Butcher tableaus, and compare their outcome. If their discrepancy is above a certain level, then the step size is reduced.

Let us e.g. consider the RK45 adaptive integration scheme, which is implemented in Matlab in the function ode45.m. At each RK step, the procedure:

- Generates two steps from x_k, let us call them x_{k+1} and $\hat{\mathrm{x}}_{k+1}$, using two different Butcher tableaus.
- Compares the two steps, i.e. computes $e = \|\mathrm{x}_{k+1} - \hat{\mathrm{x}}_{k+1}\|$
- Reduces Δt if the error e is above some tolerance, and computes the steps again until the tolerance is met

- Increases Δt a bit if the error e is significantly below the tolerance

We observe here that Δt has a “memory" in the sense that if a very low Δt was needed at a step k, then it will remain small for some steps. One ought to observe that this procedure is somewhat computationally expensive, as it computes at every RK steps two different steps. In order to limit the computational expenses, a common approach is to use two Butcher tableau that *differ only* in the b coefficients, i.e. in forming the linear combination (6.26d), while the coefficients a and c are the same such that equations (6.26a)-(6.26c) need to be computed only once. The two steps x_{k+1} and $\hat{\mathrm{x}}_{k+1}$ are therefore computed as:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} b_i \mathrm{K}_i \tag{6.44a}$$

$$\hat{\mathrm{x}}_{k+1} = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} \hat{b}_i \mathrm{K}_i \tag{6.44b}$$

The Butcher tableau for adaptive RK methods is then presented with two lines in the “b" part of the tableau, the first one for the “classic" b and the second for $\hat{b}$. E.g. the Butcher tableau used in the ode45.m Matlab function reads as[12]:

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	64448/6561	−212/729			
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	35/384	0	500/1113	125/192	−2187/6784	11/84	0
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40

One can observe that this method has $s = 7$ stages. Both steps x_{k+1} and $\hat{\mathrm{x}}_{k+1}$ are of order $o = 5$. Figure 6.18 illustrates the step size Δt selected by the ode45.m when treating the Van Der Pol oscillator (6.44).

[12] the RK method coded by this tableau is referred to as the RK45 Dormand-Prince method

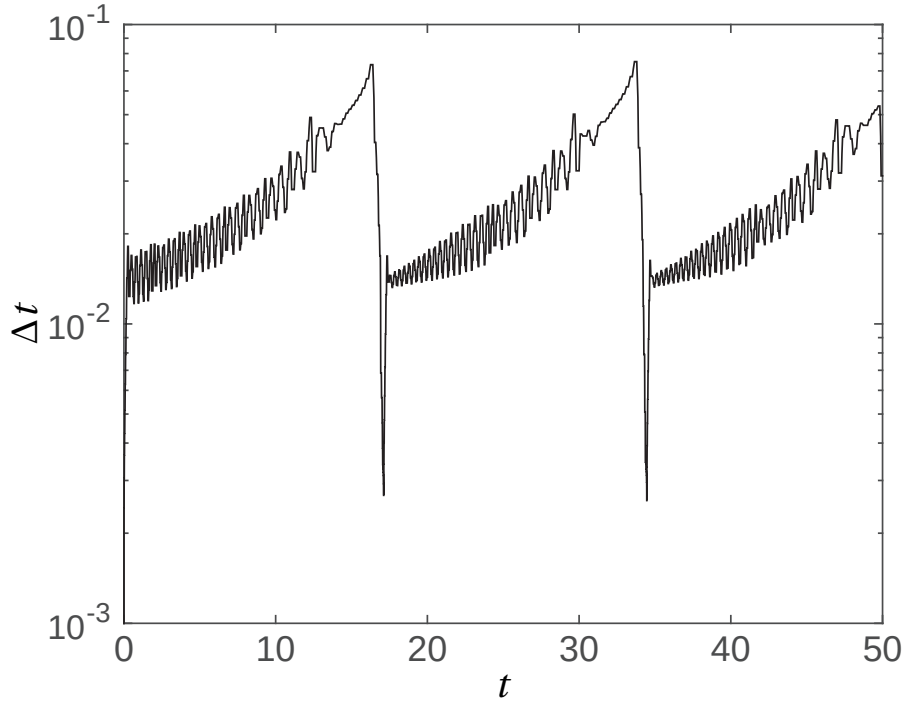


Figure 6.18: Illustration of the step sizes selected by ode45.m for simulating the Van Der Pol trajectories. Whenever the trajectories are going through a “sharp" turn, the step size drops drastically in order to keep the integration error under control. It then increases slowly again to reduce the computational burden.

One can finally note that the last line of the “a" part of the tableau is identical to the first line of the “b" part, and that the K_1 is given by:

$$K_1 = f(x_k, u(t_k)) \tag{6.45}$$

This implies that K_1 of step $k+1$ matches K_7 of step k such that the last K_7 can be reused for the next K_1.

7 Implicit Integration Methods - Runge-Kutta

In the explicit RK section, we have seen that a Butcher tableau can define explicit or implicit RK methods. We have discussed in details explicit RK methods. It is time now to discuss implicit RK methods. While implicit methods are more expensive computationally (one needs to solve equations at every RK step, as opposed to simply evaluate the K_i sequentially as in explicit methods), we will see that they have some striking advantages. Indeed, Implicit RK methods

- can achieve very high and systematic orders
- are stable regardless of the step size Δt
- are ideal for handling the simulation of DAEs

7.1 IMPLICIT EULER METHOD

Let us start with approaching implicit integrators from a basic point of view, by studying the implicit Euler method. Recall that the *explicit* Euler method uses the iteration:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}(\mathrm{x}_k, \mathrm{u}_k) \tag{7.1}$$

where $\mathrm{u}_k = \mathrm{u}(t_k)$. The *implicit* Euler method uses instead:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}(\mathrm{x}_{k+1}, \mathrm{u}_{k+1}) \tag{7.2}$$

The name “implicit" ought to be fairly clear from (7.2). Indeed, obtaining x_{k+1} from (7.2) cannot be done via a simple function evaluation, but must be done via *solving* (7.2) for x_{k+1}, i.e. one needs to find a solution to

$$\mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1}) := \mathrm{x}_k + \Delta t \mathrm{f}(\mathrm{x}_{k+1}, \mathrm{u}_{k+1}) - \mathrm{x}_{k+1} = 0 \tag{7.3}$$

in terms of x_{k+1}. Note that $\mathrm{r} \in \mathbb{R}^n$ where n is the size of the state space, i.e. $\mathrm{x} \in \mathbb{R}^n$. Solving (7.3) is typically done via applying the Newton method covered earlier in these notes. The deployment of the implicit Euler algorithm is then detailed below.

Algorithm: Implicit Euler method

Input: Initial conditions x_0, input profile $\mathrm{u}(.)$, step size Δt

for $k = 0, \ldots, N-1$ **do**

Guess x_{k+1}, one can e.g. use $\mathrm{x}_{k+1} = \mathrm{x}_k$

while $\|\mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1})\| > \mathrm{Tol}$ **do**

Compute the solution $\Delta \mathrm{x}_{k+1}$ to the linear system:

$$\frac{\partial \mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}} \Delta \mathrm{x}_{k+1} + \mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1}) = 0 \tag{7.4}$$

where r is given by (7.3). Update:

$$\mathrm{x}_{k+1} \leftarrow \mathrm{x}_{k+1} + \alpha \Delta \mathrm{x}_{k+1} \tag{7.5}$$

for some step size $\alpha \in]0, 1]$ (a full step $\alpha = 1$ generally works for implicit integrators)

return $\mathrm{x}_{1,\ldots,N}$

Note that this procedure is significantly more complex than the simple update (7.1) of the explicit Euler method. In particular, computing the Newton step (7.4) requires computing the Jacobian $\frac{\partial \mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}}$ and forming its matrix factorization (i.e. solving the linear system). This procedure must be repeated at *each* RK step $\mathrm{x}_k \to \mathrm{x}_{k+1}$, i.e. many times in a complete simulation. Note that the Jacobian $\frac{\partial \mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}}$ reads as:

$$\frac{\partial \mathrm{r}(\mathrm{x}_{k+1}, \mathrm{x}_k, \mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}} = \frac{\partial \mathrm{f}(\mathrm{x}_{k+1}, \mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}} - I \tag{7.6}$$

and requires computing the Jacobian on the model dynamics $\frac{\partial \mathrm{f}(\mathrm{x}_{k+1},\mathrm{u}_{k+1})}{\partial \mathrm{x}_{k+1}}$.

The implicit Euler method has a **global error of order 1**, i.e.

$$\|\mathrm{x}_N - \mathrm{x}(T)\| \leq c\Delta t \tag{7.7}$$

for some $c > 0$ and for Δt sufficiently small. Hence the order of the implicit Euler method is identical to the explicit Euler method.

7.1.1 Stability of the implicit Euler method

Let us now unpack one of the main motivations for using, in some cases, implicit methods for simulation a model. Let us come back to the stability issue of the explicit Euler method (7.1), and recall that the iteration is unstable on the (stable) test system

$$\dot{x} = \lambda x \tag{7.8}$$

for

$$\Delta t \lambda > 2 \tag{7.9}$$

We can make a very similar computation for implicit Euler scheme, i.e. we observe that the Implicit Euler method (7.2) deployed on the test system (7.8) reads as

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \lambda \mathrm{x}_{k+1} \tag{7.10}$$

or equivalently

$$\mathrm{x}_{k+1} = \frac{1}{1-\lambda\Delta t}\mathrm{x}_k \tag{7.11}$$

Hence the Implicit Euler method (7.2) is stable if

$$|1 - \lambda\Delta t| > 1 \tag{7.12}$$

Observe that $\mathscr{R}(\lambda) < 0$ is required in order for (7.8) to be stable, and it follows that (7.12) always holds. This result is quite striking. Indeed, it entails that **the implicit Euler method is stable regardless of how "fast" the time constants of the model are**. This property is called **A-stability**, and means that the whole left-hand complex plane is stable (as opposed to the limited regions depicted in Figure 6.14). A-stability allows one to treat stiff dynamics without taking special care of the instability of the method, i.e. by taking "fairly large" steps Δt despite the fast time constant λ of the model.

