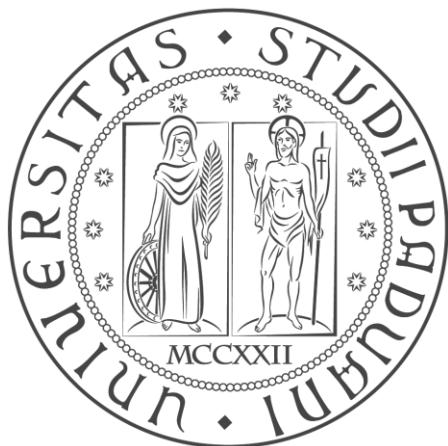


Simscape based robot digital twin development supported by symbolic modeling



Student: Soldà Enrico

Mt. v2087499

Relator Paolo Boscariol

Academic year 2023/2024

Summary

Summary	1
Abstract.....	5
Premises.....	5
Introduction	5
Reasons for this work.....	6
Why matlab and Simulink-Simscape environment.....	7
Short work roadmap	9
On modeling robots	10
Kinematics	11
Rigid body motion.....	28
Number of degree of freedom.....	36
Lie Groups and Algebra of transformations and rotations	43
Screw coordinates	55
Links kinematics	56
Semi transformations (inboard and outboard) and more details in frames propagation.....	64
Differential kinematics equations	66
Kinematic differential equations	67
Direct and inverse kinematics	67
Dynamics	73
Newton Euler algorithm customization	87
Reduction of complexity of equations: complexity considerations	102
Use of inertia matrix for determining free motion.....	102
Simplifications of model	103
Sensitivity and dynamic matrices, dynamic coefficient and identification.....	108
Inertia matrix.....	109
Generalization of inertia matrix concept.....	110
A carnot theorem generalization	115
Inertia matrix calculation method: direct and indirect.....	116
Inertia matrix inversion.....	122
Inertial matrix examples and special case.....	123
Pure inertia revolute joints only robot	124
Prismatic joints	124
A special case: planar robot.....	128
Actuation and Actuators	131
Joint actuation and control	134
Parameters identification and filtering	147
Parameters identification and estimation (learning).....	148
Robot parameters identification	150

Modeling robot dynamics as function of inertial parameters.....	166
Trajectory adjustment	176
Simscape modelling	206
Simscape/Simulink modeling process.....	207
Forces modelling	217
Mechanical systems simulation and visualization.....	219
Conventional body frames	221
Structure of a link representative block	221
Geometrical modeling, CAD importing and inertial estimate	222
Transformation block parameters	232
Solver initial conditions	235
Simulations with simscape models: trade off time accuracy.....	239
Model instrumenting	239
Requirements and test points	241
Inverse dynamics model (IDM)	251
Direct or forward dynamics and simulation (FD or DD)	251
Hybrid/mixed inverse-direct dynamics with simscape models	252
Gripper modelling	253
Code of Practices (COP)	254
Solver Parameters.....	260
Limit Penetration Depth	260
Flexible bodies.....	260
Code generation	266
Simulink/Simscape based identification without analytical models	268
Reinforcement learning.....	279
How does reinforcement learning works using simscape	279
Observations	284
Agents	284
Policy.....	285
Policy as Markov Decision Process (MDP)	287
Reinforcement learning and identification	288
Robot Learning	288
Inverse Reinforcement Learning (IRL)	289
Case study I: SCARA robot modeling and identification.....	291
Overcoming physical and experimental limitations using digital twins	304
From offline least squares to recursive least square and online estimate.....	306
SCARA robot direct dynamics and inertia matrix inversion	306
Multiphysics	308
Temperature effects on friction parameters	308
Additional modeling	308
identification using recursive least square in matlab environment	309
Identification using online LS estimator in Simulink/Simscape environment....	310
Kalman filter using simulink model.....	313

Case study II: UR5 articulated manipulator 6dof robot	314
Conversion of URDF files to simscape and automatic instrumenting	357
SIMSCAPE multiphysics.....	358
Automatic generation of tree like models	358
Dynamic model manipulation	358
Identification of model.....	359
Case study: importing and simulation of Valkyrie humanoid robot	360
Future works and developments	362
Bibliography.....	364

Abstract

Robotic systems involve different physical phenomena and require accurate modeling for assuring high level operative performances, justifying the use of digital twins for different operative phases. Here we propose the use of Simscape as a modeling environment for development of digital twins models involving different aspects from kinematics and dynamics, offline and online estimates up to control. The potential of this environment is multiple and some not complete examples are provided. We underline the essential role of symbolic manipulation, that in robotics has always played a major role, other environments are not able to support it as effectively, in spite of its not specific design for robotic applications. Development of even complex and advanced applications is illustrated.

Premises

In this work we do not have the opportunity, because of lack of time and resources, for deepening all subjects potentially affected by the core topic of this work. Consequently we do not pretend to be complete and exhaustive in the treatment of the matter here, since it would require a by far more extensive work, we've no place nor time for. Indeed Simscape modeling could touch a vast range of applications and study cases of interest for robotics as well for collateral applications. Digital twin subject is by itself a wide subject that would deserve a specific treatment, we cannot cover completely on this work, especially in regard to operative aspects involving communications and synchronization in real time and near real time that are of no secondary relevance. On the other side the lack of physical cyber systems targets availability limits the opportunity for a complete implementation. For these reasons we focus this work on the preliminary part of digital modeling, that's in truth, as we're going to show, also the most relevant in terms of utility and reusability. In author opinion, a few months of full time work could lead to the development of a full suite for studying [multibody systems](#) both analytically and numerically with robotic specific functionalities.

Introduction

Under a historical stream of increasing digitalization and applications of robotics in a variety of fields even outside strict industrial environment, adoption of digital twins of increasingly reliable, representative and with extended functionalities is a natural expectation.

This work is related to a survey on applicability, opportunities and lack of application of Simscape in Matlab®/simulink® environment for developments of digital twins for robotic applications. Throughout hereafter pages we will see how different tools of the Matlab environment could effectively support the development of physical systems, digital twins supported and supporting symbolic modeling of systems. We want to underline from one side advantages and disadvantages of numerical models and how they can synergically be integrated. In robotics the use of symbolic modeling ages back to the aurora of the applicative field motivated by practical needs even more than academic reasons. Nonetheless still today, where across almost any engineering field numerical methods and simulations play prominent role on the scene, we still see large adoption and application of symbolic modeling, of which we want to underline both advantages and disadvantages, adopting a perspective not

separated but flanking and supporting numerical side, not ignoring other primary thematic related, especially when dealing with digital twins like geometric modeling and physical modeling. Use of symbolic approach has taken a rather large space in this work for underlining potential of a well structured analytical approach could extend its effective usability, at the price of a complexity not always easy to manage. Besides, numerical methods are more direct to results but lack physical understanding. Then a trade off exists between them a digital twin could effectively realize offering both opportunities in a complementary way.

Reasons for this work

Recently the adoption of digital twins for robotics has known increasing interest in academic literature for a vast range of applications and uses.

Traditionally symbolic modeling of robot dynamics has far origins in early years of robotics. Nonetheless this is not an old and un-actual topic but still very present.

In recent years most of literature moved towards numerical methods and approaches both for modeling, simulation. One of the most promising and actually employed tools is Simscape, based on the MATLAB®/Simulink® environment allowing for a variety of applications.

This environment is one of most performing and almost universally recognized for technical applications both in academic and industrial fields.

In spite of the fact many predecessors existed, like Mathematica® and others, none of them reached the same success level.

Simscape allows to quickly build up and simulate a multibody or multiphysics model of a system and simulate it.

There is also a chance to partially or totally automatize the process exploiting scripts and command line instructions, easing considerably both use and applications variety.

One lack of the system is the lack of transparency into underlying model equations.

On the other side in robotics literature history the symbolic modeling and deployment of system equations has soon been relegated into a second stage because of fast rising complexity of model, often taking pages of equations still unmanageable even with modern computers and time taking even for just a few degrees of freedom in serial chains systems. Willing to apply this approach to large multibody systems would be prohibitive and unviable. In this work we'll show how some simple ploy would definitively dodge this issue leading to fast and direct models deployment in symbolic way avoiding most of the limits encountered with traditional systems and applicable to arbitrary [topologies](#) of multibodies.

This is of primary importance especially nowadays and in future perspective where all around spreading of complex robotic architectures like humanoid robots or legged robots and many others coming to the scene of robotics, like snakelike shapes, work shapes, soft robots and many others couldn't deal with limits of symbolic model management.

If one solution was found in the past using numerical models, advantages of analytical and formal models should not be underestimated, because their generalities could by far simplify engineers' work and understanding of system properties.

In addition, the proposed approach takes advantage of larger flexibility in case system topology changes, increasing generality and reusability.

In addition the knowledge of mathematical model of a system is a preliminary step using traditional methods

Another relevant issue regards the complexity of model equations, often unmanageable and difficult to read, but hiding an underlying structure that could be difficult to see without the support of a proper symbolic formalism adoption, is eased by the method proposed here.

Often models arising are by far too complex to be managed even automatically and even automatic simplification could fail in finding most proper representation for intermediate variables leading to lower readability and understanding.

One route for easing the work is the adoption of the inertia matrix, whose structure is rather regular and allows us to understand system patterns with limited complexity, in spite of the fact it contains all the information to define system dynamics from an inertial point of view. In addition model identification requires the insight on relevant parameters of system inertia could be obtained in different ways, one is the direct parametrization of mass matrix, or using approach developed from Khalil and Gautier exploiting sequential nature of most serial chain robotis. Inertia matrix based approach would be more general but less automatic.

Completing with other system dynamics matrices one gets the full picture in a synthetic way.

Symbolic models allow to support easy and straightforward approach to both direct and inverse robot dynamics in an almost automatic way.

If in the past there were a lot of concerns for symbolic manipulation and management of symbolic models, today this is a rather easy and straightforward procedure to be applied even in an ordinary laptop, although some issues and tricks are still needed to keep them usable for arbitrary systems.

Why matlab and Simulink-Simscape environment

Could naturally arise a question why such commercial tools like matlab and Simulink®/Simscape® have been used in a place of free tools or traditional programming languages. This is a careful and purposed choice, because, although many physical engines are already available and used all around the world, they don't leave you much control

On the other side, symbolic manipulation in Matlab® is a widely proved and used process, with many decades of heritage. From strictly programming point of view Matlab® language has proven to be by far superior performance choice in technical environment, not only because of its huge availability of toolboxes, free scripts, a large community of developers, and not last relevant commercial interest in having working products and a well working customer support all around the world.

Though Matlab® is not a standard and is continuously developing and improving, it is used all around the world, with effective results, by a huge community of users in academic, industrial, technical and amateur fields.

Reasons for this success should not be neglected, and founding on many critical success factors, eventually decree its success:

- Use of interpreted language
- Extensive use of algebraic and array based programming facilitating and providing compact and powerful code
- Text based execution (no compiling, no linking), easy and intuitive what you see is what you do
- Debugging process available
- Optimized functions and libraries
- Availability of compilation feature
- Others

Although in the past robotic tools have been developed in many languages like C++, Fortran (in the early era), nowadays often Python is used, but many professional users are converging to the adoption of Matlab® for parts or the whole process.

Indeed its syntax makes it a natural choice for writing code involving matrices and linear transformations, together with many other features and the presence of built in technical tools easy to use.

Most of previous projects for robotics modeling and symbolic development appear to have failed, disappeared or abandoned, like SYMORO, Autolev, and many others.

Today almost everybody would be able to develop its own model rather rapidly and use it or customize it for his own purposes, with few simple lines of codes.

One witnessing of this trend is the use of some specialists like Featherstone.

In the author's experience speed deficiency and memory limitations are not sensible neither a key issue, in comparison to the ease of code development , reuse, understandability and debugging, and moreover, when failing in Matlab® usually most of codes are not viable neither when written in compiled and executable languages. Some applications like PDE simulations and FEM analysis have realistically been left out of range, but for most others applications proven to be powerful enough to fulfill almost every required task.

On the other side Matlab has proven to be a not exclusive environment, where also developers using C++, Java and other languages are able to provide functionalities and packages in an effective and productive way. In recent years matlab had become more and more a integrated work environment more than a classic language, where data collection, processing, analysis coexist with algorithms and models development and even with real time interfaces able to read serial and usb inputs from real devices in closed HIL (Hardware In the Loop) mode.

The choice of this environment is then becoming more and more natural for engineers working with electronics, controls, mechanics,dynamics and simulations.

Indeed Simulink, that's the root environment for Simscape® users, is the preferred environment for most control and automation engineers. Since the mathematics involved in this work are closely connected with control design, this choice is once more perfectly natural.

Short work roadmap

As anticipated, in absence of time and resources, we focused on a restricted set of study cases in order to give an idea of the potential a complete development of the subject could lead to. Topics covered are results of a set of activities not preliminarily planned, then resulting overall appearance could result not completely organic, but the core purpose is to give a panoramic of potential applications beyond a specific applicative target. After introduction of fundamental concepts in robotics (namely kinematics and dynamics) with references for deepening specialistic and related topics in modern robotics. Specifically we dedicate a special attention to the inertia matrix concept as a cardinal concept in modern dynamic robot modeling, where symbolic manipulation automation allows significant support. In this work we put a special attention to robot identification problems, as one of most advanced and powerful the employment of analytical models and symbolic modeling and base parameters determination problems. If most other modeling tasks (kinematics and dynamics) are actually covered without using explicit analytical models, in modern literature identification problems using mathematical models is still reputed to be the standard and best method in research. Symbolic manipulation allows for better insight into model understanding and management still unparalleled by numerical models. We then highlight the power of a better integration between these two approaches, still lacking even in advanced tools and environments. We go then for presenting fundamentals concepts used in system modeling under the Simscape environment, presenting fundamental elements used in this work, like joints, forces blocks, body geometry, inertia blocks and transformation measurement blocks, inertia measurement blocks, physical signals and conversions blocks, frame transformation blocks and contact forces blocks. In this section we present also some user specific tools and practices suggested for a reusable and ready for use modeling following robotic notations. Finally we go for illustrating a bunch of case studies where notions presented in precedent chapters are applied to some simple cases, presenting also some collateral applications like on line filtering and estimation of parameters, and analysis of real data acquisitions with mixed model-numerical techniques, and finally we propose a workflow automation tool for analyzing a robotic system defined by standard URDF files. A final consideration and future developments section is given illustrating what evolution perspectives are, premising for this work final conclusions.

About modeling robots

Developing a digital twin in robotics usually means first defining and posing the basis for a system model, accounting for its fundamental characteristics, in terms of kinematics, dynamics, and in case other physical effects are accounted for like thermal effects and electric behavior, a suitable modelisations shall be given. In the introductory part of this work we illustrate fundamentals of robot modeling as treated in field literature, in order to contextualize required preliminary knowledge. Most literature exploit fundamental notions of analytical mechanics and mathematical language for system description and modeling in the purpose of making it predictable. We'll see some alternative perspectives arise later by the use of modern tools of development.

Dynamic Models of Rigid-Body Systems

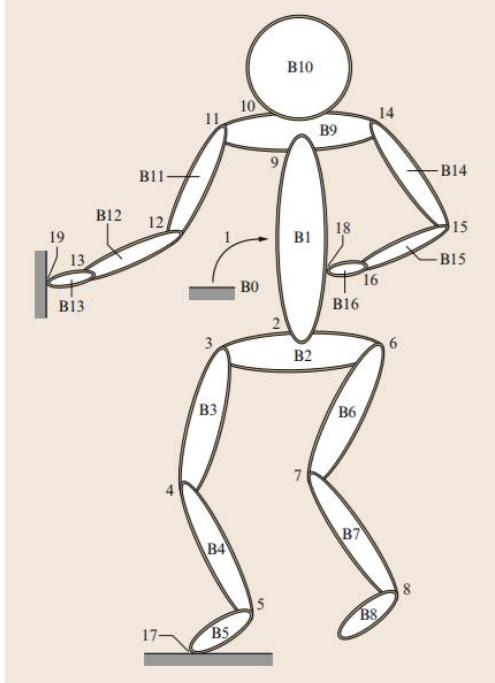
A basic rigid-body model of a robot mechanism has four components: a *connectivity graph*, link and *joint geometry parameters*, *link inertia parameters*, and a *set of joint models*. To this model, one can add various force-producing elements, such as *springs*, *dampers*, *joint friction*, *actuators*, and *drives*. The actuators and drives, in particular, may have quite elaborate dynamic models of their own. It is also possible to add extra motion freedoms to model *elasticity* in the joint bearings or links See also [Handbook of robotics] (specifically Chapter 11). This section describes a basic models. More on this topic can be found in field books such as [Rigid Body Dynamics Algorithms], [Modeling, Identification and Control of Robots], [Dynamics of Multibody Systems].

Connectivity

A connectivity graph is an undirected graph in which each node represents a rigid body and each arc represents a joint. The graph must be connected; and exactly one node represents a fixed base or reference frame. If the graph represents a mobile robot (i. e., a robot that is not connected to a fixed base), then it is necessary to introduce a fictitious 6-DOF joint between the fixed base and any one body in the mobile robot. The chosen body is then known as a floating base. If a single graph is to represent a collection of mobile robots, then each robot has its own floating base, and each floating base has its own 6-DOF joint. Note that a 6-DOF joint imposes no constraints on the two bodies it connects, so the introduction of a 6-DOF joint alters the connectivity of the graph without altering the physical properties of the system it represents. In graph-theory terminology, a *loop* is an arc that connects a node to itself, and a *cycle* is a closed path that does not traverse any arc more than once. In the connectivity graph of a robot mechanism, loops are not allowed, and cycles are called kinematic loops. A mechanism that contains kinematic loops is called a closed-loop mechanism; and a mechanism that does not is called an open-loop mechanism or a kinematic tree. Every closed-loop mechanism has a *spanning tree*, which defines an open-loop mechanism, and every joint that is not in the spanning tree is called a loop-closing joint. The joints in the tree are called tree joints. The fixed base serves as the root node of a kinematic tree, and the root node of any spanning tree on a closed-loop mechanism. A kinematic tree is said to be *branched* if at least one node has at least two children, and unbranched otherwise. An *unbranched* kinematic tree is also called a kinematic *chain*, and a branched tree can be called a *branched kinematic chain*. A typical industrial robot arm, without a gripper, is a

kinematic chain, while a typical humanoid robot is a kinematic tree with a floating base. In a system containing N_B moving bodies and N_J joints, where N_J includes the 6-DOF joints mentioned above, the bodies and joints are numbered as follows. First, the fixed base is numbered body 0. The other bodies are then numbered from 1 to N_B in any order such that each body has a higher number than its parent. If the system contains kinematic loops then one must first choose a spanning tree, and commit to that choice, since the identity of a body's parent is determined by the spanning tree. This style of numbering is called a *regular numbering scheme*. Having numbered the bodies, we number the tree joints from 1 to N_B such that joint i connects body i to its parent. The loop-closing joints, if any, are then numbered from $N_B + 1$ to N_J in any order. Each loop-closing joint k closes one independent kinematic loop, and we number the loops from 1 to N_L (where $N_L = N_J - N_B$ is the number of independent loops) such that loop l is the one closed by joint $k = N_B + l$. Kinematic loop l is the unique cycle in the graph that traverses joint k , but does not traverse any other loop-closing joint. For an unbranched kinematic tree, these rules produce a unique numbering in which the bodies are numbered consecutively from base to tip, and the joints are numbered such that joint i connects bodies i and $i - 1$. In all other cases, regular numberings are not unique. Although the connectivity graph is undirected, it is necessary to assign a direction to each joint for the purpose of defining joint velocity and force. This is necessary for both tree joints and loop-closing joints. Specifically, a joint is said to connect from one body to another. We may call them the predecessor $p.i$ and successor $s.i$ for joint i , respectively. Joint velocity is then defined as the velocity of the successor relative to the predecessor; and joint force is defined as a force acting on the successor. It is standard practice (but not a necessity) for all tree joints to connect from the parent to the child. The connectivity of a kinematic tree, or the spanning tree on a closed-loop mechanism, is described by an N_B -element array of parent body numbers, where the i -th element $p.i$ is the parent of body i . Note that the parent $p.i$ for body i is also the predecessor $p(i)$ for joint i , and thus the common notation. Many algorithms rely on the property $p(i) < i$ to perform their calculations in the correct order. The set of all body numbers for the children of body i , $c(i)$, is also useful in many recursive algorithms.

The connectivity data for kinematic loops may be described in a variety of ways. A representation that facilitates use in recursive algorithms includes the following conventions. Loop-closing joint k joins bodies $p(k)$ (the predecessor) and $s(k)$ (the successor). The set $LR(i)$ for body i gives the numbers of the loops for which body i is the root. Using the property $p(i) < i$ for bodies in the spanning tree, the root of a loop is chosen as the body with the lowest number. In addition, the set $LB(i)$ for body i gives the numbers of the loops to which body i belongs but is not the root. An example of a closed-loop system is given in Fig. 3.3. The system consists of a humanoid mechanism with topologically varying contacts, with the environment and within the mechanism, which form closed loops. The system has $N_B = 16$ moving bodies and $N_J = 19$ joints with $N_J - N_B = 3$ loops. The main body (1) is considered to be a *floating base* for this mobile robot system. It is connected to the fixed base (0) through a fictitious 6-DOF joint (1). To complete the example, the loop-closing joint and the body numbers $p(k)$ and $s(k)$ as well as the root body for each loop are given in Table 3.2. The body-based sets $c(i)$ and $LB(i)$ are given in Table 3.3. Note that $LR(0) = \{1,3\}$ and $LR(1) = \{2\}$ and all other LR sets are null for this example.



Humanoid robot example. Note: to distinguish between body numbers and joint numbers in this figure, body numbers are preceded by a B for clarity

After topological description and definition of a robot system, we recall kinematics notion used in model development.

Kinematics

Kinematics definition

Kinematics is a subfield of physics and mathematics, developed in [classical mechanics](#), that describes the [motion](#) of **points**, **bodies** (**objects**), and systems of bodies (**groups of objects**) without considering the **forces** that cause them to move. We are going to synthesize some basilar notions, but redirect to literature for deepening: [Kinematics](#).

Usually in mechanics kinematics of *geometrical* points is first introduced defining quantities like

- *Position* (relative to a point and a reference frame, is usually described by means of a vectorial function \Re^n with $n=3$ in common space)

- *Trajectories* (as sequence of positions over time, usually assumed to be continuous on time, such that definitions like derivatives exist up to a second or higher order)
- *Velocity* as vectorial rate of change of position
- Accelerations as vectorial rate of change of speed

Higher order derivatives are seldomly accounted for since in physical systems these terms are not involved in defining system dynamic evolution laws.

In robotics, we assign one or more frames to each link of the robot and each object of the workcell. Thus, transformation of frames is a fundamental concept in the modeling and management of a robot. It enables us to:

- compute the location, position and orientation of robot links relative to each other, describe the position and orientation of objects in workspace;
- specify the trajectory and velocity of the end-effector of a robot for a desired task;
- describe and control the forces when the robot interacts with its environment;
- implement sensory-based control using information provided by various sensors, each having its own reference frame.

In this chapter, we present a notation that allows us to describe the relationship between different frames and objects of a robotic cell. This notation, called homogeneous transformation, has been widely used in computer graphics to compute the projections and perspective transformations of an object on a screen. Currently, this is also being used extensively in robotics. We will show how the points, vectors and transformations between frames can be represented using this approach. Then, we will define the differential transformations between frames as well as the representation of velocities and forces using screws.

Points and vectors definitions

We want to introduce informally a set of basilar geometric objects we will use.

Considering an arbitrary dimension space we assume to be able to perform a point to point difference and name it vector. A vector can be applied to a point or less. We've following properties (see [A Mathematical Introduction to Robotic Manipulation]).

1. Sums and differences of vectors are vectors.
2. The sum of a vector and a point is a *point*.
3. The difference between two points is a vector.
4. The sum of two points is *meaningless*.

A vector has a length, and angle between vectors is defined intuitively (alternative definitions are possible). In addition we introduce some basic binary operations over vectors like

- Scalar product between vectors as $a \cdot b = \|a\| \|b\| \cos \angle ab$
- Vector product (indicated somewhere with \wedge or \times) $a \times b = (\perp ab) \|a\| \|b\| \sin \angle ab$

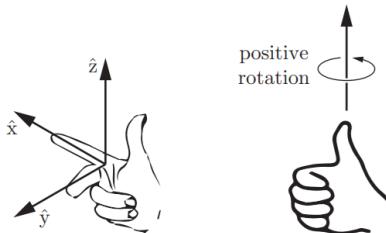
Frames definition

One convenient to describe position and orientation of a rigid body in space way is to attach a reference frame to the rigid body and to develop a way to quantitatively describe the frame's position and orientation as it moves. A frame is a set of non-collinear free vectors, named

base, not associated to any point, and an origin point O, allowing for describing all points of a rigid body and of any point in space, as vector, that by linear algebra we know we can write as linear combination of frame base.

$$u = a_1 e_1 + a_2 e_2 + a_3 e_3 = \sum_{k=1}^3 a_k e_k$$

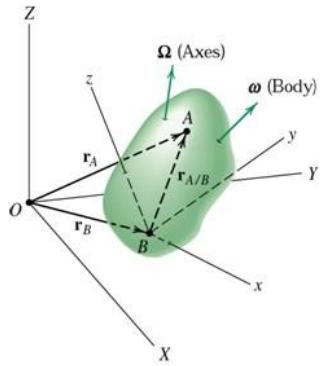
Most used frames are orthonormal in order to make calculations easier. That means, if e_k are frames base vectors $e_i \cdot e_j = \delta_{ij}$ with Kronecker delta. If base vector are unitary length we can indicate them using hat notation \hat{e}_k . Remark introduction of a frame can be used also to introduce metrics and reference lengths. Introduction of a base frame allows for description of points and free vectors inside same formalism. Required number of base vectors depend on dimension of embedding space. Euclidean space account for 3 (space) dimensions. In euclidean space we can introduce as many frames as we need, body attached (to any number of body) or moving. Rigid body motion respect any object or respect a background can be seen as a transformation occurring between two time events of observation. Similarly we can describe body position and orientation respect two frames as a transformation as well. Indeed following galilean relativity, and human intuition, description of a body position, orientation and motion is relative to observator frame. In mechanics not all frames are the same, since for dynamics we shall refer to [inertial frames](#) class of equivalence of frames moving steadily with respect any inertial frame. We can of course use arbitrary frame for motion description, but the law of dynamics make use of inertial quantities. A frame attached to a body and fixed respect is named body frame. Frames used are conventionally *right handed*.



Basic requirements for transformations between frames are

- Preserve distances between points
- Preserve vector length (norm or magnitude)
- Preserve angles between vectors (orientation)
- Are linear (distributive)

Homogeneous transformations are suitable type of transformations respecting these properties.



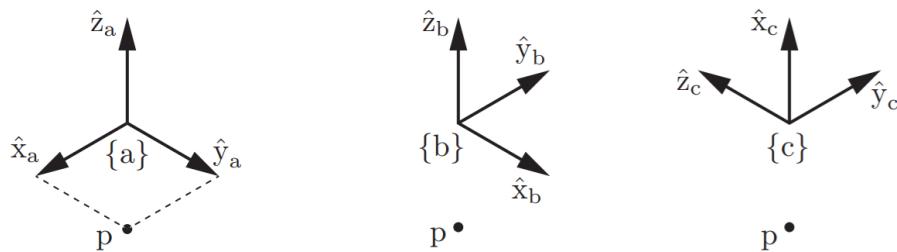
Use of rotation matrix in kinematics

A fundamental point to keep in mind is that rotation matrices can be used in different ways

- ❖ To represent a **frame orientation**
- ❖ To change reference frame where **vectors** and other tensorial objects (as multivectors) are **represented**
- ❖ To **rotate** vectors (hence frames), keeping fixed representation frame

Commented [1]: Dots?

The equivalence between three perspectives is due to the fact that a rotation of a vector in a fixed frame can be considered equivalent to the rotation of the frame in opposite direction due to the relative nature of rotation measure in kinematics. From a dynamical point of view this equivalence do not last anymore necessarily since rotation of inertially munited objects results in many case (exception done for some corner case) in forces and torques, due to the fact frames are not all equivalent but a subclass of **privileged frames** called **inertial frames** exist.

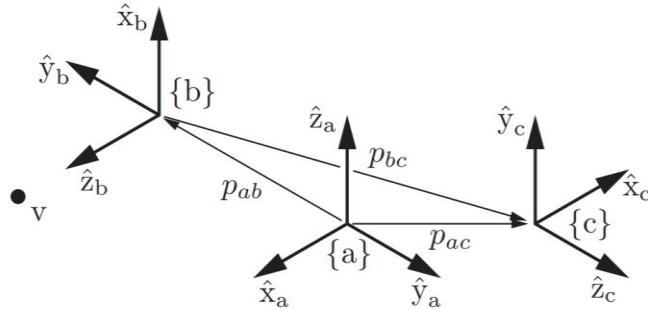


The same space and the same point p represented in three different frames with different orientations.

Use of transformation matrices

Similarly to what seen for rotation transformation matrices, transforming vector in space considering their application point. We can use for homogeneous transformation matrices one

- ❖ To represent the **configuration** (position and orientation) of a rigid body or a frame in space;
- ❖ To **change** the **reference** frame in which a vector or frame is represented;
- ❖ To **displace** a vector or frame



Three reference frames in space, and a point v that can be represented in $\{b\}$ as $v_b = (0, 0, 1.5)$.

Rigid body kinematics

Extension from points to systems of points is rather immediate and usually based on simplification assumptions: all distances between points belonging to the same body are at fixed distance from each other and consequently positions are fixed on any frame attached to. Rigid body kinematics as well as frame kinematics are governed by rules. For any point having a position ρ in body frame

$$\frac{d\rho\hat{\rho}}{dt} = \dot{\rho}\hat{\rho} + \rho \frac{d\hat{\rho}}{dt}$$

$$\frac{d}{dt}(a_1\hat{e}_1 + a_2\hat{e}_2 + a_3\hat{e}_3) = \dot{a}_1\hat{e}_1 + \dot{a}_2\hat{e}_2 + \dot{a}_3\hat{e}_3 + a_1 \frac{d\hat{e}_1}{dt} + a_2 \frac{d\hat{e}_2}{dt} + a_3 \frac{d\hat{e}_3}{dt}$$

Since derivative of a versor are normal to versor itself, we can write (using linearity)

$$\frac{d\hat{e}_k}{dt} = \omega \wedge \hat{e}_k$$

$$\frac{d\hat{\rho}}{dt} = \omega \wedge \hat{\rho}$$

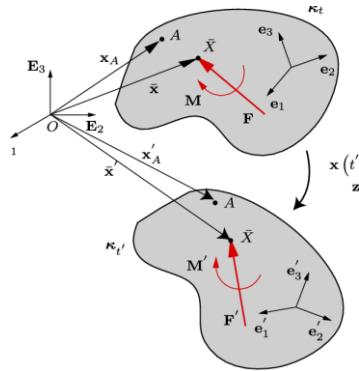
Constant distance condition lead to

$$dP = dP_0 + d\rho = \left(\dot{P}_0 + \omega \wedge \rho + \dot{\rho}^0 \right) dt = (v_0 + \omega \wedge \rho) dt$$

Similarly occurs for accelerations of frames and rigid bodies

$$a = \ddot{\rho} + \dot{\omega} \wedge \rho + 2\omega \wedge \dot{\rho} + \omega \wedge \omega \wedge \rho$$

Same equations are valid for any generic frame and for an arbitrary number of points.



