Università degli Studi di Trento

Department of Industrial Engineering

Mechatronic Systems Simulation

Prof.: Bertolazzi Enrico, Biral Francesco

Course Notes

Matteo Dalle Vedove
matteo.dallevedove@studenti.unitn.it

Academic Year 2021-2022
March 29, 2022

Contents

Part I

Differential Algebraic Equations

Chapter 1

Ordinary Differential Equations and Numerical Solutions

To start the description of the **differential algebraic equations** **DAEs** it's firstly necessary to recall what **ordinary differential equations** **ODEs** are, what they mean and how to solve them.

In particular ordinary differential equations is a particular equation that involves a function (for example $y(x)$ depending from the independent variable x) and it's derivative as shown in this example:

$$y''(x) + xy'(x) + y(x) = \sin x \qquad \leftrightarrow \qquad y'' + xy' + y = \sin x$$

Ordinary differential equations (or system of ODEs) can be written in a ***standard form*** made by the **differential part**, where the first derivative is a function of itself, and the **initial condition** that set a specified value of the solution of the problem:

$$\begin{cases} y' = f(x, y(x)) = f(x, y) & : \text{differential part} \\ y(a) = y_a & : \text{initial condition} \end{cases} \tag{1.1}$$

Initial conditions are mandatory: the solution of the differential part lonely gives a *family* of solutions (parametric results) whose specific value can be determined only by knowing the value of the function at certain time. Considering the simple case of the differential $x' = 0$ with independent variable t, then the general solution is the class of all the constant functions $x(t) = c$ (with $c \in \mathbb{R}$). If a boundary condition is set (example $x(1) = 3$) then we can chose the particular solution (in this case $c = 3$ and so $x(t) = 3$).

Ordinary differential equations can also come in system as in the following example (with t as dependent variable) composed by 2 differential and 2 initial condition terms:

$$\begin{cases} x' = x + y \\ y' = e^x - y \\ x(0) = 0 \\ y(0) = 1 \end{cases}$$

Vectorial notation Considering the system of $n = 3$ differential equation depending from the independent variable t in the form

$$\begin{cases} z'(t) = x(t) + z(t) \\ w'(t) = z(t) \\ x'(t) = w(t)z(t) + t(t) \\ x(0) = w(0) = z(0) = 1 \end{cases}$$

then it can be rewritten in a vectorial form; considering in fact the substitutions $x(t) = y_1(t)$, $w(t) = y_2(t)$ and $z(t) = y_3(t)$ we have obtain the system

$$\begin{cases} y_3' = y_1 + y_3 \\ y_2' = y_3 \\ y_1' = y_2 y_3 + t \\ y_1(0) = y_2(0) = y_3(0) = 1 \end{cases} \Rightarrow \begin{cases} y_1' = y_2 y_3 + t \\ y_2' = y_3 \\ y_3' = y_1 + y_3 \\ y_1(0) = y_2(0) = y_3(0) = 1 \end{cases}$$

Considering the vector $\boldsymbol{y} = (y_1, y_2, y_3)$ we can so rewrite the original system of ODEs as

$$\begin{cases} \boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}) \\ \boldsymbol{y}(0) = \mathbf{1} \end{cases} \tag{1.2}$$

where

$$\boldsymbol{f} : \mathbb{R}^{n+1} \to \mathbb{R}^n = \begin{pmatrix} y_2 y_3 + t \\ y_3 \\ y_1 + y_3 \end{pmatrix}$$

In this case the domain of the function $\boldsymbol{f}$ is a vector of dimension $n + 1$: n related to the number of ODEs and 1 for the time dependency.

Order of an ODE The **order** of an ordinary differential equation is equal to the maximum order derivative appearing in the differential part of the system; considering as example

$$\begin{cases} y'' + y' + z' = 0 \\ z' + t = 0 \end{cases}$$

the order of such ordinary differential system is equal to 2 (associated to the term y'').

In general numerical methods are defined for 1^{st} order ODEs and so it's necessary (but most importantly possible) to convert any generic differential equation into a system of 1^{st} order ODEs by performing a change of variable.

Example 1.1: reduction to a system of ODEs of first order

Given the system of ODEs of the 3^{rd} order in the independent variable t defined as

$$\begin{cases} x''' + y' = x^2 + t \\ y'' + x = t^2 + 1 \\ x(0) = 0 \qquad y(0) = 0 \\ x'(0) = 0 \qquad y'(0) = 2 \\ x''(0) = 2 \end{cases}$$

the reduction to a system of first order ODEs is made by introducing the variable $z = x'$; defining instead the function $x'' = z' = w$ we also have that $z' = w$ hence $x''' = z'' = w'$; finally we can set

$y' = p$ hence $y'' = p$. The system so becomes

$$\begin{cases} w' + p = x^2 + t \\ p' + x = t^2 + 1 \\ x' = z \\ z' = w \\ y' = p \\ x(0) = 0 \qquad y(0) = 0 \\ z(0) = 0 \qquad p(0) = 2 \\ w(0) = 2 \end{cases}$$

This system present 3 more differential terms and so seems *more difficult*, however this formulation is numerically more suitable for the computation.

1.1 Existence of the solution

Given the general system of n ordinary differential equations in the standard form

$$\begin{cases} \boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}) \\ \boldsymbol{y}(a) = \boldsymbol{y}_a \end{cases}$$

using **Peano's theorem** we can state that if the vectorial map $\boldsymbol{f} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is continuous, then a solution exists in the neighbourhood of the point $(t = a, \boldsymbol{y} = \boldsymbol{y}_a)$. This theorem provide a sufficient condition to determine if a solution exists, but it doesn't state that the solution is unique.

Example 1.2: ODE with multiple solution

Considering the ordinary differential equation in the independent variable t

$$\begin{cases} y' = \sqrt{|y|} \\ y(0) = 0 \end{cases}$$

we can see that the map $f(y) = \sqrt{|y|}$ is continuous for all $y \in \mathbb{R}$, hence for Peano's theorem a solution must exists for t *sufficiently close* to 0. In particular we can observe that the function

$$y(t) = 0$$

is a solution of the system, in fact it matches the initial condition and we have that it's derivative $y' = \frac{dy}{dt} = 0$ is equal to the function $f(y) = \sqrt{|0|} = 0$.
However this is not the lonely solution, considering in fact the function

$$y(t) = \frac{t^2}{4}\text{sign}(t) = \begin{cases} \frac{t^2}{4} & t \geq 0 \\ -\frac{t^2}{4} & t < 0 \end{cases}$$

Observing that the derivative of such function can be regarded as

$$y'(t) = \begin{cases} \frac{t}{4} & t \geq 0 \\ -\frac{t}{2} & t < 0 \end{cases} \quad \Rightarrow \quad y'(t) = \frac{|t|}{2}$$

we can also check that the provided solution solves the differential part of the system, in fact

$$\sqrt{|y(t)|} = \sqrt{\left|\frac{t^2}{4}\text{sign}(t)\right|} = \sqrt{\frac{t^2}{4}} = \frac{|t|}{2} = y'(t)$$

In general when solving system of differential equation we want to be sure that the solution exists (using as example Peano's theorem) but that is also unique. In order to do so we have to defined the **Lipschitz continuity**:

$$f : \mathbb{R}^n \to \mathbb{R} \text{ is Lip. cont.} \quad \text{if } \exists L \in \mathbb{R} \quad \text{such that} \quad \big\|f(\boldsymbol{x}) - f(\boldsymbol{y})\big\| \leq L\big\|\boldsymbol{x} - \boldsymbol{y}\big\| \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R} \tag{1.3}$$

We can so state that a system of ordinary differential equation in the standard form has one solution if the map $\boldsymbol{f} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is continuous (from Peano) and is also lipschitzian.

Example 1.3: Lipschitz continuity

Considering the problem of example 1.2 we can prove that the system has multiple solution by checking that's not Lipschitz continuous. Known that the map $f(t, y) := \sqrt{|y|}$ is continuous we can apply Lipschitz definition in the particular case when one point is the origin of the the axis:

$$\begin{aligned} \big|f(0,y) - f(0,0)\big| &\leq L\big|x - 0\big| \\ \Big|\sqrt{|y|} - \sqrt{|0|}\Big| L &\leq |y| \\ \cancel{\sqrt{|y|}} &\leq L \cancel{\sqrt{|y|}} \sqrt{|y|} \\ L &\geq \frac{1}{\sqrt{|y|}} \end{aligned}$$

Observing that for $y \to 0$ the denominator converges to zero this means that L must diverge to ∞ meaning that the function is not lipschitzian: the solution exists (Peano's theorem) but it might not unique (for Lipschitz).

1.2 Taylor expansion

The main tool used for numerical approximate solution of (system of) ordinary differential equation is the **Taylor series expansion** that's used to approximate a function $f : \mathbb{R} \to \mathbb{R}$ of class $\mathcal{C}^\infty$ as a polynomial in the neighbourhood of a specific point x_0; given h as the deviation from the point x_0 for *small* values of h we can approximate the function

$$\begin{aligned} f(x_0 + h) &\approx a_0 + a_1 h + a_2 h^2 + a_3 h^3 + \cdots + a_n h^n \\ &\approx a_0 + \sum_{k=1}^{\infty} a_k h^k \end{aligned} \tag{1.4}$$

Evaluating both members allows to obtain the first coefficient $a_0 = f(x_0)$ of the Taylor expansion, in fact

$$f(x_0) = a_0 + \sum_{k=1}^{\infty} a_k 0^k = a_0$$

By differentiation we can also obtain other coefficients

$$\begin{aligned} f'(x_0 + h) &= a_1 + \sum_{k=2}^{\infty} a_k h^{k-1} k & \xrightarrow{h=0} \quad a_1 &= f'(x_0) \\ f''(x_0 + h) &= 2a_2 + \sum_{k=3}^{\infty} a_k h^{k-2} k(k-1) & \xrightarrow{h=0} \quad 2a_2 &= f''(x_0) \end{aligned}$$

Considering that in general each coefficient can be computed as $a_k = f^{(k)}(x_0)/k!$ we can better rewrite the Taylor series expansion of equation 1.4 as

$$f(x_0 + h) \approx \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} h^k \tag{1.5}$$

Truncation From a numerical standpoint the computation of the Taylor series is truncated to a order n and so we can use the formulation

$$f(x_0 + h) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} h^k + R_n(h) \tag{1.6}$$

where R_n is the reminder due to the truncation of the series that can be evaluated in multiple ways:

- using Peano's formulation the more formal definition of the reminder is

  $$R_n(h) = \int_{x_0}^{x_0+h} f^{(n+1)}(s) \frac{(s-h)^n}{n!}\, ds$$

  This formulation is still complex and numerically *unusable*;

- the Lagrange reminder in the form

  $$R_n(h) = f^{(n+1)}(\zeta) \frac{h^{n+1}}{(n+1)!} \qquad \text{with } \zeta \in (x_0, x_0 + h)$$

- the *big O* notation $R_n(h) = \mathcal{O}(h^{n+1})$; in particular we denote $g(x) = \mathcal{O}(f(x))$ if

  $$\exists C \in \mathbb{R} \quad \text{such that} \quad |g(x)| \leq C|f(x)|$$

- the *small o* notation $R_n(h) = o(h^n)$; we say that $g(x) = o(f(x))$ if f is lipschitzian (equation 1.3) and we have that

  $$\lim_{x \to 0} \frac{o(f(x))}{g(x)} = 0$$

Common Taylor expansions Examples of notable series expansion on the point $x_0 = 0$ for well-known functions are the exponential, the cosine and sine:

$$\begin{aligned} e^h &= \sum_{k=0}^{\infty} \frac{h^k}{k!} = 1 + h + \frac{h^2}{2} + \frac{h^3}{3!} + \frac{h^4}{4!} + \dots \\ \cos h &= \sum_{k=0}^{\infty} -1^k \frac{h^{2k}}{2k!} = 1 - \frac{h^2}{2!} + \frac{h^4}{4!} - \frac{h^6}{6!} + \dots \\ \sin h &= \sum_{k=0}^{\infty} -1^k \frac{h^{2k+1}}{(2k+1)!} = h - \frac{h^3}{3!} + \frac{h^5}{5!} - \frac{h^7}{7!} + \dots \end{aligned} \tag{1.7}$$

Existence of the expansion Considering the definition of the lagrangian reminder in the form $f^{(m)}(\zeta)\frac{h^m}{m!}$ if we truncate the series to higher order (bigger value of m) we observe that for $h < 1$ the term $h^m/m!$ tends to zero, and so if we ensure that the m-th derivative doesn't diverge we have that the Taylor series converge to the *real* function. However this sometimes can fail: considering as example the function

$$f(x) = \begin{cases} e^{-\frac{1}{x^2}} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

it's proven that the function is *smooth* ($f \in C^\infty$) and that the k-th derivative is in the form

$$\left(e^{-\frac{1}{x^2}}\right)^{(k)} = e^{-\frac{1}{x^2}} \frac{p_1(x)}{p_2(x)} \qquad p_1, p_2 \text{ polynomials}$$

and converges to zero for $x \to 0$. By applying the definition of the Taylor expansion in $x_0 = 0$ we so have that

$$f(0+h) = \sum_{k=0}^{\infty} f^{(k)}(0)\frac{h^k}{k!} = 0$$

This expansion correctly models the left-hand side of the function f but not the right side, and so the Taylor expansion fails.