Let us reuse the stiff model (6.42) and apply the implicit Euler method to simulate it. The outcome is illustrated in Fig. 7.1 for $\Delta t = 1 \cdot 10^{-3}$ and $\Delta t = 2 \cdot 10^{-2}$.

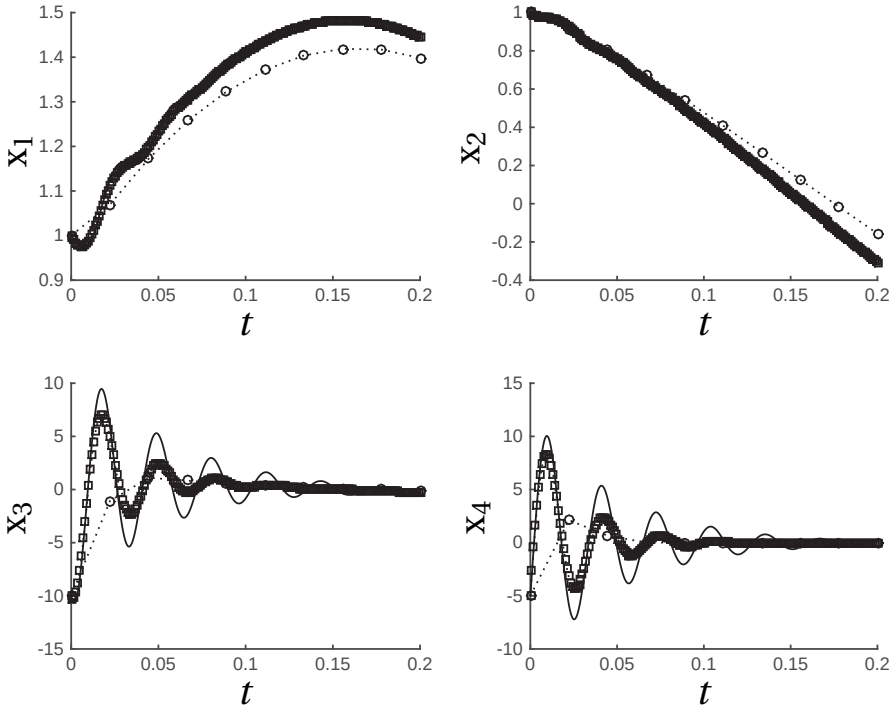


Figure 7.1: Illustration of the trajectories of the stiff system (6.42) and its simulation via the implicit Euler method using $\Delta t = 2 \cdot 10^{-2}$ (round markers) and $\Delta t = 1 \cdot 10^{-3}$ (square markers). One can observe that the implicit Euler, for "long" steps remains stable and approximates the fast dynamics by decaying to their damped values.

7.2 Implicit Runge-Kutta methods

Let us now study higher-order implicit Runge-Kutta methods. As already explained in the explicit RK section, a Butcher tableau that is not lower-diagonal describes an implicit method.

Then the RK equations (6.26) recalled here:

$$\mathrm{K}_1 = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{u}(t_k + c_1\Delta t)\right) \tag{7.13a}$$

$$\vdots$$

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{u}(t_k + c_i\Delta t)\right) \tag{7.13b}$$

$$\vdots$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{u}(t_k + c_s\Delta t)\right) \tag{7.13c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} b_i\mathrm{K}_i \tag{7.13d}$$

are implicit in $\mathrm{K}_{1,\dots,s}$, as $a_{ij} \neq 0$ for some $j \geq i$. In this case we need to solve (7.13a)-(7.13c) numerically, typically using the Newton method. more specifically, similarly to (7.3), we write:

$$\mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.)) := \begin{bmatrix} \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{u}(t_k + c_1\Delta t)\right) - \mathrm{K}_1 \\ \vdots \\ \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{u}(t_k + c_i\Delta t)\right) - \mathrm{K}_i \\ \vdots \\ \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{u}(t_k + c_s\Delta t)\right) - \mathrm{K}_s \end{bmatrix} = 0 \tag{7.14}$$

and deploy a Newton method on the “unknown"

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_1 \\ \vdots \\ \mathrm{K}_s \end{bmatrix} \tag{7.15}$$

to solve (7.14). Note that $\mathrm{r} \in \mathbb{R}^{n \cdot s}$ and $\mathrm{K} \in \mathbb{R}^{n \cdot s}$ where n is the dimension of the state space (i.e. $\mathrm{x} \in \mathbb{R}^n$) and s the number of stages of the IRK method. The Newton method then works as follows:

Algorithm: IRK for explicit ODEs

Input: Initial conditions x_0, input profile $\mathrm{u}(.)$, Butcher tableau, step size Δt

for $k = 0, \ldots, N-1$ **do**

Guess for K (one can e.g. use $\mathrm{K}_i = \mathrm{x}_k$)

while $\|\mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.))\| > \mathrm{Tol}$ **do**

Compute the solution $\Delta \mathrm{w}$ to the linear system:

$$\frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.))}{\partial \mathrm{K}} \Delta \mathrm{K} + \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.)) = 0 \tag{7.16}$$

with r given by (7.14). Update:

$$\mathrm{K} \leftarrow \mathrm{K} + \alpha \Delta \mathrm{K} \tag{7.17}$$

for some step size $\alpha \in]0, 1]$ (a full step $\alpha = 1$ generally works for implicit integrators)

Take RK step:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} b_i \mathrm{K}_i \tag{7.18}$$

return $\mathrm{x}_1, \ldots, \mathrm{x}_N$

The main computational complexity of this procedure is typically solving the linear system (7.16) which involves the (possibly dense) Jacobian matrix $\frac{\partial \mathrm{r}(K, \mathrm{x}_k, \mathrm{u}(.))}{\partial K}$ of size $\mathbb{R}^{n \cdot s \times n \cdot s}$. Forming the Jacobian matrix can also be fairly expensive in terms of computational complexity as it requires evaluating the Jacobians of the system dynamics:

$$\frac{\partial \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij} \mathrm{K}_j, \mathrm{u}(t_k + c_i \Delta t)\right)}{\partial \mathrm{K}_l} \in \mathbb{R}^{n \times n} \tag{7.19}$$

for $i, l = 1, \ldots, s$.

7.3 Stability & order of IRK methods

Similarly to the implicit Euler method, **IRK methods are A-stable**, meaning that they can "survive" (i.e. they are stable for) any fast dynamics. Some of them have even stronger form of stability (L-stability), but we will not detail this in these notes. Concerning the order of IRK methods, the picture here is fairly striking. Indeed, recall that explicit RK methods achieve an order $o = s$ (number of stages) for $s \leq 4$ and then the order "stalls" and does not increase as fast as s. This was bad news for high-order explicit RK methods in terms of efficiency (complexity vs. accuracy).

In contrast, **implicit RK methods can achieve $o = 2s$ for any number of stages s.**

We can then revisit the “order table" that we saw in the explicit RK section, comparing the explicit RK methods (ERK) to the implicit RK methods (IRK)

Order	Stages required	Order	Stages required
ERK2	2	IRK2	1
ERK4	4	IRK4	2
ERK6	7	IRK6	3
ERK8	11	IRK8	4

and readily see that IRK methods requires dramatically less stages to achieve the same order as ERK methods. Let us illustrate this statement in Figure 7.2.

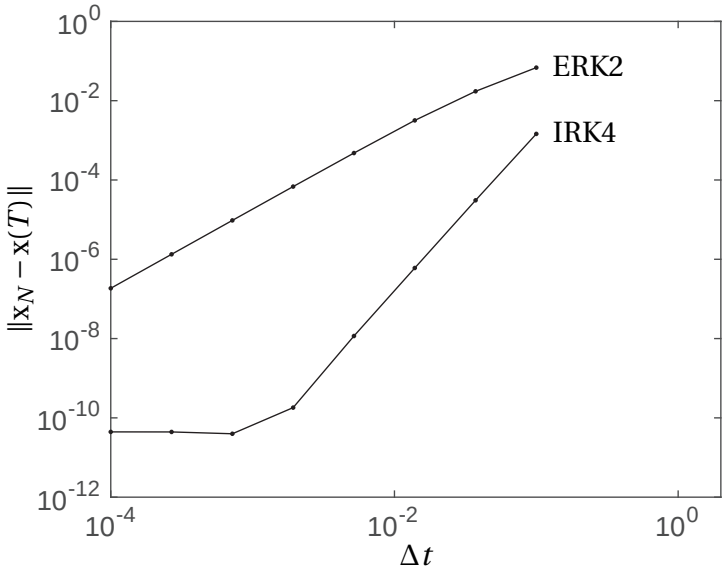


Figure 7.2: Comparison of the global error of the explicit RK2 method (6.19) and an implicit RK4 method, as obtained from numerical experiments. Note that both methods require $s = 2$ stages. One can see the high order $o = 2 \cdot s$ of the IRK method to the order $o = s$ of the ERK method. One can also observe how the accuracy of the IRK method stops decreasing at about $10^{-10} - 10^{-11}$ where it reaches the accuracy of the linear algebra.