So far we've considered the kinematics under only the hypothesis of constant distance. More complex constraints can arise in real applications like motion under guides.

Different types of constraints are possible. We can have *unilateral* or *bilateral* constraints. In robotics, many application constraints are *bilateral*. Bilateral constraints always work and act in all but a set of directions on a [tangent bundle](#) of constraint variety (assumed [differential manifold](#)). Bilateral constraints are mathematically expressed by means of equations

$$b(q, t) = 0 \quad \text{or equivalently} \quad b(q, t) = c$$

leading by consequence a condition over time written as (also in term of partial derivatives), that imply constraints on speeds

$$\dot{b}(q_k, t) = 0 \iff \frac{\partial b}{\partial q_k} \dot{q}_k = 0 \iff A(q_k, t) \dot{q}_k = 0$$

Joints are modeled as bilateral constraints. Unilateral constraints are defined by means of inequalities:

$$b(q, t) \geq 0$$

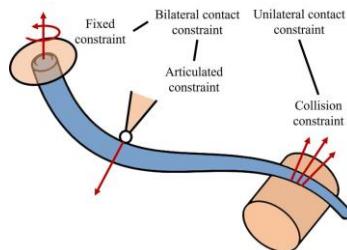
and are active or act only when the condition of equality is reached. When condition >0 is met, it can be considered as free motion.

Constraints types

Contact and collision constraints (or contacts, see also [contact mechanics](#)) like walls and limits, are usually [unilateral](#) and define conditions that arise and work only when condition is satisfied, where motion occurs as free otherwise. Mathematically unilateral constraint are expressed by means of inequalities

Unilateral constraints allow to define a system configuration (all points positions) specifying coordinates specified by coordinates. Differentiating over time we get relations between coordinates derivatives and point speeds and vector angular speeds, through linear relations with matrices called jacobians.

time dependence of constraints is of most importance in mechanics. Constraints not dependent on times are said [scleronomous](#). Systems whose constraints depend on time are defined [rheonomic](#). Observe as system parts and subsystems could have different nature, most general condition wins in defining overall characterization.



Soft constraint vs hard constraint

Mechanical system constraint could be *hard*, when related to strict conditions like equalities and inequalities already seen. From these conditions additional equations arises system shall comply with, coming from mathematical operations applied, time derivative one for all. Besides other constraints are possible, *soft* constraints, *relaxing* imposed conditions (see also [constrained optimization](#)), by means of relaxation of imposed conditions. Effect can be obtained using a penalty function. Bilateral and unilateral constraint can be though as limit case of infinite penalty function. On these types of constraints, condition is allowed to be violated up to some measure, for instance by means of a penalty rule. Penalty rule is one of ways usually used for mathematically modeling them. Soft constraints allow modeling deformable constraints. Soft constraints could regard both external constraint as well system internal flexibilities. Indeed they're of increasing importance and application in modern robotics, where managing deformable, flexible, soft objects, as well as contact dynamics, on surfaces, or even robot body deformability ([soft robotics](#) and [Handbook of robotics] Chapter 21) are no more negligible.

Holonomic constraints

This class of classic constraints are in most general case time dependant, but in any case involving only coordinates or if derivatives are included, they're *integrable* so that can be reconducted to classic coordinates, like in case of pure rolling without skid. Even free motion can be modeled as *holonomic* constraint. Holonomic constraints can be reconducted to the following form for every point k of the system, given n degree of freedom

$$r_k = r_k(q_1, \dots, q_n, t)$$

First consequence of holonomic constraints is the connection between generalized coordinates and generalized speeds as time derivative of, making them not independent.

In case also time dependance drops, we've *scleronomic* constraints. When this is not the case, the constraints are said to be *rheonomic*. Differentiating respect to time leads to the following constraint equation in the velocity domain:

$$\dot{r}_k = \frac{\partial r_k}{\partial q_h} \dot{q}_h + \frac{\partial r_k}{\partial t}$$

where first term can be put in relation with Pfaffian constraint concept. A condition for acceleration can be obtained also:

$$\ddot{r}_k = \frac{\partial^2 r_k}{\partial q_h \partial q_i} \dot{q}_h \dot{q}_i + \frac{\partial^2 r_k}{\partial q_h} \frac{\partial \dot{q}_h}{\partial q_i} \dot{q}_i + \frac{\partial^2 \dot{r}_k}{\partial q_h} \dot{q}_h + \frac{\partial r_k}{\partial q_h} \ddot{q}_h + \frac{\partial^2 r_k}{\partial t^2}$$

Non-holonomic constraints

In case constraints includes not integrable derivatives of coordinates, we get a constraint that cannot be reconducted to holonomic form by integration, then generally they take form

$$r_k = r_k(q_1, \dots, q_n, \dot{q}_1, \dots, \dot{q}_n, t)$$

with the exception of rolling contact, all of the constraints associated with the joints discussed in sections can be expressed mathematically by equations containing *only* the joint position variables. These are called holonomic constraints. The number of equations, and hence the number of constraints, is 6-n, where n is the number of degrees of freedom of the joint. The constraints are intrinsically part of the axial joint model. A nonholonomic constraint is one that cannot be expressed in terms of the position variables alone, but includes the time derivative of one or more of those variables. These constraint equations cannot be integrated to obtain relationships solely between the joint variables. The most common example in robotic systems arises from the use of a wheel or roller that rolls without slipping on another member. Nonholonomic constraints, particularly as they apply to wheeled robots, are discussed in more detail in Chap. 24 of [Handbook of robotics]

Commented [2]: repeated

Differential kinematics equations and inequalities

The imposition of derivatives on constraint equations allow to write differential equations that are valid for the system independently from other system characteristics like inertial properties and applied forces, typical of dynamic problems. The use of these differential equations is often very useful and allow to define subspace where motion solutions should occur a priori from any other consideration and ignoring even effective system dynamics. Differential kinematics is the study of system motion deriving from differentiation of constraints, lead us writing conditions for (absolute and relative)

- Position and displacement
- Speeds
- Accelerations
- Higher order derivatives (rarely used): jerk, snap or jounce, crackle, pop

Jerk is related to accelerations change and is related with bouncing and other:

Jerk (also known as **jolt**) is the rate of change of an object's acceleration over time. It is a vector quantity (having both magnitude and direction). Jerk is most commonly denoted by the symbol j and expressed in m/s^3 (SI units) or standard gravity per second (g/s).
<https://en.wikipedia.org/wiki/Jerk>

Differentiate	Displacement	m
Differentiate	Velocity	ms^{-1}
Differentiate	Acceleration	ms^{-2}
Differentiate	Jerk	ms^{-3}
Differentiate	Snap	ms^{-4}
Differentiate	Crackle	ms^{-5}
Differentiate	Pop	ms^{-6}

Usually these equations have at least a *quasi-linear structure* less complex than dynamic equations making it easier to manage **them with respect** constraints and dynamic equations themselves.

Pfaffian constraints

Constraints of type below are called Pfaffian constraints, where *not necessarily* A derives from a constraint partial derivative. In that last case, the constraint function can be seen as the integral of speed Pfaffian constraint A. Hence is an integrable constraint. Otherwise a generic non integrable constraint can be defined.

$$A(q_k)\dot{q}_k = 0$$

Configurations and definition

For a given value of vector of generalized coordinates q_k we say we've a configuration of the system. Space defined on value assumed by q is named *configuration space*.

The *configuration* of a robot is a complete specification of the position of every point of the robot.

The minimum number n of real-valued coordinates needed to represent the configuration is the *number of degrees of freedom* (dof) of the robot.

The n-dimensional space containing all possible configurations of the robot is called the *configuration space* (C-space). The configuration of a robot is represented by a point in its C-space.

When setting all generalized coordinates to zero we get a special configuration of the system sometimes defined as *reference configuration*. As from the name it's usually supposed to have a special meaning.

Manipulability ellipsoid

The manipulability ellipsoid, whose shape indicates the ease with which the robot can move in various directions, is also derived from the Jacobian.

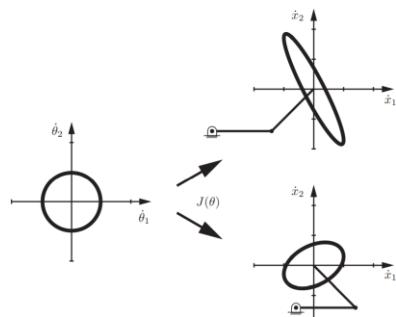


Figure 5.3: Manipulability ellipsoids for two different postures of the 2R planar open chain.

Generalized coordinates and topology induced

The choice of generalized coordinates adopted is not worthless or effectless, since some choices could lead to different *kind of singularities* in mapping forth and back from geometrical space from/to mechanism to coordinate space and its decomposability/composability. Where task coordinate space is defined by geometrical space properties, *configuration space* is much more complex and its topology and geometry is determined by adopted choice.

We say that two spaces are *topologically equivalent* if one can be continuously deformed into the other without cutting or gluing.

Some C-spaces (not all!) can be expressed as the *Cartesian product* of two or more spaces of lower dimension; that is, points in such a C-space can be represented as the *union* of the representations of points in the lower-dimensional spaces.

Examples of topologies are given in [Modern Robotics: Mechanics, Planning, and Control]

system	topology	sample representation
point on a plane	\mathbb{E}^2	(x, y)
spherical pendulum	S^2	latitude longitude $[-180^\circ, 180^\circ] \times [-90^\circ, 90^\circ]$
2R robot arm	$T^2 = S^1 \times S^1$	θ_2 θ_1 $[0, 2\pi] \times [0, 2\pi]$
rotating sliding knob	$\mathbb{E}^1 \times S^1$	θ $\mathbb{R} \times [0, 2\pi]$

Figure 2.2: Four topologically different two-dimensional C-spaces and example coordinate representations. In the latitude-longitude representation of the sphere, the latitudes -90° and 90° each correspond to a single point (the South Pole and the North Pole, respectively), and the longitude parameter wraps around at 180° and -180° ; the edges with the arrows are glued together. Similarly, the coordinate representations of the torus and cylinder wrap around at the edges marked with corresponding arrows.



Figure 2.9: An open interval of the real line, denoted (a, b) , can be deformed to an open semicircle. This open semicircle can then be deformed to the real line by the mapping illustrated: beginning from a point at the center of the semicircle, draw a ray that intersects the semicircle and then a line above the semicircle. These rays show that every point of the semicircle can be stretched to exactly one point on the line, and vice versa. Thus an open interval can be continuously deformed to a line, so an open interval and a line are topologically equivalent.

Representations

To perform computations, we must have a numerical *representation* of the space, consisting of a set of real (or complex) numbers. We are familiar with this idea from linear algebra – a vector is a natural way to represent a point in a Euclidean space. It is important to keep in mind that the representation of a space involves a choice, and therefore it is not as fundamental as the topology of the space, which is independent of the representation. A choice of n coordinates, or parameters, to represent an n -dimensional space is called an explicit parametrization of the space. Such an explicit parametrization is valid for a particular range of the parameters. Usually representations have singularities. Singularities are a potential result of mapping between spaces having different topology. In proximity of a singularity a small step results large changes in coordinates.

If you can assume that the configuration never approaches a singularity of the representation, you can ignore this issue. If you cannot make this assumption, there are two ways to overcome the problem. Use more than one coordinate chart on the space, where each coordinate chart is an explicit parametrization covering only a portion of the space such that, within each chart,

there is no singularity. If we define a set of singularity-free coordinate charts that overlap each other and cover the entire space, the charts are said to form an *atlas* of the space. An advantage of using an atlas of coordinate charts is that the representation always uses the minimum number of numbers. Use an implicit representation instead of an explicit parametrization. An implicit representation views the n-dimensional space as embedded in a Euclidean space of more than n dimensions. A disadvantage of this approach is that the representation has more numbers than the number of degrees of freedom. Another advantage is that while it may be very difficult to construct an explicit parametrization, or atlas, for a closed-chain mechanism, it is easy to find an implicit representation, using loop closure equations.

Generalized Coordinates

In a robotic mechanism consisting of N bodies, $6N$ coordinates are required to specify the position and orientation of all the bodies relative to a coordinate frame. Since some of those bodies are jointed together, a number of constraint equations will establish relationships among some of these coordinates. In this case, the $6N$ coordinates can be expressed as functions of a smaller set of coordinates q that are all independent. The coordinates in this set are known as generalized coordinates, and motions associated with these coordinates are consistent with all of the constraints. The joint variables q of a robotic mechanism form a set of generalized coordinates

Natural coordinates

In spite of the fact general degree of freedom balance together with constraints allow a considerably wide space of possible choices for generalized coordinates, some of them results to be more natural than others because of their intuitive and physically immediate meaning, in spite of potential usefulness, for example in simplifying governing dynamics equations. Some example of these more natural choices are

- Joint position, directly related to actuators state, leading to joint space
- Position and orientation of specific points/frames. When it is end effector/s leads to task space
- Joint position respect global frame, useful for expressing absolute joint position

Joint space

The space in which the location of all the links of a robot are represented is called joint space, or configuration space. We use the joint variables, $q \in \mathbb{R}^N$, as the coordinates of this space. Its dimension N is equal to the number of independent joints and corresponds to the number of degrees of freedom of the mechanical structure. In an open chain robot (simple or tree structured), the joint variables are generally independent, whereas a closed chain structure implies constraint relations between the joint variables. Unless otherwise stated, we will consider that a robot with N degrees of freedom has N actuated joints.

Task space

The location, position and orientation, of the end-effector is represented in the *task space*, or *operational space*. We may consider as many task spaces as there are end-effectors. In general more than one end effector frames and workspace are possible. Cartesian coordinates are used to specify the position in \mathbb{R}^3 and the rotation group $SO(3)$ for the

orientation. Thus the task space is equal to cartesian product space $\mathbb{R}^3 \times SO(3)$. An element of the task space is represented by the vector $X \in \mathbb{R}^M$, where M is equal to the maximum number of independent parameters that are necessary to specify the location of the end-effector in space. Consequently, $M < 6$ and $M < N$.

At a given joint configuration q , the rank r of the Jacobian matrix J_n - hereafter written as J to simplify the notation - corresponds to the number of degrees of freedom of the end-effector. It defines the dimension of the accessible task space at this configuration. The number of degrees of freedom of the task space of a robot, M , is equal to the maximum rank, r_{max} which the Jacobian matrix can have at all possible configurations. Noting the number of degrees of freedom of the robot as N (equal to n for serial robots), the following cases are considered [Gorla 84]:

- if $N = M$, the robot is *non-redundant* and has just the number of joints required to provide M degrees of freedom to the end-effector;
- if $N > M$, the robot is *redundant* of order $(N - M)$. It has more joints than required to provide M degrees of freedom to the end-effector;
- if $r < M$, the Jacobian matrix is *rank deficient*. The robot is at a singular configuration of order $(M-r)$. At this configuration, the robot cannot generate an end-effector velocity along some directions of the task space, which are known as *degenerate directions*. When the matrix J is square, the singularities are obtained by the zeros of $\det(J) = 0$, where $\det(J)$ indicates the determinant of the Jacobian matrix of the robot. They correspond to the zeros of $\det(J^T J) = 0$ for redundant robots (based on pseudoinverse concept).

The relationship between the variables representing the configuration of an articulated manipulator and those describing an assigned task in the appropriate space can be established at the position, velocity or acceleration level. In particular, consideration of the first-order task kinematics inevitably results in discussion of the task Jacobian matrix, which is central to many redundancy resolution techniques.

Extensive treatment of task workspace can be found in [Handbook of robotics]

Redundancy

A robot is classified as redundant when the number of degrees of freedom of its task space is less than the number of degrees of freedom of its joint space. This property increases the volume of the reachable workspace of the robot and enhances its performance. We will see that redundant robots can achieve a secondary objective besides the primary objective of locating and moving the end effector with desired velocity. Notice that a simple open chain is redundant if it contains any of the following combinations of joints:

- more than six joints;
- more than three revolute joints whose axes intersect at a point;
- more than three revolute joints with parallel axes;
- more than three prismatic joints, - prismatic joints with parallel axes;
- revolute joints with collinear axes.

For an articulated mechanism with several end-effectors, redundancy is evaluated by comparing the number of degrees of freedom of the joint space acting on each end-effector and the number of degrees of freedom of the corresponding task space; another type of redundancy may occur when the number of degrees of freedom of the task is less than the number of degrees of freedom of the robot.

Singular configurations

For all robots, redundant or not, it is possible that at some configurations, called singular configurations, the number of degrees of freedom of the end-effector becomes less than the dimension of the task space. For example, this may occur when:

- the axes of two prismatic joints become parallel;
- the axes of two revolute joints become collinear;
- the origin of the end-effector lies on a line that intersects all the joint axes.

Mathematical condition to determine the number of degrees of freedom of the task space of a mechanism as well as its singular configurations

Typically an articulated manipulator robot can present following singular positions:

- **shoulder singularity:** takes place when there is alignment between arm principal link and shoulder vertical axis. This means that one can always find a solution, but when leaving this configuration, a small change in the desired location may require a significant variation in shoulder position is implied, impossible to realize due to the speed and acceleration limits of the actuator;
- **wrist singularity:** takes place when axes of wrist joints are collinear.
- **elbow singularity:** occurs when elbow reach maximum extension, aligning the main arm link.

The above-mentioned singularities are classified as *first order singularities*. Singularities of higher order may occur when several singularities of first order take place simultaneously.

Choosing the number of degrees of freedom of a robot

A non-redundant robot must have six degrees of freedom in order to place an arbitrary object in space. However, if the manipulated object exhibits revolution symmetry, five degrees of freedom are sufficient, since it is not necessary to specify the rotation about the revolution axis. In the same way, to locate a body in a plane, one needs only three degrees of freedom: two for positioning a point in the plane and the third to determine the orientation of the body. From these observations, we deduce that:

- the number of degrees of freedom of a mechanism is chosen as a function of the shape of the object to be manipulated by the robot and of the class of tasks to be realized;
- a necessary but insufficient condition to have compatibility between the robot and the task is that the number of degrees of freedom of the end effector of the robot is equal to or more than that of the task.

Jacobians and their use

Analytical derivation of basic jacobian

By definition we can express under a generic change of variable, as a transformation between two spaces, that under assumptions of analyticity, or at least of differentiability and continuity, as is classically done in mechanics and in robotics

$$X(q) = X$$

It's then easy to connect time changes of vector quantities like position and orientation in terms of derivative connected in two spaces between a jacobian matrix

$$\dot{X} = J(q)\dot{q}$$

When jacobian is related to geometrical and physical quantities is also called to be a **kinematic jacobian**.

Calculation of jacobians allow essentially to transit from one space (natural space of geometry and physical quantities) to a coordinate space where is more natural and easy to act for different reasons.

Using jacobians we can calculate for any robot configuration allowed from kinematic and mechanical constraints, speed, generalized forces, accelerations in an easy way.

$$f = J\tau$$

$$\tau = J^T f$$

Jacobian can be calculated respect any arbitrary reference frame, so one to pass from one frame to another one use a 6x6 rotation matrix

$${}^j J_P = \begin{pmatrix} {}^j R_i & 0_{3x3} \\ 0_{3x3} & {}^j R_i \end{pmatrix} {}^i J_P$$

Remarkably formalism naturally enclose in 6-vector notation of twists.

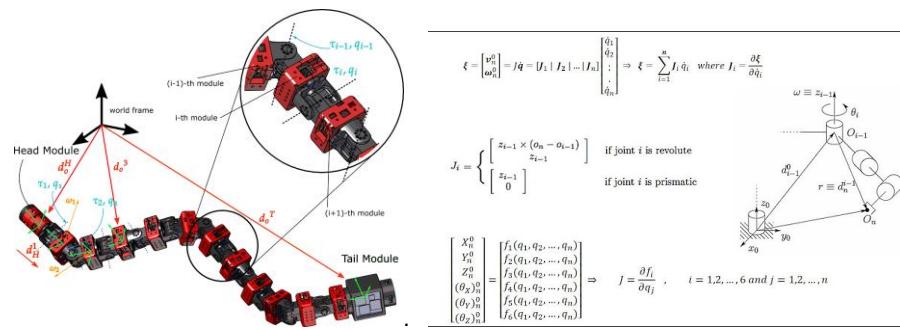
Geometrical derivation of basic jacobian

Jacobian is the matrix connecting a generic point of a robot speed with coordinates speed. It can be carried out geometrically considering for a given configuration instantaneous speeds induced from every joint. This can be carried out with simple formulations coming from rigid motion theory. Is possible to prove indeed that any point speed can be written as the sum of speed vector coming from any joint motion, like in virtual displacement case, but considering speeds. This method is more intuitive and do not requires analytical or numerical differentiations, that especially for many degree of freedom, may result to be long, tedious and

time taking. This technique results then to be simpler, faster, and more intuitive. Anyway general case requires to know point position vectors in generic configuration

Mathematically geometric jacobian take form of a $6 \times N$ matrix, whose first three row are cross product of joint axis versor versus distance vector of point (we're considering jacobians) from it, for revolute joints, or just joint versor for prismatic joints. Second three rows are not null only for revolute joints, and is given from joint axis versor. Using commonly adopted nomenclature in literature where prismatic joints have $\sigma_i = 1$ and $\bar{\sigma}_i = 1 - \sigma_i$ and vice versa.

$$J_P = \begin{pmatrix} \sigma_1 z_1 + \bar{\sigma}_1 (z_1 \wedge r_{1P}) & \dots & \sigma_n z_n + \bar{\sigma}_n (z_n \wedge r_{nP}) \\ \bar{\sigma}_1 z_1 & \dots & \bar{\sigma}_n z_n \end{pmatrix}$$



Jacobian inverse

In order to solve the linear system of equations in the joint rates obtained by decomposing of Jacobian mapping into its component equations when v_N is known, it is necessary to invert the Jacobian matrix. The equation becomes

$$\dot{q} = J^{-1}(q)v_N$$

Since J is a 6×6 matrix, *numerical inversion* is not very attractive. It is quite possible for J to become *singular* ($|J| = 0$), in which case the inverse does not exist. More information on singularities can be found in textbooks and other sections. Even when the Jacobian matrix does not become singular, it may become *ill-conditioned*, leading to degraded performance in significant portions of the manipulator's workspace. Most industrial robot geometries are simple enough that the Jacobian matrix can be *inverted analytically*, leading to a set of explicit equations for the joint rates. This greatly reduces the number of operations needed as compared to numerical inversion. For more complex manipulator geometries, though,

numerical inversion is the *only solution option*. The Jacobian of a redundant manipulator is not square, so it *cannot* be inverted. However discussions on how various pseudoinverses can be used in such cases are presented in sections.

Jacobians singularities

A robot configuration q is singular if the task Jacobian matrix j_t is rank-deficient there. Considering the role of J_t , it is easy to realize that at a singular configuration it is impossible to generate end effector task velocities or accelerations in certain directions. Further insight can be gained by looking at equations [], which indicates that a singularity may be due to a loss of rank of the transformation matrix T and/or of the geometric Jacobian matrix J .

Expression of R depends on the adopted minimal representation of the orientation, a configuration at which T is singular is then referred to as a representation singularity.

A configuration at which J is singular is referred to as a kinematic singularity.

Rigid body motion

Rigid motion in euclidean space

Most of transformations in Euclidean space can be represented as [group](#) of transformations, since are close, associative (they can be composed), have an identity, and an inverse transformation. Since these groups are parametrized by means of continuous parameters they're Lie groups, and Lie algebra and exponential formula are applicable. Usually they're not commutative (non-abelian groups), excepted for pure rotations.

Rotations group $SO(3)$

We're interested mainly on \mathbb{R}^3 space rotations for applicative reasons. A rotation is a motion preserving at least a point, and allow to change frame of reference having a common origin (or less, just ignoring it) point. Rotations conserve distance (*isometric* transformation) and angles between vectors (*isogonic*). Rotations can be represented using matrixes called *rotation matrix* whose components are scalar products of frames versors on initial and final frame. Identity matrix correspond to a null rotation and is idempotent operator on \mathbb{R}^3 euclidean space vectors. Matrix elements can be parametrized as seen in other sections. Rotation is then a linear operator transformation.

As consequence of our considerations we can apply repeatedly the this passage composing a sequence of of rotation transformations leading to a composite rotation. Usually we don't sum directly rotation matrices since this is not a meaningful operation from geometrical standpoint.

These matrixes are by definition *orthonormal* since its transpose correspond to inverse rotation matrix, and is by force unique, consequently we can see that

$$R^{-1}R = R^T R = I \Rightarrow \det(R) = \pm 1$$

The sign distinguish between proper (+1) and improper (-1) rotations. Conventionally proper rotations keep determinant 1 and belongs to connected subgroup of geometrically meaningful rotations, coming from right-handed convention.

Resulting matrices set belongs to a *non abelian* group on matrix multiplication operation (since product of elements are still rotation matrix) with identity *neutral* element. Its element are expression of SO_3 group of special (determinant =1) orthogonal square real matrix of dimension 3. Planar rotations can be represented by means of SO_2 that's a subgroup of SO_3 .

Any orientation configuration of a rigid body as well of a frame free to rotate respect another frame fixed, can be represented by means of a unique rotation matrix. Under this condition we can map SO_3 on configuration space of system and any trajectory is a trajectory in SO_3 defined as curve or rotation matrix depending on time $R(t) \in SO_3$. Rigorously configuration space shall include all and only those valid configurations having a correspeticve unique representation.

Alternative use of rotation matrix is for transformation (rotation) of a vector components from one frame to another one. If vector have some specific direction could result unchanged by rotation, resulting in a fixed direction, and is a rotation eigenvector. Implications are seen later in Euler theorem.

A common formalism used is to provide as right pedix the index of provenance frame and as left apex destination frame. This notation is very useful to make expressions readable, but could generate confusion when other notations are used, like some conventions used in robotics.

Rotation is distributive when applied to external products, resulting in commuting operators

$$R(u \wedge v) = Ru \wedge Rv$$

Roto Translation group SE(3): Euclidean group

In this section we want to describe arbitrary motions of a rigid body in a reference space.

Extension to higher dimensions: SE(n)

Although not strictly physically meaningful since our space is known to be described with three independent spatial coordinates, is remarkable that reasonament could be extended quite easily and straightforward to an arbitrary number of dimensions, remembering the definition of rotation matrix and taking matrix formalism for rototranslations. indeed equation

$$uP = Rv + P$$

is still equivalent to, for an arbitrary number of dimensions (or vector components)

$$u = Tv = \begin{pmatrix} R & P \\ 0 & 1 \end{pmatrix}$$

Where R will be a nxn matrix defined as the scalar product of basis vectors (a and b respectively)

$$R = \begin{pmatrix} a_1 \cdot b_1 & \dots & a_1 \cdot b_n \\ \vdots & \ddots & \vdots \\ a_n \cdot b_1 & \dots & a_n \cdot b_n \end{pmatrix}$$

In an n-dimensional euclidean space we need n base vectors non collinear, usually chosen orthonormal, to define any vector in this space, for any chosen basis. Indeed still last

$$v = v_1 a_1 + \dots v_n a_n = w_1 b_1 + \dots w_n b_n$$

leading by projection on one of the basis

$$w_1 = (v_1 a_1 + \dots + v_n a_n) \cdot b_1 \\ \vdots$$

$$w_n = (v_1 a_1 + \dots + v_n a_n) \cdot b_n$$

or exploiting linearity

$$\begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} a_1 \cdot b_1 & & a_1 \cdot n \\ & \ddots & \vdots \\ a_n \cdot b_1 & \dots & a_1 \cdot b_1 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

And vice-versa.

Although in a n dimensional space their components, in case of rotations around one coordinate axis is less intuitive. On the other side all properties valid for 3-D space extend to N -D spaces (including orthonormality, inversion as transpose, preservation of norm and angles between vectors). Consequently we still have a group of roto translation matrixes which structure is similar to euclidean group in 3-D.

Affine transformations

Using an [augmented matrix](#) and an augmented vector, it is possible to represent both the translation and the linear map using a single [matrix multiplication](#). The technique requires that all vectors be augmented with a "1" at the end, and all matrices be augmented with an extra row of zeros at the bottom, an extra column—the translation vector—to the right, and a "1" in the lower right corner.

$$\begin{bmatrix} y \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

equivalent to

$$y = Ax + b$$

The price to be paid for the convenience of having a homogeneous or linear representation of the rigid body motion is the increase in the dimension of the quantities involved from 3 to 4. The last row of the matrix of equation appears to be "extra baggage" as well. However, in the graphics literature, the number 1 is frequently replaced by a scalar constant which is either greater than 1 to represent **dilation** or less than 1 to represent **contraction**. Also, the row vector of zeros in the last row may be replaced by some other row vector to provide "perspective transformations." In both these instances, of course, the transformation represented by the augmented matrix no longer corresponds to a rigid displacement.

Tensorial notation

Any roto translation in space can be represented using tensorial notation, very useful in dealing with physical and mechanical quantities assuming form of vectors and tensors

For instance rigid bodies inertia are tensors, where linear speeds are vectors as position vectors as well. Angular speeds are more complex objects denoted as biaxial vectors.

The use of tensorial notation is not useful only in kinematics but also in dynamics, since all quantities involved (forces, speeds, accelerations, torques and angular accelerations, inertial tensors, rotation matrixes) have tensorial nature. One special attention is deserved to angular speed and momenta that rigorously speaking have a bivector nature, although they can be represented as vectors, they're not. This will be more clear when speaking of spatial vector notation which formalism is more close to.

Tensorial notation is sometimes named multivector since can be effectively manage as a multiindex vector behaving as a multiple vector. That's due to the linearity nature of their transformation rules under a frame transformation. Indeed a tensor can be composed such that

- sum of tensor are tensors (provided their order matches)
- products of tensors are tensors of higher order

Tensor constitutes a vectorial space under real field. Particular tensors of order 0 are scalar, implying that.

Using Einstein notation indexes could be summed up using contraction leading to order reduction. A simple example is scalar product leading to vector norm.

$$s = a_i b_i$$

Any tensor under a generic homogeneous transformation behaves like

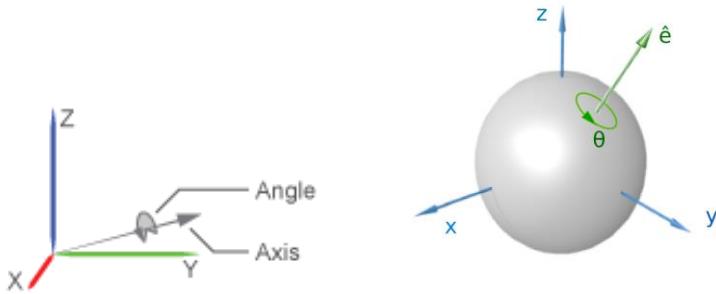
$$u_i = r_{ij} u_j$$

then is subject to linear relations, whose conditions of preserving angles and moduli of vectors lead to need of orthonormality for transformation r .

One relevant aspect of tensorial formalism is that 0-tensor also called scalars transform as identity under transformations, leading to constant scalars also called invariants. Invariant quantities are of most interest for physical purpose. An invariant can be built up by means of tensor contraction (like work or energy invariant or vector magnitude).

Euler theorem

Euler theorem: Any orientation $R \in SO(3)$ is equivalent to a *rotation* about a *fixed axis* $e \in \mathbb{R}^3$ through an angle $\theta \in [0, 2\pi]$.