1.2.1 Multi-variable functions

Until now we have defined the Taylor expansion of function with one variable in the form $f(x)$, but such concept should be extended to function of multiple variables. Considering the simple case of $f : \mathbb{R}^2 \to \mathbb{R}$ in order to perform the Taylor series we have to ensure a continuity up to an order m for the function, meaning

$$f(x,y) \in C^m \qquad \Leftrightarrow \qquad \frac{\partial^{i+j}}{\partial x^i \partial y^j} f(x,y) \in C \quad \forall i+j \leq m \tag{1.8}$$

We have the formal definition of the Taylor series for functions of one variable (equation 1.5) and so, as idea, we can *slice* the function f passing through a point (x_0, y_0) with a direction $(x - x_0, y - y_0) = (d_x, d_y)$ using a function

$$g(t) = f(x_0 + td_x, y_0 + td_y)$$

The idea is so to compute the Taylor series of this function in the neighbourhood of $t = 0$:

$$g(t) = g(0) + g'(t)t + g''(0)\frac{t^2}{2!} + g'''(0)\frac{t^3}{3!} + \dots$$

The term $g'(t)$relates to the total derivative of $f(x_0 + td_x, y_0 + td_y)$ respect to the variable t, meaning that

$$\begin{aligned}
g'(t) &= \frac{d}{dt} f\big(x_0 + t\,d_x, y_0 + t\,d_y\big) \\
&= \frac{\partial f(\dots)}{\partial x}\frac{d(x_0 + t\,d_x)}{dt} + \frac{\partial f(\dots)}{\partial y}\frac{d(y_0 + t\,d_y)}{dt} \\
&= \frac{\partial f(\dots)}{\partial x} d_x + \frac{\partial f(\dots)}{\partial y} d_y = \frac{\partial f(\dots)}{\partial x}(x - x_0) + \frac{\partial f(\dots)}{\partial y}(y - y_0) \\
g'(0) &= \frac{\partial f(x_0, y_0)}{\partial x}(x - x_0) + \frac{\partial f(x_0, y_0)}{\partial y}(y - y_0)
\end{aligned}$$

Using a similar methodology it's possible to compute the second order derivative of g as

$$\begin{aligned}
g''(t) &= \frac{d}{dt} g'(t) = \frac{d}{dt}\left(\frac{\partial f(\dots)}{\partial x}(x - x_0) + \frac{\partial f(\dots)}{\partial y}(y - y_0)\right) \\
&= \frac{\partial^2 f(\dots)}{\partial x^2}(x - x_0)^2 + \frac{\partial^2 f(\dots)}{\partial y^2}(y - y_0)^2 + 2\frac{\partial^2 f(\dots)}{\partial x\,\partial y}(x - x_0)(y - y_0)
\end{aligned}$$

Considering as more general statement to express the Taylor series respect to a point (x_0, y_0) moving with values $h = td_x$ and $k = td_y$ the series truncated to the second order is so

$$\begin{aligned}
f(x_0 + h, y_0 + k) = & f(x_0, y_0) + \left.\frac{\partial f}{\partial x}\right|_{(x_0,y_0)} h + \left.\frac{\partial f}{\partial y}\right|_{(x_0,y_0)} k \\
& + \frac{1}{2}\left.\frac{\partial^2 f}{\partial x^2}\right|_{(x_0,y_0)} h^2 + \frac{1}{2}\left.\frac{\partial^2 f}{\partial y^2}\right|_{(x_0,y_0)} k^2 + \left.\frac{\partial^2 f}{\partial x\,\partial y}\right|_{(x_0,y_0)} hk
\end{aligned} \tag{1.9}$$

This representation can be compacted using a vectorial/matrix notation condensing (x_0, y_0) in the vector $\boldsymbol{x}_0$ and the increment $(h, k) = \boldsymbol{h}$ we can define

$$f(\boldsymbol{x}_0 + \boldsymbol{h}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)\,\boldsymbol{h} + \frac{1}{2}\boldsymbol{h}^t\,\nabla^2 f(\boldsymbol{x}_0)\,\boldsymbol{h} + R_3(\|\boldsymbol{h}\|) \tag{1.10}$$

where $\nabla f(\boldsymbol{x}) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$ is the **gradient** of the function (and is a row vector) and $\nabla^2 f(\boldsymbol{x})$ is the **hessian matrix** of f. Note that this formulation is general and is valid for any multi-variable function $f : \mathbb{R}^n \to \mathbb{R}$ assuming that's at least C^2.

Higher order Taylor expansion In order to perform Taylor expansion with order greater than 2 it's necessary to use a **tensor** notation; in particular the expansion of a function $f : \mathbb{R}^n \to \mathbb{R}$ up to the order m is described by the equation

$$f(\boldsymbol{x}_0 + \boldsymbol{h}) = \sum_{k=0}^{m} \sum_{|\alpha|=k} \partial_\alpha f(\boldsymbol{x}_0) \frac{\boldsymbol{h}^\alpha}{\alpha!} + R_m \tag{1.11}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is the multi-index vector that consists of non-negative integers; the condition $|\alpha| = k$ based on the norm $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$ choses all combination of α satisfying such relation and the multi-index variable is used for the computation following the expressions

$$\partial_\alpha := \partial_{x_1}^{\alpha_1} \partial_{x_2}^{\alpha_2} \ldots \partial_{x_n}^{\alpha_n} f(x) \qquad\qquad \boldsymbol{h}^\alpha := h_1^{\alpha_1} h_2^{\alpha_2} \ldots h_n^{\alpha_n}$$

1.3 Numerical methods

1.3.1 Taylor series based

In practise (systems of) ordinary differential equations are numerically solved by computes using algorithms some of which are based on the Taylor series expansion. Considering for the simplicity a first order ordinary differential equation in the standard form (equation 1.1)

$$\begin{cases} y' = f(x, y) \\ y(a) = y_a \end{cases}$$

If we consider $y(x)$ the solution of the ODE system, that can be expanded up to the second order with Taylor as

$$y(x + h) = y(x) + y'(x)h + \mathcal{O}(h^2) = y(x) + f(x, y)h + \mathcal{O}(h^2)$$

The most basic idea to numerically compute the solution is to subdivide the interval $[a, b]$ of integration into N pieces having each a width $h = \frac{b-a}{N}$; this discretization of the x axis determines so the sequence of point $x_k = a + hk$. With this idea in mind we can see that the yet computed Taylor expansion can be regarded as

$$y(x_{k+1}) = y(x_k) + f\big(x_k, y(x_k)\big)h + \mathcal{O}(h^2) \tag{1.12}$$

Numerical methods determines a sequence of output y_k that tends to approximate the real behaviour of the solution, hence $y_k \approx y(x_k)$. The simplest numerical method to solve the ordinary differential equation is by simply using equation 1.12 neglecting the reminder:

$$y_{k+1} = y_k + f(x_k, y_k)h \tag{1.13}$$

Error Numerical methods are approximation of the analytical solutions, hence intrinsically contains an error that should be somehow defined in order to determine *how bad* or *good* the numerical solution is. If we new define the error ϵ_k on the k-th step as the difference between the analytical and numerical solution

$$\varepsilon_{k+1} = y(x_{k+1}) - y_{k+1} = y(x_k) - y_k + \Big(f\big(x_k, y(x_k)\big) - f(x_k, y_k)\Big)h + \frac{y''(\zeta_k)}{2}h^2 \tag{1.14}$$

where the reminder $\mathcal{O}(h^2)$ as been substituted with the lagrangian one and hence $\zeta_k \in (x_k, x_{k+1})$. Considering that the function f is assumed to be lipschitzian, then it means that exists $L \in \mathbb{R}$ such that

$$\left|f\big(x_k, y(x_k)\big) - f(x_k, y_k)\right| \leq L\,|y(x_k) - y_k|$$

Considering this inequality, knowing that $y(x_k) - y_k = \epsilon_k$ and using the triangular inequality we can rewrite equation 1.14 as

$$|\varepsilon_{k+1}| = |\varepsilon_k| + hL\big|y(x_k) - y_k\big| + \frac{h^2}{2}|y''(\zeta_k)| = A|\varepsilon_k| + B \tag{1.15}$$

where $A = 1 + hL$ and $B = \frac{h^2}{2}M_2$. In particular M_2 is the constant that bounds the second derivative of y in the domain of integration, meaning

$$M_2 = \sup_{x\in[a,b]}\{y''(x)\}$$

Starting with the theoretical assumption that $\varepsilon_0 = 0$ (the initial error is null given the initial condition) and observing that $A \to 1$ and $B \to 0$, then we have that $|\varepsilon_1| \leq A0 + B = B$; the sequent error is so $|\varepsilon_2| \leq A|\varepsilon_1| + B = B(1 + A)$. Computing $|\varepsilon_3| \leq B(1 + A + A^2)$ it's possible to prove by induction that the error has a formulation

$$|\varepsilon_k| \leq \big(1 + A + A^2 + \cdots + A^{k-1}\big)B$$

The maximum error E_h of this numerical method is so determined by considering the maximum error respect to all discretization steps:

$$E_h = \max_{k=0,\ldots,N}|\varepsilon_k| \leq \max_{k=0,\ldots,N}\big(1 + A + A^2 + \cdots + A^{k-1}\big)B = \big(1 + A + A^2 + \cdots + A^{N-1}\big)B$$

Considering the geometrical series determined by $A + A^2 + \ldots$ we obtain that such sequence sums to the value $\frac{1-A^N}{1-A}$ and so the error can be considered as

$$E_h \leq \frac{A^N - 1}{A - 1}B = \frac{A^N - 1}{1 + hL - 1}\frac{h^2}{2}M_2 = \frac{A^h - 1}{L}\frac{h}{2}M_2$$

All we need now is to quantity the error related to the term A^N; considering that the Taylor series of the exponential sequence $e^x = 1 + x + \frac{x^2}{2!} + \ldots$ we have that such quantity is always greater than $1 + x$, and so knowing that $A = 1 + hL$ we have that $(1 + hL)^N \leq \left(e^{hL}\right)^n = e^{LNh}$. Observing that $Nh = b - a$ we can finally state the total error as

$$E_h \leq \frac{e^{L(b-a)}M_2}{2L}h = Ch \tag{1.16}$$

where $C \in \mathbb{R}$ is a constant, meaning that for $h \to 0$ the error presents the expected behaviour of approaching zero. By a computation point of view this method isn't that good, because in order to halve the error we have to halve also the integration step n (doubling the number of intervals N). If we would have considered other methods truncated to higher orders of derivation what we would have obtained is an error in the form

$$E_h = Ch^p$$

hence by dividing by 2 h the error would have been reduces by 2^p.

System of ODEs Considering the more general case of a system of ordinary differential equation in the standard form using the vectorial notation

$$\begin{cases}\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})\\ \boldsymbol{y}(a) = \boldsymbol{y}_a\end{cases}$$

the yet described method can be still used by expanding each component of $\boldsymbol{y}$, meaning that the numerical solution can be approximated as

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\boldsymbol{f}(x_k, \boldsymbol{y}_k)$$

and the error can be regarded as $E_h = \max\|\boldsymbol{y}(x_k) - \boldsymbol{y}_k\| \leq Ch$.

Implicit method: back-backward Euler

The numerical method described until now is the **explicit Euler integration** for determining the solution of ordinary differential equations; the formulation as provided in equation 1.13 (page 8) is computationally lightweight (because by knowing x_k, y_k at the current stage allows to automatically compute y_{k+1}), however is unstable and can quickly diverges from the analytical solution.

A way to solve such problematic is by using **implicit method** that are constructed by performing the Taylor expansion *from the left*. Considering as example the **Euler back-backward** method, the computed Taylor series is

$$y(x-h) = y(x) - hy'(x) + \frac{h^2}{2}y''(\zeta) = y(x) - hf(x,y) + \frac{h^2}{2}y''(\zeta)$$

This gives origin to the iterative numerical method defined as

$$y_{k-1} = y_k - hf(x_k, y_k) \tag{1.17}$$

where the solution of the current output y_k is implicitly defined as function of the current *position* x_k and the previous value y_{k-1}. This formulation increases the computational complexity (at each iteration a non-linear system has to be solved in order to determine the implicit solution y_k) but strongly increases the robustness of the algorithm.

Other methods based on the Taylor series

In general given the ordinary differential equation

$$\begin{cases} y' = f(x,y) \\ y(a) = y_a \end{cases}$$

in order to have a solution we assume that f is continuous and lipschitzian; given $y(x)$ the exact solution of the problem, we can apply the Taylor expansion on such result obtaining

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots + \frac{h^p}{p!}y^{(p)}(x) + \mathcal{O}(h^{p+1})$$

Knowing that $y(x)$ is the solution of the ODE, then we have that $y'(x) = f\big(x, y(x)\big)$; considering now so it's derivative we have that

$$\begin{aligned} y''(x) &= \frac{d}{dx}y'(x) = \frac{d}{dx}f\big(x,y(x)\big) = \frac{\partial f}{\partial x}\big(x,y(x)\big) + \frac{\partial f}{\partial y}\big(x,y(x)\big)y'(x) \\ &= \frac{\partial f}{\partial x}\big(x,y(x)\big) + \frac{\partial f}{\partial y}\big(x,y(x)\big)f\big(x,y(x)\big) \end{aligned}$$

This allows to rewrite the Taylor expansion as

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}f\right) + \cdots + \frac{h^p}{p!}y^{(p)}(x) + \mathcal{O}(h^{p+1})$$

The previously described explicit Euler method was determined by neglecting the terms with order higher then h, but in this case we have the possibility to express also y'' as function of f and x, y increasing hence the numerical accuracy. The numerical method is so

$$y_{k+1} = y_k + hf(x_k, y_k) + \frac{h}{2}\left(\frac{\partial f(x_k,y_k)}{\partial x} + \frac{\partial f(x_k,y_k)}{\partial y}f(x_k,y_k)\right) \tag{1.18}$$

where so in this case we dropped an error in the form $\mathcal{O}(h^3)$. Increasing the order of the numerical method increases the solution but requires the symbolical computation of the derivatives $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$; we can

carry on the process to explicitly determine all the derivative $y^{(p)}$ (up to a certain order) as function of x, y and partial derivatives of f. Considering as example the third derivative of y we see that

$$\begin{aligned} y'''(x) &= \frac{d}{dx}y''(x) \\ &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x\,\partial y}y' + \frac{\partial^2 f}{\partial x\,\partial y}f + \frac{\partial^2 f}{\partial y^2}yf + \frac{\partial f}{\partial y}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}y'\right) \end{aligned}$$

We can see that numerical method based on the Taylor series are conceptually easy but requires *a lot* of symbolical computation in order to express the derivatives as function of the known variables.