Unfortunately, this striking difference does not mean that IRK methods are necessarily better than the ERKs. In order to unpack that statement, let us investigate the efficiency of IRK methods (complexity for a given accuracy), using similar lines as what we did for the ERK methods. Let us assume that we want the global error to be limited to some given number Tol

- Then for an integrator of order o (even), we need:

$$\|\mathrm{x}_N - \mathrm{x}(T)\| \leq c\Delta t^o \leq \mathrm{Tol} \tag{7.20}$$

such that the step size Δt is limited to:

$$\Delta t \leq \left(\frac{\text{Tol}}{c}\right)^{\frac{1}{o}} \tag{7.21}$$

- In order to carry out a simulation on the time interval $[0,\ T]$, the number of integrator step required per simulation time T is:

$$N = \frac{T}{\Delta t} \geq T\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} \tag{7.22}$$

- The similarity with ERK methods stops here. Indeed, the complexity of IRK methods is typically dominated by solving the linear system (7.16). The complexity of solving this system is typically in the order of the cube of the size of the Jacobian matrix if the matrix is dense[13], which is $n \cdot s$, where n is the state size. This system must typically be solved several times, say m in order to reach a good accuracy in solving the IRK equations. The complexity per time step is then in the order of:

$$\mathscr{C} = \mathcal{O}\left(mn^3 s^3\right) \tag{7.23}$$

We deduce from this simple reasoning that the computational cost per unit of simulation time is of the order

$$\frac{\mathscr{C}}{T} = \mathcal{O}\left(Nmn^3 s^3\right) = \mathcal{O}\left(\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} mn^3 s^3\right) = \mathcal{O}\left(\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} \frac{mn^3 o^3}{8}\right) \tag{7.24}$$

One can put (7.24) in contrast with the complexity of explicit RK methods(6.38), recalled here:

$$\frac{n}{T} \geq s\left(\frac{\text{Tol}}{c}\right)^{-\frac{1}{o}} \tag{7.25}$$

but the comparison is arguably a bit difficult, as we are trying to compare evaluations of the model dynamics f in explicit methods to forming and solving linear systems in implicit methods. We can, however, compare the explicit and implicit approaches via numerical experiments. E.g. Fig. 7.3 depicts the computational time vs. integration accuracy for explicit and implicit RK methods of various orders (for Δt fixed) for the Van Der Pol oscillator (6.43). The observations we make in this specific example are fairly consistent for different models. Implicit methods can be a bit more computationally expensive than explicit ones, though the difference is typically mild. This picture is dramatically changed for stiff systems, where implicit integrators are typically required to have a numerically stable integration scheme.

[13] the cube can be lowered if the Jacobian is sparse or structured, which can be the case for IRK methods. Linear Algebra packages are very efficient at finding such structures, and lower the complexity of solving the linear system. Hence in numerical experiments, one often observe a lower complexity.

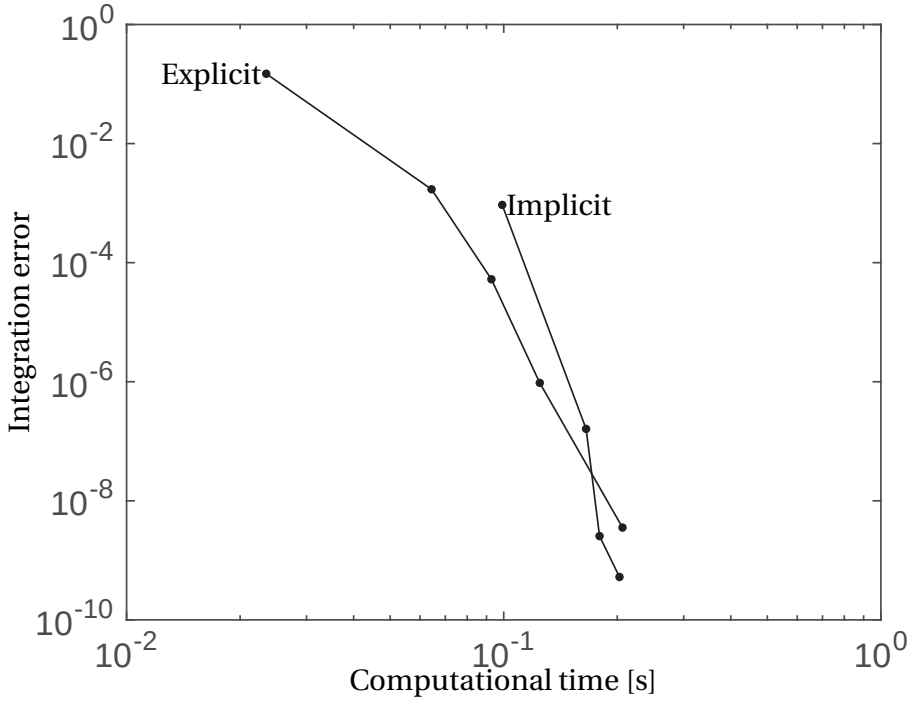


Figure 7.3: Illustration of the computational time vs. accuracy for IRK and ERK methods for the VanDerPol system (6.44) with $\Delta t = 10^{-2}$. One can observe the different behaviors of the implicit and explicit methods, and that even though implicit methods reach very high order for few stages, the computational cost of solving linear systems at every RK step makes the IRK method typically more expensive than the ERK ones. Note that these results are only illustrative, and may change depending on the coding language, computer architecture, system, etc.

7.4 Collocation methods

There are different families of implicit RK methods, but in this course we will investigate the family of collocation methods, which have very strong properties, and which have a fairly intuitive interpretation. The key idea behind collocation methods is to approximate the model trajectories via polynomials. This is intrinsically an interpolation problem.

7.4.1 Polynomial interpolation

Interpolation based on polynomials typically uses a linear combination:

$$\mathrm{p}(\tau, \mathrm{K}) = \sum_{i=1}^{s} \mathrm{K}_i \ell_i(\tau) \tag{7.26}$$

of a collection of polynomials $\ell_i \in \mathbb{R}$ built on the interval $\tau \in [0,1]$ and weighted by the parameters $\mathrm{K}_{1,\dots,s} \in \mathbb{R}^n$. Note that $\mathrm{p} \in \mathbb{R}^n$ as well. A common choice of polynomial collection

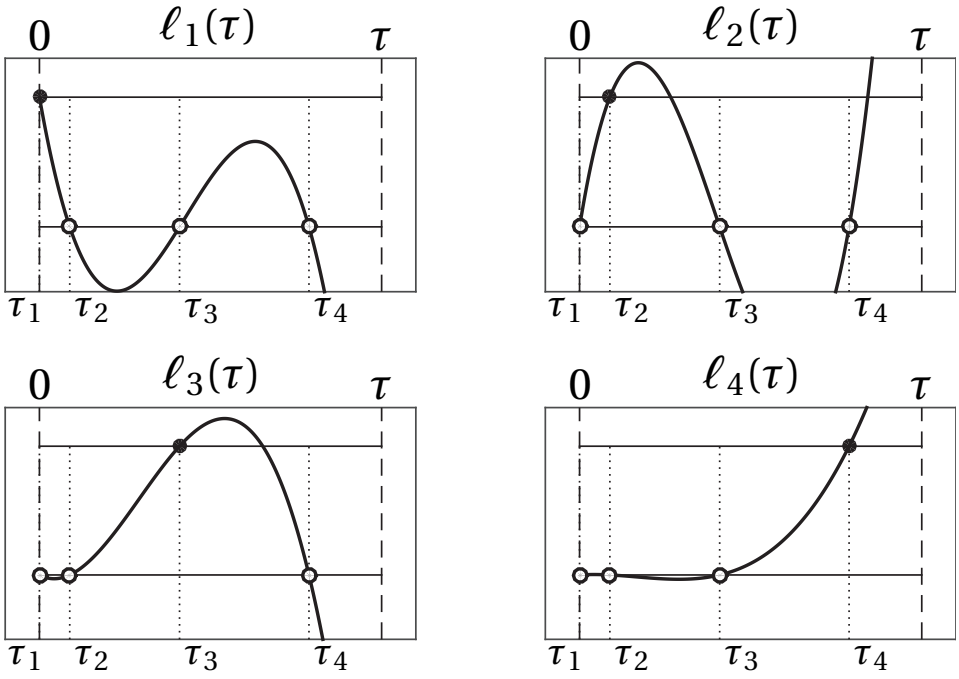


Figure 7.4: Illustration of the Lagrange polynomials for $s = 4$. The grid $\tau_{1,\dots,4}$ is displayed via the vertical dotted lines (note that $\tau_1 = 0$). Property (7.29) is readily visible.