This method of representing a rotation is also known as the *equivalent axis representation* or rotation axis representation. Resulting intuitive interpretation has profound consequences onto basic rigid transformations kinematics and consequently on rigid bodies kinematics.

The proof is rather easy using eigenvalues analysis of rotation matrix.

From mechanics and geometry we know for any orthonormal matrix R we have a set of eigenvalues and eigenvectors such that

$$Rv = \lambda v$$

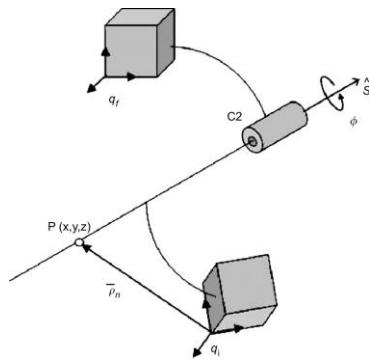
where eigenvalues are necessarily ± 1 . Thus

$$Rv = \pm v$$

We can then say for any rotation a fixed direction v exist such that it is not changed under rotation transformations. This holds for an arbitrary number of dimensions, but number and type of eigenvalues and eigenvectors could change. In 3D case we've one eigenvalue with multiplicity 2 and one with multiplicity one. Extending reasoning with a translation along this vector direction we get a rigid motion. It can be proven that any rigid motion could be written as a roto translation of this type. This give birth to screw theory and plucker vectors.

Chasles - Mozzi theorem

[Chasles-Mozzi theorem](#): Every rigid body motion can be realized by a *rotation* about a fixed axis combined with a translation parallel to that axis. This motion is also known as screw and the relative axis is named [screw axis](#) or helical axis, or twist axis. See [Screw theory](#) for deepenings. Euler's theorem can be considered a special case of.



The geometric meaning of a screw is expressed succinctly in this theorem. Its proof follows directly from the definition of the attributes of a twist.

We can then state that for every couple of rigid bodies or frames, four parameters are enough to define a transformation, giving an explanation for denavit hartenberg required number of parameters, or as well twists, screw or plucker vectors (where this idea is more subtle because of 6 vector formalism).

Indeed one could imagine that a generic homogeneous transformation would be parametrized with six parameters , three translation components, and three for describing a rotation, or any other combinations. This argument do not last because of the connection between frames explained in chasles theorem. Henceforth we could say that not all homogenous transformations elements are freely chosen, but at least two remain constrained.

Referring to Fig. 2, consider a rigid body motion which consists of rotation about an axis ω in space through an angle of θ radians, followed by translation along the same axis by an amount d . Such a motion is called a screw motion, since it is reminiscent of the motion of a screw, in so far as a screw rotates and translates about the same axis. By this analogy, the pitch of the screw is defined as the ratio of translation to rotation, $h = d/\theta$, assuming that $\theta \neq 0$. Thus, the net translational motion after rotating by θ radians is $h\theta$.

As aforementioned, the axis of rotation is $\omega \in \mathbb{R}^3$, $\|\omega\| = 1$, and $r_o \in \mathbb{R}^3$ is the vector for a point R_o on the axis. Assuming that the rigid body rotates about ω with angular velocity $h\omega$ together with a translational velocity $d\omega = h\theta\omega$, then the velocity of the tip point, $p(t)$, is

$$\dot{p}(t) = \omega \times (p(t) - r_o) + h\omega \quad (28)$$

This equation can be conveniently converted into homogeneous coordinates by defining a 4×4 matrix $[S]$ as

$$[S] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \quad (29)$$

with $v = -\omega \times r_o + h\omega$. Using Eq. (29), Eq. (28) can be rewritten in the homogeneous form as

$$\begin{bmatrix} \dot{p} \\ 0 \end{bmatrix} = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = [S] \begin{bmatrix} p \\ 1 \end{bmatrix} \quad (30)$$

That is

$$\dot{p} = [S] p \quad (31)$$

The solution of the differential equation in Eq. (31) is given by

$$\dot{p}(t) = e^{[S]t} \dot{p}(0) \quad (32)$$

where $e^{[S]t}$ is the matrix exponential of the 4×4 matrix $[S]t$, defined by

$$e^{[S]t} = \mathbf{I} + [S]t + \frac{([\mathcal{S}]t)^2}{2!} + \frac{([\mathcal{S}]t)^3}{3!} + \dots \quad (33)$$

If the body rotates about the axis ω at unit velocity for θ units of time, then the net transformation is given by

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \mathbf{I} + [\mathcal{S}]\theta + [\mathcal{S}]^2 \frac{\theta^2}{2!} + [\mathcal{S}]^3 \frac{\theta^3}{3!} + \dots \quad (34)$$

where, $[\mathcal{S}]^2 = \begin{bmatrix} [\omega]^2 & v \\ 0 & 0 \end{bmatrix}$, $[\mathcal{S}]^3 = \begin{bmatrix} [\omega]^3 [\omega]^2 & v \\ 0 & 0 \end{bmatrix}$, $[\mathcal{S}]^4 = \begin{bmatrix} [\omega]^4 [\omega]^3 & v \\ 0 & 0 \end{bmatrix}$, \dots . Such that Eq. (34) becomes

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} \mathbf{Q}(\theta) & v \\ 0 & 1 \end{bmatrix} \quad (35)$$

with $\mathbf{Q}(\theta) = \mathbf{I}\theta + [\omega] \frac{\theta^2}{2!} + [\omega]^2 \frac{\theta^3}{3!} + \dots$; and using the identity $[\omega]^3 = -[\omega]$, $\mathbf{Q}(\theta)$ can be simplified to

$$\begin{aligned} \mathbf{Q}(\theta) &= \mathbf{I}\theta + [\omega] \frac{\theta^2}{2!} + [\omega]^2 \frac{\theta^3}{3!} + \dots \\ &= \mathbf{I}\theta + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots \right) [\omega] + \left(\frac{\theta^3}{3!} - \frac{\theta^5}{5!} + \frac{\theta^7}{7!} - \dots \right) [\omega]^2 \\ &= \mathbf{I}\theta + (1 - \cos \theta) [\omega] + (\theta - \sin \theta) [\omega]^2 \end{aligned} \quad (36)$$

Hence, considering that $v = -\omega \times r_o + h\omega$, $\mathbf{Q}(\theta)v$ can be derived as follows by using the identities that $[\omega]^2 \omega = 0$ and $[\omega]^3 = -[\omega]$ as follows,

$$\begin{aligned} \mathbf{Q}(\theta)v &= (\mathbf{I}\theta + (1 - \cos \theta) [\omega] + (\theta - \sin \theta) [\omega]^2)(-\omega \times r_o + h\omega) \\ &= \mathbf{I}\theta(-\omega \times r_o + h\omega) + (1 - \cos \theta) [\omega](-\omega \times r_o + h\omega) + (\theta - \sin \theta) [\omega]^2(-\omega \times r_o + h\omega) \\ &= -[\omega]r_o \theta + h\theta\omega - (1 - \cos \theta) [\omega]^2 r_o + \theta + (\theta - \sin \theta) [\omega] r_o + \theta \\ &= h\theta\omega - (1 - \cos \theta) [\omega]^2 r_o - \sin \theta [\omega] r_o \\ &= (\mathbf{I} - e^{[\omega]\theta}) r_o + h\theta\omega = (\mathbf{I} - \mathbf{R}) r_o + dw \end{aligned} \quad (37)$$

which is the same as $\mathbf{A}\mathbf{q}$ that is obtained in Eqs. (10) and (11).

Substituting Eq. (37) into Eq. (35), it yields

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & \mathbf{Q}(\theta)v \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & (\mathbf{I} - e^{[\omega]\theta}) r_o + h\theta\omega \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & (\mathbf{I} - \mathbf{R}) r_o + dw \\ 0 & 1 \end{bmatrix} \quad (38)$$

Further, considering that $\omega^T v = \omega^T (-\omega \times r_o) + h\omega^T \omega$ such that $h = \omega^T v$, and $r_o = \omega \times v$, Eq. (38) can also be written as

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & (\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & [(\mathbf{I} - e^{[\omega]\theta}) [\omega] + \omega\omega^T \theta] v \\ 0 & 1 \end{bmatrix} \quad (39)$$

which is the same as the one obtained in Ref. [2].

In the case of pure translation which complies with $\|\omega\| = 0$, and $\|v\| = 1$, it has

$$[S] = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \quad (40)$$

which leads to $\mathbf{R} = e^{[\omega]\theta} = \mathbf{I}$, and $\mathbf{Q}(\theta) = \mathbf{I}\theta$ where $\theta = d$ is the amount of translation, such that

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \begin{bmatrix} \mathbf{I} & dv \\ 0 & 1 \end{bmatrix} \quad (41)$$

In the case that $d = 0$ and $\|\omega\| = 1$, which corresponds to the pure rotational motion, Eqs. (38) and (39) become

$$\mathbf{T} = e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\omega]\theta} & (\mathbf{I} - e^{[\omega]\theta}) r_o \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & (\mathbf{I} - e^{[\omega]\theta}) (\omega \times v) \\ 0 & 1 \end{bmatrix} \quad (42)$$

where r_o is the vector for a point on the screw axis as shown in Fig. 2.

Commented [3]: eliminate

Poinsot theorem

Every collection of wrenches applied to a rigid body is equivalent to a force applied along a fixed axis plus a torque about the same axis. The proof is constructive. Let $F = (f, \tau)$ be the net wrench applied to the object, is easy to derive by definitions application corresponding quantities at least of a free parameter. See [A Mathematical Introduction to Robotic Manipulation] for details.

The dual of Chasles' theorem, which showed that every twist could be generated by a screw, is called Poinsot's theorem. This is second major result upon which screw theory is founded concerns the representation of forces acting on rigid bodies. Poinsot is credited with the discovery that any system of forces acting on a rigid body can be replaced by a single force applied along a line, combined with a torque about that same line. Such a force is referred to as a wrench. Wrenches are dual to twists, so that many of the theorems which apply to twists can be extended to wrenches.

Screw concept

Screw motion concept derives from theorem seen above. Every rigid motion can be seen as a rotation around a fixed axis direction and a translational motion along it. Consequently a screw is defined by means of a versor, a rotation magnitude and a translation for a total of four parameters.

Twist concept

Similar concept to screw motion for speeds, lead to twist concept, consisting of an angular speed around a direction and a translational speed along this direction. Formally one can obtain a twist from a screw with unitary rotation speed $\omega = 1$, multiplying by angular speed. Any rigid-body configuration can be achieved by starting from the fixed (home) reference frame and integrating a constant twist for a specified time.

$$\mathcal{V} = S\dot{\theta} \quad S = (\omega \quad v_x \quad v_y)$$

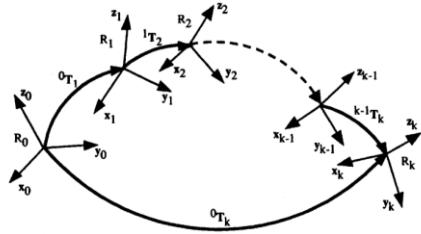
can also be considered a set of exponential coordinates.

Reciprocal screw

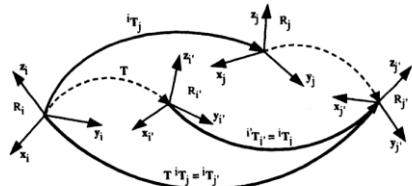
The dot product between *twists* and *wrenches* gives the instantaneous power associated with moving a rigid body through an applied force. We carry out all calculations relative to a single coordinate frame and omit the use of subscripts. A *wrench* F is said to be *reciprocal* to a twist V if the instantaneous power is zero: $F \cdot V = 0$. Since both twists and wrenches can be represented by screws, we can use this to define the notion of reciprocal screws. Two screws S_1 and S_2 are reciprocal if the twist V about S_1 and the wrench F along S_2 are reciprocal.

Composition of transformations

The use of transformations composition is of fundamental importance in robotics since is usual approach to build up any transformation from body to body frames transformations through subsequent transformations compositions. The mechanism is easy, each link is associated to a proper frame whose definitions is standardized using some convention and for every joint a definition of frame transformations for a follower (or conversely predecessor) frame respect a base (or follower) frame is defined as function of joint nature (see also Simscape® section for details). Using transformation composition rules along any path consisting of sequences of connected frames) allows for the definition of path extreme frames transformation calculation in a rather mechanical and automatized way easy to program. The same approach is not restricted to serial open chain but is virtually applicable to any mechanism provided there are congruence conditions satisfied. In open chain sequence path is unique (spanning tree concept), where in closed chain multiple path exist. Remarkably is not strictly necessary intermediate frames belongs to to a rigid link, just definition of frame to frame transformation are required, but in robotics for multiple reasons links frames are considered preferred method. Path can be followed forward and or backward, following left multiplication or right multiplication rule.



Composition of transformations: multiplication on the right



Composition of transformations: multiplication on the left

The rule of composition of transformations are generally valid for any system of frames, consequently is applicable to any mechanical system of rigid and/or non rigid bodies. In addition we can imagine that a general frame is not connected to any rigid body, and its orientation depend on time and other parameters as we like, if required. Serial open chain are more simple because we can imagine to have a single frame for each rigid body and the relative position and attitude of each frame depend on joints connecting rigid body sequence. This case is simple since a unique path can be defined from one frame to another. Even if a tree of rigid bodies is more complex since multiple paths exist for each branch, a general iterative operations sequence exist that allow to systematically calculate transformation matrix associated to each rigid body by convention.

Algorithm 3.1 Recursive computation of link transformations

$$\left\{ \begin{array}{l} \mathbb{I}_{T_{n+1}} = \mathbf{I} \\ \text{for } k = n \dots 1 \\ \quad \mathbb{I}_{T_k} = \mathbb{I}_{T_{k+1}} \cdot {}^{k+1}\mathbb{T}_k \\ \text{end loop} \end{array} \right.$$

Each transformation matrix is fully defined choosing up to six parameters properly chosen. In spite of generality, this deduction lead to an overestimate of the number of parameter strictly required for defining a rigid transformation between frames. Consequently a free choice of transformation matrix reveals to be partially arbitrary. Chasles theorem give us inference for minimal set of parameters for a rigid transformation between frames of just four parameters.

Using Denavit Hartenberg parameterisation this will become evident, as in case of screw motion.

Number of degree of freedom

Considering a system of N rigid bodies we've, in absence of constraints in free 3D space motion

$$N_{dof} = 6N$$

or in planar case

$$N_{dof} = 3N$$

Considering constraints between bodies by means of single degree of freedom joints, reduces overall degree of freedom, assuming open chain condition

$$N_{dof} = N_{joint} = N$$

assuming a single joint for each link. This last formula is true also in planar case.

If more constraints arise because, for instance of closure conditions for loops, mechanism number of degree of freedom is given by Gruebler's formula.

$$N_{dof} = N(N - g) + g$$

Grubler criterion

The **Chebychev–Grübler–Kutzbach criterion** determines the number of [degrees of freedom](#) of a [kinematic chain](#), that is, a coupling of rigid bodies by means of mechanical constraints. These devices are also called [linkages](#).

The Kutzbach criterion is also called the *mobility formula*, because it computes the number of parameters that define the configuration of a linkage from the number of links and joints and the degree of freedom at each joint.

Interesting and useful linkages have been designed that violate the mobility formula by using special geometric features and dimensions to provide more mobility than predicted by this formula. These devices are called [overconstrained mechanisms](#).

The result is that the mobility of a system formed from n moving links and j joints each with freedom f_i for $i = 1, \dots, j$, is given by

$$M = 6n - \sum_{i=1}^j (6 - f_i) = 6(N - 1 - j) + \sum_{i=1}^j f_i$$

Chebychev-Grubler-Kutzbach criterion

In *planar case* each link has 3 degree of freedom and general formula can be reduced to (Gruebler formula for $m=3$)

$$M = 3(N - 1 - j) - \sum_{i=1}^j f_i$$

For *open-chain robots* such as the industrial manipulator, each joint is independently actuated and the dof is simply the sum of the freedoms provided by each joint.

For *closed chains* like the Stewart–Gough platform, Grubler's formula is a convenient way to calculate a lower bound on the dof. Unlike open-chain robots, some joints of closed chains are not actuated.

Minimal representation

The choice of set of parameters used to represent frames transformations, point and rigid body kinematics is almost free and determined by considerations. Different equivalent coordinate sets are also possible, as well as mapping between them. In other sections we've seen that the number of parameters to set in order to define unambiguously the position and orientation of a frame is not as one may think at first six but four. Nevertheless the choice of more parameters is not forbidden but implies some *constraints* among parameters used, as well one can choose sets of parameters that are not independent but in minimal number. The choice of a minimal set of parameters is worthy because it reduces the number of conditions to consider, and resulting evaluation complexity. It's the same reason why we prefer often to use Euler's angle for defining rotations because it is more intuitive, although prone to singularities.

Choice of coordinates: generic and rotations representation

From historical literature a variety of different coordinates systems for representing frames and rigid body attitude, has been developed and in most of cases there's freedom to chose one or another without great issues. By the way, in truth differences exist when dealing with properties of coordinates choice. Translational subgroup of transformations is usually virtually arbitrary and impactless. Just remark standard choice assume a set of orthonormal vectors as most practical (although not strictly speaking mandatory) choice. The choice for representing rotations group is much harder and not unique. Key troubles one can find with a rotation parametrization regard essentially

- *Globality*
- *Non Singularities*
- Computational efforts (trigonometric and exponential functions)
- Rules for *factoring*
- *Physical meaning*
- *Minimality*

From math we've that a minima of three parameters is required to represent attitude, although more can be used under proper constraints. But **any minimal representation cannot be at same time global and nonsingular**.

On the other side, here we've the advantage of non minimal representation: they can be *non singular* and *global* at same time. One example is quaternions.

Second issue is rather common in most traditional axis rotation based coordinate systems, since there are configurations without a specific parameter set corresponding to singularities (non invertibility) potentially dangerous when dealing with arbitrary motions.

First issue regards the coverage of possible kinematic states and orientations.

Rodriguez and Euler representations are global but singular. Cayley 3-parameters are non singular but not global. The choice of parameters used for representing attitudes is key for an effective formalization of system kinematics.

We go now for illustrating some of most commonly and traditionally used system of parameters used for describe kinematics

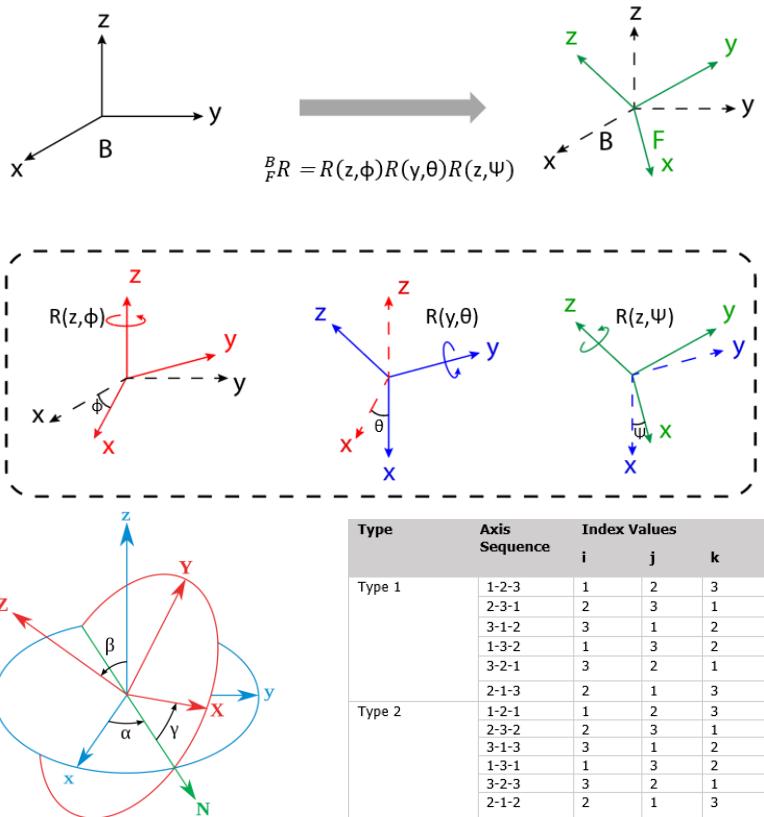
Euler angles

Euler angles are the most classic 3-parameter representation of rotation, and as said and easily shown, have singularities. Euler angles representation are sequence of rotations around the current axis, then order is almost always relevant. Rotation axis can also occur in non consecutive pairs. The number of possible sequence are 12,

$$R_x(\theta) = e^{\hat{x}\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{pmatrix}$$

$$R_y(\theta) = e^{\hat{y}\theta} = \begin{pmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{pmatrix}$$

$$R_z(\theta) = e^{\hat{z}\theta} = \begin{pmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



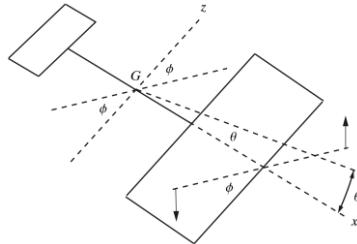
$$R_{ijk}(\phi, \theta, \psi) = \begin{bmatrix} i & j & k \\ c\theta & s\theta & -\alpha_{ij}s\theta c\phi \\ s\psi s\theta & c\psi c\phi - s\psi c\theta s\phi & \alpha_{ij}(c\psi s\phi + s\psi c\theta c\phi) \\ \alpha_{ij}c\psi s\theta & -\alpha_{ij}(s\psi c\phi + c\psi s\theta s\phi) & -s\psi s\phi + c\psi c\theta c\phi \end{bmatrix}$$

$$\alpha_{ij} = \sum_{k=1}^3 \epsilon_{ijk} = \begin{cases} +1 & \text{for } j \text{ cyclic successor of } i \\ -1 & \text{for } i \text{ cyclic successor of } j \\ 0 & \text{otherwise} \end{cases}$$

from [Schuster 93] where an extensive treatise on different attitude representation is reported.

The 1-2-1, 1-3-1, 2-3-2, 2-1-2, 3-1-3, and 3-2-3 sets of Euler angles are known as the symmetric sets, whereas the other six sets are known as asymmetric sets. The latter sets

are also known as the Cardan angles, Tait angles, or Bryan angles. Tait's original discussion (of what we would call 1-2-3 Euler angles) can be seen in Section 12 of his 1868 paper [Tait 1898]. In his seminal text [Bryan 11] on aircraft stability that was published in 1911, Bryan introduced what we would refer to as a 2-3-1 set of Euler angles. It is interesting to recall that the Wright brothers' first successful flight was in 1903.



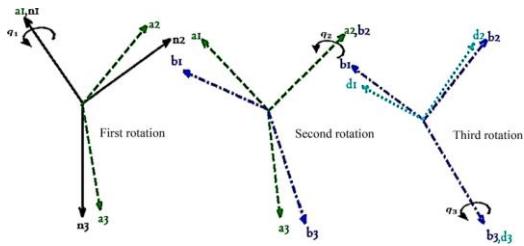
To avoid singularity associated with the most famous minimum element attitude parametrization, Euler angle set by making use of the method of sequential rotation.

$C_{ij} = \pm 1$	Singular Euler Angle Set	Euler angle sets farthest away from singularity
C_{11}	1-2-1,1-3-1	2-3-1, 3-2-1, 1-3-2, 1-2-3
C_{12}	2-3-1	2-1-2, 2-3-2, 2-1-3, 1-3-2, 3-1-2, 3-2-1
C_{13}	3-2-1	1-2-1, 1-3-1, 2-3-1, 3-1-2, 3-1-3 3-2-3
C_{21}	1-3-2	2-1-2, 2-3-2, 3-1-2, 1-2-1, 1-3-1 3-2-1
C_{22}	2-1-2,2-3-2	3-1-2, 3-2-1, 1-3-2, 2-3-1
C_{23}	3-1-2	3-2-1, 3-1-3, 3-2-3, 2-1-2, 2-3-2, 1-3-2
C_{31}	1-2-3	3-1-3, 3-2-3, 2-1-3, 1-3-1, 1-2-1, 1-3-2
C_{32}	2-1-3	3-1-3, 3-2-3, 2-1-2, 2-3-2, 1-2-3, 2-3-1
C_{33}	3-1-3,3-2-3	2-1-3, 1-2-3, 3-1-2, 3-2-1

Source: [How-to-avoid-singularity-when-using-Euler-angles-Singla-Mortari](#)

Cardan angles

Cardan angles, also known as Tait-Bryan angles representation, are similar to euler angles and are the composition of three rotations around coordinate axes, but this time rotation occurs around initial frame axis.



Further deepening about Euler and Bryan angles are found in [Euler angles](#) and in [Schuster 93].

Cayley 3- parameters

There exists a mapping $\phi: \text{SU}(2) \rightarrow \text{SO}(3)$ which is an [epimorphism](#) by virtue of its algebraic properties and a double [covering](#) by virtue of its topological properties. (Restricted to some neighborhood of the identity matrix, is an isomorphism). in other words, $\text{SO}(3)$ and $\text{SU}(2)$ are locally isomorphic.) Each matrix $V \in \text{SU}(2)$ may be written as

$$\begin{pmatrix} \alpha & \beta \\ -\bar{\beta} & \bar{\alpha} \end{pmatrix}$$

where α, β are complex numbers such that $\alpha^2 + \beta^2 = 1$. These are taken to be the Cayley–Klein parameters

In [mathematics](#), a [covering group](#) of a [topological group](#) H is a [covering space](#) G of H such that G is a topological group and the covering map $p: G \rightarrow H$ is a [continuous group homomorphism](#). The map p is called the [covering homomorphism](#). A frequently occurring case is a [double covering group](#), a [topological double cover](#) in which H has [index](#) 2 in G ; examples include the [spin groups](#).

$$e^{-i(\psi+\phi)/2} \cos\left(\frac{1}{2}\theta\right)$$

$$\mathbf{A} = \begin{bmatrix} \frac{1}{2}(a^2 - \gamma^2 + \delta^2 - \beta^2) & \frac{1}{2}i(\gamma^2 - \alpha^2 + \delta^2 - \beta^2) & \gamma\delta - \alpha\beta \\ \frac{1}{2}i(a^2 + \gamma^2 - \beta^2 - \delta^2) & \frac{1}{2}(a^2 + \gamma^2 + \beta^2 + \delta^2) & -i(\alpha\beta + \gamma\delta) \\ \beta\delta - \alpha\gamma & i(\alpha\gamma + \beta\delta) & a\delta + \beta\gamma \end{bmatrix}$$

The group $\text{SU}(2)$ is isomorphic to the group of quaternions with norm 1, then one can replace the Cayley–Klein parameters by the Euler–Rodriguez parameters, four real numbers

Axis and angle parameters

One very useful rotations group representation, coming out directly from Euler theorem, gives a rotation by means of a versor orientation (then two parameters are required, considering unitarity condition) and rotation angle.

$$\begin{pmatrix} u_x^2(1 - c\theta) + c\theta & u_xu_y(1 - c\theta) - u_zs\theta & u_xu_z(1 - c\theta) + u_ys\theta \\ u_xu_y(1 - c\theta) + u_zs\theta & u_y^2(1 - c\theta) + c\theta & u_yu_z(1 - c\theta) - u_xs\theta \\ u_xu_z(1 - c\theta) - u_ys\theta & u_yu_z(1 - c\theta) + u_xs\theta & u_z^2(1 - c\theta) + c\theta \end{pmatrix}$$

DCM: Direct Cosine Matrix

Direct cosine matrix is just a matrix composed with scalar product of frames vectors. Consequently it represent any rotation without singularities, but need to specify all nine parameters, although they're not independent since as for rotation matrix orthonormality conditions sussist, leading to three independent degrees of freedom.

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} v_1 \cdot u_1 & v_1 \cdot u_2 & v_1 \cdot u_3 \\ v_2 \cdot u_1 & v_2 \cdot u_2 & v_2 \cdot u_3 \\ v_3 \cdot u_1 & v_3 \cdot u_2 & v_3 \cdot u_3 \end{pmatrix}$$

This representation is then

Quaternions

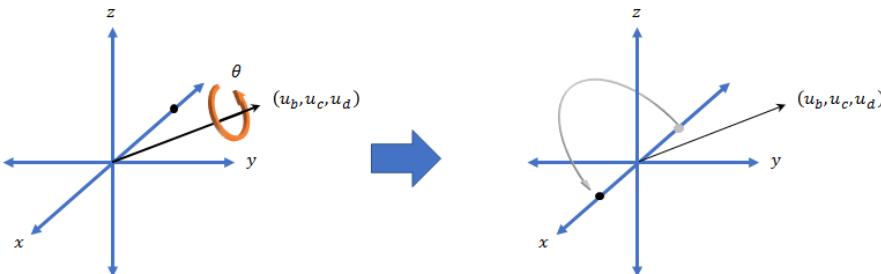
The quaternions are also called Euler parameters or Olinde-Rodrigues parameters. In this representation, the orientation is expressed by four parameters that describe the orientation by a rotation of an angle θ ($0 < \theta < \pi$) about an axis of unit vector u . We define the quaternions as:

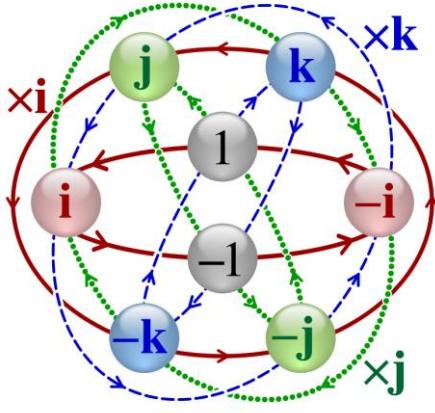
$$q = (u_x s\theta \quad u_y s\theta \quad u_z s\theta \quad c\theta)^T$$

satisfying property

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

Being not minimal and avoiding use of trigonometric functions are recently most used for representing rotations in mechanics, especially in aerospace applications.





Multiplication table
Non commutativity is
emphasized by
colored squares

x	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

SU(2) representation: Cayley-Klein parameters

The unitary Cayley-Klein matrices form a group, denoted by $SU(2)$, the group of special unitary matrices in 2 dimensions. The group identity element is I_{2x2} , and the inverse is given by the Hermitian conjugate. Like the Euler-Rodrigues symmetric parameters, there are two sets of the Cayley-Klein parameters corresponding to every rotation, which differ from one another only by an overall sign. Noting the similarities in the elements of the Cayley-Klein matrix, this has the general form

$$Q = \begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix}$$

$$H = \begin{pmatrix} \alpha & \beta \\ -\beta^* & \alpha^* \end{pmatrix}$$

$$\alpha\alpha^* + \beta\beta^* = 1$$

Topology of rotations and rototranslations

Orthonormal matrix group $O(3)$ is topologically structured into two connected area, characterized by different determinant ± 1 . That's why usually additional condition is imposed of positive determinant, leading to $SO(3)$ group. Usually Rotations group can be represented in many ways, depending on used parametrization, but used parameterization and representation, but topology is not affected. One is by using group representations matrix $SO(3)$, $SU(2)$, and as subgroups of $SE(3)$. These representations include theirs subrepresentations for reduced dimensionality like $SO(2)$, $SU(1)$ (complex representation of rotations using gauss plane).