Example 1.4: explicit Euler scheme of the second order

Given the ODE

$$\begin{cases} y' = xy + x \\ y(0) = 1 \end{cases}$$

to express the 2^{nd} order Euler scheme we need to compute the partial derivatives

$$\frac{\partial f}{\partial x} = y + 1 \qquad\qquad \frac{\partial f}{\partial y} = x$$

and hence using equation 1.18 we obtain the method

$$y_{k+1} = y_k + h\big(x_k y_k + x_k\big) + \frac{h^2}{2}\Big[y_k + 1 + x_n\big(x_n y_n + x_n\big)\Big]$$

Higher order methods for this problem can be implemented explicitly computing the derivatives

$$\begin{aligned} y'(x) &= xy(x) + x \\ y''(x) &= y(x) + xy'(x) + 1 \\ y'''(x) &= y'(x) + y'(x) + xy''(x) = 2y'(x) + xy''(x) \\ y^{(4)}(x) &= 2y''(x) + y''(x) + xy'''(x) = 3y''(x) + xy'''(x) \\ &\vdots \end{aligned}$$

The idea is so to determine the numerical method as

$$y_{k+1} = y_k + hy'_k + \frac{h^2}{2}y''_k + \frac{h^3}{6}y'''_k + \frac{h^4}{24}y^{(4)}_k + \dots$$

where

$$\begin{aligned} y'_k = x_k y_k + x_k \quad &\to \quad y''_k = y_k + x_k y'_k + 1 \\ \to \quad y'''_k = 2y'_k + x_k y''_k \quad &\to \quad y^{(4)}_k = 3y''_k + xy'''_k \quad \to \quad \dots \end{aligned}$$

Economic Taylor scheme Recalling the higher order method shown in the previous example, a way to simplify the notation on the numerical method based on the Taylor series we consider the scheme

$$y_{k+1} = y_k + hy'_k + \frac{h^2}{2}y''_k + \dots + \frac{h^p}{p!}y^{(p)}_k$$

where the numerical derivatives are recursively defined considering that $y'_k = f(x_k, y_k)$:

$$\begin{aligned}
y'_k = D_1\big(x, y(x)\big) &= f(x,y) \\
y''_k = D_2\big(x, y(x), y'(x)\big) &= \frac{\partial D_1}{\partial x}(x,y) + \frac{\partial D_1}{\partial y}(x,y)\, y'(x,y) \\
y'''_k = D_3\big(x, y(x), y'(x), y''(x)\big) &= \frac{\partial D_2}{\partial x}(x,y) + \frac{\partial D_2}{\partial y} y' + \frac{\partial D_2}{\partial y'} y'' \\
&\vdots \\
y^{(p)}_k = D_p\big(x, y(x), y'(x), \dots, y^{(p-1)}(x)\big) &= \frac{\partial D_{p-1}}{\partial x} + \frac{\partial D_{p-1}}{\partial y} + \cdots + \frac{\partial D_{p-1}}{\partial y^{(p-2)}} y^{(p-1)}
\end{aligned}$$

1.3.2 Runge-Kutta

Considering the statements seen for numerical methods derived from the Taylor series expansion, we can see that the function f can be expanded by parameters α, β as

$$f(x+\alpha, y+\beta) = f(x,y) + \frac{\partial f}{\partial x}\alpha + \frac{\partial f}{\partial y}\beta + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\alpha^2 + \frac{\partial^2 f}{\partial x\, \partial y}\alpha\beta + \frac{1}{2}\frac{\partial^2 f}{\partial y^2}\beta^2 + \mathcal{O}\Big(\sqrt{\alpha^2+\beta^2}^{\,3}\Big)$$

The idea that originates the Runge-Kutta method is so to combine many evaluation of $f(x+\alpha, y+\beta)$ in order to match the results provided by the Taylor series; in general this match determines an error that however should be at maximum comparable to the order $\mathcal{O}(x^n)$ required. Considering the expansion previously discussed

$$y(x+h) = y(x) + hf\big(x, y(x)\big) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f\right) + \mathcal{O}(h^3)$$

we can consider the term

$$(i): \qquad h\, f\big(x, y(x)\big) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f\big(x, y(x)\big)\right)$$

and it's Taylor expansion

$$\begin{aligned}
h\omega f\big(x, y(x)\big) + h\gamma f\big(x+\alpha h, y(x)+\beta h\big) &= h\omega f + h\gamma\left(f + \frac{\partial f}{\partial x}\alpha h + \frac{\partial f}{\partial y}\beta h + \mathcal{O}(h^2)\right) \\
(ii): \qquad &= h\omega f + h\gamma f + h^2\gamma\alpha\frac{\partial f}{\partial x} + h^2\gamma\beta\frac{\partial f}{\partial y} + \mathcal{O}(h^3)
\end{aligned}$$

By now subtracting (ii) to (i) we determine

$$hf(1-\omega-\gamma) + h^2\frac{\partial f}{\partial x}\left(\frac{1}{2} - \gamma\alpha\right) + h^2\frac{\partial f}{\partial y}\left(\frac{f}{2} - \beta\gamma\right) + \mathcal{O}(h^3)$$

In order to reduce the error with a threshold $\mathcal{O}(h^3)$ we have to set to zero all the multiplicative terms of f (and it's derivatives) determining the following non-linear system:

$$\begin{cases} 1-\omega-\gamma = 0 \\ \frac{1}{2} - \gamma\alpha = 0 \\ \frac{f}{2} - \beta\gamma = 0 \end{cases}$$

If we determine parameters $\omega, \gamma, \alpha, \beta$ that satisfy such conditions we have that the expansion

$$y(x+h) = y(x) + \omega\, f\big(x, y(x)\big) + \gamma\, f\big(x+\alpha h, y(x)+\beta h\big) + \mathcal{O}(h^3)$$

matches the result obtained with Taylor; the system has 3 equation but the unknowns are 4, having so the a parametric solution in the form $\omega = 1-\gamma$, $\alpha = \frac{1}{2\gamma}$ and $\beta = \frac{f}{2\gamma}$ is always satisfied. Substituting this in the original expression we have that all the 2^{nd} order numerical method that do not use *explicitly* the partial derivatives of $f(x,y)$ are

$$y(x+h) = y(x) + h(1-\gamma)f(x,y(x)) + h\gamma f\left(x + \frac{h}{2\gamma}, y(x) + \frac{h}{2\gamma}f(x,y(x))\right) + \mathcal{O}(h^3)$$

determining the numerical method

$$y_{k+1} = y_k + h(1-\gamma)f(x_k,y_k) + h\gamma f\left(x_k + \frac{h}{2\gamma}, y_k + \frac{h}{2\gamma}f(x_k,y_k)\right) \tag{1.19}$$

Definition The idea of the **Runge-Kutta** method is use a combination of *displacements* in order to match *as much as possible* the Taylor expansion of the exact solution; in particular the numerical steps are written as

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \sum_{i=1}^{s} b_j \boldsymbol{k}_j \tag{1.20}$$

where the s vectors $\boldsymbol{k}_j$ (where s is the order of the Runge-Kutta method) are obtained as

$$\begin{cases} \boldsymbol{k}_1 = h\,\boldsymbol{f}\left(x_k + c_1 h, y_k + \sum_{j=1}^{s} A_{1j}\boldsymbol{k}_j\right) \\ \boldsymbol{k}_2 = h\,\boldsymbol{f}\left(x_k + c_2 h, y_k + \sum_{j=1}^{s} A_{2j}\boldsymbol{k}_j\right) \\ \quad\vdots \\ \boldsymbol{k}_s = h\,\boldsymbol{f}\left(x_k + c_s h, y_k + \sum_{j=1}^{s} A_{sj}\boldsymbol{k}_j\right) \end{cases} \tag{1.21}$$

where the coefficients c_j, b_j, A_{ij} are computed in such a way that $y_{k+1} - y(x_{k+1}) = \mathcal{O}(h^p)$ (so the error between the computed value and the theoretical solution) where p is as large as possible. Such values are already tabled in the ***Runge-Kutta tableaus*** represented as

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^t \end{array} \tag{1.22}$$

where $\boldsymbol{c} = (c_1, \ldots, c_s)$, $\boldsymbol{b} = (b_1, \ldots, b_s)$ (with $\boldsymbol{c}, \boldsymbol{b} \in \mathbb{R}^s$) and $A \in \mathbb{R}^{s\times s}$.

Runge-Kutta of order 4 A tableau for the Runge-Kutta method of order 4 is defined as

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & -1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array}$$

where the non-represented terms are zeros. Recalling equation 1.20 as the definition of the Runge-Kutta, we have that the numerical method derived from that is

$$y_{k+1} = y_j + \frac{1}{8}k_1 + \frac{3}{8}k_2 + \frac{3}{8}k_3 + \frac{1}{8}k_4$$

where

$$\begin{cases} k_1 &= h\,f(x_k, y_k) \\ k_2 &= h\,f\left(x_k + \frac{1}{3}h, y_k + \frac{1}{3}k_1\right) \\ k_3 &= h\,f\left(x_k + \frac{2}{3}h, y_k - \frac{1}{3}k_1 - k_2\right) \\ k_4 &= h\,f\left(x_k + h, y_k + k_1 - k_2 + k_3\right) \end{cases}$$

In this case the method is **explicit**: in fact we can sequentially compute k_1 (that depends on the knowns h, x_k, y_k) and sequentially k_2, then k_3 and lastly k_4.

Euler methods If we consider the *strange* tableau

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

what we obtain is the explicit Euler method, in fact having $k_1 = h\,f(x_k, y_k)$ determines the method

$$y_{k+1} = y_k + 1k_1 = y_k + hf(x_k, y_k)$$

and perfectly matches the definition provided in equation 1.13 at page 8. Considering now instead the tableau

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

it determines an **implicit** method: we in fact have that $k_1 = h\,f(x_k + h, y_k + k_1)$ where the solution is implicit in k_1; considering that such relation can be regarded as $k_1 = hf(x_{k+1}, y_{k+1})$ what we determine is the implicit Euler method (equation 1.17, page 10)

$$y_{k+1} = y_k + h\,f\big(x_{k+1}, y_{k+1}\big)$$

If we in general have that all the elements A_{ij} with $j \geq j$ are all zeros, then the Runge-Kutta scheme is explicit meaning that all the coefficients $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_s$ can be computed consecutively. *Graphically* this means that the matrix A must have non-zero terms only below its principal matrix not included (it means that A_{ii} must always be zero).

One step methods

Usually we refer to ***one step methods*** the ones that allow to explicitly compute the next step as function of the current step in the form

$$y_{k+1} = \phi(x_k, y_h, h)$$

An example is the explicit Euler method (equation 1.13, page 8) that's characterized by the function $\phi(x_k, y_k, h) = y + hf(x_k, y_k)$. Also implicit methods can be one step; considering the implicit Euler (equation 1.17, page 10) it can be considered as

$$y_{k+1} = y_k + K_1 \qquad \text{with } K_1 = h\,f\big(x_{k+1}, y_k + K_1\big)$$

the value K_1 is formally a function of the parameters x_k, y_k, h, hence

$$G\big(K_1, x_k, y_k, h\big) = K - h\,f\big(x_k + hy_k + K\big) \qquad \Rightarrow \qquad \phi(x, y, h) = y + K(x, y, h)$$

In general all Runge-Kutta methods (both explicit and implicit) can be formally written in the form $y_{k+1} = \phi(x_k, y_k, h)$ and so are one step methods.

Error propagation

Known that each Runge-Kutta is a one-step method, then we can express the **local truncation error** $\tau_k(h)$ as the difference between the theoretical computed value and the numerical result obtained:

$$y\big(x_{k+1}\big) = \phi\big(x_k, y(x_k), h\big) + \tau_k(h) \qquad \Rightarrow \qquad \tau(h) = y(x+h) - \phi\big(x, y(x), h\big)$$

Considering the **error** $\varepsilon_k = y\big(x_k\big) - y_k$ as the difference between the analytical solution and the numerical approximated one, we can regard the two cases as

$$\begin{aligned} (i)\,: &\qquad y_{k+1} = \phi(x_k, y_k, h) \\ (ii)\,: &\qquad y\big(x_{k+1}\big) = \phi\big(x_k, y(x_k), h\big) + \tau_k(h) \end{aligned}$$

performing the difference $(i) - (ii)$ what we obtain is

$$\varepsilon_{k+1} = \Big(\phi\big(x_k, y(x_k), h\big) - \phi(x_k, y_k, h)\Big) + \tau_k(h)$$

Considering the simple case of the explicit Euler method characterized by a function $\phi(x, y, h) = y + h\, f(x, y)$, the difference in the parenthesis is in the form $\phi(x, z, h) - \phi(x, y, h) = z - y + h\big(f(x, z) - f(x, y)\big)$; computing it's absolute value we have and considering that f is lipschitzian we have

$$\big|\phi(x, z, h) - \phi(x, y, h)\big| \leq |z - y| + h\big|f(x, z) - f(x, y)\big| \leq (1 + hL)|z - y|$$

We can so rewrite the magnitude of the error as

$$|\varepsilon_{k+1}| \leq (1 + hL)\big|y(x_k) - y_k\big| + |\tau_k(h)| \leq (1 + hL)|\varepsilon_k| + |\tau_k(h)|$$

MIN 7.08

Chapter 2

Introduction to Algebraic Differential Equations

The **algebraic differential equations** **DAEs** can be regarded as a system of ordinary differential equations combined with *general* algebraic equations; as example a DAE system is

$$\begin{cases} y' = f(x,y) \\ y(a) = y_a \\ f(x,y) = 0 \\ x^2 - y = 3 \end{cases}$$

In general for ODEs and algebraic equations a lot of numerical methods have been implemented with consolidated theory regarding existence, stability... The problem is that when combining such theories in the algebraic differential equations, the numerical results that we might wanna retrieve are a *nightmare* to compute.

DAE with an example: simple pendulum Considering the simple pendulum of a mass m fixed by a bar of length l to a pivot point; considering such center as the origin of a reference frame, the coordinates of the mass can be described as function of using the minimal number of coordinates (associated in this case to lagrangian coordinate θ as the angle between the bar and the vertical line) as

$$x = l\sin\theta \qquad\qquad y = -l\cos\theta$$

The idea is that this coordinates satisfy the constraint $x^2 + y^2 = l^2$, meaning that the mass m can move only on the circle of radius l. Taking the velocities

$$\dot{x} = \frac{dx}{dt} = l\cos\theta\,\dot{\theta} \qquad\qquad \dot{y} = \frac{dy}{dt} = l\sin\theta\,\dot{\theta}$$

Computing the kinematic and potentials energy as

$$T = \frac{m}{2}(\dot{x}^2 + \dot{y}^2) = \frac{m}{2}l^2\dot{\theta}^2 \qquad\qquad V = mgy = -mgl\cos\theta$$

With this we can build the lagrangian $\mathcal{L} = T - V = \frac{m}{2}l^2\dot{\theta}^2 + mgl\cos\theta$ and using the Euler-Lagrange equation (following the minimal action principle) the differential equation describing the motion is

$$\frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{\theta}} - \frac{\partial\mathcal{L}}{\partial\theta} = ml^2\ddot{\theta} + mgl\sin\theta = l\ddot{\theta} + g\sin\theta = 0$$

where the ordinary differential equation, in order to be solved/integrated, requires the initial conditions $\theta(0) = \theta_0$ and $\dot{\theta}(0) = \dot{\theta}_0$. Introducing $\omega = \dot{\theta}$ we can reduce the system of ODEs to the first order

that can be numerically solved:

$$\begin{cases} \dot{\theta} = \omega \\ l\dot{\omega} + g\sin\theta = 0 \\ \theta(0) = \theta_0, \quad \omega(0) = \omega_0 = \dot{\theta}_0 \end{cases}$$

Observe that we obtained the solution as an ordinary differential equation because we found the *minimal* set of coordinates which describes the system.