$\ell_{1,\dots,s}$ is the so called Lagrange polynomials, which are constructed as:

$$\ell_i(\tau) = \prod_{j \neq i} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{7.27}$$

based on a grid $\tau_{1,\dots,s} \in [0,1]$ (see Figure 7.4). These polynomials have two interesting features

- They are orthogonal, i.e.

$$\int_0^1 \ell_i(\tau)\ell_j(\tau)\,\mathrm{d}\tau = 0 \quad \text{if} \quad i \neq j \tag{7.28}$$

- They satisfy:

$$\ell_i(\tau_j) = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \tag{7.29}$$

  resulting in the following property on p:

$$\mathrm{p}(\tau_i, \mathrm{K}) = \mathrm{K}_i, \quad i = 1, \dots, s \tag{7.30}$$

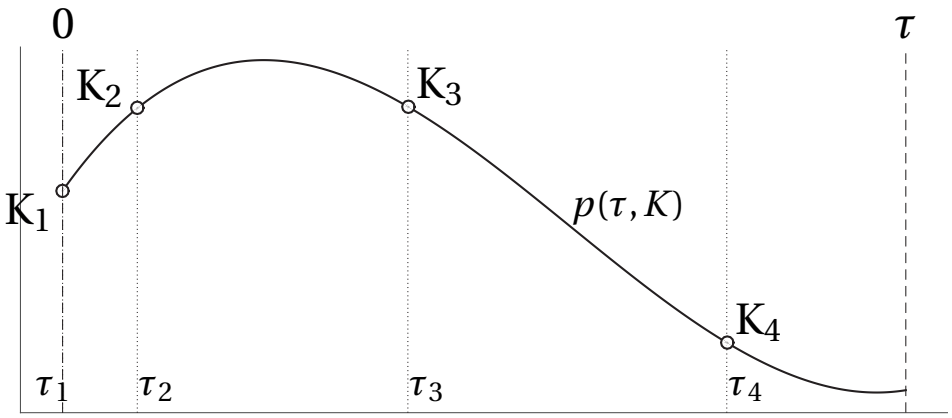


Figure 7.5: Illustration of the polynomial $\mathrm{p}(t,K) = \sum_{i=1}^{s} \mathrm{K}_i \ell_i(\tau)$ for $s = 4$ and for an arbitrary coefficient vector $\mathrm{K} \in \mathbb{R}^4$, and $\mathrm{K}_i \in \mathbb{R}$. One can observe the property (7.30), i.e. $p(\tau_i, v_k) = \mathrm{K}_i$.

7.4.2 Interpolation of the trajectories

The key idea behind the collocation method is to approximate[14] the true model trajectories by using[15]:

$$\dot{\mathrm{x}}(t_k + \tau \cdot \Delta t) = \mathrm{p}(\tau, \mathrm{K}) = \sum_{i=1}^{s} \mathrm{K}_i \ell_i(\tau) \tag{7.31}$$

on the interval $[t_k,\, t_{k+1}]$ (corresponding to $\tau \in [0,\, 1]$) by selecting the coefficients $\mathrm{K}_{1,\ldots,s} \in \mathbb{R}^n$. Note that each interval $[t_k,\, t_{k+1}]$ (for $k = 0, \ldots, N-1$) will have its own, possibly different coefficients

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_1 \\ \vdots \\ \mathrm{K}_s \end{bmatrix} \tag{7.32}$$

where $\mathrm{K}_i \in \mathbb{R}^n$.

We should now formulate equations that let us determine the parameters K. The key idea here is to enforce the dynamics on the grid points $\tau_1, \ldots, \tau_s$, i.e. we want to determine the $\mathrm{K}_{1,\ldots,s}$ such that:

$$\dot{\mathrm{x}}\left(t_k + \Delta t \cdot \tau_j\right) = \mathrm{p}\left(\tau_j, \mathrm{K}\right) = \mathrm{f}\left(\mathrm{x}\left(t_k + \Delta t \cdot \tau_j\right), \mathrm{u}\left(t_k + \Delta t \cdot \tau_j\right)\right) \tag{7.33}$$

[14] sometimes the method is presented by considering that the trajectories $\mathrm{x}(t)$ themselves are approximated by the polynomials instead of their time derivatives $\dot{\mathrm{x}}(t)$. This does not change much the mathematics of the collocation schemes.

[15] Note that in the following developments we use the label x(.) to denote the trajectories provided by the integrator.

holds for $j = 1, \ldots, s$ (i.e. on each grid point τ_j). We first observe that

$$\dot{\mathrm{x}}\left(t_k + \tau_j \cdot \Delta t\right) = \mathrm{p}\left(\tau_j, \mathrm{K}\right) = \mathrm{K}_j \tag{7.34}$$

holds by construction from (7.30). In order to further specify the equations above, we need to relate $\mathrm{x}\left(t_k + \tau_j \cdot \Delta t\right)$ (i.e. the integral of (7.31)) to the coefficients K. We do this next. We first observe that:

$$\mathrm{x}(t_k + \tau \cdot \Delta t) = \mathrm{x}_k + \int_0^{\tau \cdot \Delta t} \dot{\mathrm{x}}(t_k + \nu)\, \mathrm{d}\nu \tag{7.35}$$

where x_k is the initial state of interval $[t_k,\, t_{k+1}]$. We can then make a change of variable $\nu = \xi \cdot \Delta t$, yielding $\mathrm{d}\nu = \Delta t \cdot \mathrm{d}\xi$, we can then rewrite (7.35) as:

$$\mathrm{x}(t_k + \tau \cdot \Delta t) = \mathrm{x}_k + \Delta t \int_0^{\tau} \dot{\mathrm{x}}(t_k + \Delta t \cdot \xi)\, \mathrm{d}\xi \tag{7.36}$$

We then use (7.31) in (7.36) to get:

$$\mathrm{x}(t_k + \tau \cdot \Delta t) \approx \mathrm{x}_k + \Delta t \int_0^{\tau} \sum_{i=1}^{s} \mathrm{K}_i \ell_i(\xi) \mathrm{d}\xi = \tag{7.37a}$$

$$= \mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i \int_0^{\tau} \ell_i(\xi) \mathrm{d}\xi \tag{7.37b}$$

$$= \mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i L_i(\tau) \tag{7.37c}$$

where we define:

$$L_i(\tau) = \int_0^{\tau} \ell_i(\xi) \mathrm{d}\xi \tag{7.38}$$

Note that $L_i(\tau)$ can be easily computed explicitly as it is simply the integration of the polynomial ℓ_i. We can then use (7.37c) in (7.33) to get:

$$\underbrace{\dot{\mathrm{x}}\left(t_k + \Delta t \cdot \tau_j\right)}_{=\mathrm{K}_j} = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i L_i(\tau_j),\, \mathrm{u}\left(t_k + \Delta t \cdot \tau_i\right)\right) \tag{7.39}$$

which should hold for all grid points $j = 1, \ldots, s$. It follows that on a given interval $[t_k,\, t_{k+1}]$

the **collocation equations** read as:

$$\mathrm{K}_1 = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i L_i(\tau_1),\, \mathrm{u}\left(t_k + \Delta t \cdot \tau_1\right)\right) \tag{7.40a}$$

$$\vdots \tag{7.40b}$$

$$\mathrm{K}_j = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i L_i(\tau_j),\, \mathrm{u}\left(t_k + \Delta t \cdot \tau_j\right)\right) \tag{7.40c}$$

$$\vdots \tag{7.40d}$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i L_i(\tau_s),\, \mathrm{u}\left(t_k + \Delta t \cdot \tau_s\right)\right) \tag{7.40e}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \cdot \sum_{i=1}^{s} \mathrm{K}_i L_i(1) \tag{7.40f}$$

At this stage, it is very useful to compare (7.40) to the RK equations (6.26) and realise that they are identical if one defines:

$$a_{ji} = L_i(\tau_j), \qquad b_i = L_i(1), \qquad c_j = \tau_j \tag{7.41}$$

We also observe that if one picks the grid points $\tau_1, \ldots, \tau_s$, then the polynomials $\ell_{1,\ldots,s}(\tau)$ are defined and so is their integrals $L_i(\tau)$. **It follows that the coefficients** (7.41) **of the Butcher tableau are entirely defined via the grid points** $\tau_1, \ldots, \tau_s$. We also construe here the role of the variables $\mathrm{K}_{1,\ldots,s}$ in RK methods. Indeed, (7.39) shows us that these variables are holding the state derivatives $\dot{\mathrm{x}}$ at the grid points $\tau_1, \ldots, \tau_s$.

Note that not all IRK methods are collocation methods, as one could choose a dense Butcher tableau with coefficients that do not satisfy (7.41) for any choice of grid points $\tau_1, \ldots, \tau_s$. Indeed, one can easily see that the Butcher tableau has $s^2 + 2s$ degrees of freedom, while the grid point selection offers only s degrees of freedom, such that not all choice of coefficients a, b, c can arise from (7.41).

We now need to briefly discuss how to choose the grid points $\tau_1, \ldots, \tau_s$ effectively. There are a few possible choices for a given number of stages s. The most commonly used for treating ODEs is the Gauss-Legendre method, which chooses the grid points $\tau_1, \ldots, \tau_s$ as the roots of the polynomial:

$$P_s(\tau) = \frac{1}{s!} \frac{\mathrm{d}^s}{\mathrm{d}\tau^s} \left[\left(\tau^2 - \tau\right)^s \right] \tag{7.42}$$

i.e. we select the grid points $\tau_1, \ldots, \tau_s$ such that:

$$P_s(\tau_j) = 0, \quad j = 1, \ldots, s \tag{7.43}$$

This selection rule may sound mysterious. Its motivation, though, has solid and deep roots in the Gauss quadrature theory, but we will not discuss this further here. Equipped with this rule, we have a procedure to build IRK methods (i.e. their Butcher tableau):

1. Select the number of stages s
2. Find the roots of (7.42) to get $\tau_1, \ldots, \tau_s$
3. Build the polynomials $\ell_1(\tau), \ldots, \ell_s(\tau)$ according to (7.27)
4. Build their integrals $L_i(\tau)$ according to (7.38)
5. Evaluate $L_i(\tau_j)$, $b_i = L_i(1)$ for $i, j = 1, \ldots, s$
6. Build a computer code to solve the collocation equations (7.40)

This procedure **builds an IRK method with order $o = 2s$, and A-stable**.

An additional benefit of using polynomial interpolation for simulation models is that the polynomials provide us with an approximation of the state trajectories at any point in time, i.e. unlike other E/IRK methods, which deliver only the states on the time grid $t_{0,\ldots,N}$, collocation methods can be prompted to deliver the states in-between, using (7.36), i.e. the true state trajectories (say $\mathrm{x}^{\mathrm{true}}$) is approximated at any time point in $[0, T]$ by:

$$\mathrm{x}^{\mathrm{true}}(t_k + \Delta t\tau) \approx \mathrm{x}_k + \Delta t \int_0^\tau \dot{\mathrm{x}}(t_k + \Delta t \cdot \xi)\,\mathrm{d}\xi = \mathrm{x}_k + \Delta t \sum_{i=1}^{s} \mathrm{K}_i \int_0^\tau \ell_i(\xi)\mathrm{d}\xi \tag{7.44}$$

Note that the order of approximation $o = 2s$ unfortunately only holds on the grid $t_{0,\ldots,N}$, such that this approximation is typically a bit worse than order $2s$.