Comparison between representations

Method	Advantages	Disadvantages
Euler Angles	-No redundant parameters -Clear physical interpretation	-Singularities -Trig functions -No convenient product rule
Direction Cosine	-No singularities -Good physical representation -Convenient product rule	-Six redundant parameters -Trig functions
Quaternions	-No singularities -No trig functions -Convenient product rule	-One redundant parameter -No physical meaning

Parametrization	Dimension	Attitude Matrix	Kinematic Equations	Singularities	Constraints
DCM, (C_{ij})	9	$C = [C_{ij}]$	$\dot{C} = -[\omega]C$	None	$C^T C = I$
EA (θ_i)	3	$C = \begin{bmatrix} \text{transcendental functions of } \theta_i \\ \theta_i \end{bmatrix}$	$\dot{\theta} = \begin{bmatrix} \text{transcendental functions of } \theta_i \\ \theta_i \end{bmatrix} \omega$	$\theta_i = \pm \frac{\pi}{2}$	None
ERSP (q_i)	4	$C = \begin{bmatrix} \text{algebraic functions of } q_i \\ q_i \end{bmatrix}$	$\dot{q} = \begin{bmatrix} \text{linear functions of } q_i \\ q_i \end{bmatrix} \begin{Bmatrix} 0 \\ \omega \end{Bmatrix}$	None	$q^T q = 1$
RP (r_i)	3	$C = \begin{bmatrix} \text{quadratic functions of } r_i \\ r_i \end{bmatrix}$	$\dot{r} = \begin{bmatrix} \text{non-linear functions of } r_i \\ r_i \end{bmatrix} \omega$	$\phi = \pm \pi$	None
MRP (ϕ_i)	3	$C = \begin{bmatrix} \text{cyclic functions of } \phi_i \\ \phi_i \end{bmatrix}$	$\dot{\phi} = \begin{bmatrix} \text{non-linear functions of } \phi_i \\ \phi_i \end{bmatrix} \omega$	$\phi = \pm 2\pi$	None

Source: [How-to-avoid-singularity-when-using-Euler-angles \[Singla-Mortari-Junkins 05\]](#)

Screws

A screw motion is characterized by a line axis rotation and a subsequent translation of magnitude depending on pitch along same line

$$gp = q + e^{\hat{\omega}\theta} (p - q) + h\theta\omega$$

In this section, we explore some of the geometric attributes associated with a twist $\xi = (\nu, \omega)$. These attributes give additional insight into the use of twists to parameterize rigid body motions. We begin by defining a specific class of rigid body motions, called screw motions, and then show that a twist is naturally associated with a screw. Consider a rigid body motion which consists of rotation about an axis in space through an angle of θ radians, followed by translation along the same axis by an amount d as shown in Figure 2.7a. We call such a motion a screw motion, since it is reminiscent of the motion of a screw, in so far as a screw rotates and translates about the same axis. To further encourage this analogy, we define the pitch of the screw to be the ratio of translation to rotation, $h := d/\theta$ (assuming $\theta \neq 0$). Thus, the net translational motion after rotating by θ radians is $h\theta$.

A screw S consists of an axis l , a pitch h , and a magnitude M . A screw motion represents rotation by an amount $\theta = M$ about the axis l followed by translation by an amount $h\theta$ parallel to the axis l . If $h = \infty$ then the corresponding screw motion consists of a pure translation along the axis of the screw by a distance M . To compute the rigid body transformation associated with a screw, we analyze the motion of a point $p \in \mathbb{R}^3$, as shown in Figure 2.8. The final location of the point is given by

Screw and twist correspondence

Given a screw with axis l , pitch h , and magnitude M , there exists a unit magnitude twist ξ such that the rigid motion associated with the screw is generated by the twist $M\xi$.

Commented [4]: Keep or leave, check notation

Screw coordinates

There is not a simple one-to-one mapping between the twist coordinates for the joints of a robot manipulator and the Denavit-Hartenberg parameters. This is because the twist coordinates for each joint are specified with respect to a single base frame and hence do not directly represent the relative motions of each link with respect to the previous link.

However, in almost all instances it is substantially easier to construct the joint twists ξ_i directly, by writing down the direction of the joint axes and, in the case of revolute joints, choosing a convenient point on each axis. Indeed, one of the most attractive features of the product of exponentials formula is its usage of only two coordinate frames, the base frame S, and the tool frame T. This property, combined with the geometric significance of the twists ξ_i , make the product of exponentials representation a superior alternative to the use of Denavit-Hartenberg parameters.

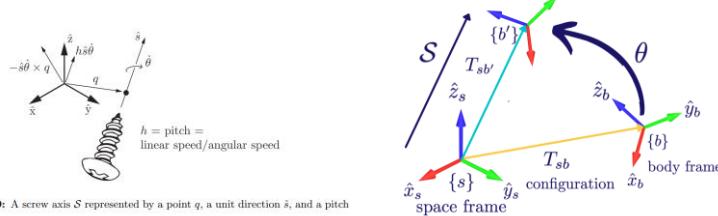


Figure 3.19: A screw axis S represented by a point q , a unit direction \hat{s} , and a pitch h .

Lie Groups and Algebra of transformations and rotations

Let X be an $n \times n$ [real](#) or [complex matrix](#). The exponential of X , denoted by e^X or $\exp(X)$, is by definition the $n \times n$ matrix given by the [power series](#)

$$e^X = \exp(X) = \sum_{k=0}^{\infty} \frac{1}{k!} X^k$$

where X^0 is defined to be the identity matrix I with the same dimensions as X . The series always converges, so the exponential of X is well-defined. Equivalently

$$\begin{aligned} \exp(X) &= e^X = \lim_{k \rightarrow \infty} \left(I + \frac{X}{k} \right)^k \\ \exp(X) &= \lim_{k \rightarrow \infty} \left(I + \frac{X}{k} \right)^k \approx \prod_{n=1}^k \underbrace{\left(I + \frac{X}{k} \right)}_{T_\epsilon} \\ &\quad k \rightarrow \infty \implies T_\epsilon \rightarrow I \end{aligned}$$

Leading to interpretation of exponential formula as application of infinite repetition of infinitesimal transformations.

See [Matrix exponential](#) for details.

By [Jacobi's formula](#), for any complex square matrix the following [trace identity](#) holds:

$$\det(e^X) = e^{\text{Tr}(X)}$$

This formula demonstrates that a matrix exponential is always an [invertible matrix](#), follows from the fact that the right hand side of the above equation is always on-zero. The matrix exponential of a real symmetric matrix is positive definite.

$$x^T e^S x = x^T e^{S/2} e^{S/2} x = x^T \left(e^{S/2} \right)^T e^{S/2} x = z^T z \geq 0$$

Even if X and Y do not commute, the exponential e^{X+Y} can be computed by the [Lie product formula](#)

$$e^{X+Y} = \lim_{k \rightarrow \infty} \left(e^{\frac{X}{k}} e^{\frac{Y}{k}} \right)^k$$

Using a large finite k to approximate the above is basis of the [Suzuki-Trotter expansion](#).

In the other direction, if X and Y are sufficiently small (but not necessarily commuting) matrices, we have

$$e^X e^Y = e^Z \Rightarrow Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] - \frac{1}{12}[Y, [X, Y]] + \dots$$

where Z may be computed as a series in [commutators](#) of X and Y . Commuting case where X and Y can swap effectless, is only a special case.

Exponential formula results to be

Using matrix exponential formalism is easy to see of a transformation characterized by continuity on parameters can be expressed as exponential of an algebra of generators

$$T = \sum_{k=1}^N \alpha_k G_k$$

$$e^T \approx I + T + \frac{1}{2}T^2 + \frac{1}{3!}T^3 + \dots + \frac{1}{n!}T^n + \dots$$

As intuition suggest for bijective applications, an inverse function exist for exponential matrices, that we can call log. This works also for complex matrix exponential inversion, leading to complex logarithm concept.

$$\log(I + A) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{A^k}{k!}$$

R ∈ ℝ ^{n×n}	log	θ ∈ ℝ u ∈ ℝ ⁿ
rotation matrix	exp	angle & axis

Application to rotation group

Exponential rotations

Considering a common case in robotics like a single axis revolute joint, it realizes a rotation around an axis. Kinematic differential equations results in:

$$\dot{q} = \omega \wedge q(t) = \tilde{\omega}q(t)$$

can be integrated as linear time invariant differential equations, whose integral is

$$q(t) = e^{\tilde{\omega}t}q_0$$

with initial state in t=0 q_0 , and $e^{\tilde{\omega}t}$ is a matrix exponential operator, that must coincide with a rotation matrix

$$R(\omega, t) = e^{\tilde{\omega}t} = I + \tilde{\omega}t + \frac{\tilde{\omega}^2}{2!}t^2 + \frac{\tilde{\omega}^3}{3!}t^3 + \frac{\tilde{\omega}^4}{4!}t^4 + \dots = R(\omega, \theta)$$

The proof of orthonormality is binded to the skew symmetric nature of $\tilde{\omega}$, leading

$$R^{-1} = e^{-\tilde{\omega}t} = e^{\tilde{\omega}^T t} = \left(e^{\tilde{\omega}t} \right)^T = R^T$$

In addition continuity of determinant function on t, together with $\det(I) = 1$ for identity matrix lead to

$$\det(R) = 1$$

Geometrically skew symmetric matrices correspond to rotation unitary axis vector, as highlighted from tensorial bi-vector formalism.

a subjective mapping exist by construction between rotation matrix expressions and exponential formulation, leading to expression of rotation matrix as function of axis vector components and rotation parameter q.

$$\begin{aligned} e^{\tilde{\omega}q} &= \begin{bmatrix} 1 - (\omega_2^2 + \omega_3^2)v_\theta & \omega_1\omega_2v_\theta - \omega_3s_\theta & \omega_1\omega_3v_\theta + \omega_2s_\theta \\ \omega_1\omega_2v_\theta + \omega_3s_\theta & 1 - (\omega_1^2 + \omega_3^2)v_\theta & \omega_2\omega_3(1 - cq) - \omega_1s_\theta \\ \omega_1\omega_3v_\theta - \omega_2s_\theta & \omega_2\omega_3v_\theta + \omega_1s_\theta & 1 - (\omega_1^2 + \omega_2^2)v_\theta \end{bmatrix} \\ &= \begin{bmatrix} \omega_1^2v_\theta + c_\theta & \omega_1\omega_2v_\theta - \omega_3s_\theta & \omega_1\omega_3v_\theta + \omega_2s_\theta \\ \omega_1\omega_2v_\theta + \omega_3s_\theta & \omega_2^2v_\theta + c_\theta & \omega_2\omega_3v_\theta - \omega_1s_\theta \\ \omega_1\omega_3v_\theta - \omega_2s_\theta & \omega_2\omega_3v_\theta + \omega_1s_\theta & \omega_3^2v_\theta + c_\theta \end{bmatrix} \\ e^{\tilde{\omega}q} &= \begin{bmatrix} \omega_1^2(1 - cq) + cq & \omega_1\omega_2(1 - cq) - \omega_3sq & \omega_1\omega_3(1 - cq) + \omega_2sq \\ \omega_1\omega_2(1 - cq) + \omega_3sq & \omega_2^2(1 - cq) + cq & \omega_2\omega_3(1 - cq) - \omega_1sq \\ \omega_1\omega_3(1 - cq) - \omega_2sq & \omega_2\omega_3(1 - cq) + \omega_1sq & \omega_3^2(1 - cq) + cq \end{bmatrix} \end{aligned}$$

resulting in common properties of notation matrix

$$\text{Tr}(R) = r_{11} + r_{22} + r_{33} = 1 + 2 \cos(\theta)$$

$$\theta = a \cos \left(\frac{\text{Tr}(R) - 1}{2} \right)$$

$$w = \frac{1}{2s\theta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

Base and generators

Basic generators dependents on adopted coordinates systems and on the choice of generators, since any linear combination of generators could be used to define an alternative generators algebra. Often simpler generator are preferred for sake of simplicity, corresponding to those aligning to coordinate axis.

$$L_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad L_y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad L_z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Vector field arising from linear combinations of generators gives a skew matrix. Taking exponential of skew matrix, we've for a generic rotation having magnitude θ :

$$\begin{aligned} \exp([\hat{\omega}]\theta) &\approx I + [\hat{\omega}]\theta + [\hat{\omega}]^2 \frac{\theta^2}{2!} + [\hat{\omega}]^3 \frac{\theta^3}{3!} + \dots \\ &\approx I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots \right) [\hat{\omega}] + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots \right) [\hat{\omega}]^2 \end{aligned}$$

$$\begin{aligned} \sin(\theta) &\approx 0 + \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \dots \\ \cos(\theta) &\approx 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \dots \end{aligned}$$

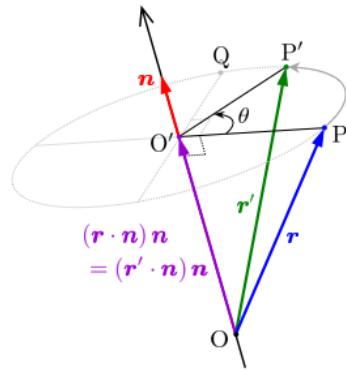
Taking them together we get a closed form formula for rotations, that in spite of apparent approximation of series expansions, PoE reveals to be exact.

$$\text{Rot}(\hat{\omega}, \theta) = \exp([\hat{\omega}]\theta) = I + \sin(\theta)[\hat{\omega}] + (1 - \cos(\theta))[\hat{\omega}]^2$$

This formula is well known as Rodriguez formula, usable in form of matrix or as well to single vector, or even frames and tuples of vectors $\{F\}$, apply:

$$\begin{aligned} R &= I + \sin(\theta)[\omega \times] + \frac{1 - \cos(\theta)}{2} [w \times]^2 \\ R\{F\} &= \{F\} + \sin(\theta)[\omega \times]\{F\} + (1 - \cos(\theta))[w \times]^2\{F\} \\ Rv &= v + \sin(\theta)[\omega \times]v\{F\} + (1 - \cos(\theta))[w \times]^2v \\ v &= v + \sin(\theta)k \times v + \frac{1 - \cos(\theta)}{2}k(k \cdot v) \end{aligned}$$

where properties of vector product has been used. Same formula can be derived from some elementary considerations of vectorial algebra and geometry of rotations



$$\begin{aligned} O'P' = r' &= OO' + O'P \quad + P'P \\ &= n(n \cdot r) + (r - n(n \cdot r)) \cos \phi + (r \times n) \sin \phi \end{aligned}$$

As consequence of exponential formalism we've the definition of logarithm of an operators and of matrixes, as well the logarithm of rotation matrixes, intended as inverse operation, found in literature.

Lie group for transformations

Rigid body and frames related homogeneous transformations can be composed to form new rigid body transformations. As for other groups a Lie representation and algebra exist, with generators and exponential formulas associated. A general transformation can be obtained by composition of intermediate transformations resulting in a composition formula based on products of generators exponential. Homogeneous and affine transformations are characterized by

- *Closure*: Composition of two homogeneous transformations (rigid body motion) is a transformation (rigid body motion)
- *Identity* matrix 4x4 corresponds to an identity transformation
- *Inverse*
- Transformations composition is *associative*

Let $g_{bc} \in SE(3)$ be the configuration of a frame C relative to a frame B, and g_{ab} the configuration of frame B relative to another frame A. Then, using equation, the configuration of C relative to frame A is given by

Twists

What done with rotations group can be extended the method to rototranslations, leading to a linear time invariant differential equations

$$\dot{\bar{p}} = \hat{\xi} \bar{p}$$

$$\hat{\xi} = \begin{pmatrix} -\tilde{\omega} & \omega \wedge q \\ 0 & 1 \end{pmatrix}$$

whose solution is similar to exponential for rotations but leading to a different structure

$$\begin{aligned} \bar{p} &= e^{\tilde{\xi}t} \bar{p}_0 \\ e^{\tilde{\xi}t} &= I + \tilde{\xi}t + \frac{\tilde{\xi}^2}{2!} t^2 + \frac{\tilde{\xi}^3}{3!} t^3 + \frac{\tilde{\xi}^4}{4!} t^4 + \dots \end{aligned}$$

for a revolute joint, where for a prismatic joint we've

$$e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \quad \omega \neq 0$$

Logarithm of homogeneous transformation

As in rotations case is possible to define the logarithm of a homogeneous transformations T as a matrix twist.

$$e^{[S]\theta} = T = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix} \implies [S]\theta = \begin{pmatrix} [\omega] & v \\ 0 & 0 \end{pmatrix} \theta = \log(T)$$

Generators of group for homogeneous transformation

Just as it is convenient to have a skew-symmetric matrix representation of an angular velocity vector, it is convenient to have a matrix representation of a twist, as shown in equation. In a similar way to what was done for rotations we can adopt any basis for generators of Lie algebra associated to homogeneous transformations group of rototranslations. Twist coordinates are introduced as two vectors stacked leading to a six vector that allows the definition of matrix $\hat{\xi}$ by wedge operation and inverse:

$$\begin{pmatrix} \tilde{\omega} & v \\ 0 & 0 \end{pmatrix} \sim = \begin{pmatrix} v \\ \omega \end{pmatrix} = \nu$$

related to homogeneous transformation derivatives

$$T^{-1}\dot{T} = \begin{bmatrix} \tilde{\omega} & v \\ 0 & 0 \end{bmatrix} = [\mathcal{V}]$$

The set of all 4×4 matrices of this form is called $\text{se}(3)$ and comprises the matrix representations of the twists associated with the rigid-body configurations $\text{SE}(3)$.

Jacobian in twist coordinates

Using twist coordinates one can write jacobian

$$J^s = [\xi'_1 \dots \xi'_n]$$

jacobian columns correspond to twists associated to each joint.

Product of exponential (PoE)

The formulation which we present here is the product of exponentials formula, which represents the kinematics of an open-chain mechanism as the product of exponentials of twists. This setting works whenever the joints of the robot consist of either revolute, prismatic, or helical joints, which is the case for practically all commercially available robot manipulators. It provides a global, geometric representation of the kinematics of a manipulator which greatly simplifies the analysis of the mechanism and provides a very structured parameterization for openchain robots. A more geometric description of the kinematics can be obtained by using the fact that motion of the individual joints is generated by a *twist* associated with the joint axis. The **product of exponentials (POE)** method is a [robotics convention](#) for mapping the links of a spatial [kinematic chain](#). It is an alternative to [Denavit–Hartenberg](#) parameterization. While the latter method uses the *minimal number of parameters* to represent joint motions, the former method has a number of advantages: uniform treatment of prismatic and revolute joints, definition of only two reference frames, and an easy geometric interpretation from the use of screw axes for each joint. The product of exponentials method uses only two [frames of reference](#): the base frame S and the tool frame T . Constructing the Denavit–Hartenberg parameters for a robot requires the careful selection of tool frames in order to enable particular cancellations, such that the twists can be represented by four parameters instead of six. In the product of exponentials method, the joint twists can be constructed directly without considering adjacent joints in the chain. This makes the joint twists easier to construct, and easier to process by computer. In addition, revolute and prismatic joints are treated uniformly in the POE method, while they are treated separately when using the Denavit–Hartenberg parameters. Moreover, there are multiple conventions for assigning link frames when using the Denavit–Hartenberg parameters.

$$g_{sd} = g_{l_0 l_1}(q_1) g_{l_1 l_2}(q_2) g_{l_2 l_3}(q_3) \dots g_{l_{n-1} l_n}(q_n) g_{l_n t}$$

There is not a one-to-one mapping between twist coordinate mapping in both methods, but algorithmic mapping from POE to Denavit–Hartenberg has been demonstrated. Each joint matrix can be easily be built up by exponentiation once known its generator .

$$e^{\hat{\xi}_k q_k} = \begin{pmatrix} e^{\hat{\omega}_k q_k} & t \\ 0 & 1 \end{pmatrix}$$

Composition rule for forward kinematics just comes out by matrix multiplication of 4x4 transformation matrix arising from previous step

$$g_d = e^{\hat{\xi}_1 q_1} e^{\hat{\xi}_2 q_2} e^{\hat{\xi}_3 q_3} \dots e^{\hat{\xi}_n q_n} g_{st}(0) = g_d(q_1, q_2, \dots, q_n)$$

$$g_d = \prod_{k=1}^{N_{dof}} e^{\hat{\xi}_k q_k} g_{st}(0)$$

The ξ_i 's must be numbered sequentially starting from the base. Formula gives the configuration of the tool frame (as well as of any intermediate) independently of the order in which the rotations and translations are actually performed.

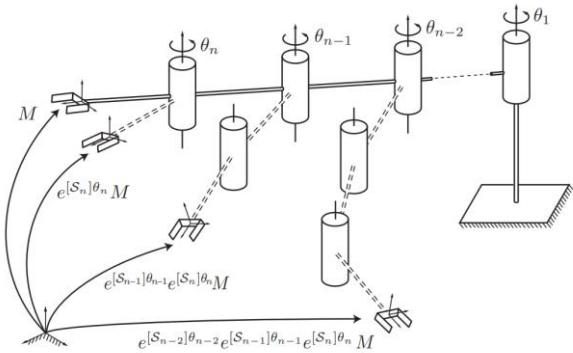


Figure 4.2: Illustration of the PoE formula for an n -link spatial open chain.

Considerations about pros and cons of PoE formalism

The payoff for the product of exponentials formalism is shown in this chapter in the context of an elegant formulation of a set of *canonical problems* for solving the inverse kinematics problem: the problem of determining the joint angles given the position and orientation of the end-effector or gripper of the robot. Algebraic geometry, which we discuss briefly.

Another payoff of using the product of exponentials formula for kinematics is the ease of differentiating the kinematics and to obtain the manipulator Jacobian.

The columns of the manipulator Jacobian have the interpretation of being the twist axes of the manipulator, in similar way to geometrical Jacobian. As a consequence, it is easy to geometrically characterize and describe the singularities of the manipulator. The product of exponentials formula is also used for deriving the kinematics of robots with one or more closed kinematic chains, such as a Stewart platform or a four-bar planar linkage.

Derivation of the product of exponentials formula for the forward kinematics of an arbitrary open-chain manipulator. We concentrate on the most general case, where the end-effector configuration lies in SE(3).

However, in almost all instances it is substantially easier to construct the joint twists ξ_j directly, by writing down the direction of the joint axes and, in the case of revolute joints, choosing a convenient point on each axis. Indeed, one of the most attractive features of the product of exponentials formula is its usage of only two coordinate frames, the base frame S, and the tool frame T. This property, combined with the geometric significance of the twists ξ_j , make the product of exponentials representation a superior alternative to the use of Denavit-Hartenberg parameters.

for g near the identity, then the functions $(\theta_1, \theta_2, \dots, \theta_n)$ are called the canonical coordinates of the second kind.

Derivative exponential formula

It's easy to prove that derivative of transformations with POE formalism lead to a eigenvector formula

$$\begin{aligned} \frac{d}{d\theta} \left(e^{\hat{\xi}\theta} \right) &= \frac{d}{d\theta} \left(I + \hat{\xi} \frac{\theta}{1!} + \hat{\xi}^2 \frac{\theta^2}{2!} + \hat{\xi}^3 \frac{\theta^3}{3!} + \dots + \hat{\xi}^n \frac{\theta^n}{n!} \right) = \hat{\xi} + 2\hat{\xi}^2 \frac{\theta}{2!} + 3\hat{\xi}^3 \frac{\theta^2}{3!} + \dots + n\hat{\xi}^n \frac{\theta^{n-1}}{n!} \\ \hat{\xi} + 2\hat{\xi}^2 \frac{\theta}{2!} + 3\hat{\xi}^3 \frac{\theta^2}{3!} + \dots + n\hat{\xi}^n \frac{\theta^{n-1}}{n!} &= \hat{\xi} \left(I + \hat{\xi}\theta + \hat{\xi}^2 \frac{\theta^2}{2!} + \dots + \hat{\xi}^{n-1} \frac{\theta^{n-1}}{(n-1)!} \right) = \hat{\xi} e^{\hat{\xi}\theta} \end{aligned}$$

Campbell-Baker-Hausdorff formula

Product of exponential is related to Lie algebra vector field of generators by means of Campbell-Baker-Hausdorff formula. Consequently product operation out is generally not commutative since Lie brackets generally are not. But under specific assumptions, like nullity of Lie brackets, lead to transformations commutativity.

$$e^{g_1} e^{g_2} = e^{g_1 + g_2 + \frac{1}{2}[g_1, g_2] + \frac{1}{12}([g_1[g_1, g_2]] - [g_2[g_1, g_2]])} \dots$$

$$\exp(g_1) \exp(g_2) = \exp \left(g_1 + g_2 + \frac{1}{2}[g_1, g_2] + \frac{1}{12}([g_1[g_1, g_2]] - [g_2[g_1, g_2]]) \dots \right)$$

Base generators vectors of a Lie algebra are not all linear independent, because of structural relations.

To get a minimal independent base of vector some selection rules shall be applied, leading to Philip Hall coordinates:

Philip Hall coordinates

Given a set of generators $\{g_1, \dots, g_m\}$, we define the length of a Lie product recursively as

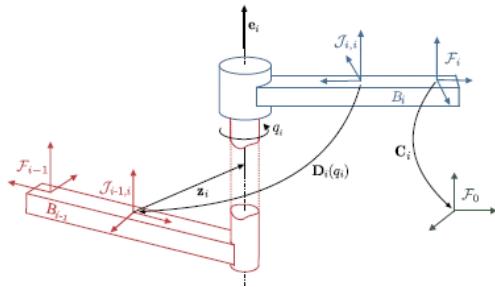
$$l(g_i) = 1$$

$$l([A, B]) = l(A) + l(B)$$

A Philip Hall basis is an ordered set of Lie products $H = \{B_i\}$ satisfying:

1. $g_i \in H$
2. $[B_i, B_j] \in H$
3. $B_i, B_j \in H$ and $B_i < B_j$ and
either $B_j = g_k$ for some k or $B_j = [B_l, B_r]$ with $B_l, B_r \in H$ and $B_l \leq B_i$

A Philip Hall basis which is nilpotent of order k can be constructed from a set of generators using this definition. The simplest approach is to construct all possible Lie products with length less than k and use the definition to eliminate elements which fail to satisfy one of the properties. In practice, the basis can be built in such a way that only condition 3 need be checked



Synthetic synoptic of kinematics relations

Rotations	Rigid-Body Motions
$R \in SO(3) : 3 \times 3$ matrices	$T \in SE(3) : 4 \times 4$ matrices
$R^T R = I, \det R = 1$	$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix},$ where $R \in SO(3), p \in \mathbb{R}^3$
$R^{-1} = R^T$	$T^{-1} = \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix}$
change of coordinate frame: $R_{ab}R_{bc} = R_{ac}, R_{ab}p_b = p_a$	change of coordinate frame: $T_{ab}T_{bc} = T_{ac}, T_{ab}p_b = p_a$
rotating a frame {b}:	displacing a frame {b}:
$R = \text{Rot}(\dot{\omega}, \theta)$	$T = \begin{bmatrix} \text{Rot}(\dot{\omega}, \theta) & p \\ 0 & 1 \end{bmatrix}$
$R_{sb'} = RR_{sb}$: rotate θ about $\dot{\omega}_s = \dot{\omega}$	$T_{sb'} = TT_{sb}$: rotate θ about $\dot{\omega}_s = \dot{\omega}$ (moves {b} origin), translate p in {s}
$R_{sb'} = R_{sb}R$: rotate θ about $\dot{\omega}_b = \dot{\omega}$	$T_{sb'} = T_{sb}T$: translate p in {b}
unit rotation axis is $\dot{\omega} \in \mathbb{R}^3$, where $\ \dot{\omega}\ = 1$	"unit" screw axis is $\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6$, where either (i) $\ \omega\ = 1$ or (ii) $\omega = 0$ and $\ v\ = 1$
for a screw axis $\{q, \dot{s}, h\}$ with finite h ,	
$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \dot{s} \\ -\dot{s} \times q + h\dot{s} \end{bmatrix}$	
angular velocity is $\omega = \dot{\omega}\dot{\theta}$	twist is $\mathcal{V} = \mathcal{S}\dot{\theta}$
for any 3-vector, e.g., $\omega \in \mathbb{R}^3$,	for $\mathcal{V} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6$,
$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in so(3)$	$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \in se(3)$
identities, $\omega, x \in \mathbb{R}^3, R \in SO(3)$:	(the pair (ω, v) can be a twist \mathcal{V} or a "unit" screw axis \mathcal{S} , depending on the context)
$[\omega] = -[\omega]^T, [\omega]x = -[x]\omega,$ $[\omega][x] = ([x][\omega])^T, R[\omega]R^T = [R\omega]$	
$RR^{-1} = [\omega_a], R^{-1}\dot{R} = [\omega_b]$	$\dot{T}T^{-1} = [\mathcal{V}_a], T^{-1}\dot{T} = [\mathcal{V}_b]$
change of coordinate frame: $\dot{\omega}_a = R_{ab}\dot{\omega}_b, \omega_a = R_{ab}\omega_b$	change of coordinate frame: $\mathcal{S}_a = [\text{Ad}_{T_{ba}}]\mathcal{S}_b, \mathcal{V}_a = [\text{Ad}_{T_{ba}}]\mathcal{V}_b$
exp coords for $R \in SO(3)$: $\dot{\omega}\theta \in \mathbb{R}^3$	exp coords for $T \in SE(3)$: $\mathcal{S}\theta \in \mathbb{R}^6$
$\exp : [\dot{\omega}]\theta \in so(3) \rightarrow R \in SO(3)$	$\exp : [\mathcal{S}]\theta \in se(3) \rightarrow T \in SE(3)$
$R = \text{Rot}(\dot{\omega}, \theta) = e^{[\dot{\omega}]\theta} =$	$T = e^{[\mathcal{S}]\theta} = \begin{bmatrix} e^{[\dot{\omega}]\theta} & * \\ 0 & 1 \end{bmatrix}$
$I + \sin \theta [\dot{\omega}] + (1 - \cos \theta)[\dot{\omega}]^2$	where $* = (I\theta + (1 - \cos \theta)\omega) + (\theta - \sin \theta)[\omega]^2 v$
$\log : R \in SO(3) \rightarrow [\dot{\omega}]\theta \in so(3)$ algorithm in Section 3.2.3.3	$\log : T \in SE(3) \rightarrow [\mathcal{S}]\theta \in se(3)$ algorithm in Section 3.3.3.2
moment change of coord frame: $m_a = R_{ab}m_b$	wrench change of coord frame: $\mathcal{F}_a = (m_a, f_a) = [\text{Ad}_{T_{ba}}]^T \mathcal{F}_b$

Links kinematics

Joints kinematics description

Unless explicitly stated otherwise, the kinematic description of robotic mechanisms typically employs a number of idealizations. The links that compose the robotic mechanism are assumed to be perfectly rigid bodies having surfaces that are geometrically perfect in both position and shape. Accordingly, these rigid bodies are connected together at joints where their idealized surfaces are in ideal contact without any clearance between them. The respective geometries of these surfaces in contact determine the freedom of motion between the two links, or the joint kinematics. A kinematic joint is a connection between two bodies that constrains their relative motion. Two bodies that are in contact with one another create a simple kinematic joint. The surfaces of the two bodies that are in contact are able to move over one another, thereby permitting relative motion of the two bodies. Simple kinematic joints are classified as *lower pair joints* if contact occurs over surfaces and as higher pair joints if contact occurs only at points or along lines. A joint model describes the motion of a frame fixed in one body of a joint relative to a frame fixed in the other body. The motion is expressed as a function of the joint's motion variables, and other elements of a joint model include the rotation matrix, position vector, free modes, and constrained modes. The free modes of a joint define the directions in which motion is allowed. They are represented by the $6n_i$ matrix Φ_i whose columns are the Plücker coordinates of the allowable motion. This matrix relates the spatial velocity vector across the joint

to the joint velocity vector \dot{q}_i

$$v_{rel,j} = \Phi_i \dot{q}_i$$

In contrast, the constrained modes of a joint define the directions in which motion is not allowed. They are represented by the $6(6 - n_i)$ matrix Φ_i^c that is complementary to Φ_i . Tables contain the formulas of the joint models for all of the joints described in this section. They are used extensively for the dynamic analysis presented in literature. Additional information on joints can be found in modern textbooks.