As alternative approach we could have used simpler independent ordinary differential equations and add some constraints; considering the independent mass m described by the point (x, y) in the plane and constrained to move on a circle of radius l, it's kinetic and potential energies are still $T = \frac{m}{2}(\dot{x}^2 + \dot{y}^2)$ and $V = mgy$ (note that no transformation in terms of θ has been applied), then the lagrangian is

$$\mathcal{L} = T - V = \frac{m}{2}(\dot{x}^2 + \dot{y}^2) - mgy$$

The constraint is described by the equation $\phi(x, y) = x^2 + y^2 - l^2 = 0$. Adding to the constraint the least action principle stating that the functional $\mathcal{A} = \int_{t_0}^{t_1} \mathcal{L}(x, y, \dot{x}, \dot{y}, t)\, dt$ we can find the *stationary point* of the action $\mathcal{A}$ that's subject to $\phi(x, y) = 0$. Expanding the definition

$$\int_{t_0}^{t_1} \mathcal{L}(x, y, \dot{x}, \dot{y}, t) - \lambda\phi(x, y)\, dt$$

we can build the hamiltonian $\mathcal{H} = \mathcal{L} - \lambda\phi$ that, after the first variation, determines the system

$$\begin{cases} \frac{d}{dt}\frac{\partial \mathcal{H}}{\partial \dot{x}} - \frac{\partial \mathcal{H}}{\partial x} & = m\ddot{x} + \lambda x = 0 \\ \frac{d}{dt}\frac{\partial \mathcal{H}}{\partial \dot{y}} - \frac{\partial \mathcal{H}}{\partial y} & = m\ddot{y} + \lambda y = -mg \\ \phi(x, y) & = x^2 + y^2 - l^2 = 0 \end{cases}$$

that is a **differential algebraic equation**; introducing $\dot{x} = u$ and $\dot{y} = v$ we can simplify to a differential algebraic equation

$$\begin{cases} m\dot{u} + \lambda x = 0 \\ m\dot{v} + \lambda y = -mg \\ \dot{x} = u, \qquad \dot{y} = v \\ x^2 + y^2 - l^2 = 0 \end{cases} \tag{2.1}$$

Introduction to numerical methods for DAEs Considering the example of the pendulum, we can rewrite the differential equations as

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{u} \\ \dot{v} \\ 0 \end{pmatrix} = \begin{pmatrix} u \\ v \\ -\lambda x/m \\ -\lambda y/m - g \\ x^2 + y^2 - l^2 \end{pmatrix}$$

Defining $\boldsymbol{z} = (x, y, u, b, \lambda)$, such relation is similar to the form $\dot{\boldsymbol{z}} = \boldsymbol{F}(t, \boldsymbol{z})$. The main idea is in fact to **transform DAEs to ODEs**; note in fact that the last equation is the lonely one that's not already a ordinary differential equation: deriving it in time determines

$$\frac{d}{dt}(x^2 + y^2 - l^2) = 2x\dot{x} + 2y\dot{y} = 2xu + 2yv$$

but stull we observe that the variable λ is missing in the equation. Deriving one more time respect to t we obtain

$$\begin{aligned} \frac{d}{dt}(2xu + 2yv) &= 2\dot{x}u + 2x\dot{u} + 2\dot{y}v + 2y\dot{v} = 2u^2 + 2v^2 - 2x\frac{\lambda x}{m} - 2y\left(\frac{\lambda y}{m} + g\right) \\ &= 2(u^2 + v^2) - \frac{2\lambda}{m}(x^2 + y^2) - 2yg \end{aligned} \tag{2.2}$$

By substituting the different known relations for $\dot{x}, \dot{y}, \dot{u}, \dot{v}$ we so obtain a derivative that's function of λ, but not of $\dot{\lambda}$. Deriving one more time respect to the variable t

$$\begin{aligned}\frac{d}{dt}(2.2) &= 4\left(u\dot{u} + v\dot{v}\right) - \frac{4\lambda}{m}\left(x\dot{x} + y\dot{y}\right) - 2\dot{y}g - \frac{2}{m}\dot{\lambda}\left(x^2 + y^2\right) \\ &= 4\left(-u\frac{\lambda x}{m} - v\frac{\lambda y}{m} - vg\right) - \frac{4\lambda}{m}\left(xu + yv\right) - 2vg - \frac{2}{m}\dot{\lambda}\left(x^2 + y^2\right) = 0\end{aligned}$$

Solving for $\dot{\lambda}$ so gives

$$\dot{\lambda} = \frac{-4\lambda\left(xy + yv\right) - 3vmg}{x^2 + y^2}$$

We can so rewrite the differentia algebraic system in equation 2.1 as a system of ODE only as

$$\begin{cases} \dot{x} = u \\ \dot{y} = v \\ \dot{u} = -\frac{l}{m}x \\ \dot{v} = -\frac{\lambda}{m}y - g \\ \dot{\lambda} = \frac{-4\lambda\left(xy+yv\right)-3vmg}{x^2+y^2} \end{cases}$$

With such definition we can use numerical methods to solve the form $\dot{\boldsymbol{z}} = \boldsymbol{F}(t, \boldsymbol{z})$. This formulations however introduces some problems:

- the initial condition on λ is not set. This problem can be overcome considering that given x, the coordinate y is constrained by $x^2 + y^2 = l^2$. If we moreover know $\dot{x} = u$ then using the derivative of the constraint $2xy + 2yv = 0$, then also $v = \dot{y}$ is constrained. Using the second derivative of the constraint (equation 2.2) we can finally solve for λ_0. We see that the initial conditions must satisfy the *original* constraints and the *hidden ones* determined by the derivatives of the algebraic equations.
- another problem is that if we considered a constraint in the form

  $$\phi(x, y) = x^2 + y^2 - l^2 + a + bt + ct^2 = 0$$

  in order to obtain the ODE equivalent system we have to derive ϕ three times over time resulting in the cancellations of the polynomial terms $a + bt + ct^2$ (we in fact would have obtained the same ODE system).

2.1 Linear differential algebraic equations

Starting from the simplest cases of study, a generic differential algebraic equation can be written as

$$\begin{cases} \boldsymbol{F}(t, \boldsymbol{y}, \boldsymbol{y}') = 0 \\ \boldsymbol{y}(a) = \boldsymbol{y}_a \end{cases}$$

with $\boldsymbol{y}(t) \in \mathbb{R}^n$. We can say that the map $\boldsymbol{F}$ is **linear** in y if it can be expressed as linear combination of $\boldsymbol{y}'$ in the form

$$\boldsymbol{F}(t, \boldsymbol{y}, \boldsymbol{y}') = \mathbf{E}(t, \boldsymbol{y})\boldsymbol{y}' + \boldsymbol{G}(t, \boldsymbol{y}) \tag{2.3}$$

Moreover the map $\boldsymbol{F}$ is linear in both $\boldsymbol{y}$ and $\boldsymbol{y}'$ if it can be regarded as

$$\boldsymbol{F}(t, \boldsymbol{y}, \boldsymbol{y}') = \mathbf{E}(t)\boldsymbol{y}' + \mathbf{A}(t)y - \boldsymbol{C}(t) \tag{2.4}$$

where in general $\mathbf{E}, \mathbf{A}$ are matrices that can sometimes be singular. The map $\boldsymbol{F}$ is said **linear with constant coefficients** if it happens that the matrices $\mathbf{E}, \mathbf{A}$ are time independent.

Example 2.1: linear DAEs

An example of linear differential algebraic equation is the one that can be described as

$$\begin{bmatrix} 1 & t \\ t^2 & 2 \end{bmatrix} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} + \begin{bmatrix} \sin t & \cos t \\ t^2 & 1 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} e^t \\ 1+t \end{pmatrix}$$

Observe that if the matrix $\boldsymbol{E}$ in non singular, expression 2.3 corresponds to a *simple* ordinary differential equation: it can be in fact rewritten as

$$\boldsymbol{y}' = -\mathbf{E}^{-1}(t, \boldsymbol{y})\boldsymbol{G}(t, \boldsymbol{y}) = \boldsymbol{f}(t, \boldsymbol{y}) \tag{2.5}$$

For the moment we can assume that if $\mathbf{E}$ is singular the system is not an ODE. In this first part we will focus on linear differential algebraic equations in order to exploit linear algebra tools to ease the calculations.

Jordan normal form As a recall from the linear algebra, given a matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ there exists always a non-singular matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{T}^{-1}\mathbf{B}\mathbf{T} = \mathbf{J}$$

where $\mathbf{J}$ is the **Jordan matrix form** defined as

$$\mathbf{J} = \begin{bmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_m \end{bmatrix} \qquad \text{where } J_k = \begin{bmatrix} \lambda_k & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_k \end{bmatrix} \tag{2.6}$$

Regular pencil Given the matrices $\mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n}$, the couple $(\mathbf{B}, \mathbf{C})$ is a **regular pencil** if

$$f(\lambda) = \det\left(\mathbf{B} - \lambda\mathbf{C}\right) \neq 0$$

is not identically null, or equivalently if there exists a λ such that $f(\lambda) \neq 0$. Considering as example the matrices

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

is not a regular pencil, in fact the polynomial $f(\lambda) = \det(\mathbf{B} - \lambda\mathbf{C}) = \det \begin{bmatrix} 1-\lambda & 1 \\ 0 & 0 \end{bmatrix} = 0$ always evaluates to zero.

Nilpotent matrix A matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ is **nilpotent** of order p if

$$\mathbf{B}^p = 0 \quad \text{and} \quad \mathbf{B}^j \neq 0 \; \forall j < p \tag{2.7}$$

where $\mathbf{B}^p$ is the product $\mathbf{B}\mathbf{B}\ldots\mathbf{B}$ p times. Considering as example

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B}^2 = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B}^3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

we have that such matrix $\mathbf{B}$ is nilpotent of order 3. Observe that if the matrix $\mathbf{B}$ is non-singular, than it can't be nilpotent.

Kronecker normal form If we consider two regular pencil matrices $(\mathbf{B}, \mathbf{C}) \in \mathbb{R}^{n\times n}$, then there exists two non-singular matrices $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{PBQ} = \begin{bmatrix} \mathbf{N} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \qquad \text{and} \qquad \mathbf{PCQ} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{J} \end{bmatrix} \tag{2.8}$$

where $\mathbf{N}$ is a nilpotent matrix, $\mathbf{I}$ is the identity matrix and $\mathbf{J}$ is a Jordan normal form matrix. Considering that the *blocks* $\mathbf{N}, \mathbf{J}$ can be empty, as extreme cases we have $\mathbf{PBQ} = \mathbf{I}$, $\mathbf{PCQ} = \mathbf{J}$, $\mathbf{PBQ} = \mathbf{N}$ and $\mathbf{PCQ} = \mathbf{I}$.

2.1.1 Usage of the Kronecker normal form

To ease the computation of linear differential algebraic equation, we can use the Kronecker normal form assuming that the couple of matrices $(\mathbf{E}, \mathbf{A})$ (equation 2.4) are a regular pencil (in order not to have an *inconsistent* DAE).

Example 2.2: inconsistent DAE

Using the matrices defined used in the theory of regular pencil, we can build a DAE system of the form

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} t \\ 1 \end{pmatrix}$$

the associated system is

$$\begin{cases} y_1' + y_2' + y_2 = t \\ 0 = 1 \end{cases}$$

that's inconsistent.

With such assumption we can compute the Kronecker normal form $\mathbf{PEQ} = \begin{bmatrix} \mathbf{N} & \\ & \mathbf{I} \end{bmatrix}$ and $\mathbf{PAQ} = \begin{bmatrix} \mathbf{I} & \\ & \mathbf{J} \end{bmatrix}$; premultiplying so equation 2.4 by $\mathbf{P}$ results in $\mathbf{PE}\boldsymbol{y}' + \mathbf{PA}\boldsymbol{y} = \mathbf{P}\boldsymbol{C}$. Performing the change of variable $\mathbf{Q}\boldsymbol{z} = \boldsymbol{y}$ (hence $\boldsymbol{z} = \mathbf{Q}^{-1}$) and observing that $\mathbf{Q}\boldsymbol{z}' = \boldsymbol{y}'$ we obtain the expression $\mathbf{PEQ}\boldsymbol{z}' + \mathbf{PAQ}\boldsymbol{z} = \mathbf{P}\boldsymbol{C}$ on top of which we can apply the Kronecker normal form:

$$\begin{bmatrix} \mathbf{N} & \\ & \mathbf{I} \end{bmatrix} \boldsymbol{z}' + \begin{bmatrix} \mathbf{I} & \\ & \mathbf{J} \end{bmatrix} \boldsymbol{z} = \mathbf{P}\boldsymbol{C}$$

Splitting both vectors $\boldsymbol{z} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\mathbf{P}\boldsymbol{C} = (\boldsymbol{d}, \boldsymbol{e})$ we can rewrite this expression as

$$\begin{bmatrix} \mathbf{N} & \\ & \mathbf{I} \end{bmatrix} \begin{pmatrix} \boldsymbol{\alpha}' \\ \boldsymbol{\beta}' \end{pmatrix} + \begin{bmatrix} \mathbf{I} & \\ & \mathbf{J} \end{bmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{e} \end{pmatrix}$$

The associated linear system representation is

$$\begin{cases} i) & \mathbf{N}\boldsymbol{\alpha}' + \boldsymbol{\alpha} = \boldsymbol{d}(t) \\ ii) & \boldsymbol{\beta}' + \mathbf{J}\boldsymbol{\beta} = \boldsymbol{e} \end{cases}$$

$ii)$ represent a *standard* system of ordinary differential equations; the term $i)$ is instead more complex but we can invert the relation to obtain $\boldsymbol{\alpha} = \boldsymbol{d} - \mathbf{N}\boldsymbol{\alpha}'$. Observing that the k-th derivative in time of this expression evaluates to $\boldsymbol{\alpha}^{(k)} = \boldsymbol{d}^{(k)} - \mathbf{N}\boldsymbol{\alpha}^{k+1}$, we can substitute the derivatives $\boldsymbol{\alpha}^{(k)}$ determining

$$\begin{aligned} \boldsymbol{\alpha} &= \boldsymbol{d} - \mathbf{N}\boldsymbol{\alpha}' = \boldsymbol{d} - \mathbf{N}(\boldsymbol{d}' - \mathbf{N}\boldsymbol{\alpha}'') = \boldsymbol{d} - \mathbf{N}\boldsymbol{d}' + \mathbf{N}^2\boldsymbol{\alpha}'' = \boldsymbol{d} - \mathbf{N}\boldsymbol{d}' + \mathbf{N}^2(\dots) = \dots \\ &= \boldsymbol{d} - \mathbf{N}\boldsymbol{d}' + \mathbf{N}^2\boldsymbol{d}'' - \mathbf{N}^3\boldsymbol{d}''' + \mathbf{N}^4\boldsymbol{d}^{(4)} - \mathbf{N}^5\boldsymbol{d}^{(5)} + \dots \end{aligned}$$

This series is infinite, however being $\mathbf{N}$ a nilpotent matrix of order p we have only that the first p terms remains and so

$$\boldsymbol{\alpha} = \boldsymbol{d} - \mathbf{N}\boldsymbol{d}' + \mathbf{N}^2\boldsymbol{d}'' + \cdots + (-1)^{p-1}\mathbf{N}^{p-1}\boldsymbol{d}^{(p-1)} + \cancel{(-1)^{p}\mathbf{N}^{p}\boldsymbol{d}^{(p)}} = \sum_{j=0}^{p-1}(-1)^j\mathbf{N}^j\boldsymbol{d}^{(j)} \tag{2.9}$$

With this relation we determined $\boldsymbol{\alpha}$ without using the initial conditions (as was required for the DEA solution using the conversion to ODE), depends only on $\boldsymbol{d}$ (and it's derivatives up to the $p-1$ order). Also observe that $ii)$ is a *regular* ODE, hence the initial values $\boldsymbol{\beta}(0)$ must be specified.