7.5 RK METHODS FOR IMPLICIT ODES

We have so far discussed numerical integration methods for models in the explicit ODE form:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) \tag{7.45}$$

Some models, however, are easier to treat in an implicit ODE form:

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{x}, \mathrm{u}) = 0 \tag{7.46}$$

This is e.g. the case for models of complex mechanical system arising in Lagrange mechanics, taking the form (see (2.40)), where we use $\mathrm{v} \equiv \dot{\mathrm{q}}$:

$$\begin{bmatrix} I & 0 \\ 0 & W(\mathrm{q}) \end{bmatrix} \underbrace{\begin{bmatrix} \dot{\mathrm{q}} \\ \dot{\mathrm{v}} \end{bmatrix}}_{\equiv \dot{\mathrm{x}}} = \begin{bmatrix} \mathrm{v} \\ \mathrm{Q} + \nabla_{\mathrm{q}} \mathscr{L} - \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q})\mathrm{v} \right) \mathrm{v} \end{bmatrix} \tag{7.47}$$

and where the symbolic inverse of matrix $W(\mathrm{q})$ is very complex to write. In that case, it is best to *avoid* trying to form the explicit version of (7.47), i.e.

$$\begin{bmatrix} \dot{\mathrm{q}} \\ \dot{\mathrm{v}} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & W(\mathrm{q}) \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{v} \\ \mathrm{Q} + \nabla_{\mathrm{q}} \mathscr{L} - \frac{\partial}{\partial \mathrm{q}} \left(W(\mathrm{q})\mathrm{v} \right) \mathrm{v} \end{bmatrix} \tag{7.48}$$

and work with the implicit form (7.47) directly. One trivial, but not necessarily effective approach would be to use an explicit RK method while forming the matrix inverse numerically in (7.48) at every integrator step.

A usually more effective approach is to use an implicit RK method directly on the implicit equation (7.47). Note that we can easily write (7.47) as (7.46), using:

$$\mathrm{F}(\dot{\mathrm{x}},\mathrm{x}) = \begin{bmatrix} I & 0 \\ 0 & W(\mathrm{q}) \end{bmatrix} \dot{\mathrm{x}} - \begin{bmatrix} \mathrm{v} \\ \nabla_{\mathrm{q}}\mathscr{L} - \frac{\partial}{\partial \mathrm{q}}\big(W(\mathrm{q})\mathrm{v}\big)\mathrm{v} \end{bmatrix} = 0 \tag{7.49}$$

We should then focus on treating (7.46) numerically in implicit integrators. Interestingly enough, the IRK equations (6.26) apply almost directly to implicit models, using only a minor modification. Recall that (6.26) for an explicit model of the form $\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x},\mathrm{u})$ reads as:

$$\mathrm{K}_1 = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{u}(t_k + c_1\Delta t)\right) \tag{7.50a}$$

$$\vdots$$

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{u}(t_k + c_i\Delta t)\right) \tag{7.50b}$$

$$\vdots$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{u}(t_k + c_s\Delta t)\right) \tag{7.50c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_i\mathrm{K}_i \tag{7.50d}$$

Recall also that the variables $\mathrm{K}_{1,\dots,s}$ in RK method are holding the state derivatives $\dot{\mathrm{x}}$ at the grid points $\tau_1,\dots,\tau_s$. The modification of the IRK equations for treating (7.46) therefore read as:

$$\mathrm{F}\left(\mathrm{K}_1,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j,\ \mathrm{u}(t_k + c_1\Delta t)\right) = 0 \tag{7.51a}$$

$$\vdots$$

$$\mathrm{F}\left(\mathrm{K}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k + c_i\Delta t)\right) = 0 \tag{7.51b}$$

$$\vdots$$

$$\mathrm{F}\left(\mathrm{K}_s,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j,\ \mathrm{u}(t_k + c_s\Delta t)\right) = 0 \tag{7.51c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_j\mathrm{K}_j \tag{7.51d}$$

The equations to solve are then

$$\mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.)) := \begin{bmatrix} \mathrm{F}\left(\mathrm{K}_1,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j,\ \mathrm{u}(t_k + c_1\Delta t)\right) \\ \vdots \\ \mathrm{F}\left(\mathrm{K}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k + c_i\Delta t)\right) \\ \vdots \\ \mathrm{F}\left(\mathrm{K}_s,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j,\ \mathrm{u}(t_k + c_s\Delta t)\right) \end{bmatrix} = 0 \tag{7.52}$$

The rest works as IRK schemes for explicit ODEs (see algorithm “IRK for explicit ODEs").

Here it is useful to reflect on the IRK method in comparison to using an ERK method for solving the implicit ODE (7.46). Indeed, since the ODE does not readily deliver the state derivative $\dot{\mathrm{x}}$, even when using an explicit method, one would have to solve the implicit equation for $\dot{\mathrm{x}}$ at every stage of the RK step, typically via a Newton method. We can then observe that

- Deploying an explicit RK method would then requiring solving s implicit equations of size n (the size of the state space), in order to get an order $o = s$ (for $s \leq 4$).
- Deploying an implicit IRK method requires solving one implicit equation of size $n \cdot s$, in order to get an order $o = 2s$ (for collocation methods).

Comparing formally the efficiency of the two approaches can be tricky, but the bottom line here is that if the ODE model is implicit, using an implicit RK method to perform the simulation can be a very good choice, independently of questions of stiffness.

7.6 RK METHODS FOR IMPLICIT DAEs

In order to close this section, it remains to be discussed how to treat DAEs numerically. We have in fact already all the tools required to tackle this problem, we only need to clarify a few points carefully. Let us consider DAEs in the fully implicit form (5.22) recalled here:

$$\mathrm{F}(\dot{\mathrm{x}}, \mathrm{z}, \mathrm{x}, \mathrm{u}) = 0 \tag{7.53}$$

DAEs can be treated very similarly to implicit ODEs, but we need to understand how the algebraic variables z are treated here.

At every time step, the algebraic variables z ought to be considered as “free" variables that need to be determined independently of the other time steps, and they need to be adjusted so that (7.53) holds. More specifically, while an implicit ODE ought to be treated via imposing

$$\mathrm{F}\left(\mathrm{K}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k + c_i\Delta t)\right) = 0 \tag{7.54}$$

for $i = 1, \dots, s$ of each RK step $\mathrm{x}_k \to \mathrm{x}_{k+1}$, equivalently for a DAE of the form (7.53), we ought to impose

$$\mathrm{F}\left(\mathrm{K}_i,\ \mathrm{z}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k + c_i \Delta t)\right) = 0 \tag{7.55}$$

for $i = 1, \dots, s$ of each RK step $\mathrm{x}_k \to \mathrm{x}_{k+1}$. Note that here the variables $\mathrm{z}_i \in \mathbb{R}^m$ operate as "unknown" that must be determined alongside the variables $\mathrm{K}_i \in \mathbb{R}^n$. The complete IRK equations for DAEs then read as:

$$\mathrm{r}(\mathrm{w}, \mathrm{x}_k, \mathrm{u}(.)) := \begin{bmatrix} \mathrm{F}\left(\mathrm{K}_1,\ \mathrm{z}_1,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j,\ \mathrm{u}(t_k + c_1 \Delta t)\right) \\ \vdots \\ \mathrm{F}\left(\mathrm{K}_i,\ \mathrm{z}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k + c_i \Delta t)\right) \\ \vdots \\ \mathrm{F}\left(\mathrm{K}_s,\ \mathrm{z}_s,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j,\ \mathrm{u}(t_k + c_s \Delta t)\right) \end{bmatrix} = 0 \tag{7.56}$$

where we gathered the $\mathrm{K}_i, \mathrm{z}_i$ variables in:

$$\mathrm{w} = \begin{bmatrix} \mathrm{K}_1 \\ \vdots \\ \mathrm{K}_s \\ \mathrm{z}_1 \\ \vdots \\ \mathrm{z}_s \end{bmatrix} \tag{7.57}$$

Note that if $\mathrm{x} \in \mathbb{R}^n$ and $\mathrm{z} \in \mathbb{R}^m$, then function F returns vectors of dimension $\mathbb{R}^{n+m}$. It follows that $\mathrm{w} \in \mathbb{R}^{s(n+m)}$. Moreover, the "residual" function r in (7.56) returns a vector of dimension $\mathbb{R}^{s(n+m)}$, and that solving (7.56) must provide the variables w. The set of equations (7.56) must be solved at each RK step $\mathrm{x}_k \to \mathrm{x}_{k+1}$ in order to build the complete simulation on the time interval $[0,\ T]$, and is done as previously using the Newton method.

For the sake of clarity, let us write down explicitly the pseudo-code required to perform the numerical simulation of a fully implicit DAE model.