Lower Pair Joints

[Lower pair joints](#) are mechanically attractive since wear is spread over the whole surface and lubricant is trapped in the small clearance space (in non idealized systems) between the surfaces, resulting in relatively good lubrication. As can be proved from the requirement for surface contact, there are only *six possible forms* of lower pair joints: *revolute*, *prismatic*, *helical*, *cylindrical*, *spherical*, and *planar joints*.

For more details see [Handbook of robotics] section 2.3.1 and [Lower pair joints](#).

Link geometry

When two bodies are connected by a joint, a complete description of the connection consists of a *description of the joint itself*, and the locations of two coordinate frames, one in each body, which specify where in each body the *joint is located*. If there are N_J joints in the system, then there are a total of $2N_J$ joint-attachment frames. One half of these frames are identified with the numbers 1 to N_J , and the remainder with the labels J_1 to J_{N_J} . Each joint i connects from frame J_i to frame i . For joints 1 to N_B (i. e., the tree joints), frame i is rigidly attached to body i . For joints $N_B + 1$ to N_J , frame k for loop-closing joint k will be rigidly attached to body $s(k)$. The second coordinate frame J_i is attached to the predecessor $p(i)$ for each joint i , whether it is a tree joint or a loop-closing joint. Coordinate frame J_i provides a base frame for joint i in that the joint rotation and/or translation is defined relative to this frame. Figure shows the coordinate frames and transforms associated with each joint in the system. The overall transform from frame $p(i)$ coordinates to frame i coordinates for a tree joint is given by

$${}^i X_{p(i)} = {}^i X_{J_i} {}^{J_i} X_{p(i)} = X_J(i) X_L(i)$$

The transform $X_L(i)$ is a fixed link transform which sets the base frame J_i of joint i relative to $p(i)$. It may be used to transform spatial motion vectors from $p(i)$ to J_i coordinates. The transform $X_J(i)$ is a variable joint transform which completes the transformation across joint i from J_i to i coordinates. Similarly, the overall transform from frame $p(k)$ coordinates to frame k coordinates for a loop-closing joint is given by

$${}^k X_{p(k)} = {}^k X_{J_k} {}^{J_k} X_{p(k)} = X_J(k) X_{L1}(k)$$

An additional transform $X_{L2}(k)$ is defined from frame $s(k)$ coordinates to frame k coordinates and is given by

$$X_{L2}(k) = {}^k X_{s(k)}$$

Link and joint geometry data can be specified in a variety of different ways. The most common method is to use Denavit–Hartenberg parameters (original work [Denavit-Hartenberg 55]). However, standard Denavit–Hartenberg parameters are not completely general, and are insufficient for describing the geometry for a branched kinematic tree, or for a mechanism containing certain kinds of multi-DOF joints. A modified form of Denavit–Hartenberg parameters [Introduction to Robotics: Mechanics and Control] is used for single-DOF joints in some prominent book (including [Handbook of robotics] and many others). The parameters have been extended for branched kinematic trees [Modeling, Identification and Control of Robots] and closed-loop mechanisms.

Link inertias

The link inertia data consists of the masses, positions of centers of mass, and rotational inertias of each link in the mechanism. The inertia parameters for link i are expressed in coordinate frame i , and are therefore constants.

Joint models

The relationship between connected links is described using the general joint model of Roberson and Schwertassek [Dynamics of Multibody Systems]. For a kinematic tree or spanning tree on a closed-loop mechanism, an $n_i \times 1$ vector, \dot{q}_i , relates the velocity of link i to the velocity of its parent, link $p(i)$, where n_i is the number of degrees of freedom at the joint connecting the two links. For a loop-closing joint in a closed-loop mechanism, the relationship is between the velocity of link $s(i)$ (the successor) and the velocity of link $p(i)$ (the predecessor). In either case, the relationship is between the velocity of coordinate frames i and J_i .

....continue....

Kinematic tree

The dynamics of a kinematic tree is simpler, and easier to calculate, than the dynamics of a closed-loop mechanism. Indeed, many algorithms for closed-loop mechanisms work by first calculating the dynamics of a spanning tree, and then subjecting it to the loop closure constraints. This section describes the following dynamics algorithms for kinematic trees: the recursive [Newton–Euler](#) algorithm (RNEA) for inverse dynamics, the articulated-body algorithm (ABA) for forward dynamics, the composite-rigid-body algorithm (CRBA) for calculating the joint-space inertia matrix (JSIM), and two algorithms to calculate the operational-space inertia matrix (OSIM). Implementations of the first three can be found in most of robotic textbooks.

Kinematic Loops

The above algorithms apply only to mechanisms having the connectivity of kinematic trees, including unbranched kinematic chains. A final algorithm is provided later for the forward dynamics of closed loop systems, including parallel robot mechanisms. The algorithm makes use of the dynamic equations of motion for a spanning tree of the closed-loop system, and supplements these with loop-closure *constraint equations*. Three different methods are outlined to solve the resulting linear system of equations. Method 2 is particularly useful if $n \gg n^c$, where n^c is the number of constraints due to the loop-closing joints. This method offers the opportunity to use $O(n)$ algorithms on the spanning tree [Baraff 96]. The section ends with an efficient algorithm to compute the loop-closure constraints by transforming them to a single coordinate system. Since the loop-closure constraint equations are applied at the acceleration level, standard *Baumgarte stabilization* [Baumgarte 72] is used to prevent the accumulation of position and velocity errors in the loop-closure constraints

Algorithm 3.7 Algorithm to calculate loop-closure constraints

```

inputs: model , RNEA partial results
outputs: L, l
model data : NB, p(i), NJ, p(k), s(k), LB(i), XL1(k),
XL2(k), Kp, Kv
RNEA data :  $\Phi_i$ ,  ${}^iX_{p(i)}$ , vp(k), vs(k), ap(k)vp, as(k)vp

for i = 1 to NB do
    if p(i)  $\neq$  0 then
         ${}^iX_0 = {}^iX_{p(i)} {}^{p(i)}X_0$ 
    end if
    if LB(i)  $\neq$  null then
         ${}^0\Phi_i = {}^iX_0^{-1} \Phi_i$ 
    end if
end for
L = 0
for k = NB + 1 to NJ do
    i = p(k)
    j = s(k)
    while i  $\neq$  j do
        if i > j then
            Lk,i =  ${}^0\Phi_i$ 
            i = p(i)
        else
            Lk,j =  ${}^0\Phi_j$ 
            j = p(j)
        end if
    end while
    ae =  ${}^{s(k)}X_0^{-1} a_{s(k)}^{vp} - {}^{p(k)}X_0^{-1} a_{p(k)}^{vp}$ 
    ve =  ${}^{s(k)}X_0^{-1} v_{s(k)} - {}^{p(k)}X_0^{-1} v_{p(k)}$ 
    pe = x_to_vec ( ${}^{p(k)}X_0^{-1} X_{L1}^{-1}(k) X_{L2}(k) {}^{s(k)}X_0$ )
    lk = -ae - Kv ve - Kp pe
end for

```

Conventions in robotics

Beyond traditional kinematics description of systems of rigid bodies, the use of extensive notation of robotics is worthy of attention from specialists of different sectors, not last aerospace, since is much more complete and transparent, avoiding most of misunderstanding possibly arising in writing and reading equations of both kinematics and dynamics.

That's not a case since complexity and variety of frames used easily yield confusion and errors,

A fundamental distinction should be made on what's reference for quantity evaluation (namely relative speed or position of a point or frame) and the frame used for its description (components). This applies for vectors as well for any tensor.

Another relevant convention is the enumeration of links from 0 or 1 at the root up to tip. This convention is not strictly mandatory but just for making simpler schematics. Indeed when dealing with tree-like structures or multibody systems lose immediately its intuitiveness and advantage. Some author even use reverse bodies indexing (see Jain)

In robotics there're also universally used conventions like the use of one degree of freedom joint with axis aligned to z. This is indeed restrictive, but in the aim of avoiding errors, is a widely adopted convention.

Elementary transform sequence (ETS)

Elementary transforms are canonic rotations or translations about, or along, the x-, y- or z-axes. The amount of rotation or translation can be a constant or a variable. A variable amount corresponds to a joint. This approach is general enough to be able to describe any serial-link robot manipulator. For a branched manipulator we can use ETS to describe the connections between every parent and child link pair.

$${}^W\xi_E = \underbrace{{}^W\xi_0}_{\xi_B} \oplus {}^0\xi_1 \oplus {}^1\xi_2 \oplus \cdots \oplus {}^{N-1}\xi_N \oplus \underbrace{{}^N\xi_E}_{\xi_T}$$

$$T_1 \oplus T_2 \oplus T_3 \oplus T_4 \dots \oplus T_n$$

References: [Manipulator Differential Kinematics Part 1: Kinematics, Velocity, and Applications](#).

[Manipulator Differential Kinematics: Part 2: Acceleration and Advanced Applications](#)

Denavit Hartenberg conventions

Also Denavit Hartenberg is a line representation, and one of most diffused and used in robotics because minimal, basing on common normal between two line.

Rather than attaching reference frames to each link in an arbitrary fashion, in the Denavit–Hartenberg convention a set of rules for assigning link frames

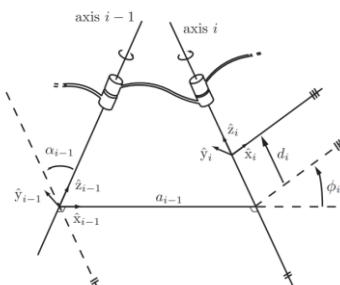


Figure C.1: Illustration of the Denavit–Hartenberg parameters.

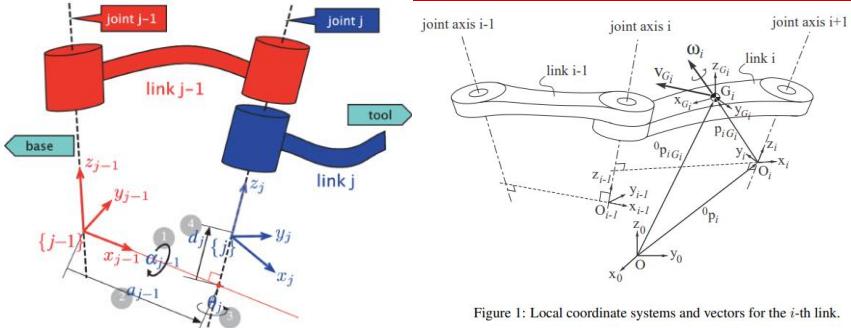


Figure 1: Local coordinate systems and vectors for the i -th link.

The first rule is that the $\hat{z}_i \hat{z}_i$ -axis coincides with joint axis i and the $\hat{z}_{i+1} \hat{z}_i$ -axis coincides with joint axis $i+1$. The direction of positive rotation about each link's \hat{z} -axis is determined by the right-hand rule. Once the \hat{z} -axis direction has been assigned, the next rule determines the origin of the link reference frame. First, find the line segment that orthogonally intersects both the joint axes $\hat{z}_{i+1} \hat{z}_i$ and $\hat{z}_i \hat{z}_i$. For now let us assume that this line segment is unique; the case where it is not unique (i.e., when the two joint axes are parallel), or fails to exist (i.e., when the two joint axes intersect), is addressed later. Connecting joint axes $i+1$ and i by a mutually perpendicular line, the origin of frame $\{i\}$ is then located at the point where this line intersects joint axis $i+1$. Determining the remaining \hat{x} - and \hat{y} -axes of each link reference frame is now straightforward: the $\hat{x}_i \hat{x}_i$ -axis is chosen to be in the direction of the mutually perpendicular line pointing from the $(i+1)$ -axis to the i -axis. The \hat{y} -axis is then uniquely determined from the cross product $\hat{x} \times \hat{y} = \hat{z}$. Figure depicts the link frames $\{i\}$ and $\{i+1\}$ chosen according to this convention. Having assigned reference frames in this fashion for links i and $i+1$, we now define four parameters that exactly specify $T_{i+1,i}$ and $T_{i+1,i}$:

- The length of the mutually perpendicular line, denoted by the scalar a_{i+1} , is called the link length of link $i+1$. Despite its name, this link length does not necessarily correspond to the actual length of the physical link.
- The link twist α_{i+1} is the angle from $\hat{z}_{i+1} \hat{z}_i$ to $\hat{z}_i \hat{z}_i$, measured about $\hat{x}_{i+1} \hat{x}_i$.
- The link offset d_i is the distance from the intersection of $\hat{x}_i \hat{x}_i$ and $\hat{z}_i \hat{z}_i$ to the origin of the link- i frame (the positive direction is defined to be along the $\hat{z}_i \hat{z}_i$ -axis).
- The joint angle θ_i is the angle from $\hat{x}_{i+1} \hat{x}_i$ to $\hat{x}_i \hat{x}_i$, measured about the $\hat{z}_i \hat{z}_i$ -axis.

Commented [5]: symbology

These parameters constitute the Denavit–Hartenberg (D–H) parameters. For an open chain with n one-degree-of-freedom joints, the $4n$ D–H parameters are sufficient to completely describe the forward kinematics. In the case of an open chain with all joints revolute, the link lengths a_{i+1} , twists α_{i+1} , and offset parameters d_i are all constant, while the joint angle parameters θ_i act as the joint variables. We now consider the cases where the mutually perpendicular line is undefined or fails to be unique, or where some of the joints are prismatic; finally, we consider how to choose the ground and end-effector frames.

$$(\alpha_i, a_i, d_i, \theta_i)$$

$${}^{i-1}T_i = \begin{pmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & r_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & r_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and its inverse remembering

$$T = \begin{pmatrix} R & p \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$T^{-1} = \begin{pmatrix} R^T & -R^T p \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$${}^iT_{i-1} = \begin{pmatrix} c\theta_i & s\theta_i & 0 & r_i \\ -s\theta_i c\alpha_i & c\theta_i c\alpha_i & s\alpha_i & -d_i s\alpha_i \\ s\theta_i s\alpha_i & -c\theta_i s\alpha_i & c\alpha_i & d_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^iT_{i-1} = \begin{pmatrix} c\theta_i & s\theta_i & 0 & r_i \\ -s\theta_i c\alpha_i & c\theta_i c\alpha_i & s\alpha_i & -d_i s\alpha_i \\ s\theta_i s\alpha_i & -c\theta_i s\alpha_i & c\alpha_i & -d_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Typical case includes those where only an offset and rotations around θ_i axis occur, leading to simplified transformation matrix and inverse

$${}^{i-1}T_i \Big|_{\substack{d_i=0 \\ \alpha_i=0}} = \begin{pmatrix} c\theta_i & -s\theta_i & 0 & r_i c\theta_i \\ s\theta_i & c\theta_i & 0 & r_i s\theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denavit-Hartenberg matrix can be written as product of two screw

Some considerations about DH parameters in special conditions

When Adjacent Revolute Joint Axes Intersect: If two adjacent revolute joint axes intersect each other then a mutually perpendicular line between the joint axes fails to exist. In this case the link length is set to zero, and we choose $\hat{x}_{i+1} \wedge \hat{x}_i$ to be perpendicular to the plane spanned by $\hat{z}_{i+1} \wedge \hat{z}_i$. There are two possibilities, both of which are acceptable: one leads to a positive value of the twist angle φ_i while the other leads to a negative value.

When Adjacent Revolute Joint Axes Are Parallel: The second special case occurs when two adjacent revolute joint axes are parallel. In this case there exist many possibilities for a mutually perpendicular line, all of which are valid (more precisely, a one-dimensional family of mutual perpendicular lines is said to exist). A useful guide is to try to choose the mutually perpendicular line that is the most physically intuitive and that results in as many zero parameters as possible.

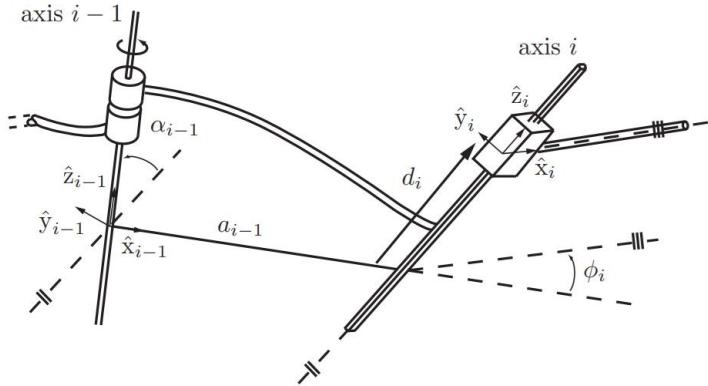
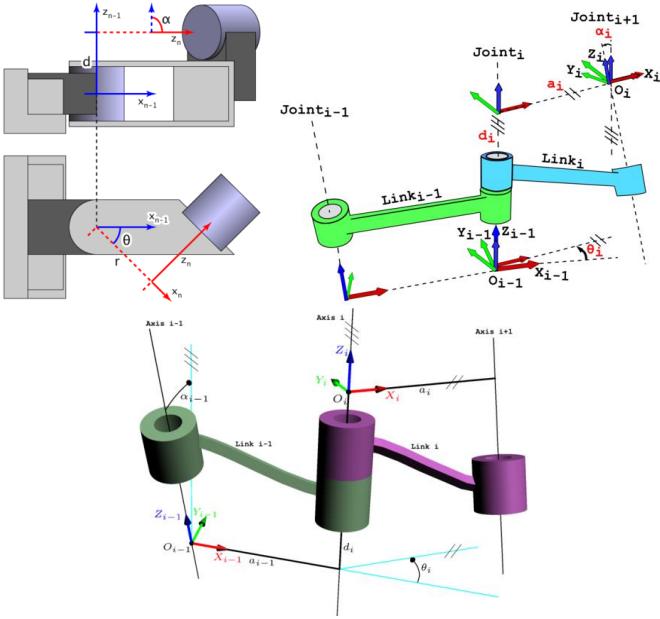


Figure C.2: Link frame assignment convention for prismatic joints. Joint $i - 1$ is a revolute joint, while joint i is a prismatic joint.

Assigning the Ground and End-Effector Frames: Our frame-assignment procedure described thus far does not specify how to choose the ground and final link frames. Here, as before, a useful guideline is to choose initial and final frames that are the most physically intuitive and that simplify as many D–H parameters as possible. This usually implies that the ground frame is chosen to coincide with the link-1 frame in its zero (rest) position; in the event that the joint is revolute this choice forces $a_0 = \alpha_0 = d_1 = 0$, while for a prismatic joint we have $a_0 = \alpha_0 = 1 = 0$. The end-effector frame is attached to some reference point on the end-effector, usually at a location that makes the description of the task intuitive and natural and also simplifies as many of the D–H parameters as possible (e.g., their values become zero). It is important to realize that arbitrary choices of the ground and end-effector frames may not always be possible, since there may not exist a valid set of D–H parameters to describe the relative transformation. We elaborate on this point below.

Semi transformations (inboard and outboard) and more details in frames propagation
The importance of a detailed frames systems definition usually occur later in studies, but for sake of precision we want to introduce it immediately. Older text books adopt just a frame per link assuming z axis in correspondence of joint axis. Modern textbooks prefer to choice of a more differentiated system of frames. Its role in propagation of kinematics (relative position and pose, speed and accelerations) is very important.



Denavit Hartenberg tables

Denavit hartenberg parameters are usually placed in table layout for increasing readability. Although strict order is not mathematically mandatory but neither commonly well respected, we suggest to keep under attention specific sequence, convention and order used for, usually specified. We report some example of tables used:

We want to remark some implicit assumption of this formalism: each frame is subsequent to preceding one. There's no need for each frame to be associated to a link, although most of cases is, so introduction of intermediate and support or virtual frames is possible, but specification is opportune, and sometimes provided. Table lines where free joint variables are used are not always, but should be indicated for clarity. That's because usually joint are reported for better explainability using a schematic representation. Otherwise implicitly, in absence of virtual frames, to each row a joint correspond, conventions impose aligned with axis z, and usually revolute type. Angles are expressed by choice in degree or radians, where unit of measure for lengths should be specified somewhere but not always indicated in table. Sometimes manufacturers integrate DH tables with further indications regarding relevant informations about links.

Link	a_i	α_i	d_i	θ_i
1	0	90°	0.0891	θ_1
2	-0.425	0°	0	θ_2
3	-0.392	0°	0	θ_3
4	0	90°	0.109	θ_4
5	0	-90°	0.0946	θ_5
6	0	0°	0.0823	θ_6

Differential kinematics equations

In robotics we can deal with different types of differential equations, relating different quantities, and expressing different types of relations among their derivatives.

Those related to geometrical constraints and their derivatives respect, for instance, time (one of most common case because of implications for motion), are known as differential kinematics equations. Given geometrical relationships between system point positions as function of free degree of freedom, expressed by means of generalized coordinates, we can get some relations among them and their derivatives

More generally we do not need of introducing any generalized coordinates, but still in vectorial and tensorial formalism we can define kinematics differential relations.

One simple case regard rotations and homogeneous transformation matrix derivative

Adjoint representation

Using 6x6 formalism we can express velocity relations between frames as

The 6×6 matrix which transforms twists from one coordinate frame to another is referred to as the *adjoint transformation* associated with g , written Ad_g . Thus, given $g \in SE(3)$ which maps one coordinate system into another, $Ad_g : R^6 \rightarrow R^6$ $Ad_g : \mathcal{R}^6 \rightarrow \mathcal{R}^6$ is given as

$$Ad_g = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix} \in \mathbb{R}^{6 \times 6}$$

A more extensive treatment of spatial motion notation and

$$Ad_T = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \in \mathbb{R}^{6 \times 6}$$

Applicative example

A simple example is given by 1 dof manipulator. Consider the one degree of freedom manipulator. The configuration of the coordinate frame B relative to the fixed frame A is given by

$$g(t) = \begin{pmatrix} \cos \theta(t) & -\sin \theta(t) & 0 & -l_2 \sin \theta(t) \\ \sin \theta(t) & \cos \theta(t) & 0 & l_1 + l_2 \cos \theta(t) \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The spatial velocity of the rotating rigid body is given by

$$V^s = \begin{bmatrix} v^s \\ \omega^s \end{bmatrix} = \begin{bmatrix} -\dot{R}R^T p + \dot{p} \\ (\dot{R}R^T)^\vee \end{bmatrix}$$

Kinematic differential equations

Application of time derivative to kinematic expressions and equations and inequalities allow the derivation of further constraints and relations valid for a study system. These differential equations are more general than dynamic differential equations and provide additional informations that are sometimes enough alone to determine motion solution.

In addition constitution of differential kinematics relations allows for mapping kinematic quantities like angular and linear speed and acceleration to other quantities like rotation matrixes

Direct and inverse kinematics

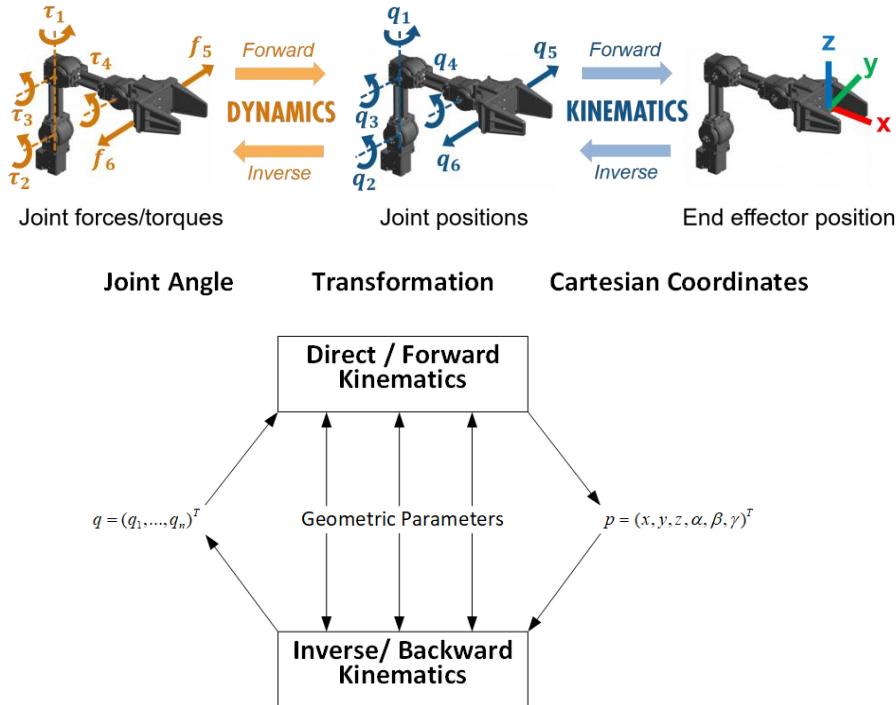
As said kinematics allows to define positions and speeds of any point as function of coordinates (joint positions) and their derivatives. This is forward kinematics.

A [reverse or inverse kinematics problem](#) can be defined as finding joint positions and speeds corresponding to specific positions and speed of system points, usually of one or many end effector. In kinematics forward problem is defined practically with a unique mapping from joint positions into spatial positions.

$$r(t_k) = r(q_1(t_k), q_2(t_k), q_3(t_k), \dots, q_n(t_k), t)$$

Inverse kinematics problems are often *not unique* and multiple solutions exist for a specified end effector. Sometimes solutions to inverse problems do not exist. That means kinematic mapping is not always bijective and invertible. Consequently choice of joint position requires

additional conditions for assuring a solution. The more degrees of freedom are involved, tendentially, the more solutions potentially exist .



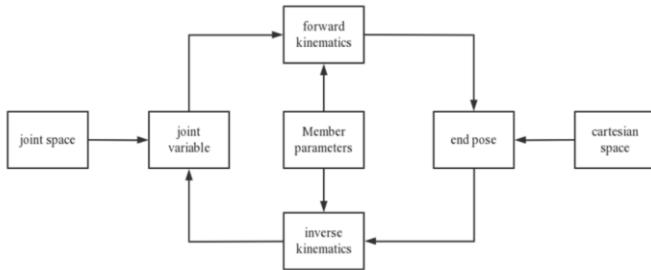
[Forward kinematics](#) problem are then solvable by direct calculation for instance using homogeneous transformations

$${}^0T_6 = {}^0T_1{}^1T_2{}^2T_3{}^3T_4{}^4T_5{}^5T_6$$

where inverse kinematics are solvable in closed form only for few degree of freedom or under special assumptions. In general case we need to solve nonlinear equations systems and resolve ambiguity for multiple solutions allowed.

$$\begin{aligned} q_1(t_k) &= f_1(r(t_k)) \\ q_2(t_k) &= f_2(r(t_k)) \\ &\vdots \\ q_n(t_k) &= f_n(r(t_k)) \end{aligned}$$

In following picture we propose an overview on connection between different topics in kinematics:



Forces and torques representations: wrench

A collection of forces and moments acting on a body can be reduced to a wrench \mathbf{f}_i at point O_i , which is composed of a force f_i at O_i and a moment m_i about O_i :

$$\mathbf{f}_i = \begin{pmatrix} f_i \\ m_i \end{pmatrix}$$

Note that the vector field of the moments constitutes a screw where the vector of the screw is f_i . Thus, the wrench forms a screw. Consider a given wrench ${}^i\mathbf{f}_i$, expressed in frame R. For calculating the equivalent wrench ${}^j\mathbf{f}_j$, we use the transformation matrix between screws such that:

$$\begin{pmatrix} {}^j\mathbf{m}_j \\ {}^j\mathbf{f}_j \end{pmatrix} = {}^j\mathbb{T}_i \begin{pmatrix} {}^i\mathbf{m}_i \\ {}^i\mathbf{f}_i \end{pmatrix}$$

equivalent to

$${}^j\mathbf{f}_j = {}^jR_i {}^i\mathbf{f}_i {}^j\mathbf{m}_j = {}^jR_i^i (\mathbf{m}_j + {}^i\mathbf{f}_i \times {}^iP_j)$$

It is often more practical to permute the order of \mathbf{f} and \mathbf{m} . In this case, equation becomes:

$${}^j\mathbf{f}_i = \begin{pmatrix} {}^j\mathbf{f}_j \\ {}^j\mathbf{m}_j \end{pmatrix} = {}^j\mathbb{T}_i^T \begin{pmatrix} {}^i\mathbf{f}_i \\ {}^i\mathbf{m}_i \end{pmatrix} = {}^j\mathbb{T}_i^T {}^i\mathbf{f}_i$$

Robot designation by means of string

A common way to represent a robot derives from Assur group mechanism representation. Kinematic structure of planar mechanisms addresses the study of attributes determined exclusively by the joining pattern among the links forming a mechanism. The system group classification is central to the kinematic structure and consists of determining a sequence of kinematically and statically independent-simple chains which represent a modular basis for the kinematics and force analysis of the mechanism.

A first distinction shall be made between structural representation and graph representation.

Structural representation describe mechanism in terms of

Graph representation just describe types of connections between links.

Commented [6]: Eliminate

Simply listing the type of joint in ordered sequence of presence in the mechanism is enough to define and identify its overall architecture.

Unfortunately this is sufficient only for planar mechanism. For more general mechanism in tridimensional space we need to define also joint axis orientation in space. Nonetheless a very similar representation is used in robotics, assuming open chain mechanisms, for describing joints and wrists kinematics. For example:

- PPP.. describes a robot composed of prismatic joints, possibly a cartesian robot
- RRRR.. describe a robot composed of revolute joints, possibly an articulated manipulator
- RRPR could describe a SCARA robot

This description is particularly useful for wrists/shoulder descriptions, where relative joint displacement is null or small and have little effect on kinematics, respect links kinematics.

Completing with additional information allowing to define relative orientation of joint axis allow for a short description of robot kinematics.

Symbolic Kinematics

Symbolic manipulation of models is helpful in defining kinematics equations can be used for determining frames position and orientation beyond end effectors, valid for any point of the system considered of interest. These include naturally joint frames, center of mass and body frames used in dynamic considerations as well for robot encumbered space.

Using symbolic tools and symbolic parameters it is easy to derive general expressions for transformation matrices that, assembled and combined together using sequence rules, lead to desired kinematic expressions.

Using time dependent symbolic parameters allow for automatization of derivation and differentiation of expressions in vectorial or matrix form straightforwardly, obtaining differential kinematics equations in first and second or higher order, used for instance in speeds and accelerations definitions.

Since used parameterization is absolutely arbitrary, different choices are possible and can be adopted. The use of symbolic expressions for kinematics can be much useful for general study avoiding simulations, and reusable for different purposes, like operative workspace exploration

The use of symbolic manipulation results anyway in expressions often too long and complex for manual inspection even for a limited number of degrees of freedom, in the order of tens. Suggestion is to use and introduce whenever possible intermediate variables able to simplify kinematic (and dynamic models). Indeed the number of terms is prone to increase factorially.