The order of the nilpotency of the matrix $\mathbf{N}$ is the so called **index** of the differential algebraic equation (for the linear ones) and is a sort of *measure* of the *difficulty* of solving numerically the DAE. In particular if $p = 0$ then what we have is a system of ordinary differential equation while if $p = n$ we have a set of only algebraic equations.

Part II

Modelling & Simulation

Chapter 3

Introduction

Mechatronics multi-body domain systems **MBDS** are *complex* systems that interacts and embeds **multiple physical domain**, such the thermal, controls, electrical, hydraulic...
Such system are intrinsically complex and their **modelling** and consequent **simulation** requires a high level of (mathematical) abstraction and modularisation; to improve the management of the model complexity it's necessary to decompose the initial MBDS into sub-system with low interaction with each other in order to focus independently on various aspect of the complex system.

From a mathematical standpoint multi-body domain systems are usually described by **differential algebraic equations** DAE (described in part I), a combination of ordinary differential and algebraic equations that can be used, as example, to perform **kinematic analysis** (to determine position/velocity/acceleration of components in the system) or **dynamic simulation** both off-line (to estimate the behaviour of the system) or online (with the **hardware in loop** HIL method allowing to control the mechatronic system).

System modelling The **model** is the process that, starting from real world observation, construct an **abstract representation** (mainly mathematical) **validated** with physical experiments that allow to generate **simulations** in order to accomplish the **analysis** of the results obtained. The realisation of a model-based analysis of a mechatronic system's dynamic can per performed through this step:

- generate a qualitative system model in order to understand the general behaviour;
- determine the domain specific models;
- specify the mathematical dynamic of the models yet generated;
- at this stage we so have a model that can predict the behaviour of real mechatronic systems.

The modelling of the system can be performed using two types of approaches:

- the **qualitative** model is a *black box* that for a given input determines an output; implementation of this modelling are **neural networks** that firstly need to be trained in order to predict the functionality of the system. It's proven in fact that any system can be described, within a certain accuracy range, by a proper neural network; the problem of this approach is that each time a parameter changes, the neural network must be re-trained and so it doesn't allow to state general assertion regarding the functionality of the multi-body system;
- a **quantitative** approach, based on differential algebraic equations, allows instead to have a more general meaning; the system is in fact described by a set of parameters, variables, equations and constraints that can somehow be symbolically manipulated or numerically solved.

All models are so approximation of the real behaviour of the system and for that reason must be validated using experiments/testing and/or using the past experience gained; every model has a limited range of validity on which results are reliable within a certain standard. A good model must

be *simple* (it's useless to have an over-complex system to analyse every little detail) and captures the critical property of the system is modelling.

Mathematical models can be *small* or *large*, *simple* or *complicated*, static or dynamic (regarding time-variant), deterministic or stochastic, qualitative or quantitative, linear or non-linear, continuous or discrete-time...

Simulations For the modelling and consequent simulation of multi-body systems several general-purpose simulation modelling and languages have been developed that can be classified according the following criteria:

- graph or language based;
- procedural or declarative (based on equation) models;
- multi or single-domain modelling respect to specific problem;
- continuous or discrete (hybrid solutions also exists): this is related to the way on how subsystems of different domain interact together. In a **co-simulation** approach the communication interval between the system is discrete and introduces a delay between the system that can cause instability, while with a **unified** approach the solver solves the full set of equation simultaneously increasing accuracy/stability but also numerical complexity;
- functional or object oriented.

Usually a symbolic approach for the modelling is preferred because it allows to be *more general*: we don't have to specify to the model what is the input and what's the output (that's instead decided at simulation time by reversing all relations) and allows to perform mathematical simplifications; usually for fast/real-time numerical simulation analytical formulas are converted in optimized `C`, `C++` code.

Chapter 4

Rigid Body Kinematics and Rotation Matrix

Kinematic is the study of purely geometrical aspects of individual positions and motions of rigid bodies, hence cause of motions (forces and torques) are not considered. In particular we define **bodies** as **rigid** if for every point $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$ in it the mutual distance between them and the angle represented by the vector connecting each pair of point are constant (or can be approximated as so):

$$\boldsymbol{P}_1\boldsymbol{P}_2 = \text{const.} \qquad \text{and} \qquad \angle\{\boldsymbol{P}_1\boldsymbol{P}_2, \boldsymbol{P}_1\boldsymbol{P}_3\} = \text{const.} \qquad \forall \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3 \text{ points} \in \text{body}$$

4.1 Rotation matrix

To describe the **pose** (*position*) and **attitude** (*orientation in space*) of a body respect to another one (or respect to ground) we use the so called **reference frame** $\mathfrak{R}$ describing the **configuration** (the set of parameters that fully describes the system) of the body in the space. Considering planar systems, each body has 3 degrees of freedom hence the minimum number of parameters of the configuration is 3; in the spatial case the number increases to 6.

Typically bodies are described in **local** (or **moving**) reference frames $\mathfrak{R}^b$ that are obtained as roto-translations of a **world** (or **ground**) reference frame $\mathfrak{R}^w$. Given in fact the coordinates $\boldsymbol{x}_b$ of a point $\boldsymbol{P}$ described in the local frame of the body, the absolute position respect to the world frame can be obtained as

$$\boldsymbol{x}_w = \boldsymbol{x}_0 + \mathbf{R}\boldsymbol{x}_b \tag{4.1}$$

where $\boldsymbol{x}_0$ is the origin of the local frame respect to the world frame (representing so the **pose**) and $\mathbf{R}$ is a **rotation matrix** that describes the **attitude** of the body in ground coordinates.

Rotation matrix The rotation matrix, as stated, represent the attitude of a body respect to a given absolute reference. In order to specify such *angular orientation* what we need to do is to describe the vectorial base $\mathbf{e}^b$ of the body in terms of the ground vectorial base $\mathbf{e}^w$ as

$$\mathbf{e}^b = \mathbf{R}\mathbf{e}^w \tag{4.2}$$

In particular the nine elements of the matrix $\mathbf{R}$ are the generalized coordinates that describes the angular orientation of the body in the base $\mathbf{e}^w$. In particular if we consider that the 3 versor that generates the base $\mathbf{e}^b$ expressed in the world frame are $\hat{\boldsymbol{\imath}}_b = (i_{b,x}, i_{b,y}, i_{b,z})$, $\hat{\boldsymbol{\jmath}}_b = (j_{b,x}, j_{b,y}, j_{b,z})$, $\hat{\boldsymbol{k}}_b = (k_{b,x}, k_{b,y}, k_{b,z})$ then the matrix $\mathbf{R}$ can be regarded as

$$\mathbf{R} = \begin{bmatrix} \hat{\boldsymbol{\imath}}_b & \hat{\boldsymbol{\jmath}}_b & \hat{\boldsymbol{k}}_b \end{bmatrix} = \begin{bmatrix} i_{b,x} & j_{b,x} & k_{b,x} \\ i_{b,y} & j_{b,y} & k_{b,y} \\ i_{b,z} & j_{b,z} & k_{b,z} \end{bmatrix} \quad \text{or alternatively} \quad \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \tag{4.3}$$

An important aspect to keep in mind is that a base, to be so, must have that the scalar product between any different versor is zero and between the same direction must be unitary, mathematically

$$\hat{\boldsymbol{e}}_i \cdot \hat{\boldsymbol{e}}_j = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

This means that the rotation matrix $\mathbf{R}$, in order to be a proper base, must withstand the following six constraints described by all the possible combination i, j:

$$\sum_{k=1}^{3} r_{ik} r_{jk} = \delta_{ik} \qquad i, j \in \{1, 2, 3\} \tag{4.4}$$

A base is also characterized by being *right-handed* (the order of the versor can be described using the *right hand rule*) and mathematically this means imposing $\hat{\boldsymbol{e}}_1 \cdot (\hat{\boldsymbol{e}}_2 \times \hat{\boldsymbol{e}}_3) = 1$. Overall the rotation matrix $\boldsymbol{R}$ has **9 generalized coordinates** (elements of the matrix), but only **3** of them are **independent** because we have **6 constraints** determined by equation 4.4. This intuitively proves the fact that in order to describe the *orientation* of a rigid body in space only 3 parameters are required.

As it will be shown different (and more practical ways) to represent such rotation matrix will be present in order to have just 3 parameters (and not 9 subjected to 6 constraints, increasing the complexity of the problem).

Another important property related to the rotation matrix $\mathbf{R}$ is that it belongs to the **special orthogonal matrices** $SO(N)$ of order N (and in our case of spatial kinematic we have $N = 3$) that's characterized by having the inverse of $\mathbf{R}$ equals to it's transposed; we in fact have tat

$$\mathbf{R}^{-1}\mathbf{R} = \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathcal{I} \qquad \Leftrightarrow \qquad \mathbf{R}^{-1} = \mathbf{R}^T \tag{4.5}$$

By this means equation 4.1 can be inverted in order to express the coordinates $\boldsymbol{x}_b$ of a point in the body reference frame as function of the ground coordinates $\boldsymbol{x}_w$ as

$$\boldsymbol{x}_b = \mathbf{R}^{-1}(\boldsymbol{x}_w - \boldsymbol{x}_0) = \mathbf{R}^T\boldsymbol{x}_w - \mathbf{R}^T\boldsymbol{x}_0 \tag{4.6}$$

4.1.1 Transformation matrix

The **transformation matrix** $\mathfrak{R}$ is a 4×4 matrix that's used to fully describe the roto-translation transformation that has to be applied in order to determine the world coordinates $\boldsymbol{x}_w = (x_w, y_w, z_w)$ of a point knowing it's *position* in the local frame $\boldsymbol{x}_b = (x_b, y_b, z_b)$, it's pose $\boldsymbol{o}^w = (x_0, y_0, z_0)$ and orientation $\mathbf{R}$ respect to the global frame:

$$4.1 \mapsto \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \left[\begin{array}{ccc|c} & \mathbf{R} & & \begin{matrix} x_0 \\ y_0 \\ z_0 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{pmatrix} x_b \\ y_b \\ z_b \\ 1 \end{pmatrix} \tag{4.7}$$

We can see that this construction directly determines the direct transformation described in equation 4.1, but we can also determine the transformation matrix of the inverse transformation as

$$4.6 \mapsto \begin{pmatrix} x_b \\ y_b \\ z_b \\ 1 \end{pmatrix} = \left[\begin{array}{ccc|c} & \mathbf{R}^{-1} & & -\mathbf{R}^{-1}\boldsymbol{o}^w \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{4.8}$$

Reference frames as sequence of transformations Given a world reference frame $\mathfrak{R}^w$ that's hence fixed, local frames associated to bodies are described in such immovable space using a combination of roto-translations. Given for example two transformation $\mathfrak{R}_1, \mathfrak{R}_2$ the order on which we perform the transformations heavily affects the final frame, and in general

$$\mathfrak{R}_1\mathfrak{R}_2 \neq \mathfrak{R}_2\mathfrak{R}_1$$

Pure translation If we consider a reference frame $\mathfrak{R}_2$ defined as pure translation of a frame $\mathfrak{R}_1$ (that's also a pure translation of a world reference frame $\mathfrak{R}^w$), as shown in figure 4.1, defined by the transformation matrices

$$\mathfrak{R}_1 = \left[\begin{array}{ccc|c} & \mathcal{I} & & \begin{matrix} x_{01}^w \\ y_{01}^w \\ z_{01}^w \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad \mathfrak{R}_2 = \left[\begin{array}{ccc|c} & \mathcal{I} & & \begin{matrix} x'_{02} \\ y'_{02} \\ z'_{02} \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

in order to describe the second reference frame into global coordinate system we have to perform the operation $\mathfrak{R}_1\mathfrak{R}_2$. Performing algebraically the operation we indeed retrieve the intuitive result of a transformation matrix $\mathfrak{R}_2^w$ with no rotation ($\mathcal{R} = \mathcal{I}$) and center of the base in $\boldsymbol{x}_1^w + \boldsymbol{x}'_2$, in fact

$$\mathfrak{R}_2^w = \mathfrak{R}_1\mathfrak{R}_2 = \left[\begin{array}{ccc|c} & \mathcal{I} & & \begin{matrix} x_{01}^2 + x'_{02} \\ y_{01}^2 + y'_{02} \\ z_{01}^2 + z'_{02} \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

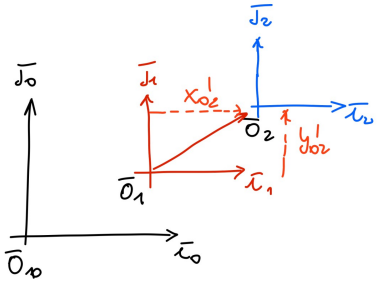


Figure 4.1: multiple transformations of pure translation; in this case, for sake of simplicity, the planar case has been considered.