Algorithm: IRK for Fully Implicit DAEs

Input: Initial conditions x_0, input profile $\mathrm{u}(.)$, Butcher tableau, step size Δt

for $k = 0, \ldots, N-1$ **do**

Guess for w (one can e.g. use $\mathrm{K}_i = \mathrm{x}_k$, $\mathrm{z}_i = 0$)

while $\|\mathrm{r}(\mathrm{w}, \mathrm{x}_k, \mathrm{u}(.))\| > \mathrm{Tol}$ **do**

Compute the solution $\Delta\mathrm{w}$ to the linear system:

$$\frac{\partial \mathrm{r}(\mathrm{w}, \mathrm{x}_k, \mathrm{u}(.))}{\partial \mathrm{w}} \Delta\mathrm{w} + \mathrm{r}(\mathrm{w}, \mathrm{x}_k, \mathrm{u}(.)) = 0 \tag{7.58}$$

with r given by (7.56). Update:

$$\mathrm{w} \leftarrow \mathrm{w} + \alpha \Delta\mathrm{w} \tag{7.59}$$

for some step size $\alpha \in]0, 1]$ (a full step $\alpha = 1$ generally works for implicit integrators)

Take RK step:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_i \mathrm{K}_i \tag{7.60}$$

return $\mathrm{x}_{1,\ldots,N}$

Note that the remarks at the end of section 7.5 also hold here, i.e. in order to deploy an explicit RK method on the DAE (7.53), one would have to solve it for $\dot{\mathrm{x}}$ and z at every stage of the RK steps, typically requiring the deployment of a Newton method.

7.6.1 RK METHOD FOR SEMI-EXPLICIT DAE MODELS

Adapting the approach presented above to the semi-explicit DAE case is fairly straightforward. Let us do it here nonetheless. Recall that semi-explicit DAEs are in the form:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{z}, \mathrm{x}, \mathrm{u}) \tag{7.61a}$$

$$0 = \mathrm{g}(\mathrm{z}, \mathrm{x}, \mathrm{u}) \tag{7.61b}$$

Here we ought to impose at each $i = 1, \ldots, s$:

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{z}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij} \mathrm{K}_j,\ \mathrm{u}(t_k + c_i \Delta t)\right) \tag{7.62a}$$

$$0 = \mathrm{g}\left(\mathrm{z}_i,\ \mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij} \mathrm{K}_j,\ \mathrm{u}(t_k + c_i \Delta t)\right) \tag{7.62b}$$

hence we form the RK equations:

$$\mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.)) := \begin{bmatrix} \mathrm{f}\left(\mathrm{z}_1,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{1j}\mathrm{K}_j,\ \mathrm{u}(t_k+c_1\Delta t\right)-\mathrm{K}_1 \\ \mathrm{g}\left(\mathrm{z}_1,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{1j}\mathrm{K}_j,\ \mathrm{u}(t_k+c_1\Delta t\right) \\ \vdots \\ \mathrm{f}\left(\mathrm{z}_i,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k+c_i\Delta t\right)-\mathrm{K}_i \\ \mathrm{g}\left(\mathrm{z}_i,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{ij}\mathrm{K}_j,\ \mathrm{u}(t_k+c_i\Delta t\right) \\ \vdots \\ \mathrm{f}\left(\mathrm{z}_s,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{sj}\mathrm{K}_j,\ \mathrm{u}(t_k+c_s\Delta t)\right)-\mathrm{K}_s \\ \mathrm{g}\left(\mathrm{z}_s,\ \mathrm{x}_k+\Delta t\sum_{j=1}^{s} a_{sj}\mathrm{K}_j,\ \mathrm{u}(t_k+c_s\Delta t\right) \end{bmatrix} = 0 \tag{7.63}$$

And apply the same procedure as described several times above to perform the simulation.

Let us know make a connection between the problem of DAE differential index and numerical simulations. Recall that a semi-explicit DAE (7.61) is of index 1 (i.e. an “easy" DAE) if the Jacobian $\frac{\partial \mathrm{g}}{\partial \mathrm{z}}$ is full rank (i.e. invertible) on the system trajectories. In order to make a simple point, let us consider an IRK method having a single stage $s = 1$ to treat the DAE. In that simple case, we observe that the RK equations (7.63) boil down to:

$$\mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.)) = \begin{bmatrix} \mathrm{f}(\mathrm{z}_1,\ \mathrm{x}_k+\Delta t a_{11}\mathrm{K}_1,\ \mathrm{u}(t_k+c_1\Delta t)-\mathrm{K}_1 \\ \mathrm{g}(\mathrm{z}_1,\ \mathrm{x}_k+\Delta t a_{11}\mathrm{K}_1,\ \mathrm{u}(t_k+c_1\Delta t) \end{bmatrix} = 0 \tag{7.64}$$

and that the Jacobian of these equations read as:

$$\frac{\partial \mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.))}{\partial \mathrm{w}} = \begin{bmatrix} \frac{\partial \mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.))}{\partial \mathrm{K}_1} & \frac{\partial \mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.))}{\partial \mathrm{z}_1} \end{bmatrix} = \begin{bmatrix} \Delta t a_{11}\frac{\partial \mathrm{f}(\mathrm{x},\mathrm{u})}{\partial \mathrm{x}}-I & \frac{\partial \mathrm{f}(\mathrm{x},\mathrm{u})}{\partial \mathrm{z}} \\ \Delta t a_{11}\frac{\partial \mathrm{g}(\mathrm{x},\mathrm{u})}{\partial \mathrm{x}} & \frac{\partial \mathrm{g}(\mathrm{x},\mathrm{u})}{\partial \mathrm{z}} \end{bmatrix} \Bigg|_{\substack{\mathrm{x}=\mathrm{x}_k+\Delta t a_{11}\mathrm{K}_1 \\ \mathrm{z}=\mathrm{z}_1 \\ \mathrm{u}=\mathrm{u}(t_k+c_1\Delta t)}} \tag{7.65}$$

Suppose then that we let $\Delta t \to 0$, i.e. we investigate the behavior of the method for an “infinite" accuracy (i.e. very high). Then the Jacobian matrix tends to:

$$\lim_{\Delta t\to 0} \frac{\partial \mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.))}{\partial \mathrm{w}} = \begin{bmatrix} -I & \frac{\partial \mathrm{f}}{\partial \mathrm{z}} \\ 0 & \frac{\partial \mathrm{g}}{\partial \mathrm{z}} \end{bmatrix} \tag{7.66}$$

which is full rank (invertible) if and only if $\frac{\partial \mathrm{g}(\mathrm{x},\mathrm{u})}{\partial \mathrm{z}}$ is full rank. Recall that we need to deploy a Newton method in order to solve $\mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.)) = 0$, where we need the Jacobian $\frac{\partial \mathrm{r}(\mathrm{w},\mathrm{x}_k,\mathrm{u}(.))}{\partial \mathrm{w}}$ to be invertible, and therefore requires $\frac{\partial \mathrm{g}(\mathrm{x},\mathrm{u})}{\partial \mathrm{z}}$ to be full rank. We can therefore conclude from this simple example that

The IRK methods we have investigated should not be deployed on DAEs of index above 1.

Note that if $\Delta t > 0$, the rank of the Jacobian matrix may be nonetheless full, and one may conclude that the method is nonetheless fine. However, the small analysis above shows us that “something is intrinsically wrong" when deploying the IRK schemes we have studied on high-index DAEs (index above 1). We will leave out of this course further analysis of this question. Before closing this section, it ought to be underlined here that one can deploy some of the RK schemes we have investigated on high-index DAEs, and get a trajectory computed from the method. I.e. there may not be a clear sign when deploying the method that “something is not right". The trajectories obtained, however, are typically non-sensical. Hence one ought to check the index of the DAE before trusting the output of a classic RK method.

8 SENSITIVITY OF SIMULATIONS

Our last section will deal with a topic that is crucial in many fields of engineering that make use of numerical simulations. In many problems it is important not only to compute accurate and reliable simulations for an ODE or a DAE, but also to get some information on how this simulation would be affected by a change in the model parameters. To be a bit more formal, let us consider an ODE depending on a fixed parameter p:

$$\dot{\mathrm{x}} = \mathrm{f}\big(\mathrm{x}, \mathrm{p}\big), \qquad \mathrm{x}(0) = \mathrm{x}_0 \tag{8.1}$$

Note that we will omit the inputs in our developments here, and discuss in the end what role they play. Suppose that we have computed a simulation of (8.1), i.e. we have a sequence:

$$\mathrm{x}_k \approx \mathrm{x}(t_k) \tag{8.2}$$

on a given time grid $t_0, \ldots, t_N$. Clearly, if one was to modify “a bit" either the initial conditions x_0 and/or the parameters p, then the sequence $\mathrm{x}_{1,\ldots,N}$ would be affected. For nonlinear dynamics f, one cannot assess the impact of changing the parameters and/or initial conditions x_0 on the simulation without redoing the entire simulations. However, one can use first-order arguments and compute the approximate effect of changing the parameters and initial conditions i.e. one can try to compute:

$$\frac{\partial \mathrm{x}_k}{\partial \mathrm{x}_0}, \qquad \frac{\partial \mathrm{x}_k}{\partial \mathrm{p}} \tag{8.3}$$

for $k = 1, \ldots, N$, evaluated at x_0, p given. The sensitivity of simulations then deals with computing these derivatives.