Numerical kinematics or hybrid numericoo-symbolic kinematics is a possible mitigation but limits meanwhile model explainability and generality, is the key strength of the method, together with versatility and potential of synthesis.

For instance for a planar serial robot of arbitrary number of joints, we can easily derive a series like expression, leading to a simple variable transformation, that's also absolutely natural.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \sum_{k=1}^N \begin{bmatrix} l_k \cos(\sum_{i=1}^k \theta_i) \\ l_k \sin(\sum_{i=1}^k \theta_i) \\ 0 \end{bmatrix}$$

$$q_k = \sum_{i=1}^k \theta_i$$

Same formula results from direct application of transformation matrices for joint in sequence (also possible in symbolic)

$${}^0T_n = {}^0T_1 {}^1T_2 \dots {}^{k-1}T_k \dots {}^{n-1}T_n$$

$${}^{i-1}T_i = \begin{pmatrix} cq_i & -sq_i & 0 & L \\ sq_i & cq_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Extending to more general case we just need to apply sequentially joint DH transformation matrix

In robotics conventional use of relative position for joints make easy automatization of the derivation process but not necessarily enhance efficiency and simplicity of the resulting model.

From analytical expression easily differential kinematics equations are obtained by automatic or manual differentiation. This open to Jacobian automatic calculation based on analytical depression differentiation is quite useful in many cases.

In the same way as analytical jacobian is usually concealing underlying structure under expression complexity, can be unveiled by geometric approach, that's anyway with opportune customisation of symbolic algorithms be implemented and managed using formal methods.

The advantage of this modified symbolic method is considerable and consist in:

- Simpler and more geometrically and physically meaningful expressions, without giving up mathematical functional dependency and symbolic treatment
- Easier symbolic management and lower computational burden, speeding up otherwise complex and long operations
- Natural collection and factorization of involved terms useful in analysis and interpretation of both kinematics and dynamics.

On the other side this is not necessarily automatically managed by most of symbolic manipulation and calculus suite diffused in community, requiring potentially some algorithms

of formal manipulation customization and specific work effort for managing equations and expressions properly.

Algorithms involved are rather simple and general, allowing to manage every robot system with short codes lines. For instance every body frame transformation to global frame is given by:

Algorithm	Recursive computation of link transformations
	$\begin{cases} {}^{\mathbb{I}}\mathbb{T}_{n+1} = \mathbf{I} \\ \mathbf{for} \ k = n \dots 1 \\ \quad {}^{\mathbb{I}}\mathbb{T}_k = {}^{\mathbb{I}}\mathbb{T}_{k+1} \cdot {}^{k+1}\mathbb{T}_k \\ \mathbf{end loop} \end{cases}$

Dynamics

Dynamics and dynamic model: Generalities

In this section we face the problem of relating mechanical systems accelerations and speeds to forces and torques acting on them. This is a classic problem in mechanics known as [dynamics](#). Traditionally in robotics dynamics is together with kinematics second principal topic to be considered. When dealing with a digital twin of a robot affording this topic is of most importance because allows for design and verification of robot actuation and feasible loads and motions. As result of [Newton law of mechanics](#) characterizing mechanical system dynamics, equations of motion relating position, speeds and acceleration to forces and torques acting is a second order differential equation respect time.

In mechanics and in robotics we rarely use the raw physical notions of quantities from base mechanics, but generalized concepts coming from [Lagrangian mechanics](#) (and used in [hamiltonian mechanics](#)) are used. So we can write for a generic system:

$$\tau = f(q, \dot{q}, \ddot{q}, f_e)$$

where we assume notation to be familiar to the reader. These are known also as equations of motion because relating motion to exerted solicitations. Under this formalism very general conclusion can be carried out. These equations constitute part of a dynamic model often essential part of a robot digital twin. This set of relations can be afforded in multiple ways, explicitly or implicitly. Generally speaking, and formally this is best perspective to adopt, we may know parts of motion and part of acting solicitation, and need to solve the problem. Obviously this a feasible problem only under specific assumptions and sufficient *informations*. This is known as *hybrid dynamics* problem. See [Rigid body dynamics algorithms] chapter 9 for an extensive introduction on the subject. In literature is more common to find distinction between *direct* (or *forward*) and *inverse* dynamics. Assuming known forces and torques acting on the system then solving for resulting motion over time or conversely assuming known motion and solving for resultant forces/torques are respectively direct and inverse dynamic problem. In robotics is easy to find both situations. For example, given a planned motion one can desire to know which solicitations act on the robot or for a given solicitation or actuation expected mechanism response. Remark in both case we need to have a sufficient degree of knowledge on:

- Acting forces
- Robot mechanical properties
- Environmental condition

Consequently any model is good only in measure of the validity of assumptions considered. You can observe as this problem setup do not impose any restriction on system structure, so it applies to closed chain, kinematic tree or any kind of mechanism. We'll see some methods applicable only to open chains like RNEA, often used in robots dynamics.

Dynamic modeling

Before investigating how solving for direct or inverse dynamic problem, a preliminary tasks required in both cases is the definition of relations existing between dynamic variables and

system inputs. This will be treated in the following. The other main point is the utility of direct and inverse dynamics. Hybrid dynamics problems are more complex to manage and had

The dynamic model of robots plays an important role in their design and operation. For robot design, the inverse dynamic model can be used to select the actuators [Chedmail 90b], [Potkonjak 86], while the direct dynamic model is employed to carry out simulations for the purpose of testing the *performance* of the robot and to study the relative *merits* of possible control schemes. Regarding robot operations, the inverse dynamic model is used to compute the actuator torques, which are needed to achieve a desired motion. It is also used to *identify* the dynamic parameters that are necessary for both control and simulation applications. Several approaches have been proposed to model the dynamics of robots [Renaud 75], [Coiffet 81], [Vukobratovic 82]. The most frequently employed in robotics are the *Lagrange formulation* [Uicker 69], [Khalil 76], [Renaud 80a], [Hollerbach 80], [Paul 81], [Megahed 84], [Renaud 85] and the *Newton-Euler formulation* [Hooker 65], [Armstrong 79], [Luh 80b], [Orin 79], [Khalil 85a], [Khosia 86], [Khalil 87b], [Renaud 87]. In this section, we present the dynamic modeling of serial robots using different formulations and principles. The problem of calculating a minimum set of inertial parameters will be covered in detail in other sections. The study of the minimization of the number of operations of the dynamic model in view of its real time computation for control purposes, is not specifically a subject of this work, but have much relevance in literature and in applications. Where possible results will be generalized for tree structured and closed chain robots for sake of clarity and completeness, since nowadays as before robots with unusual configurations are increasingly common.

Direct dynamic model

The direct dynamic model describes the joint accelerations in terms of the joint positions, velocities and torques as result of acting solicitations. The computation of the direct dynamic model is employed to carry out simulations for the purpose of testing the robot performances and studying the synthesis of the control laws. During simulation, the dynamic equations are solved for the joint accelerations given the input torques and the current state of the robot (joint positions and velocities). Through integration of the joint accelerations, the robot trajectory is then determined. Although the simulation may be carried out offline, it is interesting to have an efficient direct dynamic model to reduce the simulation time. In literature many different methods have been considered, but some are more commonly found. General form of forward dynamic problem can be written as (for specific dynamic model)

$$\ddot{q} = FD(q, \dot{q}, f_e, \tau) \quad \ddot{q} = FD(model, q, \dot{q}, f_e, \tau)$$

There are two main approaches for solving the forward dynamics problem for a kinematic tree:

- form an equation of motion for the whole system, and solve it for the acceleration variables (known as Inertia Matrix method) or
- propagate constraints from one body to the next in such a way that the accelerations can be calculated one joint at a time (Propagation Method or ABA Articulated Body Algorithm)

Usually cannot be directly applied in symbolic form (usually inversion is prohibitive, unless some conditions apply).

Two most common methods adopted for are:

First is based on using a *specialized* Newton-Euler inverse dynamic model and needs to compute the *inverse of the inertia matrix M* of the robot; the second method is based on a recursive Newton-Euler algorithm that does not explicitly use the matrix M and has a computational cost that varies linearly with the number of degrees of freedom of the robot.

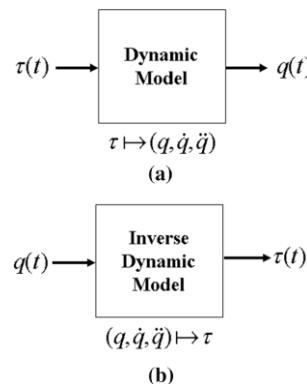
Direct dynamics problem requires then to carry out an ordinary differential equation (ODE) of second order and its solution for given boundary values (initial conditions), with a non trivial and hard to quantify numerical or analytical effort.

Inverse dynamics problems

An inverse dynamic model defines the system input as a function of the output variables. General form of inverse dynamics can be written as (for a given dynamic model):

$$\tau = ID(q, \dot{q}, \ddot{q}) \quad \tau = ID(model, q, \dot{q}, \ddot{q})$$

Inverse dynamics is usually simpler to carry out, and most of algorithms and methods effectively carry out equations in shape suitable for a direct application of inverse dynamics, where direct dynamics requires operations not always easy like inversion of inertia matrix. On the other side we can see the usual form of dynamic equations is implicit in coordinates and derivatives and explicit in applied inputs. Indeed inverse dynamics just requires expression evaluation, which effort can be deterministically quantified. In fact many algorithms of dynamic computation based on inverse dynamics provides forecasts of implied burden by formulae.



Equations of motion derivation

A lot of literature has been spent in discussing the most efficient and suitable way to derive a multibody system equation of motion, with special regard for serial open chain robots.

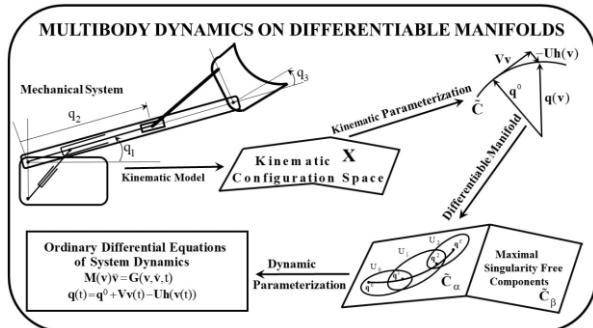
Essentially all methods have pros and cons to be considered. A more detailed illustration on literature in regard can be found in [Other methods for derivation of systems dynamics](#). Topic would deserve an extensive dissertation, especially in consideration of multibody dynamics, but for sake of brevity, we limit to fundamental notions.

Symbolic treatment of dynamics

For reasons that do not concern with digital twins and dynamic model development, symbolic treatment of dynamics account for a long legacy in robotics. Since dynamics of robots is highly non linear, classic control methods failed to perform satisfactorily, leading to control laws based on feed forward contributes and dynamic model (computed torque law), requiring a fine modeling of robot dynamics and inertial parameters identification, with additional constraints of limited resources for running typical of embedded systems. This problem was well known since very beginning of robotics, and was immediately faced, and solved using RNEA or more often Eulero Lagrange method.

The automatic symbolic computation of these models has largely been addressed in the literature [Dillon 73], [Khalil 76], [Zabala 78], [Kreuzer 79], [Aldon 82], [Cesareo 84], [Megahed 84], [Murray 84], [Kircinski 85], [Burdick86], [Izaguirre 86], [Khalil 89a]. [Khalil et al 97], which deals with all the above-mentioned models. The modeling of robots in a systematic and automatic way requires an adequate method for the description of their structure. Several methods and notations have been proposed [Denavit 55], [Sheth 71], [Renaud 75], [Khalil 76], [Borrel 79]. The most popular among these is the Denavit-Hartenberg method [Denavit 55]. This method is developed for serial structures and presents ambiguities when applied to robots with closed or tree chains. For this reason, we will use the notation of Khalil and Kleinfinger [Khalil 86a], which gives a unified description for all mechanical articulated systems with a minimum number of parameters.

General scheme of the process of deriving dynamic model is sketched in figure



Picture from [Haug 21]. First step is the definition of system kinematics and configuration space, with relative parametrization, then dynamic parametrisation occurs and lead to dynamics set of equations. This last for robotic systems as well for any multibody system.

Motion of a mass particle on a variety

We can start from particle motion recalls. The study of the motion of a mass particle on a differential variety is an easy way to figure out how dynamics of a system develops under

constraints, and how introduction of generalized coordinates arise naturally from embedding one particle motion in a differential manifold. Imposing a set of constraints on one particle motion is equivalent to set an equation introducing curvilinear coordinates on a constrained motion variety.

$$r = r(q_1, q_2, \dots, q_n, t)$$

Extension to many particle systems and then to rigid body and multibody dynamics is not straightforward but consequential.

Virtual displacements

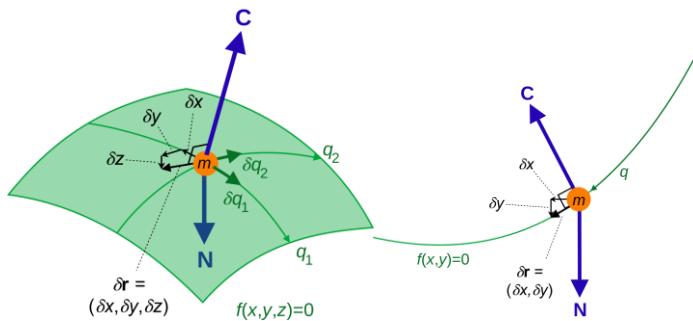
Virtual displacements is a key concept in lagrangian mechanics, and basically defines most of the properties deriving from. Virtual displacements are essentially point displacement compatible with constraints (holonomic) and represent essentially not real displacements but potential system motion. The displacement should lie on a tangent bundle of constraint variety at time instant considered, as a frozen and not moving. Indeed general displacement can be written as

$$dr_i = \sum_{k=1}^N \frac{\partial r_i}{\partial q_k} dq_k + \frac{\partial r_i}{\partial t} dt$$

just by setting $dt=0$ and imposing a set of arbitrary and *independent* variation of coordinates δq

$$\delta r_i = \sum_{k=1}^N \frac{\partial r_i}{\partial q_k} \delta q_k$$

See [Virtual displacements](#) for deepening..



Virtual works principle

Virtual work principle is one of most fundamental, and early principles used in mechanical system dynamics treatment, after Newton's laws, is probably second for importance. It just considers the product (scalar) of systems acting forces and torques by a (vectorial) quantity

called virtual displacement, that's just any possible displacement every point of the system can realize respecting imposed system constraints, as explained above. The product of virtual displacement with forces acting has dimension of a work and is called indeed *virtual work*. A null virtual work results from forces equilibria or from orthogonality with respect virtual displacements (*ideal constraints*). The constraint force delivers zero power along every direction of velocity freedom that is compatible with the motion constraints. (*Jourdain's principle of virtual power*).

D'Alembert principle

The use of D'Alembert principle in mechanics and in robotics is not too common, but is one of most direct and straightforward way to simplify equations of motion derivation exploiting projection realized with virtual works principles, avoiding explicit constraints relations and without losing equation for effectively acting forces. This way one can eliminate many of equations of the system related to vincular reaction, with significative simplifications of the model. It is convenient and useful to rederive the equations of motion without explicitly solving for the instantaneous constraint forces acting on the system. In essence, this is equivalent to projecting the motion of the system onto the feasible directions and ignoring the forces in the constrained directions, provided constraints are ideal. In doing so, we will be able to get a more concise description of the dynamics which is in a form well suited for closed-loop control. In addition it is rather intuitive since it is applied directly to physical forces and torques, without necessarily introducing through concepts like kinetic energy and potential function or generalized coordinates, speeds and forces, since direct work definition application through (not speed equivalent!) virtual displacement concept, is used. Projection of equations of motion onto the linear subspace generated by the null space of constraint equations eliminates most of the forces and torques acting on the system, allowing ignoring them and simplifying all. Results are called Lagrange-D'Alembert equations. As other work based methods it applies as well to closed chain robots or mechanisms as well to open chains, without restriction or additional treatments. Nonetheless the ideal constraint assumption, although somewhat limitative from a kinematic point of view, could be integrated in a second time with ad hoc corrections, provided we can express reactions and friction forces and torques work as function of generalized coordinates. Since the underlying idea is the same as the Eulero Lagrange method, often the second one is used, although much less intuitive, in case of non ideal constraint or in presence of non -holonomic constraints. On the other hand, D'Alembert principle intuitive nature and direct applicability could be very useful in understanding robot mechanics, although less suitable for a direct calculator of equations of motions, since all inertia forces shall be evaluated in a rather manual and difficult to automate way. On this side Kane's equations that are similar to D'Alembert principle but applied on quasi velocities, are more effective and general. In fact, it is also known as Lagrange version of D'Alembert principle. In textbooks literature, D'Alembert principle is studied specifically by few authors, as in [A Mathematical Introduction to Robotic Manipulation].

Quasi Velocities

More generally when dealing with non-integrable velocity constraints, we use quasi-velocities in place of ordinary generalized coordinates, that are indeed not derivable. Generally speaking quasi velocities can be obtained also as combinations of velocities and generalized

coordinates. One simple and remarkable example of are angular speeds. Basing on quasi-velocities we can write both a modified lagrangian set of equations or use a different set of equations derived specifically from d'alembert principle.

Quasi-Coordinates

Since quasi-velocities correspond to non-integrable to generalized coordinate quantities, we shall define a different concept for their time integral, named quasi-coordinates. In case a specific relationship exists between quasi-coordinates and generalized coordinates, it shall be a gradient function, relationship shall exist between, leading to a conservative vector field, thus a jacobian exists. Technically speaking, a necessary and sufficient condition for the existence of the quasi-coordinates, φ , is that the Riemannian manifold defined by the inertia matrix $M(q)$ be locally flat—by definition, a Riemannian manifold that is locally isometric to Euclidean manifold is called a locally flat manifold. However, that has been proven to be a very stringent condition .The fact is that the integration of quasi-velocities, in general,does not lead to quasi-coordinates. For details see [Introduction To Lagrangian Dynamics](#)

Euler-Lagrange equations

Second method mostly used in literature is the well known Euler-Lagrange method. It requires the calculator of Lagrangian function of the system, as difference between kinetic energy function T and potential function V .

$$L = T - V$$

These expressions are functions of generalized coordinates introduced precedently, as expression of mechanical constraints of the system. For serial robots there's also the advantage of allowing an explicit sequential formulation of kinetic energy that reveals to be precious in some applications like identification. We can propagate kinematics in kinematic tree from root to tip and compute each link rotational and translational speed leading to

$$T = T_t + T_r = \sum_{k=1}^n \left(\frac{1}{2} v_k^T m_k v_k + \frac{1}{2} \omega_k^T I_k \omega_k \right)$$

In this sense one can intend Euler-Lagrange method as an inverse dynamics method, because one need to define motion in order to find kinetic and potential energy. In addition kinetic and potential functions are essentially additive, and allows for decomposing system dynamics in single links contributes. General form of Euler-Lagrange equations comes out to be, including generalized external forces and nonholonomic constraints:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_k} \right) - \frac{\partial L}{\partial q_k} = Q_k + \sum_{e=1}^s \lambda_e a_{ek}$$

$$\sum_{k=1}^{N_{dof}} a_{ek} \dot{q}_k + a_{et} = 0$$

$$Q_k = \sum_{i=1}^n F_i \frac{\partial r_i}{\partial q_k}$$

To be solved using constrained optimization methods.

Note as the role of these constraint terms could be interpreted as additional generalized forces introduced in order to respect imposed constraints. Indeed one can increase the number of constraints respect those embedded in the choice of 5 generalized coordinates, using Lagrange multipliers method.

$$\underbrace{\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i}}_{\text{The "ordinary" Euler-Lagrange equations}} = \sum_j \lambda_j \frac{\partial f_j}{\partial q_i}$$

Sum over all of the constraints, labeled by the index j

Lagrange multipliers, which we have one of for each constraint

Derivative of each constraint function (f , which is a function of the coordinates)

With constraints included, the right-hand side of the Euler-Lagrange equation is no longer zero; instead, we have these constraint force terms, one for each constraint

Contrary to Newton Eulero (see next sections) method it doesn't require the calculation of all forces and torques acting on each single link, reducing considerably the number of operations as well the number of terms in equations to manage. Indeed the use of energy criterion together projection on virtual displacement associated to free degree of freedom of the system, as in D'Alembert principle, allow significant simplifications. Calculation of scalar functions like potential and kinetic energy is usually simpler than vectorial quantities.

- There's no need for a backward loop of forces calculations, since forward kinematics is enough for energy calculation
- Allow equations of motion components to be calculated
- In addition EL equations are applicable to general systems, both open or closed chains, in essentially same way
- could be also applied to integrable non holonomic system (pure rolling condition)
- is more suitable for assembling equations since different components could be introduced independently by their own nature (like potential driving forces or other generalized forces)

Nevertheless there are also some concerns that make it not always most suitable for applications of general kind.

- Requires differentiation and derivation respect time, coordinates and generalized speeds, then lagrangian shall be expressed coherently as function of

- Do not allow for calculation of reaction forces and torques
- In case non conservative forces are applied, is not always easy to manage them (although in many cases possible), because shall be expressed as function of generalized coordinates
- Less intuitive conceptually

System equivalence in lagrangian mechanics

From the theory of Lagrangian and hamiltonian mechanics we see how many different systems have the same dynamics equations, regardless of their real mechanical nature. Nonetheless it is possible by means of an opportune set of transformations to convert different mechanical systems to the same set of dynamical equations. This method is much more powerful and used in hamiltonian mechanics where canonical transformation theory is well developed and applied method for mechanics study and simplification. On the other side more basic examples are found also in lagrangian mechanics. One above all is the case of the general form of robot equations, presenting a common structure allowing a very general study. Opportune choices of coordinates allows for reconducting many systems to same or equivalent mathematical structure can be managed and solved with unique methods, Idea of automatizing these transformations is not new at all, and had many precursors, especially by means of support of symbolic calculus programs, but in the end most of attempts failed because mathematical expertise required where most of times well beyond merely algebraic manipulation, and finally out of horizon of automatic symbolic manipulation, making it impractical for general purpose.

The choice of coordinates

Apparently, when dealing with coordinate choice, it is easier to define and use relative coordinates between links, because more intuitive and easier to write. On the other hand, when writing equations we need to keep in mind dynamics and motion equations evaluate inertial forces (disregarding projection frame, maybe body, absolute or whatever) shall anyway be calculated with respect an *inertial frame*. It makes preferable where possible to use an inertial or absolute reference for them, although this choice may result in some case unnatural or more difficult to manage, or to automatise, but leading to simpler equations. A simple example is a classic planar robots with multiple joints, where it is easy to see how each joint coordinate playing inside kinetic energy terms as well inside kinematic relations, naturally considers the sum of relative angles and equates an global reference angle

Nevertheless in other less simple configuration, like in articulated robots, choice may be less simple and naturally arising by itself and mathematical passage, but at same time still not impossible. The use of joint coordinate is then a conventional method that simplify and automate equations writing for both kinematics and dynamics, reducing errors, but not necessarily lead to better equations shape.

Coordinate transformations

Given system dynamic equations with a set of coordinates, is always possible under Lagrangian formalism to perform a change of variables or coordinates transformations using

a specific mapping (not necessarily invertible in the whole domain of definition) f . Consequently Jacobian of transformation f is defined as square matrix

$$p = f(q)$$

$$J_{ij}(q) = \frac{\partial p_i}{\partial q_j} = \begin{pmatrix} \frac{\partial p_1}{\partial q_1} & \dots & \frac{\partial p_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial q_1} & \dots & \frac{\partial p_n}{\partial q_n} \end{pmatrix}$$

Following sequence of equations give the proof of equivalence and relative formulas.

$$p = f(q)$$

$$\Rightarrow (q, \dot{q}, \ddot{q}) \rightarrow (p, \dot{p}, \ddot{p})$$

$$\dot{p} = \frac{\partial f}{\partial q} \dot{q} = J(q) \dot{q}$$

$$\dot{q} = J(q)^{-1} \dot{p}$$

$$\ddot{p} = J(q) \ddot{q} + \dot{J}(q) \dot{q}$$

$$\ddot{q} = J(q)^{-1} (\ddot{p} - \dot{J}(q) J(q)^{-1} \dot{p})$$

$$J(q)^{-T} M(q) J(q)^{-1} p + J(q)^{-T} (N(q, \dot{q}) - M(q) J(q)^{-1} \dot{J}(q) J(q)^{-1} \dot{p}) = u_p$$

$$M_p(p) \ddot{p} + c_p(p, \dot{p}) + g_p(p) = u_p$$

Eventually equations shape remain the same and transformation just lead to Jacobian dependent transformations of dynamic matrices, indicated with pedix p as follow:

$$M_p = J(q)^{-T} M(q) J(q)^{-1}$$

$$c_p = J^{-T} (c - M J^{-1} \dot{J} J^{-1} \dot{p}) = J^{-T} c - M_p \dot{J} J^{-1} \dot{p}$$

$$g_p = J(q)^{-T} g$$

For more details see also [Lagrangian dynamics] from prof. A. De Luca.

Possible applications for online regression

In most of case one use an analytical expression of regressor in purpose to evaluate at samples dependance of torques on inertial parameters. This procedure could work both offline or online, but requires knowledge of regressor expression. On the other hand, just illustrated structure of equations rely only on kinematic quantities that depend on geometry of the system one can evaluate numerically without passing through a closed form analytical expression (remarkably that was also newton euler basic idea for online and real time applications to be used for control purposes). Similarly we can get online formulation of regressor and use it to perform a real time parameters estimate, because one just needs to know values of kinematic quantities in body frame, from direct measurements or from propagation, and relative jacobians. See for instance [Slotine-Li 87], [Yuan 95].

Euler equations

We go now for a presentation of a classic in mechanics of key importance for robot dynamics. Euler equations describe rotational equations of motion(that can be decoupled from translational motion, as proved by Konig theorem) expressing it in the body frame. That's of most importance since inertial quantities like the inertia tensor are constant only in the body frame, and would be complex to define dynamics with respect any other frame. Using covariant derivative for rigid body k of inertia tensor J_k we write

$$n_k = J_k \dot{\omega}_k + \omega_k \times J_k \omega_k$$

where ω_k is its angular speed vector in body frame and n_k acting external torques in body frame. Complete link dynamics includes uncoupled translational motion dynamic equations in the body frame, because projection becomes coupled. In compact form using matrices is possible to write Euler equations for a generic rigid body in body frame in the form:

$$\begin{pmatrix} m_k I & 0 \\ 0 & J_k \end{pmatrix} \begin{pmatrix} \dot{v}_k \\ w_k \end{pmatrix} + \begin{pmatrix} \omega_k \times m_k v_k \\ \omega_k \times J_k \omega_k \end{pmatrix} = \begin{pmatrix} f_k \\ n_k \end{pmatrix}$$

Remarkably this shape predisposes the use of twist and wrench formalism used in some advanced methods, like SOA, spatial vectors and Lie groups. We can do the same for all bodies of a mechanical system and express relations (connections) between bodies as transmitted forces, in place of kinematic relations, but we need to rotate every time form one body frame to other one for a correct projection. This approach is used in RNEA (Newton Euler) method for automatic calculation of equations of motion.

Recursive Newton Euler Algorithm

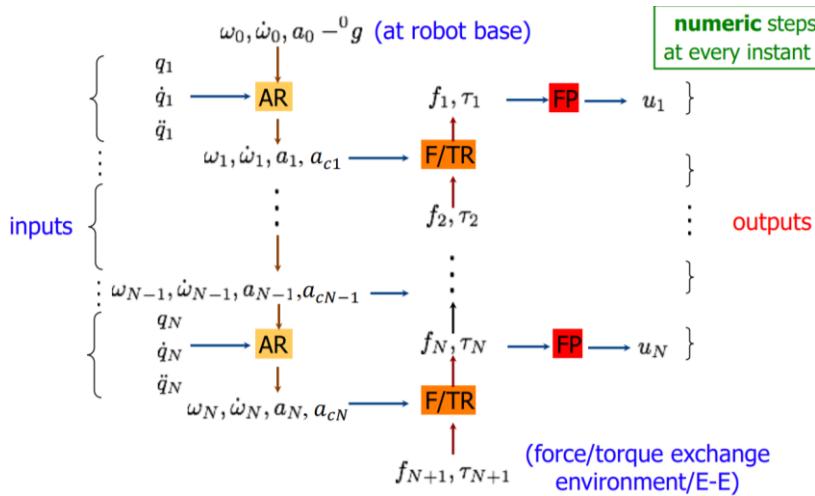
First method usually presented in literature for determination of dynamics equations is Newton Euler method. Reason of its success,

Original delineation of the method is due to Luh and Walker, [Luh et al. 80].

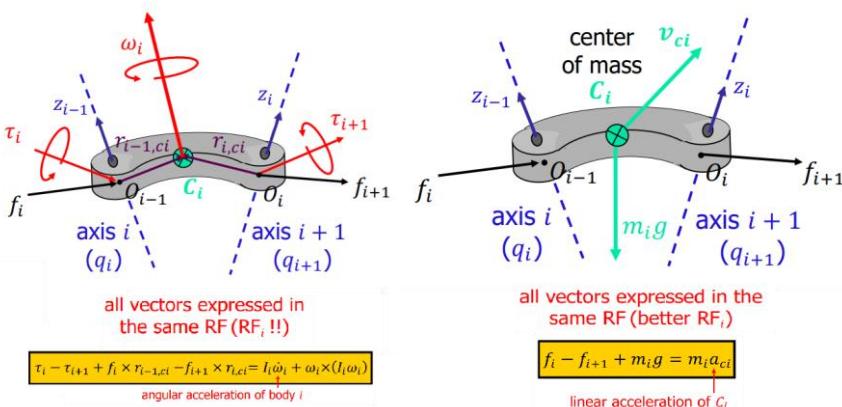
This, as others, is an inverse dynamics method that determines wrenches action on each link based on kinematic propagation. method is then two pass based:

- Forward kinematic propagation
- Backward forces and torques calculation

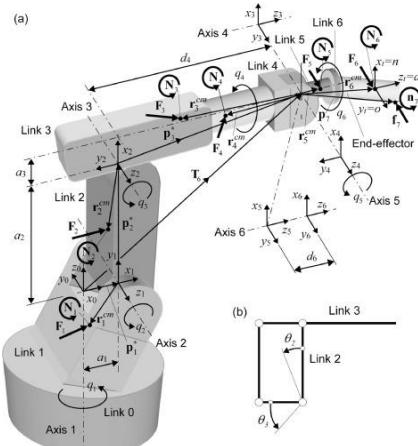
The method is recursive in the sense that it follows links order from root to tip and from tip to root, and is essentially algorithmic, so it's suitable for automatisation and computer implementation. Then is named RNEA Computational scheme is detailed as follow:



Forces and torques propagations from chain end(s) backward to the root base, shall consider body frame rotations, every link inertial forces and torques and torques/forces deriving from followers links, nonetheless all external forces and torques action on single link (gravity include).



Picture from [Newton-Euler approach]. As underlined by the author, different algorithm variants can exist, depending on application, substantially equivalent and differing for minor details. An extensive review of the topic is found in [Newton-Euler approach] and related works [De Luca 09]. In addition, as underlined in other sections, the type of kinematics representation (DH parameters, twists, PoE, SOA, transformation matrices, or others) used affects implementation of the algorithm. See [Handbook of robotics] chapter 3., [Robot and Multibody Dynamics: Analysis and Algorithms], [Rigid body dynamics algorithms], [Modern Robotics: Mechanics, Planning, and Control] and [A Mathematical Introduction to Robotic Manipulation] for different implementations. To make clear in your mind



[Handbook of robotics] presents different algorithmic implementations for the algorithms, from most traditional to most recent.