Pure rotation Considering a reference frame $\mathfrak{R}_2$ that a pure rotation (along the z axis) of an angle β respect to a reference frame $\mathfrak{R}_1$ characterized by a pure rotation of angle α respect to the reference frame $\mathfrak{R}^w$ (figure 4.2), their transformation matrices are

$$\mathfrak{R}_1 = \left[\begin{array}{ccc|c} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad \mathfrak{R}_2 = \left[\begin{array}{ccc|c} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

In order to determine the transformation of the second reference frame respect to the ground we so have to compute the product $\mathfrak{R}_1\mathfrak{R}_2$ between the transformation matrix, hence

$$\mathfrak{R}_2^w = \mathfrak{R}_1\mathfrak{R}_2 = \left[\begin{array}{ccc|c} \cos\alpha\cos\beta - \sin\alpha\sin\beta & -\cos\alpha\sin\beta - \sin\alpha\cos\beta & 0 & 0 \\ \sin\alpha\cos\beta + \cos\alpha\sin\beta & -\sin\alpha\sin\beta + \cos\alpha\cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Using so Prostaferesi's equation involving the sum of the argument in (co)sine functions we obtain the intuitive result of a pure revolution of $\alpha + \beta$ along the z axis:

$$\mathfrak{R}_2^w = \left[\begin{array}{ccc|c} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

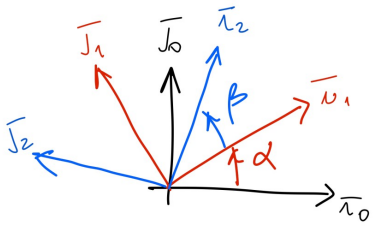


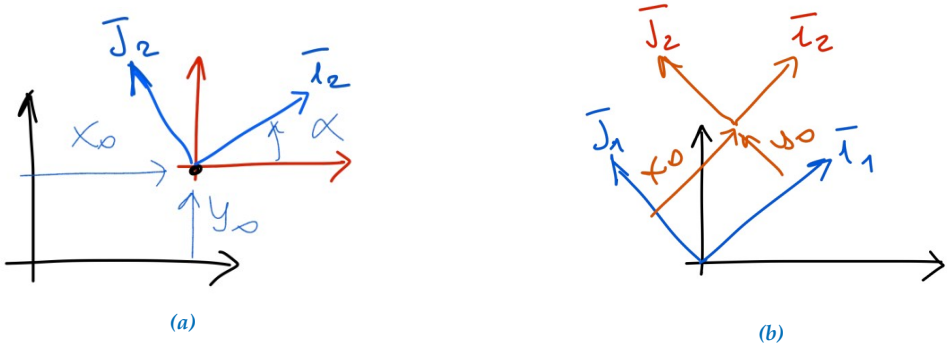
Figure 4.2: multiple transformations of pure rotation revolving the z axis.

Rotation and translation In the case of pure rotation/translation the reference frame that we have obtained was the same if we would have applied the transformation in reversed order, however this are only particular case. If we consider a translation $\mathfrak{R}_1$ and a rotation $\mathfrak{R}_2$ defined by matrices

$$\mathfrak{R}_1 = \left[\begin{array}{ccc|c} & \mathcal{I} & & \begin{matrix} x_{01}^w \\ y_{01}^w \\ 0 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad \mathfrak{R}_2 = \left[\begin{array}{ccc|c} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Intuitively the reference frame obtain by firstly applying the translation ($\mathfrak{R}_1$) is different from the one obtained by rotating first ($\mathfrak{R}_2$), as also shown in figure 4.3, in fact by performing the matrix calculations we obtain

$$\mathfrak{R}_1\mathfrak{R}_2 = \left[\begin{array}{ccc|c} & \mathcal{R}_\alpha & & \begin{matrix} x_{01}^w \\ y_{01}^w \\ 0 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad \mathfrak{R}_2\mathfrak{R}_1 = \left[\begin{array}{ccc|c} & \mathcal{R}_\alpha & & \begin{matrix} x_{01}^w \cos\alpha - y_{01}^2 \sin\alpha \\ x_{01}^2 \sin\alpha + y_{01}^w \cos\alpha \\ 0 \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$



(a) *(b)*

Figure 4.3: reference frame obtained by first translating and than rotating (a) and first rotating and then translating (b).

4.1.2 Primitive rotation matrices and sequences of rotations

As stated previously, the rotation matrix $\mathbf{R}$ (equation 4.3) contains 9 parameters 3 of which are independent (the other 6 are constrained by the conditions on the mutual orthogonality) and so a *easier* formulation can be achieved using only 3 parameters. In this sense the angular orientation of the vectorial base $\mathbf{e}^b$ can be though as the result of 3 successive rotation along axis; considering as example the sequence

$$\mathbf{e}^b = \underbrace{\mathbf{R}_1(\theta)\mathbf{R}_2(\phi)\mathbf{R}_3(\psi)}_{\mathbf{R}}\,\mathbf{e}^w \qquad \leftrightarrow \qquad \mathbf{e}^b = \mathbf{R}_x(\theta)\mathbf{R}_y(\phi)\mathbf{R}_z(\psi)\mathbf{e}^w$$

a local frame can be obtained by firstly rotating the global frame along the third (z) axis by an angle ψ. This operation determines a local frame $\mathbf{e}^{b''}$ on top of which a second rotation of angle ϕ along

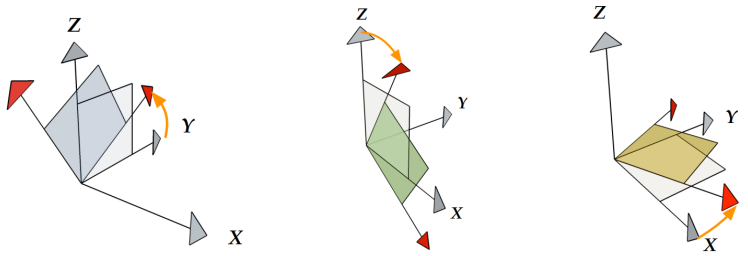


Figure 4.4: rotation matrices respect to the x (left), y (middle) and z (right) axis.

the second (y) axis is performed. That generates another vectorial space $\mathbf{e}^{b'}$ that finally determines the local base $\mathbf{e}^b$ as a rotation of an angle θ along the first (x) axis.

In particular the **primitive rotation matrices** are expressed as

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad \mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

Euler angles The **Euler angles** are a common choice in the description of the orientation in a body in space and is obtained by the sequence $z(\psi) \to x(\theta) \to z(\phi)$ (or *in numbers* by $3 \to 1 \to 3$) resulting in the transformation

$$\mathbf{R}_z(\phi)\mathbf{R}_x(\theta)\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi\cos\phi - \sin\psi\cos\theta\sin\phi & \sin\psi\cos\theta + \cos\psi\cos\theta\sin\phi & \sin\theta\sin\phi \\ -\cos\psi\sin\phi - \sin\psi\cos\theta\cos\phi & -\sin\psi\sin\phi + \cos\psi\cos\theta\cos\phi & \sin\theta\cos\phi \\ \sin\psi\sin\theta & -\cos\psi\sin\theta & \cos\theta \end{bmatrix} \tag{4.10}$$

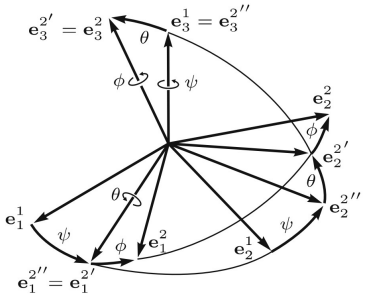


Figure 4.5: representation of the Euler angles ψ, θ, ϕ for representing the attitude of a body.

Bryan angles **Bryan** (or **Cardan**) **angles** are associated to another sequence of relative rotations carried along the sequence $1 \to 2 \to 3$ (x, y, z) resulting in

$$\mathbf{R}_z(\phi_3)\mathbf{R}_y(\phi_2)\mathbf{R}_x(\phi_1) = \begin{bmatrix} \cos\phi_2\cos\phi_3 & \cos\phi_1\sin\phi_3 + \sin\phi_1\sin\phi_2\cos\phi_3 & \sin\phi_1\sin\phi_3 - \cos\phi_1\sin\phi_2\cos\phi_3 \\ -\cos\phi_2\sin\phi_3 & \cos\phi_1\cos\phi_3 - \sin\phi_1\sin\phi_2\sin\phi_3 & \sin\phi_1\cos\phi_3 + \cos\phi_1\sin\phi_2\sin\phi_3 \\ \sin\phi_2 & -\sin\phi_1\cos\phi_2 & \cos\phi_1\cos\phi_2 \end{bmatrix} \tag{4.11}$$

Inverse sequence Until now we considered the **direct sequence** in order to describe the attitude of a local reference frame $\mathbf{e}^b$ respect to the global base $\mathbf{e}^w$, but sometime the inverse operation is required (to describe the attitude of a vector in $\mathbf{e}^b$ knowing it's coordinates in $\mathbf{e}^w$). In this case we have that the **inverse sequence** of rotation is obtained by inverting the rotation sequence and changing the sign to the angles as in this example:

$$\text{direct seq.: } \mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_x(\phi)\mathbf{R}_y(\theta) \quad \mapsto \quad \text{inverse seq.: } \mathbf{R}^{-1} = \mathbf{R}_y(-\theta)\mathbf{R}_x(-\phi)\mathbf{R}_z(-\psi) \tag{4.12}$$

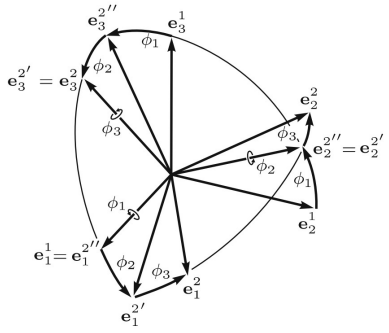


Figure 4.6: representation of the Bryan angles ϕ_1, ϕ_2, ϕ_3 for representing the attitude of a body.

Singular configuration In general the same orientation of the body can be expressed as function of different angles depending on the rotation sequence chosen, but however anyone of this representation presents a **singular configuration** (also referred s the ***gimbal lock***) where for particular values of the intermediate rotation this description doesn't allow to fully discriminate the attitude of the body. Considering as example the Euler angle representation (equation 4.10) in case of $\theta = k\pi$ (where $k \in \mathbb{Z}$), the evaluation and subsequence simplification of the matrix results in

$$\mathbf{R}_z(\phi)\mathbf{R}_x(0)\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\phi+\psi) & \sin(\phi+\psi) & 0 \\ -\sin(\phi+\psi) & \cos(\phi+\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As we can see such sequence results in a rotation matrix along the z axis of the starting reference frame of an angle $\phi + \psi$, hence we have only one parameter (instead of the required 3) to describe the orientation of the body, hence we have 2 degree of freedom left unknown.

From rotation matrix to angles In general problems what we might have is a rotation matrix $\mathbf{R}$ (whose components are described in equation 4.3) with 9 entries subjected to 6 constraints but that we want are a sequence of primitive rotations hence their associated angles. This operation can be obtained by solving the non-linear systems associated to $\mathbf{R}_i(\phi)\mathbf{R}_j(\theta)\mathbf{R}_k(\psi) = \mathbf{R}$ (where i, j, k are generically axis of rotation); such operation should be performed on the 3 *easier* expression. Considering as example the sequence of rotation $z(\psi) \to y(\theta) \to x(\phi)$ what we obtain is the equality

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\psi \\ \sin\phi\sin\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & -\sin\phi\cos\theta \\ -\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

The easier terms to match are $\cos\theta\cos\phi = u_x$, $-\cos\theta\sin\psi = v_x$ and $-\sin\phi\cos\theta = w_y$ that, simultaneously solved, determines the angles

$$\begin{aligned} \phi &= \arctan\left(-\frac{u_x}{\sqrt{u_x^2+w_y^2}}, \frac{w_y}{\sqrt{u_x^2+w_y^2}}\right) \qquad \theta = -\arccos\left(-\sqrt{u_x^2+w_y^2}\right) \\ \psi &= \pi - \arcsin\left(\frac{v_x}{\sqrt{u_x^2+w_y^2}}\right) \end{aligned} \tag{4.13}$$

In reality this is just one of the multiple solutions of the non-linear system (in this particular case symbolic solvers determine 8 different solutions for the angles that satisfy the 3 equation) hence multiple combination of angles ϕ, θ, ψ can lead to the same rotation matrix.

4.1.3 Rotation axis, Euler theorem and quaternions

As was previously stated, the rotation matrix $\mathbf{R}$ relates the relative orientation of two bases $\mathbf{e}1, \mathbf{e}2$; being that such matrix lies in the set of the special orthogonal matrices is characterised by having the

inverse equal to the transpose and has $\det \mathbf{R} = 1$. Another important property that can be obtained is by considering the **eigenvalue problem** $\mathbf{R}\boldsymbol{v}_b = \lambda \boldsymbol{v}_w$, meaning that we would like to find if there's a vector $\boldsymbol{v}$ whose orientation isn't altered after the transformation has occurred. Mathematically to compute such problem we have to first compute the eigenvalues as

$$\det\left(\mathbf{R} - \lambda \mathcal{I}\right) = p(\lambda) = 0$$

where $p(\lambda)$ is the characteristic polynomial of $\mathbf{R}$ that evaluates to

$$\begin{aligned} p(\lambda) &= -\lambda^3 + \lambda^2 \mathrm{tr}\left\{\mathbf{R}\right\} - \lambda \mathrm{tr}\left\{\mathbf{R}\right\} + \det \mathbf{R} \\ &= (\lambda - 1)\left(\lambda^2 - (\mathrm{tr}\left\{\mathbf{R}\right\} - 1)\lambda + 1\right) \end{aligned}$$

It is also proven (as shown) that such characteristic polynomial has always a unitary eigenvalue, meaning that always exists one direction of transformation that remain unaltered in both *orientation* and *elongation*: this direction is hence called **rotation axis**. Moreover the other two eigenvalues $\lambda_{2,3}$ of $p(\lambda)$ are complex conjugated and are used to determine the **rotation angle** φ of the transformation along the rotation axis:

$$\begin{aligned} \lambda_{2,3} = \frac{\mathrm{tr}\left\{\mathbf{R}\right\} - 1}{2} \pm j\sqrt{1 - \left(\frac{\mathrm{tr}\left\{\mathbf{R}\right\} - 1}{2}\right)^2} &= \cos\varphi \pm j \sin\varphi = e^{\pm j\varphi} \\ \Rightarrow \qquad \varphi = \arccos\left(\frac{\mathrm{tr}\left\{\mathbf{R}\right\} - 1}{2}\right) & \end{aligned} \tag{4.14}$$

Note that the eigensystem doesn't result in one unique solution: if we denote $\mathbf{R}(\hat{\boldsymbol{n}}, \theta)$ the rotation of an angle θ along the direction $\hat{\boldsymbol{n}}$, then the same transformation can be obtained considering the opposite axis and angle:

$$\mathbf{R}(-\hat{\boldsymbol{n}}, -\theta) = \mathbf{R}(\hat{\boldsymbol{n}}, \theta)$$

This way of defining rotations is also periodic, in the sense that we can restrict any transformation along on axis to an angle $\theta \in [0, 2\pi]$

$$\mathbf{R}(\hat{\boldsymbol{n}}, \theta + 2k\pi) = \mathbf{R}(\hat{\boldsymbol{n}}, \theta) \qquad \forall k \in \mathbb{Z}$$

Moreover if we consider a rotation with rotation angle $\theta = 0$, then such transformation is undetermined (in fact no displacement occurs). With all this premises the **Euler theorem** holds and states that *the displacement of a body-fixed base from an initial position* $\mathbf{e}^1$ *to an arbitrary final position* $\mathbf{e}^2$ *is achieved by a rotation through a certain angle about an axis which is fixed in both bases. The axis has the direction of the eigenvector associated with the eigenvalue* $\lambda = 1$ *of the direction cosine matrix* $\mathbf{R}$.