8.1 VARIATIONAL APPROACH

Before giving the answer to the question raised above, let us investigate the derivative of the trajectories $\mathrm{x}(t)$ generated by the ODE (8.1) with respect to p, x_0, i.e. let us forget for

now the numerical aspect of the problem. The question of computing

$$\frac{\partial \mathrm{x}(t)}{\partial \mathrm{p}} \qquad \frac{\partial \mathrm{x}(t)}{\partial \mathrm{x}_0} \tag{8.4}$$

can appear tricky, but let us consider the following construction. Suppose that we know the trajectory $\mathrm{x}(t)$ associated to (8.1). We observe that:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathrm{x}(t)}{\partial \mathrm{x}_0} = \frac{\partial \dot{\mathrm{x}}(t)}{\partial \mathrm{x}_0} = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\frac{\partial \mathrm{x}(t)}{\partial \mathrm{x}_0} \tag{8.5a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathrm{x}(t)}{\partial \mathrm{p}} = \frac{\partial \dot{\mathrm{x}}(t)}{\partial \mathrm{p}} = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\frac{\partial \mathrm{x}(t)}{\partial \mathrm{p}} + \frac{\partial \mathrm{f}}{\partial \mathrm{p}} \tag{8.5b}$$

where all Jacobians are evaluated at $\mathrm{x}(t)$ and p. We moreover observe that

$$\frac{\partial \mathrm{x}(0)}{\partial \mathrm{x}_0} = I, \qquad \frac{\partial \mathrm{x}(0)}{\partial \mathrm{p}} = 0 \tag{8.6}$$

Let us label:

$$A(t) = \frac{\partial \mathrm{x}(t)}{\partial \mathrm{x}_0}, \qquad B(t) = \frac{\partial \mathrm{x}(t)}{\partial \mathrm{p}} \tag{8.7}$$

We can then write the dynamic system for A and B using (8.5)-(8.6):

$$\dot{A}(t) = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}A(t) \qquad A(0) = I \tag{8.8a}$$

$$\dot{B}(t) = \frac{\partial \mathrm{f}}{\partial \mathrm{x}}B(t) + \frac{\partial \mathrm{f}}{\partial \mathrm{p}} \qquad B(0) = 0 \tag{8.8b}$$

We then observe that (8.8) is a linear time-varying system with a matrix-based state space (A and B). It is often referred to as the *variational equations* associated to (8.1). In principle, one could answer the sensitivity question raised above by performing the integration of the dynamic (8.1) together with (8.8) using any numerical integration method.

Such an approach, however, would not deliver exact derivatives. Indeed, the variational equations are expressing the sensitivity of the trajectories of the system, whereas the sequence $\mathrm{x}_{1,\dots,N}$ arises from numerical simulation and it approximates the trajectories. Moreover, a numerical integration of (8.8) is also an approximation of the trajectories $A(t)$, $B(t)$. As a result the sensitivities generated can be quite inaccurate.

For these reasons, the variational approach is not very often used to treat the sensitivity question, and rather replaced by so called *Algorithmic Differentiation* methods, which seek to compute *exact derivatives* of the inexact simulation of the dynamics. Let us briefly study these techniques next.

8.2 Algorithmic Differentiation of the explicit Euler scheme

As usual, it will be easiest to first consider the problem of sensitivity computation for the explicit Euler scheme:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}\left(\mathrm{x}_k, \mathrm{p}\right) \tag{8.9}$$

Computing (8.3) for the simulation (8.9) boils down to a careful application of the chain rule on (8.9). We observe that:

$$\frac{\partial \mathrm{x}_{k+1}}{\partial \mathrm{x}_0} = \frac{\partial \mathrm{x}_k}{\partial \mathrm{x}_0} + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\frac{\partial \mathrm{x}_k}{\partial \mathrm{x}_0}, \qquad \frac{\partial \mathrm{x}_0}{\partial \mathrm{x}_0} = I \tag{8.10a}$$

$$\frac{\partial \mathrm{x}_{k+1}}{\partial \mathrm{p}} = \frac{\partial \mathrm{x}_k}{\partial \mathrm{p}} + \Delta t \left(\frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\frac{\partial \mathrm{x}_k}{\partial \mathrm{p}} + \frac{\partial \mathrm{f}}{\partial \mathrm{p}}\right), \qquad \frac{\partial \mathrm{x}_0}{\partial \mathrm{p}} = 0 \tag{8.10b}$$

where all Jacobians are evaluated on the baseline simulation $\mathrm{x}_{1,\ldots,N}$ and for the parameter value p. We observe that (8.10) is in fact a discrete linear dynamic system, assigning dynamics to the matrices (8.3). Indeed, let us label

$$A_k = \frac{\partial \mathrm{x}_k}{\partial \mathrm{x}_0}, \qquad B_k = \frac{\partial \mathrm{x}_k}{\partial \mathrm{p}} \tag{8.11}$$

And rewrite (8.10) as:

$$A_{k+1} = \left(I + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\right) A_k, \qquad A_0 = I \tag{8.12a}$$

$$B_{k+1} = \left(I + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\right) B_k + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{p}}, \qquad B_0 = 0 \tag{8.12b}$$

This procedure is labelled Algorithmic Differentiation. An explicit Euler integrator with sensitivity generation can then be implemented using the following code:

Algorithm: Explicit Euler with sensitivity generation

Input: Initial conditions x_0, parameter p, Δt

Set $A_0 = I$, $B_0 = 0$

for $k = 0, \ldots, N-1$ **do**

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \mathrm{f}\left(\mathrm{x}_k, \mathrm{p}\right) \tag{8.13a}$$

$$A_{k+1} = \left(I + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\right)\Big|_{\mathrm{x}_k, \mathrm{p}} A_k \tag{8.13b}$$

$$B_{k+1} = \left(I + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}_k}\right)\Big|_{\mathrm{x}_k, \mathrm{p}} B_k + \Delta t \left.\frac{\partial \mathrm{f}}{\partial \mathrm{p}}\right|_{\mathrm{x}_k, \mathrm{p}} \tag{8.13c}$$

return $\mathrm{x}_{1,\ldots,N}$, $A_{1,\ldots,N}$ and $B_{1,\ldots,N}$

Note that if only the final state x_N is needed, together with the associated sensitivities A_N and B_N, then a lot of memory can be saved by reusing the memory cells assigned to storing

x, A and B.

In contrast to the variational approach, the sensitivities delivered by processing the discrete dynamics (8.8) provides the exact (up to machine precision) derivative of the sequence $\mathrm{x}_{1,\dots,N}$ delivered by the explicit Euler scheme, even though that sequence is not an exact (up to machine precision) approximation of the true trajectories.

An attentive reader, however, may have observed that if one treats both (8.1)-(8.8) using an explicit Euler scheme, then one recovers the algorithm above. Hence in this specific case, the variational equations and the Algorithmic Differentiation coincide. This observation is not true in general.

8.3 Algorithmic Differentiation of explicit Runge-Kutta methods

The explicit RK equations for (8.1) arising from a lower-diagonal Butcher tableau can be written as:

$$\mathrm{K}_1 = \mathrm{f}\big(\mathrm{x}_k, \mathrm{p}\big) \tag{8.14a}$$

$$\vdots$$

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{i-1} a_{ij}\mathrm{K}_j, \mathrm{p}\right) \tag{8.14b}$$

$$\vdots$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s-1} a_{sj}\mathrm{K}_j, \mathrm{p}\right) \tag{8.14c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_j \mathrm{K}_j \tag{8.14d}$$

(observe that the summations in the formation of the $\mathrm{K}_{1,\dots,s}$ entail by construction an explicit method). Although it is a bit tedious, one can apply chain rules to (8.14) in order to extract the sensitivities. Indeed, one can observe that:

- The differentiation of (8.14d) reads as:

$$\frac{\partial \mathrm{x}_{k+1}}{\partial \mathrm{x}_0} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) \frac{\partial \mathrm{x}_k}{\partial \mathrm{x}_0} \tag{8.15a}$$

$$\frac{\partial \mathrm{x}_{k+1}}{\partial \mathrm{p}} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) \frac{\partial \mathrm{x}_k}{\partial \mathrm{p}} + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{p}} \tag{8.15b}$$

- The differentiation of (8.14a)-(8.14c) reads as:

$$\frac{\partial \mathrm{K}_i}{\partial \mathrm{x}_k} = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \cdot \left(I + \Delta t \sum_{j=1}^{i-1} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) \tag{8.16a}$$

$$\frac{\partial \mathrm{K}_i}{\partial \mathrm{p}} = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \cdot \left(I + \Delta t \sum_{j=1}^{i-1} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{p}}\right) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{p}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \tag{8.16b}$$

A pseudo-code deploying the above principle reads as follows.

Algorithm: Explicit RK scheme with sensitivity generation

Input: Initial conditions x_0, parameter p, Δt

Set $A_0 = I$, $B_0 = 0$

for $k = 0, \ldots, N-1$ **do**

  for $i = 1, \ldots, s$ **do**

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{i-1} a_{ij}\mathrm{K}_j, \mathrm{p}\right) \tag{8.17a}$$

$$\frac{\partial \mathrm{K}_i}{\partial \mathrm{x}_k} = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \cdot \left(I + \Delta t \sum_{j=1}^{i-1} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) \tag{8.17b}$$

$$\frac{\partial \mathrm{K}_i}{\partial \mathrm{p}} = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \cdot \left(I + \Delta t \sum_{j=1}^{i-1} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{p}}\right) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{p}}\right|_{\mathrm{x}_k+\Delta t\sum_{j=1}^{i-1} a_{sj}\mathrm{K}_j,\mathrm{p}} \tag{8.17c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_j \mathrm{K}_j \tag{8.18a}$$

$$A_{k+1} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) A_k \tag{8.18b}$$

$$B_{k+1} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k}\right) B_k + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{p}} \tag{8.18c}$$

return $\mathrm{x}_{1,\ldots,N}$, $A_{1,\ldots,N}$ and $B_{1,\ldots,N}$

Note that these computations are tedious to derive and to program. Fortunately, very efficient Algorithmic Differentiation tools such as e.g. CasADi can perform these operations automatically.