Algorithm 3.1 Coordinate-free recursive Newton-Euler algorithm (RNEA) for inverse dynamics

```

 $v_0 = \mathbf{0}$ 
 $a_0 = -\mathbf{a}_g$ 
for  $i = 1$  to  $N_B$  do
     $v_i = v_{p(i)} + \Phi_i \dot{q}_i$ 
     $a_i = a_{p(i)} + \Phi_i \ddot{q}_i + \dot{\Phi}_i \dot{q}_i$ 
     $f_i = I_i a_i + v_i \times I_i v_i - f_i^c$ 
end for
for  $i = N_B$  to 1 do
     $\tau_i = \Phi_i^T f_i$ 
    if  $p(i) \neq 0$  then
         $f_{p(i)} = f_{p(i)} + f_i$ 
    end if
end for

```

Algorithm 3.2 Recursive Newton–Euler algorithm using spatial vectors

inputs: $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \text{model}, {}^0\mathbf{f}_i^c$

output: τ

model data : $N_B, \text{jtype}(i), p(i), X_L(i), I_i$

```

 $\mathbf{v}_0 = \mathbf{0}$ 
 $\mathbf{a}_0 = -\mathbf{a}_g$ 
for  $i = 1$  to  $N_B$  do
   $X_J(i) = \text{xjcalc}(\text{jtype}(i), q_i)$ 
   ${}^iX_{p(i)} = X_J(i) X_L(i)$ 
  if  $p(i) \neq 0$  then
     ${}^iX_0 = {}^iX_{p(i)} {}^{p(i)}X_0$ 
  end if
   $\Phi_i = \text{pcalc}(\text{jtype}(i), q_i)$ 
   $\dot{\Phi}_i = \text{pdcalc}(\text{jtype}(i), q_i, \dot{q}_i)$ 
   $\mathbf{v}_i = {}^iX_{p(i)} \mathbf{v}_{p(i)} + \Phi_i \dot{\mathbf{q}}_i$ 
   $\zeta_i = \dot{\Phi}_i \dot{\mathbf{q}}_i + \mathbf{v}_i \times \Phi_i \dot{\mathbf{q}}_i$ 
   $\mathbf{a}_i = {}^iX_{p(i)} \mathbf{a}_{p(i)} + \Phi_i \ddot{\mathbf{q}}_i + \zeta_i$ 
   $\mathbf{f}_i = I_i \mathbf{a}_i + \mathbf{v}_i \times I_i \mathbf{v}_i - {}^iX_0^{-T} {}^0\mathbf{f}_i^c$ 
end for
for  $i = N_B$  to 1 do
   $\tau_i = \Phi_i^T \mathbf{f}_i$ 
  if  $p(i) \neq 0$  then
     $\mathbf{f}_{p(i)} = \mathbf{f}_{p(i)} + {}^iX_{p(i)}^T \mathbf{f}_i$ 
  end if
end for

```

Algorithm 3.3 Recursive Newton–Euler algorithm in 3-D vectors, for revolute joints only

inputs: $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \text{model}$

output: τ

model data : $N_B, p(i), \mathbf{R}_L(i), {}^{p(i)}\mathbf{p}_i, m_i, \mathbf{c}_i, \bar{I}_i^{\text{cm}}$

```

 $\omega_0 = \mathbf{0}$ 
 $\dot{\omega}_0 = \mathbf{0}$ 
 $\dot{\mathbf{v}}'_0 = -\dot{\mathbf{v}}'_g$ 
for  $i = 1$  to  $N_B$  do
   ${}^i\mathbf{R}_{p(i)} = \text{rotz}(q_i) \mathbf{R}_L(i)$ 
   $\omega_i = {}^i\mathbf{R}_{p(i)} \omega_{p(i)} + \hat{z}_i \dot{q}_i$ 
   $\dot{\omega}_i = {}^i\mathbf{R}_{p(i)} \dot{\omega}_{p(i)} + ({}^i\mathbf{R}_{p(i)} \omega_{p(i)}) \times \hat{z}_i \dot{q}_i + \hat{z}_i \ddot{q}_i$ 
   $\dot{\mathbf{v}}'_i = {}^i\mathbf{R}_{p(i)} (\dot{\mathbf{v}}'_{p(i)} + \dot{\omega}_{p(i)} \times {}^{p(i)}\mathbf{p}_i$ 
   $+ \omega_{p(i)} \times \omega_{p(i)} \times {}^{p(i)}\mathbf{p}_i)$ 
   $f_i = m_i (\dot{\mathbf{v}}'_i + \dot{\omega}_i \times \mathbf{c}_i + \omega_i \times \omega_i \times \mathbf{c}_i)$ 
   $\mathbf{n}_i = \bar{I}_i^{\text{cm}} \dot{\omega}_i + \omega_i \times \bar{I}_i^{\text{cm}} \omega_i + \mathbf{c}_i \times f_i$ 
end for
for  $i = N_B$  to 1 do
   $\tau_i = \hat{z}_i^T \mathbf{n}_i$ 
  if  $p(i) \neq 0$  then
     $f_{p(i)} = f_{p(i)} + {}^i\mathbf{R}_{p(i)}^T f_i$ 
     $\mathbf{n}_{p(i)} = \mathbf{n}_{p(i)} + {}^i\mathbf{R}_{p(i)}^T \mathbf{n}_i + {}^{p(i)}\mathbf{p}_i \times {}^i\mathbf{R}_{p(i)}^T f_i$ 
  end if
end for

```

Clearly any of these formalism is suitable as well as any other, although benefits of alternative notations go beyond specific dynamics equations and shorter or longer notations.

Using spatial vector notation one can obtain a short form of Newton Euelr algorith easy to remember and to apply. See also [Robot and Multibody Dynamics: Analysis and Algorithms] and [Jain lectures], for details. First step is kinematic propagation over kinematic tree

$$\mathcal{V}(k) = \Phi^*(p(k), k)\mathcal{V}(p(k)) + H^*(k)\dot{\theta}$$

$$\alpha(k) = \Phi^*(p(k), k)\alpha(p(k)) + H(k)^*\ddot{\theta}(k) + a(k)$$

Then backward passage from tip ends to root

$$f(k) = \Phi(k, p(k)^-) f(p(k)^-) + M(k)\alpha(k) + b(k)$$

Finally body wrench is projected onto the joint axis by means of hinge map.

$$\tau(k) = H(k)f(k)$$

Remarks we've used here p for predecessors of link k and p^- for corresponding followers for each link k . Using this notation, Jain notation indexing from tip to root progressively, inverse respect classic one from root to tip, does not matter, even because in kinematic tree strict indexing order loses sense.

Newton Euler algorithm customization

The recursive Newton-Euler formulation of robot dynamics has become a standard algorithm for real time control and simulation, but also in other fields. To increase the efficiency of the Newton-Euler algorithm, we generate a customized symbolic model for each specific robot and utilize the *base dynamic parameters*. To obtain this model, we expand the recursive equations to transform them into scalar equations without incorporating loop computations. The elements of a vector or a matrix containing at least one mathematical operation are replaced by an *intermediate variable*. This variable is written in the output file, which contains the customized model [Kanade 84], [Khalil 85a]. The elements that do not contain any operation are not modified. We propagate the obtained vectors and matrices in the subsequent equations. Consequently, customizing eliminates multiplications by one (and minus one) and zero, and additions with zero. A good choice of the intermediate variables allows us to avoid redundant computations. In the end, the dynamic model is obtained as a set of intermediate variables. Those that have no effect on the desired output, the joint torques in the case of inverse dynamics, can be eliminated by scanning the intermediate variables from the end to the beginning. The customization technique allows us to reduce the computational load for a general robot, but this reduction is larger when carried out for a specific robot [Kleinfinger 86a]. The computational efficiency in customization is obtained at the cost of a software symbolic iterative structure [Khalil et al. 97] and a relatively increased program memory requirement.

In comparison to other strategies attempted this one has been proved to be the most effective for

- Mass matrix direct calculation from jacobians (using geometrical terms as configuration dependent intermediate variables)
- Euler lagrange equations efficient writing (by means of same intermediate variables definition)
- Dynamic model calculation through Newton Euler method

Base inertial parameters for simplifications

It is obvious that the use of the base inertial parameters (introduced in later sections) in a customized NewtonEuler formulation that is *linear in the inertial parameters* will reduce the number of operations because the parameters that have no effect on the model or have been grouped are set equal to zero.

By general robot, we mean:

- the geometric parameters r_i , d_i , a_i , α_i and r_n are zero (this assumption holds for any robot);
- the other geometric parameters, all the inertial parameters, and the forces and moments exerted by the terminal link on the environment can have an arbitrary real value;
- the friction forces are assumed to be negligible, otherwise, with a Coulomb and viscous friction model, we add n multiplications, 2n additions, and n sign functions.

Historical overview on methods

Before describing applicable method, and specifically one based on customized symbolic Newton-Euler formulation linear in the inertial parameters [Khalil 87b], [Kleinfinger 86a], we present a short review the main approaches in the literature. The *Lagrangian formulation* of Uicker and Kahn [Uicker 69], [Kahn 69] served as a standard robot dynamics formulation for over a decade. In this form, the complexity of the equations precluded the real time computation of the inverse dynamic model. For example, for a six degree-of-freedom robot, this formulation requires 66271 multiplications and 51548 additions [Hollerbach 80]. This led researchers to investigate either simplification or tabulation approaches to achieve *real time* implementation. The first approach *towards simplification is to neglect the Coriolis and centrifugal terms* with the assumption that they are not significant except at high speeds [Paul 72], [Bejczy 74]. Unfortunately, this belief is not valid: firstly, Luh et al. [Luh 80b] have shown that such simplification leads to *large errors* not only in the value of the computed joint torques but also in its *sign*; later, Hollerbach [Hollerbach 84a] has shown that the velocity terms have the same significance relative to the acceleration terms whatever the velocity. An alternative simplification approach has been proposed by Bejczy [Bejczy 79]. This approach is based on an exhaustive term-by-term analysis of the elements H_{ij} , C_{ijk} and Q_i so that the most significant terms are only retained. A similar procedure has been utilized by Armstrong et al. [Armstrong 86] who also proposed computing the elements H_{ij} , C_{ijk} and Q_i with a *low frequency rate* with respect to that of the servo rate. Using such an analysis for a six degree-of-freedom robot becomes very laborious and tedious.

Two methods based on a trade-off between memory space and computation time have been investigated by Raibert [Raibert 77]. In the first method *SSM (State Space Model)*, the table was carried out as a function of the joint positions and velocities (q and \dot{q}), but the required memory was too big to consider in real applications at that time. In the *Configuration Space Method (CSM)*, the table is computed as a function of the joint positions. Another technique, proposed by Aldon [Aldon 82] consists of tabulating the elements A_{ij} and Q_i and of calculating the elements C_{ijk} on-line in terms of the A_{ij} elements. This method considerably reduces the required memory but increases the number of on-line operations, which becomes proportional to n^3 . We note that the tabulated elements are functions of the load inertial parameters, which means making a table for each load. Luh et al. [Luh 80a] proposed to determine the inverse dynamic model using a Newton-Euler formulation (§ 9.5). The *complexity of this method* is $O(n)$. This method has proved the importance of the recursive computations and the organization of the different steps of the dynamic algorithm. Therefore, other researchers tried to improve the existing Lagrange formulations by introducing recursive computations. For example, Hollerbach [Hollerbach 80] proposed a new recursive Lagrange formulation with complexity $O(n)$, whereas Megahed [Megahed 84] developed a new recursive computational procedure for the Lagrange method of Uicker and Kahn. However, these methods are less efficient than the Newton-Euler formulation of Luh et al. [Luh 80a]. More recently, researchers investigated alternative formulations [Kane 83], [Vukobratovic 85], [Renaud 85], [Kazerounian 86], but the recursive Newton-Euler proved to be computationally more efficient. The most efficient models proposed until now are based on a *customized symbolic Newton-Euler formulation* that takes into account the particularity of the geometric and inertial parameters of each robot [Kanade 84], [Khalil 85a], [Khalil 87b], [Renaud 87]. Moreover, the use of the *base inertial parameters* in this algorithm reduces the computational cost by *about 25%*. We note

that the number of operations for this method is even less than that of the tabulated CSM method.

In front of much work focused on the problem of computational efficiency of the inverse dynamic model of robots in order to realize real time dynamic control. This objective is now recognized as being attained (Table 9.5).

Table 9.5. Computational cost of the inverse dynamic modeling methods

Method	Robot	General case		2RP(3R) Stanford		R(2P)(3R) TH8		6R Stäubli RX-90		
		Operations	n ddl	n=6	General	Simplified*	General	Simplified*	General	Simplified*
Raibert 78**		Mult. Add.	$n^2 + 2n^2$ $3^n + n^2$	288 252	288 252	288 252	288 252	288 252	288 252	
Luh 80b		Mult. Add.	$137n - 22$ $101n - 11$	800 595	800 595	800 595	800 595	800 595	800 595	
Hollerbach 80**		Mult. Add.	$412n - 277$ $320n - 201$	2195 1719	2195 1719	2195 1719	2195 1719	2195 1719	2195 1719	
Kane 83**		Mult. Add.	?	?	646 394	?	?	?	?	
Vukobratovic 85**		Mult. Add.	?	?	?	>372 >167	?	>193 >80	?	
Renaud 85**		Mult. Add.	?	?	?	?	?	?	368 271	
Presented method Khalil [87b]		Mult. Add.	$92n - 127$ $81n - 117$	425 369	240 227	142 99	175 162	104 72	253 238	159 113

? the number of operations is not given.
* the matrix J_j is diagonal and two components of the first moments are zero.
** forces and moments exerted by the end-effector on the environment are not considered.
> the given number of operations corresponds to the computation of the elements of A , $C_{ij,k}$ and Q .

Flexible joints effects inclusion

Models including flexibilities lumped on joints can be wrote as a perturbation of rigid body motion when assuming, as realistic, that flexibility effects are of small entity.

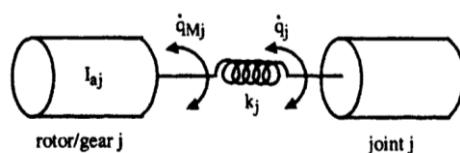


Figure 9.3. Modeling of joint flexibility

Introduction of flexibility effects are accounted for by introducing some auxiliary degree of freedom (see for instance Khalil). This approach allows for a closed form treatment using the same techniques already used for rigid body tree dynamics, also symbolically. It can be shown that treatment of flexibility as lumped on joints preserve equations structure

$$\tau = H(q)\ddot{q} + C(q, \dot{q})$$

$$\begin{pmatrix} \tau_r \\ \tau_f \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \begin{pmatrix} \ddot{q}_r \\ \ddot{q}_f \end{pmatrix} + \begin{pmatrix} C_r \\ C_f \end{pmatrix}$$

Where $H_{21} = H_{12}^T$

Assuming a value of joint elasticity we can write, respect a reference position q_0

$$\tau = -K\Delta q = -K(q - q_0)$$

In the case of a system with elasticity, the inverse dynamic model calculates the input torques and the elastic accelerations as a function of the joint positions, velocities and rigid joint accelerations. It is to be noted that the accelerations of the elastic variables cannot be specified independently. Using (68) to calculate the inverse model, we have first to calculate the acceleration elastic accelerations from the second row, then we can calculate the rigid joint torques from the first row.

Lins Flexibility body inclusion into equations of motion

Model for a robot with flexible joints the general form of the dynamic model is

$$\underbrace{\begin{pmatrix} \Gamma_r \\ \Gamma_f \end{pmatrix}}_{\Gamma} = \underbrace{\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}}_A \underbrace{\begin{pmatrix} \ddot{q}_r \\ \ddot{q}_f \end{pmatrix}}_{\ddot{q}} + \underbrace{\begin{pmatrix} H_r \\ H_f \end{pmatrix}}_H$$

where, Γ_r is the vector of rigid joint torques and Γ_f is the vector of flexible joint torques. If a joint, j is flexible then

$$\Gamma_{f_j} = -k_j \underbrace{(q_j - q_{r_j})}_{\Delta q_j}$$

where, k_j is the stiffness of the flexible joint and q_{r_j} is the reference joint position corresponding to zero elasticity force. From above for the inverse dynamic model we get

$$\begin{aligned} \ddot{q}_f &= A_{22}^{-1} (\Gamma_f - H_f - A_{12}^T \ddot{q}_r) \\ \Gamma_r &= [A_{11} \quad A_{12}] \begin{bmatrix} \ddot{q}_r \\ \ddot{q}_f \end{bmatrix} + H_r \end{aligned}$$

A three step recursive algorithm can be used for the calculation of the inverse dynamic model to calculate \ddot{q}_f and Γ_r as a function of q, \dot{q}, \ddot{q}_r or the direct dynamic model to calculate \ddot{q}_f and \ddot{q}_r in terms of Γ_r, q, \dot{q} .

A link flexibility full modeling requires to adopt coupling between different models. Usually specific programs can be used. Not all robotics softwares allow for flexibility effects inclusion.

Evaluation of flexibility effects on links requires the inclusion of elastic line differential equations or Timoshenko beam equations, making the system a [PDE](#) problem to solve with some methods. Simplifications of models are obtainable using modal decomposition approaches, discretization methods or reduced order models. Under these assumptions an [ODE](#) (Ordinary Differential Equation) or [DAE](#) (Differential-Algebraic system of equations) model as seen before can be obtained.

More details are found in [Khalil et al. 97], [Khalil 14],[Modeling, Identification and Control of Robots], [Robots with Flexible Joints], [Handbook of robotics],[Robot and Multibody Dynamics: Analysis and Algorithms].

Flexible beam effects inclusion

Academic research has worked for years also on the inclusion of flexibility effects consequent to deformations of robot joint to joint body considered traditionally as perfectly rigid, shall now include displacement coming from deformations, as depicted in figure.

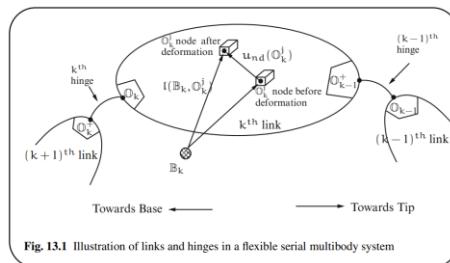
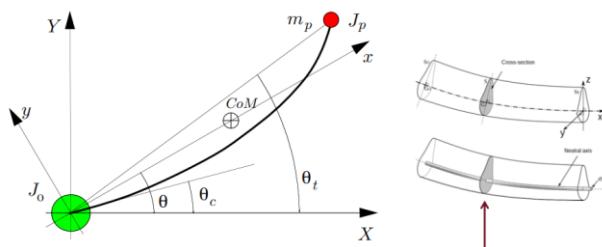


Fig. 13.1 Illustration of links and hinges in a flexible serial multibody system

Most used model is elastic line theory for slender beam (length \gg section dimensions) under small deformations, due to pure bending also known as Eulero Bernoulli theory. In this theory displacement is assumed to occur on a horizontal plane, excluding torsion and compression and shear deformation also. Extensions for the neutral beam axis (Timoshenko theory) are usually negligible.



In truth slender beam assumptions are not perfectly satisfied in many cases, but is anyway a higher order approximation. Some other issues are related to section homogeneity that is usually considered a valid hypothesis, is not necessarily true for robot segments, and could lead to some deviations. Small deformations and bending assumptions are on the other side rather realistic, unless long slender arms are considered, or lightweight materials used to save mass/costs, when no further care is used. Rigidity hypothesis indeed fail in this case, and when payload to weight is large, in case of high speed motion, large loads from environment interaction are exchanged, and when control bandwidth is increased. Some typical applicative cases are manipulation in the space field, underwater, underground, automated crane applications. Ignoring or neglecting flexibility can lead to control limitations in steady state (static) and dynamic (vibrations, poor tracking) conditions, stability issues due to non colocation between input and output (non minimum phase systems).

One way to extend treatment to multiple flexible segments is introducing any finite-dimensional discretization for $w_i(x_i, t)$

$$w_i(x_i, t) = \sum_{j=1}^{n_{eq}} \varphi_{ij}(x_i) \delta_{ij}(t)$$

the Lagrangian is given in terms of L+M generalized coordinates, with (flexible variables) and satisfies to

$$L = T - V = L(\theta_i, \delta_{ij}, \dot{\theta}_i, \dot{\delta}_{ij})$$

being τ_i the torque delivered by the actuator at joint i.

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i} = \tau_i$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\delta}_{ij}} \right) - \frac{\partial L}{\partial \delta_{ij}} = 0$$

the general dynamic model (with modal damping) is then given by (with blocks of suitable sizes)

$$\begin{pmatrix} M_{\theta\theta} & M_{\theta\delta} \\ M_{\delta\theta} & M_{\delta\delta} \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{\delta} \end{pmatrix} + \begin{pmatrix} c_\theta \\ c_\delta \end{pmatrix} + \begin{pmatrix} g_\theta \\ g_\delta \end{pmatrix} + \begin{pmatrix} 0 \\ D\dot{\delta} + K\delta \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \end{pmatrix}$$

or in the more compact form using usual equations for discrete systems

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) + \begin{pmatrix} 0 \\ D\dot{\delta} + K\delta \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \end{pmatrix}$$

as in the rigid case, the vector of centrifugal/Coriolis terms can be factorized using the Christoffel symbols. The dynamics of flexible robots can be expressed in terms of a set of dynamic coefficients $a \in \Re^p$ that summarize the mechanical (rigid + flexible) properties of the links

$$Y(\theta, \delta, \dot{\theta}, \dot{\delta}, \ddot{\theta}, \ddot{\delta})a = \begin{pmatrix} \tau \\ 0 \end{pmatrix}$$

a linear parametrization is useful for the experimental identification of a .

possible choices of the assumed modes — i.e., the basis functions $\varphi_{ij}(x_i)$ for describing the bending deformation shapes of the links

- admissible functions satisfy only geometric b.c.'s
- comparison functions (Finite Elements, Ritz-Kantorovich expansion) satisfy also natural b.c.'s
- orthonormal eigenfunctions (links models as Euler-Bernoulli beams) lead to simplifications in inertia submatrix $M_{\delta\delta}$ (block diagonal, constant)

TBC

A first problem emerge when is required to detect relevance of flexibility

The problem results to be linear when only one link is flexible, where non linearities arise when more flexible links are considered.

One obstacle in the treatment of the problem is the different nature of elastic problem governed from PDE and rigid motion governed by ODE.

General results in dynamics

We present here some general results from dynamics that can be derived in different ways, but usually Euler-Lagrange method allow a more straight derivation. Using kinetic energy definition and applying relevant derivative we get:

$$K = \frac{1}{2} \dot{q}^T H(q) \dot{q}$$

$$\sum_{j=1}^n H_{ij} \ddot{q}_j + \sum_{j=1}^n \sum_{k=1}^n C_{ijk} \dot{q}_j \dot{q}_k + \tau_{gi} = \tau_i$$

where C_{ijk} are Christoffel symbols of the first type. We can see that inertial forces come and depend only on kinetic energy K and potential deriving forces acts separately.

The definition of coefficients C is

$$C_{ijk} = \frac{1}{2} \left(\frac{\partial H_{ij}}{\partial q_k} + \frac{\partial H_{ik}}{\partial q_j} - \frac{\partial H_{jk}}{\partial q_i} \right)$$

They are functions of only the position variables, since they're linear in velocities

$$C_{ij} = \sum_{k=1}^n C_{ijk} \dot{q}_k$$

However, C is not unique, and other definitions are possible. We can prove this result last for a large variety of mechanical systems, including closed chain, not connected mechanisms, moving base systems and system with integrable constraints in general, then to multibody systems, excluding only non integrable non holonomic constraints systems, where is anyway possible with some tricks to find a EL-like shape equation. Moreover with some adjustment is possible to include flexibility effects for small displacements as part of same structure with ease. Henceforth found results are quite general and applicable. Remarkably C-related terms depend at most on squares or products of generalized speeds. Observe as equations imply an implicit form for motion coordinates. Indeed we can consider it as an inverse dynamics form, where for given motion acting wrenches are calculated. The generalized forces structure highlights a first term proportional to accelerations, depending on inertia matrix and consequently on system configuration. Second term is related to the rate of change in inertia matrix respect configuration changes. Third term is related to potential variation respect coordinate changes. Finally external forces can produce work in correspondence to coordinate changes leading to right side term.

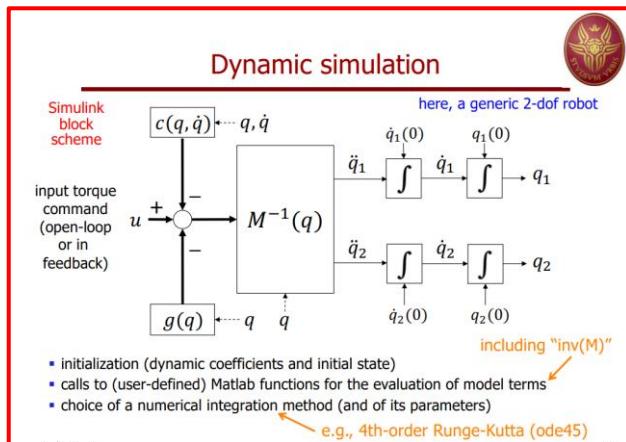
From this set we derive also an explicit formulation for direct dynamics as

$$\ddot{q} = H(q)^{-1} \left(\tau_e - \tau_g - C(q, \dot{q}) \dot{q} \right)$$

more usually rewritten as a first order system for purpose of numerical integration

$$\begin{pmatrix} \ddot{q} \\ \dot{q} \end{pmatrix} = \begin{pmatrix} H(q)^{-1} (\tau_e - \tau_g - C(q, \dot{q})\dot{q}) \\ \dot{q} \end{pmatrix}$$

In any case the model results to be highly nonlinear and coupled, then sometimes hard to solve numerically (usually closed form analytical solutions are known only for simplest cases). Remarkable inertia matrix is invertible because positive definite in consequence of kinetic energy definition. Nevermore its inversion is not easy analytically and in most of literature numerical inversion is used. On this regard see also [Robot and Multibody Dynamics: Analysis and Algorithms], [Rigid body dynamics algorithms] and [Rodriguez 87], [Rodriguez et al. 92]. Schemes of integration are diverse in literature, we just report for example one from [Lagrangian dynamics].



Not working parts of dynamics

This term C of Coriolis and centrifugal forces is important because it allows us to distinguish between terms that produce work from those that do not produce any work. Above illustrated choice of C , that is not unique, allow to achieve a proof of matrix N as skew symmetric

$$N(q, \dot{q}) = \dot{H}(q) - 2C(q, \dot{q})$$

Consequently we can use this criteria for checking equations correctness and to infer how related terms behaves against displacements, because for any given 1-vector α

$$\alpha^T N(q, \dot{q}) \alpha = 0$$

including $\alpha = \dot{q}$, consequently we can state that terms associated to N do not produce any work on the system:

$$\dot{q}^T N(q, \dot{q}) \dot{q} = 0$$

making them a useful check of consistency is introduced by property of dynamics shown, and find applications on controls too.

Considerations about the structure of equations of motion

We know that kinetic energy terms, related to inertial forces and terms of dynamics that put in place derivatives of coordinates of first and second degree, and are then those making equations becoming differential of second order (otherwise system of governing equations would be usually at most of first order due to speed related terms like viscous friction contributes), are defined by means of inertia matrix.
Consequently once inertia matrix of the system is known we can derive dynamics equations just applying Euler Lagrange method.
Developing all passages we derive following structure of equations depending on three fundamental dynamic matrixes.

$$\sum_{j=1}^n H_{ij} \ddot{q}_j + \sum_{j=1}^n \sum_{k=1}^n C_{ijk} \dot{q}_j \dot{q}_k + \tau_{gi} = \tau_i$$

First dynamic matrix is the inertia matrix itself and related all term depending on second order derivatives of generalized coordinates. These terms express changes of momenta (and consequently on energy) associated to change in generalized speeds.

Other terms consequence of the derivation of inertia matrix respect coordinates express the changes in momenta and energy due to system configuration changes causing a change in inertia. These terms consequently do not involve changes in generalized speed but just coupling of speeds pairs as couples. More precisely we can distinguish between terms depending on square of speed (centrifugal terms) and terms coupling different generalized (speeds associated to different axis in most common cases) known as coriolis terms, that are terms related to change in momenta relative to one coordinate due to change over time of another coordinate.

It is worthy to reason in a detailed way about the structure of these equations beyond what just said (that still last true for any equation of this kind).

Knowing the structure of inertia over the system and how it affects energy we can infer much more about the structure of mass matrix and other dynamic matrices

We know that energy function is build up as a sum of contributes each one deriving from one body mass and inertia.

Developing all calculation one arrives to a synthetic form equations on the shape of

Where all matrixes are determined as function of system mass and inertia distributions []

$$H_{ij}^k = \left\{ [\bar{\epsilon}_i \bar{\epsilon}_j \bar{\epsilon}_i R^{ik} I^k R^{ki} - \hat{c}^{ik} m^k \hat{c}^{jk} + \epsilon_i m^k \hat{c}^{ik} R^{ij}] + \epsilon_i R^{ij} (\epsilon_j m^k 1 - \bar{\epsilon}_j m^k \hat{c}^{jk}) u^j \right\} u^i$$

Summing up over k, we get the inertia matrix as function of rotations matrixes expressing orientation of body frames respect each other of the links, and orientation of actuation axis. In this formulation ϵ and $\underline{\epsilon}$ cannot be one together, and \hat{c}_{ij} are vectors of relative positions of mass centers respect joints.

Similarly for centrifugal terms we've

$$\beta_{ij}^k = \left\{ \bar{\epsilon}_i [\bar{\epsilon}_i (\bar{\epsilon}_i R^{ij} \hat{u}^j R^{jk} I^k R^{kj} - m^k \hat{c}^{ik} R^{ij} \hat{u}^j \hat{c}^{jk}) - \epsilon_i m^k R^{ij} \hat{u}^j \hat{c}^{jk}] u^j \right\} u^i$$

and finally for coriolis terms

$$\zeta_{ijn}^k = \left\{ \bar{\epsilon}_n [\bar{\epsilon}_i (\bar{\epsilon}_j (\bar{\epsilon}_k R^{ik} d\alpha (J^k R^{kn} u^n) R^{kj} - m^k \hat{c}^{ik} R^{in} \hat{u}^n R^{nj} \hat{c}^{jk}) + \epsilon_j m^k \hat{c}^{ik} R^{in} \hat{u}^n R^{nj}) + \epsilon_i R^{in} \hat{u}^n R^{nj} (\epsilon_j m^k 1 - \bar{\epsilon}_j m^k \hat{c}^{jk})] u^j \right\} u^i$$

This way is very precious since do not explicitly recall immediately to coordinates or components keeping vectorial expressions, probably the most convenient way to express succinctly physical and geometrical relations existing between each term. Exploding them for making explicit could quickly result in very long and cumbersome equations sets.

For more information about this formulation see [Bertrand-Bruneau 12].