Rotation matrix given an axis and the angle In practical application we might want to have that a rotation of an angle θ occurs along a certain direction $\hat{\boldsymbol{n}}$; in order so to compute the rotation matrix $\mathbf{R}(\hat{\boldsymbol{n}}, \theta)$, a way to do so is to firstly apply two generic rotation (as example $\mathbf{R}_z(\alpha)\mathbf{R}_x(\beta)$) in order to have a reference frame $\mathfrak{R}_{temp}$ whose third base component $\hat{\boldsymbol{k}}_{temp}$ is aligned with the direction $\hat{\boldsymbol{n}}$ in the ground reference frame. On such frame we can compute the rotation of the angle θ (along the z axis) but in order to obtain the final reference frame we have to apply the inverse transformation $\mathfrak{R}_{temp}^{-1} = \mathbf{R}_x(-\alpha)\mathbf{R}_z(-\beta)$ (using the result of equation 4.12). The rotation matrix can so be regarded as

$$\mathbf{R}(\hat{\boldsymbol{n}}, \theta) = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(-\beta)\mathbf{R}_x(-\alpha) \tag{4.15}$$

At this point all that's left is to determine such angles α, β in order to have the alignment of the temporary reference frame with the chosen direction $\hat{\boldsymbol{n}}$: considering that the associated rotation matrix is

$$\mathbf{R}(\alpha, \beta) = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta) = \left[\hat{\boldsymbol{i}}_2(\alpha, \beta) | \hat{\boldsymbol{j}}_2(\alpha, \beta) | \hat{\boldsymbol{k}}_2(\alpha, \beta)\right]$$

where $\hat{\boldsymbol{i}}_2, \hat{\boldsymbol{j}}_2, \hat{\boldsymbol{k}}_2$ are the versor of the temporary base $\mathbf{e}^{temp}$ in global coordinates and so in order to have that $\hat{\boldsymbol{k}}_2$ is parallel to $\hat{\boldsymbol{n}}$ we have to solve the non-linear system in α, β determined as

$$\begin{cases} \hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{i}}_2 = 0 \\ \hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{j}}_2 = 0 \end{cases}$$

Rodrigues formula Given a rotation axis $\hat{\boldsymbol{n}}$ and the rotation angle θ, the **Rodrigues formula** allows to compute the rotated coordinates $\boldsymbol{r}_w^{rot}$ of a vector with starting components $\boldsymbol{r}_b$:

$$\boldsymbol{r}_w^{rot} = \cos\theta\, \boldsymbol{r}_b + (1-\cos\theta)(\hat{\boldsymbol{n}}\cdot\boldsymbol{r}_b)\hat{\boldsymbol{n}} + \sin\theta\, \hat{\boldsymbol{n}}\times\boldsymbol{r}_b \tag{4.16}$$

Considering that $(\hat{\boldsymbol{n}}\cdot\boldsymbol{r}_b)\hat{\boldsymbol{n}}$ can be regarded as $\boldsymbol{r}_v + \hat{\boldsymbol{n}}\times(\hat{\boldsymbol{n}}\times\hat{\boldsymbol{r}}_b)$ but also that the vectorial product $\hat{\boldsymbol{n}}\times\boldsymbol{v}$ (with $\boldsymbol{v}\in\mathbb{R}^3$ is a generic vector) can be regarded as the following matrix product

$$\mathbf{N}_s\boldsymbol{v} := \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{4.17}$$

With this we can rewrite Rodrigues formula as

$$\boldsymbol{r}_w^{rot} = \Big(I + (1-\cos\theta)\mathbf{N}_s\mathbf{N}_s + \sin\theta\mathbf{N}_s\Big)\boldsymbol{r}_b \tag{4.18}$$

> **Note:** In general any vectorial product $\boldsymbol{w}\times\boldsymbol{v}$ can be reduced to a matrix multiplication in the form $\mathbf{W}\boldsymbol{v}$, where $\mathbf{W}\in\mathbb{R}^{3\times3}$ is a skew-symmetric matrix in the form shown.

By defining $q_0 = \cos(\theta/2)$ and $\tilde{\boldsymbol{q}} = (q_1,q_2,q_3) = \hat{\boldsymbol{n}}\sin(\theta/2) = \big(n_x\sin(\theta/2), n_y\sin(\theta/2), n_z\sin(\theta/2)\big)$ what we obtain are the 4 **Euler-Rodrigues parameters**; in particular $\tilde{\boldsymbol{q}}$ is a vector lying in the rotation axis (is in fact proportional by a factor $\sin(\theta/2)$ to $\hat{\boldsymbol{n}}$). Such formulation is characterized by having that

$$\sum_{i=0}^{3} q_i^2 = 1$$

The vector $\boldsymbol{q} = (q_0,\tilde{\boldsymbol{q}}) = (q_0,q_1,q_2,q_3)\in\mathbb{R}^4$ is the so called **quaternion** and allows to give a global parametrization of the rotation matrix $\mathbf{R}$ (it is said *global* because it doesn't require the definition of intermediate reference frames to determine the rotations). We can in fact show that each rotation matrix can be expressed in quaternions as

$$\mathbf{R}(\boldsymbol{q}) = \begin{bmatrix} 2q_0^2+2q_1^2-1 & -2q_0q_3+2q_1q_2 & 2q_0q_2+2q_1q_3 \\ 2q_0q_3+2q_1q_2 & 2q_0^2+2q_2^2-1 & -2q_0q_1+2q_2q_3 \\ -2q_0q_2+2q_1q_3 & 2q_0q_1+2q_2q_3 & -2q_1^2-2q_2^2+1 \end{bmatrix} \tag{4.19}$$

Such parametrization has 4 parameters, but only 3 of them are independent and so, while describing systems in such representation, the following constraint equation must always be introduced:

$$q_0^2+q_1^2+q_2^2+q_3^2 = 1$$

The cost of adding one more parameter in the description of the system (and the related constraint equation) comes with the advantage of having a representation that's always non-singular (so it doesn't have the gimbal lock problem, as for *common* rotation sequences that are intuitively easier to understand and physically interpret).

CHIEDERE LA CONFIGURAZIONE SINGOLARE PER $\theta = 0$ o $q_0 = 0$?x

4.1.4 Velocities and acceleration

Usually reference frames moves respect to each other, and so in general is important to understand how velocities (and accelerations) in local frames relates to their respective in global coordinates. For this reason we determine the **angular rate** $\boldsymbol{\omega} = (\omega_x,\omega_y,\omega_z)$ as the vector of 3 components that represents the rate of change of a reference frame with respect to another one (typically the world reference frame).

Given a vectorial base characterized by the rotation matrix $\mathbf{R}$, compute the derivative of it's components in time t determines the rate of change $\dot{\mathbf{R}}$ of the body in the world reference frame. Considering

that $\mathbf{R}^T = \mathbf{R}^{-1}$ is the inverse rotation matrix that match the world coordinates with the body one, we have that

$$\mathbf{R}^T\dot{\mathbf{R}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \mathbf{P}_\Omega \tag{4.20}$$

where $\mathbf{P}_\Omega$ is a skew-symmetric matrix usual referred as **angular operator** that's used to implement the vectorial product of $\boldsymbol{\omega}$: it happens that $\boldsymbol{\omega} \times \boldsymbol{x} = \mathbf{P}_\Omega \boldsymbol{x}$ for any vector $\boldsymbol{x} \in \mathbb{R}^3$. In particular $\boldsymbol{\omega}$ represent the projection of the rotation axis in the local reference frame; moreover if we consider $\boldsymbol{u} \in \mathbb{R}^m$ the vector of the parameters used in the description of the attitude of the body, the angular rate can be regarded as a linear combination of the rate of changes of such parameters:

$$\boldsymbol{\omega}^b = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \mathbf{E}(\boldsymbol{u})\,\dot{\boldsymbol{u}} \qquad \mathbf{E}(\boldsymbol{u}) \in \mathbb{R}^{n\times m}$$

In particular

- if we consider a set of 3 independent parameters as the 3 angles $\boldsymbol{u} = (\alpha, \beta, \theta)$ generating the rotation matrix $\mathbf{R} = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\theta)$ we can compute the angular operator $\mathbf{P}_\Omega = \mathbf{R}^T\dot{\mathbf{R}}$ determining

$$\boldsymbol{\omega}^b = \underbrace{\begin{bmatrix} \sin\beta & 0 & 1 \\ \cos\beta\sin\theta & \cos\theta & 0 \\ \cos\beta\cos\theta & \sin\beta & 0 \end{bmatrix}}_{=\mathbf{E}(\boldsymbol{u})} \underbrace{\begin{pmatrix} \dot\alpha \\ \dot\beta \\ \dot\theta \end{pmatrix}}_{=\dot{\boldsymbol{u}}}$$

- if we would have instead a quaternion representation $\boldsymbol{u} = (q_0, q_1, q_2, q_3)$, considering the definition in equation 4.19 for the rotation matrix, from the computation $\mathbf{R}^T\dot{\mathbf{R}} = \mathbf{P}_\Omega$ we would have obtained

$$\boldsymbol{\omega}^b = 2\begin{bmatrix} q_3 & q_2 & -q_1 & q_0 \\ q_2 & -q_1 & -q_0 & q_3 \\ -q_1 & q_0 & q_3 & q_2 \end{bmatrix} \begin{pmatrix} \dot q_0 \\ \dot q_1 \\ \dot q_2 \\ \dot q_3 \end{pmatrix}$$

  The number of free parameters is 3 because one is constrained by the equation $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ and so, also in this case, we have to consider the differential of such constraint in order to univocally determine the angular velocity:

$$\frac{d}{dt}(\boldsymbol{q}\cdot\boldsymbol{q} = 1) \qquad \rightarrow \qquad 2q_0\dot q_0 + 2q_1\dot q_1 + 2q_2\dot q_2 + 2q_3\dot q_3 = 0$$

We observe so that the problem of the velocities is linear if we know the actual configuration/position of the system.

Velocity Recalling equation 4.1, the global coordinates $\boldsymbol{P}^w$ of a point described by $\boldsymbol{P}^b$ in a local reference frame is defined as

$$\boldsymbol{P}^w = \boldsymbol{P}_0 + \mathbf{R}\boldsymbol{P}^b$$

where $\boldsymbol{P}_0$ is the origin of the moving frame in the global coordinates. Differentiating in time allows to determine the velocity

$$\begin{aligned} (4.1) \quad \xrightarrow{\frac{d}{dt}} \quad \dot{\boldsymbol{P}}^w &= \dot{\boldsymbol{P}}_0 + \dot{\mathbf{R}}\boldsymbol{P}^b + \mathbf{R}\dot{\boldsymbol{P}}^b = \dot{\boldsymbol{P}}_0 + \overbrace{\mathbf{R}\mathbf{R}^T}^{=\mathcal{I}}\mathbf{R}\boldsymbol{P}^b + \mathbf{R}\dot{\boldsymbol{P}}^b \\ &= \dot{\boldsymbol{P}}_0 + \mathbf{R}\mathbf{P}_\Omega\boldsymbol{P}^b + \mathbf{R}\dot{\boldsymbol{P}}^b \end{aligned} \tag{4.21}$$

where $\dot{\boldsymbol{P}}_0$ is the contribution due to the velocity in the translation of the local frame, $\mathbf{R}\mathbf{P}_\Omega\boldsymbol{P}^b$ is due to the frame angular velocity and $\mathbf{R}\dot{\boldsymbol{P}}^b$ is an optional terms that considers the relative velocity $\dot{\boldsymbol{P}}^b$ that the point might have in the local frame. The associated **transformation matrix** is so

$$\dot{\boldsymbol{P}}^w = \left[\begin{array}{c|c} \mathbf{R}\mathbf{P}_\Omega & \dot{\boldsymbol{P}}_0 \\ \hline \mathbf{0}^t & 0 \end{array}\right]\boldsymbol{P}^b + \left[\begin{array}{c|c} \mathbf{R} & \dot{\boldsymbol{P}}_0 \\ \hline \mathbf{0}^t & 1 \end{array}\right]\dot{\boldsymbol{P}}^b$$

Acceleration Differentiating 4.21 one more time respect to time allows to compute the **acceleration** of the point P in the global reference frame as

$$\xrightarrow{\frac{d}{dt}} \qquad \ddot{\boldsymbol{P}}^w = \ddot{\boldsymbol{P}}_0 + \ddot{\mathbf{R}}\boldsymbol{P}^b + \dot{\mathbf{R}}\dot{\boldsymbol{P}}^b + \dot{\mathbf{R}}\dot{\boldsymbol{P}}^b + \mathbf{R}\ddot{\boldsymbol{P}}^b \tag{4.21}$$

Considering that $\ddot{\mathbf{R}} = \frac{d}{dt}\dot{\mathbf{R}} = \frac{d}{dt}(\mathbf{R}\mathbf{P}_\Omega) = \dot{\mathbf{R}}\mathbf{P}_\Omega + \mathbf{R}\dot{\mathbf{P}}_\Omega = \mathbf{R}\mathbf{R}^T\dot{\mathbf{R}}\mathbf{P}_\Omega + \mathbf{R}\mathbf{P}_\Omega\mathbf{P}_\Omega$ what we obtain is

$$\begin{aligned}\ddot{\boldsymbol{P}}^w &= \ddot{\boldsymbol{P}}_0 + \mathbf{R}\mathbf{P}_\Omega\mathbf{P}_\Omega\boldsymbol{P}^b + \mathbf{R}\dot{\mathbf{P}}_\Omega\boldsymbol{P}^b + 2\mathbf{R}\mathbf{P}_\Omega\dot{\boldsymbol{P}}_b + \mathbf{R}\ddot{\boldsymbol{P}}^b \\ &= \ddot{\boldsymbol{P}}_0 + \underbrace{\mathbf{R}\Big(\underbrace{\mathbf{P}_\Omega\mathbf{P}_\Omega\boldsymbol{P}^b}_{\text{centripetal acc.}} + \underbrace{\dot{\mathbf{P}}_\Omega\boldsymbol{P}^b}_{\text{tangential acc.}} + \underbrace{2\mathbf{P}_\Omega\dot{\boldsymbol{P}}^b + \ddot{\boldsymbol{P}}^b}_{\text{relative acc}}\Big)}_{\text{acceleration in the local frame}}\end{aligned} \tag{4.22}$$

4.1.5 Natural coordinates

In general rotation matrices $\mathbf{R}$ can be expressed with any set of parameters we want; the idea is that the matrix $\mathbf{R} = \begin{bmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{bmatrix}$ can be described with 9 parametric components constrained by 6 equation: this determines a more redundant formulation that however generates a new way to describe the configuration of bodies. The main idea is now to use global coordinates of points (that mechanically can be associated to joints) or vector (that can represent the direction relative translation/rotation of two elements) in order to build the rotation matrix $\mathbf{R}$.