Note that the algorithm above can only work for explicit RK methods, i.e. RK methods having a lower-diagonal Butcher tableau. Recall that the implicit RK equations generally read as:

$$\mathrm{K}_1 = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{p}\right) \tag{8.19a}$$

$$\vdots$$

$$\mathrm{K}_i = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{p}\right) \tag{8.19b}$$

$$\vdots$$

$$\mathrm{K}_s = \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{p}\right) \tag{8.19c}$$

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_j \mathrm{K}_j \tag{8.19d}$$

Interestingly enough, the sensitivities of an implicit RK scheme are fairly straightforward to formulate and compute, i.e. easier than an explicit RK scheme. Recall that the implicit equations (8.19a)-(8.19c) are solved via a Newton method, i.e. we write (7.14), recalled here:

$$\mathrm{r}\left(\mathrm{K}, \mathrm{x}_k, \mathrm{p}\right) := \begin{bmatrix} \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{1j}\mathrm{K}_j, \mathrm{u}(t_k + c_1\Delta t)\right) - \mathrm{K}_1 \\ \vdots \\ \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{ij}\mathrm{K}_j, \mathrm{u}(t_k + c_i\Delta t)\right) - \mathrm{K}_i \\ \vdots \\ \mathrm{f}\left(\mathrm{x}_k + \Delta t \sum_{j=1}^{s} a_{sj}\mathrm{K}_j, \mathrm{u}(t_k + c_s\Delta t)\right) - \mathrm{K}_s \end{bmatrix} = 0 \tag{8.20}$$

and deploy a Newton method on the “unknown"

$$\mathrm{K} = \begin{bmatrix} \mathrm{K}_1 \\ \vdots \\ \mathrm{K}_s \end{bmatrix} \tag{8.21}$$

i.e. we iterate (see algorithm “IRK for explicit ODEs"):

$$\mathrm{K} \leftarrow \mathrm{K} - \alpha \left(\frac{\partial \mathrm{r}\left(\mathrm{K}, \mathrm{x}_k, \mathrm{p}\right)}{\partial \mathrm{K}}\right)^{-1} \mathrm{r}\left(\mathrm{K}, \mathrm{x}_k, \mathrm{p}\right) \tag{8.22}$$

Recall the Implicit Function theorem 5, and in particular equation (3.40). Here one ought to view

$$\mathrm{r}\left(\mathrm{K}, \mathrm{x}_k, \mathrm{p}\right) = 0 \tag{8.23}$$

as making K implicitly a function of x_k, p, and use equation (3.40) to deduce that:

$$\frac{\partial K}{\partial p} = -\frac{\partial r\left(K, x_k, p\right)}{\partial K}^{-1} \frac{\partial r\left(K, x_k, p\right)}{\partial p} \tag{8.24a}$$

$$\frac{\partial K}{\partial x_k} = -\frac{\partial r\left(K, x_k, p\right)}{\partial K}^{-1} \frac{\partial r\left(K, x_k, p\right)}{\partial x_k} \tag{8.24b}$$

One ought to observe that the matrix inverse (or rather the matrix factorization for the linear-algebra savy among the readers) needed in (8.22) can be reused in (8.24), such that the latter is inexpensive to compute. We then observe that (8.15) can be readily used to compute the sensitivities of x_{k+1} as in the explicit RK method (see (8.15)). We can summarize these observations in the following pseudo-code.

Algorithm: IRK for explicit ODEs with senstivities

Input: Initial conditions x_0, input profile u(.), Butcher tableau, step size Δt

Set $A_0 = I$, $B_0 = 0$

for $k = 0, \ldots, N-1$ **do**

Guess for K (one can e.g. use $\mathrm{K}_i = \mathrm{x}_k$)

while $\|\mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.))\| > \mathrm{Tol}$ **do**

Compute the solution $\Delta\mathrm{w}$ to the linear system:

$$\frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.))}{\partial \mathrm{K}} \Delta\mathrm{K} + \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{u}(.)) = 0 \tag{8.25}$$

with r given by (7.14). Update:

$$\mathrm{K} \leftarrow \mathrm{K} + \alpha \Delta\mathrm{K} \tag{8.26}$$

for some step size $\alpha \in]0, 1]$ (a full step $\alpha = 1$ generally works for implicit integrators)

Reuse the matrix factorization required to solve (8.25) to compute:

$$\frac{\partial \mathrm{K}}{\partial \mathrm{p}} = -\frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{p})}{\partial \mathrm{K}}^{-1} \frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{p})}{\partial \mathrm{p}} \tag{8.27a}$$

$$\frac{\partial \mathrm{K}}{\partial \mathrm{x}_k} = -\frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{p})}{\partial \mathrm{K}}^{-1} \frac{\partial \mathrm{r}(\mathrm{K}, \mathrm{x}_k, \mathrm{p})}{\partial \mathrm{x}_k} \tag{8.27b}$$

Take RK step with sensitivities:

$$\mathrm{x}_{k+1} = \mathrm{x}_k + \Delta t \sum_{j=1}^{s} b_j \mathrm{K}_j \tag{8.28a}$$

$$A_{k+1} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k} \right) A_k \tag{8.28b}$$

$$B_{k+1} = \left(I + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{x}_k} \right) B_k + \Delta t \sum_{j=1}^{s} b_j \frac{\partial \mathrm{K}_j}{\partial \mathrm{p}} \tag{8.28c}$$

return

8.5 Sensitivity with respect to inputs

Let us finish these lecture notes with discussing the computation of the sensitivity of a simulation with respect to the input u applied to the dynamics $\dot{\mathrm{x}} = \mathrm{f}(\mathrm{p}, \mathrm{u})$. Since u(.) is, in general, a profile (i.e. a function of time in the interval $[0, T]$) as opposed to a finite set of parameters p, discussing sensitivities in the form explored here is a priori inadequate. However, the classic approach in numerical simulations is to consider that the input profile u(.) is in fact parameterized by a finite set of parameters, i.e. we consider that the input

is given by:

$$\mathrm{u}(t, \mathrm{p}) \tag{8.29}$$

Any derivative in the algorithms above is then simply replaced by its chain-rule expansion:

$$\frac{\partial .}{\partial \mathrm{p}} = \frac{\partial .}{\partial \mathrm{u}} \frac{\partial \mathrm{u}}{\partial \mathrm{p}} \tag{8.30}$$

e.g.

$$\frac{\partial \mathrm{f}}{\partial \mathrm{p}} = \frac{\partial \mathrm{f}}{\partial \mathrm{u}} \frac{\partial \mathrm{u}}{\partial \mathrm{p}} \tag{8.31}$$

The input parametrization can take many forms. It can e.g. be convenient to use an approach similar to the one used in collocation-based IRK schemes. In that context, the input profile is e.g. given by:

$$\mathrm{u}\left(t_k + \tau \Delta t, \mathrm{p}\right) = \sum_{i=1}^{s} \mathrm{p}_{k,i} \ell_i(\tau) \tag{8.32}$$

where we have:

$$\mathrm{p} = \begin{bmatrix} \mathrm{p}_{0,1} \\ \vdots \\ \mathrm{p}_{0,s} \\ \vdots \\ \mathrm{p}_{N-1,1} \\ \vdots \\ \mathrm{p}_{N-1,s} \end{bmatrix} \tag{8.33}$$

Hence the input profile is piece-wise smooth, and interpolates the points $\mathrm{p}_{k,i}$ on the time intervals $[t_k,\, t_{k+1}]$ for $k = 0, N-1$. This principle is illustrated in Figure 8.1. A wide-spread alternative input parametrization is to use the piecewise constant approach, which uses:

$$\mathrm{u}(t_k + \tau \Delta t) = \mathrm{p}_k, \qquad k = 0, \ldots, N-1 \tag{8.34}$$

This principle is illustrated in Figure 8.2. All the sensitivity principles we studied above can be readily applied.

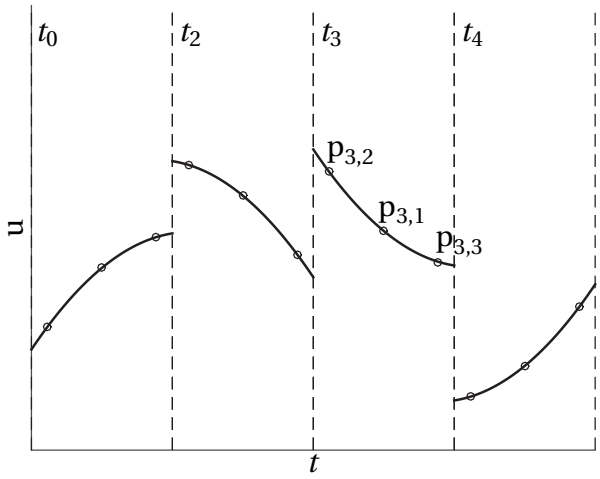


Figure 8.1: Illustration of the “collocation-based" parametrization of the inputs, for $N = 5$, and $s = 3$. Note that the input profile is piecewise-smooth.

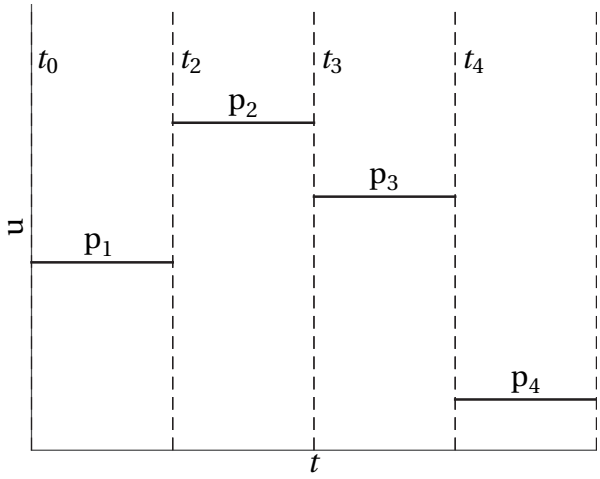


Figure 8.2: Illustration of the piecewise-constant approach to the input parametrization for $N = 4$.