The advantage of this approach is that we have at maximum quadratic equations that can easily solved numerically by calculators with the problem that quadratic constraints generates 2 solutions each.

Planar case with two points If we consider two points P_1, P_2 lying in the same rigid body (hence their distance L is fixed) characterized by global cartesian coordinates $(x_1, y_1), (x_2, y_2)$ respectively (for simplicity we consider a planar case), the whole system can be described with the coordinates $\boldsymbol{q} = (x_1, y_1, x_2, y_2)$. In order to determine $\mathbf{R}$ (that requires 3 independent parameters) as function of $\boldsymbol{q}$ (that actually has 4 parameters) we need to determine one constraint equation.

In order to determine the moving reference frame it is necessary firstly to chose one point as origin of such frame (in this case we consider P_1) and we have to orient one versor of the frame (in this case $\hat{\boldsymbol{i}}_1$) with the vector $\boldsymbol{P_1P_2}$ connecting the two points. Being the body rigid we have that the constraint equation is related to the length between the points that's fixed, in fact

$$\boldsymbol{P_1P_2} \cdot \boldsymbol{P_1P_2} = L^2$$

By what we just stated, the first versor $\hat{\boldsymbol{i}}_1$ of the moving frame in global coordinates can be computed as the normalization of the vector $\boldsymbol{P_1P_2}$, hence

$$\hat{\boldsymbol{i}}_1 = \frac{\boldsymbol{P_1P_2}}{L} = \begin{pmatrix} \frac{x_2 - x_1}{L} \\ \frac{y_2 - y_1}{L} \\ 0 \end{pmatrix} = \begin{pmatrix} i_{1x} \\ i_{1y} \\ i_{1z} \end{pmatrix}$$

The goal now is to determine the coordinates of the other versor $\hat{\boldsymbol{j}}_1$ that compose the rotation matrix that, in the planar case, has the form

$$\mathbf{R} = \begin{bmatrix} i_{1x} & j_{1x} & 0 \\ i_{1y} & j_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Knowing that a proper rotation matrix requires $\hat{\boldsymbol{i}}_1 \cdot \hat{\boldsymbol{j}}_1 = 0$ we obtain the relation

$$i_{1x}j_{1x} + i_{1y}j_{1y} = 0 \qquad \Rightarrow \qquad j_{1x} = -\frac{i_{1y}}{i_{1x}}j_{1y} = -\frac{y_2 - y_1}{x_2 - x_1}j_{1y}$$

Knowing that it must also hold $\hat{\boldsymbol{j}}_1 \cdot \hat{\boldsymbol{j}}_1 = 1$ we build the quadratic relation

$$j_{1x}^2 + j_{1y}^2 = j_{1y}^2 \left(1 + \frac{(y_2 - y_1)^2}{(x_2 - x_1)^2}\right) = 1 \qquad \Rightarrow \qquad \Rightarrow j_{1y} = \pm \frac{x_2 - x_1}{L}$$

We so have that the 2 possible formulation of the rotation matrix are

$$\mathbf{R} = \begin{bmatrix} \frac{x_2 - x_1}{L} & -\frac{y_2 - y_1}{L} & 0 \\ \frac{y_2 - y_1}{L} & \frac{x_2 - x_1}{L} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{or} \qquad \mathbf{R} = \begin{bmatrix} \frac{x_2 - x_1}{L} & \frac{y_2 - y_1}{L} & 0 \\ \frac{y_2 - y_1}{L} & -\frac{x_2 - x_1}{L} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

that are subjected to

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 = L^2$$

The problem of this formulation is that it isn't intuitive the rotation α of the reference frame respect to the global direction $\hat{\boldsymbol{i}}_0$, however this can be achieved equating $\mathbf{R}$ with the rotation matrix function of α:

$$\begin{bmatrix} \frac{x_2 - x_1}{L} & -\frac{y_2 - y_1}{L} & 0 \\ \frac{y_2 - y_1}{L} & \frac{x_2 - x_1}{L} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \sin\alpha = \frac{y_2 - y_1}{2} \quad \Rightarrow \quad \alpha = \arcsin\left(\frac{y_2 - y_1}{2}\right)$$

4.2 Multi-body approach

A system characterized by n_b free bodies (in the sense that they are not constrained together) results in a

$$6n_b \text{ degrees of freedom}$$

for the system; such value indeed represents also the minimum number of components for the **generalized coordinates** $\boldsymbol{q} \in \mathbb{R}^n$ of the system itself. Note that this is the minimum value, because if we use quaternion representation for representing the transformation matrices associated to bodies, each of them has 7 different parameters (not 6, but it also introduced the associated constraint equation).

Path and trajectory It is important to distinct the **path** as a sequence of configurations (that are independent from the concept of time) and the **trajectory** as the sequence of configurations that are changing with time. In this sense a trajectory can be regarded as a parametrization in time of a path.

Independent motions Each added mechanical joint results in a mathematical constraint that reduces the degree of freedom of the system; such joints are described by the vectorial map $\boldsymbol{\Phi}(\boldsymbol{q}) = \mathbf{0} \in \mathbb{R}^m$ that models m constraint equations. This reduces the number n_i of independent generalized coordinates:

$$\boldsymbol{q} = (\boldsymbol{q}_i, \boldsymbol{q}_d)$$

where $\boldsymbol{q} \in \mathbb{R}^n$ are the *original* generalized coordinates, $\boldsymbol{q}_i \in \mathbb{R}^{n_i}$ are the independent and $\boldsymbol{q}_d \in \mathbb{R}^{n_d}$ are the dependent coordinates (of course it must hold $n_i + n_d = n$). The total number of **degrees of freedom** (representing the minimum number of independent motion to the describe the configuration of the multi-body system) of the system can so be computed as

$$DOF = n - m$$

In general the constrained degrees of freedom strictly depends on the actual position and geometry of the system that's strictly related to time, and so given the constraint map $\boldsymbol{\Phi}(\boldsymbol{q}) : \mathbb{R}^n \mapsto \mathbb{R}^m$ (where $n > m$) the actual number of constrained degrees of freedom is

$$m = \text{rank}\left\{\frac{\partial \boldsymbol{\Phi}(\boldsymbol{q})}{\partial \boldsymbol{q}}\right\} \tag{4.23}$$

where $\partial\boldsymbol{\Phi}/\partial\boldsymbol{q} \in \mathbb{R}^{m \times n}$ is the Jacobian of the map $\boldsymbol{\Phi}$. If the Jacobian has so it's maximum rank it also means that $\det \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}} \neq 0$; every time such determinant is null we are in a situation where 1 or more joints are redundant: such condition might happen locally, but if it happens permanently it is necessary to eliminate the redundant equation in order to make the solver solve the problem.

Velocity constraints If we have that the constraint map $\boldsymbol{\Phi}$ set a conditions on the generalized coordinates $\boldsymbol{q}$ to be $\boldsymbol{\Phi}(\boldsymbol{q}) = \mathbf{0}$, then it means that also the velocities $\dot{\boldsymbol{q}}$ are constrained: deriving in time $\boldsymbol{\Phi}(\boldsymbol{q}) = \mathbf{0}$ results in fact

$$\frac{\partial \boldsymbol{\Phi}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{\Phi}(\boldsymbol{q})}{\partial t} = \mathbf{0} \tag{4.24}$$

Constraint classification The constraint map $\boldsymbol{\Phi}$ can be classified respect to different aspects; if the constraint explicitly depends on time $\boldsymbol{\Phi}$ is said **rehonomous** while in the other case is said **scleronomous**, mathematically

$$\boldsymbol{\Phi}\big(\boldsymbol{q}(t), t\big) = 0\text{: rehonomous} \qquad \boldsymbol{\Phi}\big(\boldsymbol{q}(t)\big) = 0\text{: scleronomous}$$

Constraint are said **holonomic** if it involves only the actual configuration of the system, while is **non-holonomic** if it requires also velocities or higher order derivative of $\boldsymbol{q}$:

$$\boldsymbol{\Phi}\big(\boldsymbol{q}(t)\big) = 0\text{: holonomic} \qquad \boldsymbol{\Phi}\big(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \dots\big) = 0\text{: non-holonomic}$$

4.2.1 Topology

A **multi-body** system is a collection of (multiple) bodies, joints, forces and torques; the **topology** studies in particular how such elements are connected specially involving the use od **linear graphs**.

In order to construct such graphs (as exemplified in figure 4.7) we can follow this procedure:

1. define a reference frame for each body and assign them a node (in the next figure green);
2. define auxiliary reference frames that will be used to define constraints; such elements are still nodes in the linear graph but doesn't introduce any additional coordinate because they are fixedly dependent on the relative body frame;
3. connect all the nodes with *arrow* (formally edges) representing the constraints between nodes (reference frames).

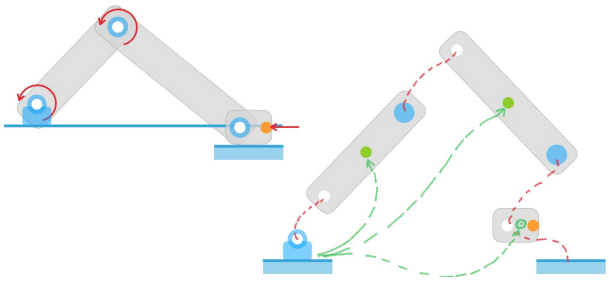


Figure 4.7: example of linear graph generation of a slider-crank mechanism where each body is characterized by a reference frame in ground coordinates (green nodes) and dashed-red lines are describing the mechanical joints.

Defining the **path** as a (finite/infinite) sequence of edges which joins a set of distinct nodes, we then have a **circuit loop** when the path begins and ends at the same vertex. In contrary a **tree** is an undirected graph in which any two vertices are connected by exactly one path.

Considering as example the linear graph in figure 4.8 we conventionally denote with h the edges associated to kinematic equation (that are strictly related to the mechanical joints), with r the edges representing fixed transformations between two reference frames in a body, with b the time-varying transformation between the body reference frame respect to ground. Auxiliary edges a might be required to describe the transformations between reference frames that are not directly connected.

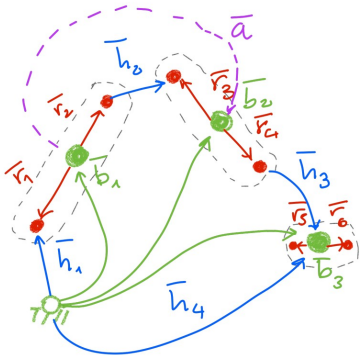


Figure 4.8: linear graph of the slider-crank mechanism of figure 4.7.

Tree and closed loops A graph is a *topological tree* if there exists exactly one path between any two nodes in the graph. If so the connectivity graph of a rigid body system is a topological tree then such system is regarded as a **kinematic tree**. Such kind of graphs have special properties that make it easy to calculate their dynamics efficiently.
Moreover a **spanning tree** of a graph G, denoted as G_t, is a sub-graph of G containing all the nodes in G together with any subset of the arcs in G such that G_t is a topological tree (as in figure 4.9); is is so proven that each connectivity graph has at least one spanning tree and if G is already a tree, then $G_t = G$ (in all other case the spanning trees are not unique).

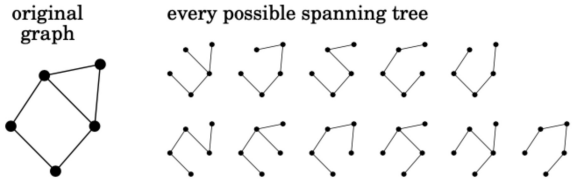


Figure 4.9: example of a graph that's not a topological tree (left) and all possible spanning trees (right).

A **cycle** is instead a closed path in a graph, hence a tree (to be so) must present no cycles. Each cycle in a connectivity graph defines a **kinematic loop**, however not all of them are useful: only the independents one must be used for the description of the system. For equation of motion only the minimal set of loops must be used. If a system has a kinematic loop then it's referred as **closed-loop system**.

4.2.2 Global and recursive approaches

Recursive formulation

In a **recursive approach** to solve the kinematic of a system we firstly need to generate a linear graph that's a tree in order so to have an open-loop system. The idea is so to determine a sequence of nodes (starting from ground for simplicity) and then follow the kinematic chain by applying all the transformation. Following such sequence we build reference frames on top of each others: each edge h is so a transformation matrix that introduces one or more generalized coordinates that are used to describe the motion of the system (hence describe the relative motion of the joint). Other edges r are instead fixed and are used to *move* inside the body.

Such method is characterized by having the minimal number of dependent coordinates; in fact if the linear graph is a tree then the number of coordinates represents the degrees of freedom of the system, while if the multi-body system has some loops we have the introduction of constraints equations.

AGGIUNGERE ESEMPIO

Global formulation

Using a **global approach** each body is initially considered as free and each body reference frame is described with parametric transformation matrices (respect to ground). Auxiliary reference frames are

constructed on top of them and then constraint equations h are applied to *assemble* the mechanism. This approach is characterized by having mainly global coordinates and simpler constraint equations, however it produces a higher number of dependent coordinates (way much more than the degrees of freedom of the system).