Another prominent point is that as well as kinetic energy, dynamic matrices present a structure that depend linearly on inertial parameters, leading to a structure of equations of motion that depend on almost linearly dependent from them, inferring structure of equations of motions that once again are linear on inertial parameters, depend at most quadratically with integer powers from geometrical factors. Finally dependence on kinematic variables of speed and acceleration are integer power up to second order, where dependence from generalized coordinates could be expressed as trigonometric functions of combinations of subsets of generalized coordinates.

Topological effects on structure of motion equations

In addition to what was seen in the previous section, we should note the structure of motions equations is closely related to *system topology* as driving factor.

Topological descriptions of open chain mechanisms exploit some of the formalisms borrowed from *graph theory* as depicted on figure (from Jain), where nodes are links and connecting joints between bodies are edges respectively. Graph theory based representation usually neglect some relevant information of links and joints like inertia, mass, geometrical dimensions, as well as joint space orientations, focusing on connections between in a zero-dimensional perspective; we'll see of most importance, even such a minimal information set is decisive for dynamics structure. In spite of this basic description some fundamental deductions are already possible. Is as much important to understand there's a preferential

direction from tip to root base, where root is supposed to be constrained in motion to base values. Indeed we can see how each link motion is influenced by all graph follower bodies toward the root, and acting inertial forces arising as consequence of kinematic motion impact backward to each predecessor down to the root. This from force perspectives. Kinematically it works exactly in the opposite way, and each body position and speed depend on all preceding nodes and links kinematics. . We can say there's a causal chain on all branches of a tree backward from branches tips down to tree root, where kinematically forward causality relationship comes from propagation of motion.

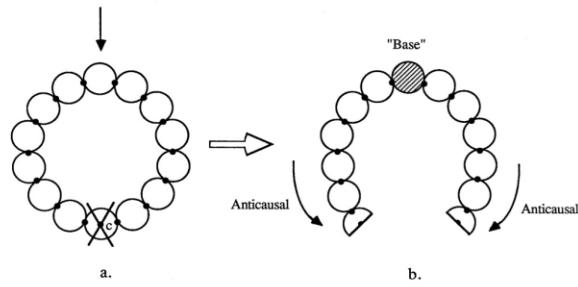
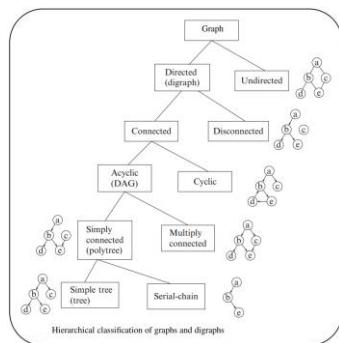


Figure 1: Closed Chain System

That means that forces and torques acting on bodies are only affected by children branches and body itself, then inertia matrices do not depend on nodes preceding node analyzed., as well kinematic quantities do not depend on follower nodes, and turns out inertia matrix dependency is of this type. Exact expression of this dependence is a function of many factors we ignored so far like joint axis orientation and inertial properties of each link as well on joint type (prismatic, revolute or helical).



Serial open chain mechanisms are the classic simple case. Each link dynamics have only one predecessor and one follower, its kinematics could depend only on self and predecessors kinematics, while its dynamic, or better acting wrenches, depend only on successors dynamics

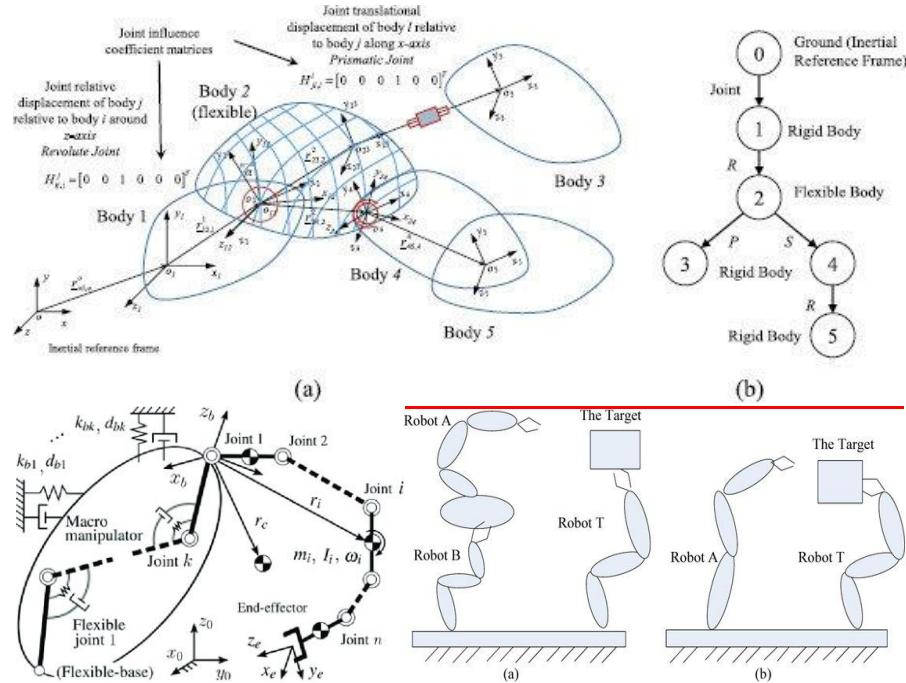
and applied wrenches. This allows the application of a simple recursive sequential algorithmic procedure for computing acting solicitations, like Newton Euler (RNEA) for computation of both kinematics and loads (in two steps, forward and backward).

By the way, even other branched open loop chains (or trees) rigid body systems can be analyzed using recursive methods as RNEA, although multiple recursion on all branches are required. Also in this case topology reflects and drives the final structure of the inertia matrix, and consequent or joint torques, whose structure is less obvious in this case.

In simple words all links of a branch kinematics depend on branch root node, and forces and torques of a branch root node are the sum of all child branches.

Remark as this recursive procedure highlights how we can imagine ideally any branch root link as a moving base for all branches, and every link as a moving base for followers links. Base motion can be treated as imposed (once solved) or treated as free floating composite base. Consequently we can extend further our speculation and imagine how each branch tip could become a base for secondary tree mounted on, **acting as an external load on end effector frame**.

For subject deepening see [Handbook of robotics], [Moving base robots](#), or [Space Robotics: Dynamics and Control].



Reasoning in terms of mechanical trees is a powerful tool since many of realizations used in robotics are suitably described based on this scheme. Some classic examples are robotic hands, humanoid robots, multiple arm manipulators, arthropods, legged robots and worm or snake shaped robots, revealing a very *convenient* balance between applicability and flexibility.

The sad truth is that complexity of models grows up quickly with (almost) exponential or power law trends, leading to unmanageable symbolic or analytic expression in few tens of degrees of freedom, leaving step to numerical methods and simplifications. It is possible to manage them in a complete way using some tricks like introduction of intermediate variables, and giving up for completely explicit symbolic expression, limiting to a formal or partial representation, or again using special geometry based factorisations.

An alternative way to reduce system complexity and increase its manageability recur to the definition and detection of some "main" bodies we can treat virtually as a (quasi-) fixed base or roots, on in other words, not significantly affected from downstream bodies mechanical forces effect, making them negligible, so that we can use them as "base" for a subtree reducing effectively working system degree of freedom and reducing analysis complexity to reasonable terms. Remark in some case, like humanoid robots, this is only partially true since each limb effectively depend dynamically only on downstream links motion.

If we think of humanoid robots and their limbs, one arm and its hand or gripper, or a base mounting multiple arms, it's clear the underlying reasoning and easing deriving. In this way results of dynamics and kinematics for a simpler and reduced number of degree of freedom can be used and reused for modeling system.

Alternative complexity reduction approaches

Here with topology we intend essentially the presence of connections between bodies, then the ability to transmit forces and torques, and consequently playing role in dynamics coupling bodies.

Since topology deserve a special place into building of dynamics even when we're not dealing with simple chain or trees, is worthy to include it directly into equations writing to get most general and easy to adjust formulation, highlighting similarities and connections between dynamics of systems differing for some limited changes.

In addition of topology intended as connections between links, we should also consider a second but not secondary factor, the nature of the joint and its orientation relative to parent and childs. Although this is not a strictly topological element, its anyway relevant for defining the final resulting dynamics.

Because of chasles theorem we know each rigid body motion could be described with just four factor, like denavit hartenberg parameters, or using twist or screw vectors.

Which one is not so fundamental, but affects both ease of understanding kinematics intuitively and treatment generality. What is important to underline is the point that traditionally in robotics we use the convention of assigning to each joint one single degree of freedom. This choice facilitate considerably problem treatment, but is limitative in some way if used uncarefully.

We suggest to use a chosen representation and apply it formally, letting all four parameters be parametric and not value fixed, and only on a later time choice which one is an effective joint variable and whose parameters. In this way is possible to get very general expressions easily adaptable to different configuration without losing generality. Of course in this way we get more complex expressions, but there are different advantages

- treatment of errors of alignment
- single treatment for different cases

We can use this approach also to allow complex joints with multiple degree of freedom. For instance without loss of generality we could use a screw as joint description assuming some

Reduction of complexity of equations: complexity considerations

Through many attempts and alternative approaches, in this work the hardness of closed form calculation of symbolic model, or similarly dynamic equations of motion, has revealed to be in many case a potentially limited task because of quickly growing number of terms and complexity of equations leading to

- slowing of calculations
- simplifications take long time and are often not optimized
- memory issues

Neither splitting equations in parts, for instance separating each link contribute lead to significative improvements. With traditional approaches, like EL and NEA, in few dof complexity rises at a level making then unmanageable in a single set of equations. Solution found by authors like Khalil and by writer himself consist in using intermediate variables in equations construction. That means we do not have anymore a set of n closed form equations but a larger set of, where expression of generalized forces, as well of kinematic and dynamic quantities are not fully explicit, but depend on intermediate variables introduced, and their effective evaluation imply a specialization to specific case. Resulting set appear to be very similar to a code branch. See also [Modeling, Identification and Control of Robots] for some examples.

Use of inertia matrix for determining free motion

As better illustrated in other sections of this work a first alternative to already presented methods for dynamics is the direct calculation of inertia matrix elements in joint space (JSIM) avoiding most of the passages leading to complete expression by means of a direct calculation with specific algorithms. Jacobians based method or CRBA or GDAHJ methods are viable ways. Indeed derivation of JSIM a posteriori from equations of motion can be a considerable burden task as much as more degrees of freedom are considered. Methods like accelerations factoring, specialized RNEA multiple runs ([Walker 82]), or jacobian of EOM, or even kinetic energy terms collection are time taking respect to a direct formulation even for few dof. Even a method based on substitution into EOM ([Modeling, Identification and Control of Robots]) using inverse dynamics is less efficient, requiring multiple evaluations. Another factor affecting its complexity is the alignment of joints axis. The more they're not aligned, the more expressions become more long and complex. This last fact does not change with adoption of different algorithms since it is related to kinematic and physics of structure itself.

From a mass matrix, the derivation of motion equations in symbolic form is rather straightforward and computationally simpler with respect to calculating of EOM using EL or RNEA and then deriving the inertia matrix. Moreover inertia matrices have properties like symmetry and recursivity whose exploitation allows for lower complexity and ease of check

out. Stratagem of adoption of intermediate variables can significantly simplify complexity of expressions using common factors, and can have further use, for instance in simplifying identification problems.

Simplifications of model

The control of a robot manipulator requires fast computation of its different models. An efficient method to reduce the computation time is to generate a symbolic customized model for each specific robot. To obtain this model, we expand the matrix multiplications to transform them into scalar equations. Each element of a matrix containing at least one mathematical operation is replaced by an intermediate variable.

The use of intermediate variables allows for great simplifications in terms of expressions length and time of computations, keeping them in reasonable limits and affordable with standard calculators commercially available.

Example of application of dynamics to simple robot cases: two link arms

Applying dynamics equations to a very simple 2 link planar arm including actuators inertia contributes one get inertia matrix as 2x2 matrix having components. The motors are located on the joint axes with centers of mass located at the origins of the respective frames.

Center of mass jacobians are for links l_1 and l_2 respectively:

$$J_P^{l_1} = \begin{pmatrix} -l_1 s_1 & 0 \\ l_1 c_1 & 0 \\ 0 & 0 \end{pmatrix} \quad J_P^{l_2} = \begin{pmatrix} -a_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ a_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 0 & 0 \end{pmatrix}$$

Resulting

$$\begin{aligned} M_{11} &= I_1 + m_1 l_1^2 + k_{r1}^2 + I_2 + m_2 (a_1^2 + l_2^2 + 2a_1 l_1 c_2) + I_{m2} + m_{m2} + a_1^2 \\ M_{12} &= M_{21} = I_2 + m_2 (l_2^2 + a_1 l_2 c_2) + k_2^2 I_{m2} \\ M_{22} &= I_2 + m_2 l_2^2 + k_2^2 I_{m2} \end{aligned}$$

Coriolis and centrifugal terms are:

$$\begin{aligned} c_{111} &= \frac{1}{2} \frac{\partial b_{11}}{\partial q_1} = 0 \\ c_{112} &= c_{121} = \frac{1}{2} \frac{\partial b_{11}}{\partial q_2} = -m_2 a_1 l_2 s_2 = h \\ c_{122} &= \frac{\partial b_{12}}{\partial q_2} - \frac{1}{2} \frac{\partial b_{22}}{\partial q_1} = h \\ c_{211} &= \frac{\partial b_{21}}{\partial q_1} - \frac{1}{2} \frac{\partial b_{11}}{\partial q_2} = -h \\ c_{212} &= c_{221} = \frac{1}{2} \frac{\partial b_{22}}{\partial q_1} = 0 \\ c_{222} &= \frac{1}{2} \frac{\partial b_{22}}{\partial q_2} = 0 \end{aligned}$$

in matrix form:

$$\begin{aligned} C(q, \dot{q}) &= \begin{pmatrix} h\dot{\theta}_2 & h(\dot{\theta}_1 + \dot{\theta}_2) \\ -h(\dot{\theta}_1 + \dot{\theta}_2) & 0 \end{pmatrix} \\ N(q, \dot{q}) &= \begin{pmatrix} 0 & -2h\dot{\theta}_1 - h\dot{\theta}_2 \\ 2h\dot{\theta}_1 + h\dot{\theta}_2 & 0 \end{pmatrix} = \begin{pmatrix} 2h\dot{\theta}_2 & h\dot{\theta}_2 \\ -h\dot{\theta}_2 & 0 \end{pmatrix} - 2 \begin{pmatrix} h\dot{\theta}_2 & h(\dot{\theta}_1 + \dot{\theta}_2) \\ -h\dot{\theta}_1 & 0 \end{pmatrix} \end{aligned}$$

gravity terms can be calculated using virtual works principle, Jacobian matrix (assuming it as static force) or as gradient of potential respect generalized coordinates, leading to, for a gravity vector of magnitude \mathbf{g} $(0 \ -g \ 0)^T$:

$$\begin{aligned} g_1 &= (m_1 l_1 + m_{m2} a_1 + m_2 a_1) g c_1 + m_2 l_2 g c_{12} \\ g_2 &= m_2 l_2 g c_{12} \end{aligned}$$

equations of motion results to be, in absence of friction terms:

$$\begin{aligned}
& (I_1 + m_1 l_1^2 + k_1^2 I_{m1} + I_2 + m_2 (a_1^2 + l_2^2 + 2a_1 l_2 c_2) + I_{m2} + m_{m2} a_1^2) \ddot{\theta}_1 + \\
& (I_2 + m_2 (l_2^2 + m_2 (l_2^2 + a_1 l_2 c_2) + k_2 I_{m2})) \ddot{\theta}_2 - \\
& 2m_2 a_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 a_1 l_2 s_2 \dot{\theta}_2^2 + (m_1 l_1 + m_{m2} a_1 + m_2 a_1) g c_1 + m_2 l_2 g c_2 = \tau_1 \\
& (I_2 + m_2 (l_2^2 + a_1 l_2 c_2) + k_2 I_{m2}) \ddot{\theta}_1 + (I_2 + m_2 l_2^2 + k_2^2 I_2) \ddot{\theta}_2 + m_2 a_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_2 = \tau_2
\end{aligned}$$

Addition of fractional terms is simple, since they act just as torques on respective joints (with opportune sign convention):

$$\boxed{
\begin{aligned}
\tau'_1 &= \tau_1 + f_{c_1} + f_{v_1} \dot{\theta}_1 + f_{v_1^2} \dot{\theta}_1^2 + \dots \\
\tau'_2 &= \tau_2 + f_{c_2} + f_{v_2} \dot{\theta}_2 + f_{v_2^2} \dot{\theta}_2^2 + \dots
\end{aligned}
}$$

$$\begin{aligned}
\tau'_1 &= \tau_1 + f_{c_1} \text{sign}(\dot{\theta}_1) + f_{v_1} \dot{\theta}_1 + f_{v_1^2} \dot{\theta}_1^2 + \dots \\
\tau'_2 &= \tau_2 + f_{c_2} \text{sign}(\dot{\theta}_2) + f_{v_2} \dot{\theta}_2 + f_{v_2^2} \dot{\theta}_2^2 + \dots
\end{aligned}$$

Some authors (Sciavicco) use a notations that distinguish between link inertial properties denoted with pedix l and actuator denoted with pedix m. in our opinion intent to include in notation distinction between different contributes is quite positive, but resulting expressions are by far heavier to read and write, considering that resulting quantities can be aggregated.

Extended denavit hartenberg tables

Denavit Hartenberg parameters are used for describing relations between two frames, usually associated to consecutive frames in a sequence, that are usually associated to subsequent links, one parent and a child. Nonetheless this applies not only to serial robots but also to general case of robots without additional constraints on joints (closure conditions for instance), like tree robots schemes, where a parent link is connected to one or more follower links. We can use extended tables to represent them, where we do not just specify denavit hartenberg parameters, but also origin frame.starting from. In this way we can represent general open robotic structures (a quite large family including complex shapes, like snakes, octopus). This approach is not different from the traditional way of describing tree robots where a vector of parents is given, usually by means of indexed links.

The use of extended denavit hartenberg tables arise quite naturally in description of a robot, since allowing collection of relevant information in a simple, ordered, schematic, and easy to read and compare tabular form.

Officially no formalized structure for this extension but we want to present our proposal.

Additional information regards origin frame, destination frame name/number, nature of the joint (revolute or prismatic), joint limits and range. Usually speed limits exist and are relevant for kinematics. Forces and torques limits exist as well, but regard more dynamics than kinematics, maybe in static, and could be included into.

In addition we can add different mechanical properties of the link relevant for other uses beyond kinematics, like for dynamics derivation like mass, center of mass vector, inertia tensor or vector.

The use of this formalism has been checked in this work and found worthy even against more commonly adopted formalisms like structures elsewhere commonly adopted. We can also note this formalism is compatible and integrable with one used by some authors for description of inertial parameters.

Remark when joint frame correspond to same link (multiple joint case), there's no need to repeat inertial properties (center of mass, inertia tensor).In case of serial chain no issue arise.

Link	Frame	α	a	d	ϑ	F_o	F	m	$m r_c$	I
Link 1		α_1	a_1	d_1	q_1					
...	

Link $n - 1$		α_{n-}	a_{n-1}	d_{n-1}	q_{n-1}		$n-1$			
Link n		α_n	a_n	d_n	q_n					

We also developed a method for converting between tabular to data structure formalism, in order to make this formalism compatible with other commonly used formalism, focused on links and grouping all relevant link informations to link data data structure.

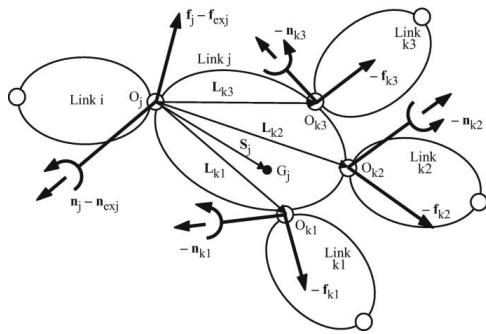
Further extensions of Denavit hartenberg tables

A further extension of DH parameters tables has been developed in order to manage more complex topologies like tree complex structure robots. For managing general structure we need to leave sequential indexing for a more complete way to define links and joints connections.

The use of extended tables allows to explicitly define both link frame index and parent frame explicitly, reducing ambiguities.

In addition each link can have more than one joint, henceforth for each link one set of DH parameters is not enough in order to define joint frames associated.

Indeed although connection of multiple links on the same joint location (but naturally associated joints need to be distinct), is possible, and maybe even not uncommon, for a kinematic tree is of course necessary, in the most general case, the presence of multiple joints (and locations) on the same link, connected to different child links, as illustrated in figure. All joints even when coincident in frame origin, can have different orientations, still following robotics convention. Typical way to generally represent multibody situations like this (even when flexible bodies are included, with reference to a undeformed shape), is as follow:



Accordingly we need to define for each link not a vector of parameters but a matrix of DH parameters, one 4-vector for each joint. Hereafter also transformation maps shall be defined based joint location and relative motion between joint frame inboard and outboard.

Customized symbolic derivation of equation of motion

A way to improve performances in deriving dynamic equations is by means of customized symbolic approach that considers specific problem geometrical elements

Using Euler-Lagrange equations make sometimes difficult to match the use of geometrical system elements appearing naturally from physical point of view, but later the need for differentiation respect generalized coordinates and time requires a customized differentiation based on system configuration. This work is well and detailedly illustrated in the work of Khalil [Khalil et al. 96].

Passages are rather long but not too complex.

Sensitivity and dynamic matrices, dynamic coefficient and identification

We've seen in detail how dynamic matrices are built and how different body inertial properties impact on each one of them, leading to characteristic dynamics structure illustrated in details in precedent paragraphs. These coefficients are often named as *dynamic coefficients* since they're product with motion kinematic quantities lead to dynamic loads acting. Now, under the light of just mentioned knowledge, we want to focalize our attention into a useful interpretation of dynamic matrices, that results to be very useful for better understanding of many topics relative mechanics, dynamics, and identification problems in robotics. We've seen how each element of these dynamic matrices is related to contributes of accelerations and speed, linearly or non linearly acting on torques as contributes. We can say that they represent some sort of sensitivity factor of torques with respect to a given acceleration or speed combination for a coriolis-centrifugal matrix.

Often it is worth thinking to them nullifying all other speeds or accelerations and considering one by one to be a set unitary value. Is clear that each element of dynamic matrices is in the end a sensitivity factor, at most depending on system mass and inertia configurations, leading. Once known torques, we could then imagine deriving our matrices just putting all terms to zero except those of interest, and in this way perform a sort of direct "identification". Unfortunately this is not always physically easy to realize due to kinematic relations existing between speeds, accelerations and coordinates in holonomic constraints making these conditions hard to realize, and meaningful mostly at calculus level. This process could then be realized only mathematically once known as full expression of joint torques, or numerically imposing quantities being zero or one as desired., and is essentially a trick for a direct calculation of dynamic matrices, used for instance for mass matrix numerical calculation. Sensitivity perspective is specially clarifying when we go to the regression problem, where the role of sensitivity factors are inverted, and regression coefficient becomes speed combinations that multiplied for the basic element of the inertial parameters vector to provide the torques for each joint. There's then a sort of duality among them, being the same expression rearranged in two different ways, although derivation is usually given by taking partial derivatives respect to parameters.

Inertia matrix

Definition of inertia matrix

Inertia matrix is fundamental element of kinetic energy definition and consequently the very element defining dynamics and inertial forces associated.

$$K = \frac{1}{2} \dot{q}^T M(q) \dot{q} = \frac{1}{2} \dot{q}_i^T M_{ij}(q) \dot{q}_j$$

There are alternative definitions for Inertia matrix we can use, one related to kinetic energy (classic and main used) and one related to the role in dynamic equations respect generalized accelerations:

$$M\ddot{q} = M_{ij}\ddot{q}_j$$

Two definitions are equivalent, as we can see looking at Euler-Lagrange equations, we have:

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} &= Q_i \\ \frac{d}{dt} \left(\frac{\partial}{\partial \dot{q}_i} \frac{1}{2} \dot{q}_i M_{ij}(q) \dot{q}_j \right) &= \frac{d}{dt} (M_{ij}(q) \dot{q}_j) = M_{ij}(q) \ddot{q}_j + \dot{M}_{ij} \dot{q}_j \\ \frac{\partial}{\partial q_i} \left(\frac{1}{2} \dot{q}_i M_{ij}(q) \dot{q}_j \right) &= \frac{1}{2} \dot{q}_i \frac{dM_{ij}(q)}{dq_i} \dot{q}_j = \frac{1}{2} \frac{dM_{ij}(q)}{dq_i} \dot{q}_i \dot{q}_j \\ \underbrace{\frac{d}{dt} \left(\frac{1}{2} \frac{\partial \dot{q}_i M_{ij} \dot{q}_j}{\partial \dot{q}_i} \right)}_{M_{ij}(q) \ddot{q}_j} - \underbrace{\frac{1}{2} \dot{q}_i \frac{\partial M_{ij}(q)}{\partial q_i} \dot{q}_j}_{\text{Coriolis-centrifugal terms}} &= Q_i \end{aligned}$$

We can see alternative definitions of Inertia matrix are equivalent and one lead to other one and both define eventually the complete dynamic parts related to inertia forces. In literature Christoffel symbols are used often to express in closed form resulting generalized inertia forces involved in coriolis and centrifugal terms

$$\begin{aligned} C_{ijk} &= \frac{1}{2} \left(\frac{\partial M_{ij}}{\partial q_k} - \frac{\partial M_{jk}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} \right) \\ C_{ij} &= \sum_{k=1}^n C_{ijk} \dot{q}_k \end{aligned}$$

this approach is faster than traditional application of Euler-Lagrange equations since

- coefficients of generalized accelerations is directly given by M
- coriolis and centrifugal terms are calculated from mass matrix
- energy function, as well as lagrangian function derivative calculation can be skipped

Terms coming from potential energy can be added separately

$$G(q) = \underbrace{\frac{d}{dt} \left(\frac{\partial U}{\partial \dot{q}_i} \right)}_0 - \frac{\partial U(q)}{\partial q_i} = \frac{\partial U}{\partial q_i} = G_i(q)$$

Inertia matrix and derivation of motion equations

It is worthy remember that having mass matrix expression, is theoretically possible to derive directly equations of motion. It is possible to account for all other forces including them on equations in later time (for instance friction terms and other non conservative forces). This is not always most practical way but often most straightforward and viable when there are no significative coupling among terms.

$$M_{ij}\ddot{q}_j + \left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_j \dot{q}_k + G_i(q) = \tau_i$$

This expression is valid for any choice of generalized coordinates. If we consider any transformation of coordinates, equations shape d not change but its content becomes even more complex and longer to evaluate, but at the same time gives more generality.

Is also useful to consider that equations of motion, in relation to inertial terms are, are more complex than inertia matrix, because including coriolis and centrifugal term. Nonetheless inertia matrix define inertial dynamics completely and involved inertial quantities, even without expressing it explicitly Irving means for understanding system dynamics in less complex terms. Considering geometry of the system is easy to see how expansion of all terms may not be convenient and hindering a more concise possible writing. We then repute useful the study of inertia matrix as preliminary for dynamics of the system.

Generalization of inertia matrix concept

Inertia matrix concept is related and dependent on adopted reference for expression of configuration and motion, that could be in terms of generalized coordinates (*joint space iberia JSIM*) or in coordinates space (*operational space inertia OSIM*). As well known, formalism allows a rather free choice of coordinates, that consequently changes the nature and shape of inertia matrix elements, although preserving essential characteristics of symmetry, positive definiteness and other properties, and equation of motion shape. Inertia matrix positive definite is a straight consequence of kinetic energy definition as, since it shall be positive definite and symmetric

$$K \geq 0 \iff \frac{1}{2} \dot{q}^T M \dot{q} \geq 0 \iff M \geq 0$$

but other properties exist, intuitive or less.

Inertia matrix properties

Here we report main properties of inertia matrix reported in literature, using progressive indexing from root to tip:

1. the inertia matrix M is *symmetric* and *positive definite*;
2. the energy of link j is a function of (q_1, \dots, q_j) and $(\dot{q}_1, \dots, \dot{q}_j)$, corresponding to links preceding links j in the kinematic tree
3. the element M_{ij} is a function of q_{k+1}, \dots, q_n , with $k = \min(i, j)$ and of the inertial parameters of links r, \dots, n r, ... n, with $r = \max(i, j)$ corresponding to links in subtree in the path of childs of j .
4. from property 2 and equation of motion, we deduce that τ_i is a function of the inertial parameters of links i, \dots, n
5. the matrix $N(q, \dot{q}) = M - 2C(q, \dot{q})$ is skew-symmetric for any choice of the matrix C given by equation[9.7][Koditschek 84], [Arimoto 84]. This property is used for the stability analysis of certain control schemes and is a check of equations correctness
6. the inverse dynamic model is *linear* in the elements of the standard inertial parameters M_j , MS_j and I_j^j . This property is exploited to *identify* the *dynamic parameters* (inertial and friction parameters), to reduce the computation burden of the dynamic model, and to develop adaptive control schemes; Observe Inearly requires reformulation of some non linear terms
7. a robot is a passive system which dissipates energy. This property is related to property 5.
8. there exist some positive real numbers a_1, \dots, a_7 such that for any values of q and \dot{q} we have [Samson 87]:

$$\|A(q)\| \leq a_1 + a_2 \|q\| + a_3 \|q\|^2$$

$$\|C(q, \dot{q})\| \leq \|\dot{q}\| (a_4 + a_5 \|q\|)$$

$$\|Q\| \leq a_6 + a_7 \|q\|$$

where $\|\cdot\|$ indicate a matrix or vector norm. If the robot has only revolute joints these become:

$$\|A(q)\| \leq a_1$$

$$\|C(q, \dot{q})\| \leq a_4 \|\dot{q}\|$$

$$\|Q\| \leq a_6$$

From properties 3 we get overall functional dependency, considering inertial parameters vector π .

$$M = \begin{bmatrix} f_{11}(q_2, \dots, q_n, \pi_1, \dots, \pi_n) & f_{12}(q_2, \dots, q_n, \pi_2, \dots, \pi_n) & f_{13}(q_2, \dots, q_n, \pi_3, \dots, \pi_n) & \dots & f_{1n}(q_2, \dots, q_n, \pi_n) \\ f_{21}(q_3, \dots, q_n, \pi_1, \dots, \pi_n) & f_{22}(q_3, \dots, q_n, \pi_2, \dots, \pi_n) & f_{23}(q_3, \dots, q_n, \pi_3, \dots, \pi_n) & \dots & f_{2n}(q_3, \dots, q_n, \pi_n) \\ & f_{j1}(q_{j+1}, \dots, q_n, \pi_j, \dots, \pi_n) & \dots & f_{jn}(q_{j+1}, \dots, q_n, \pi_n) \\ & & \ddots & & \vdots \\ & & & & f_{nn}(\pi_n) \end{bmatrix}$$

These properties are easily extended to a kinematic tree, but indexing can be violated since in a tree a sequential indexing is impossible and care shall be taken on properties reformulations, in terms of subtrees and paths on a graph.

Inertia matrix parametrization for a single link

A single link inertia matrix is a 6x6 symmetric matrix, hence can generally be fully described by

$$N_p = \frac{N^2 - N}{2} + N = \frac{N(N-1)}{2} + N = \frac{N(N+1)}{2}$$

real parameters, corresponding to **21 free real parameters**. These parameters are not unique, since any linear combination or even non linear reparameterization is possible, like in rotation matrix case. This result comes from general considerations. By the way, link inertial properties can be fully defined by means of a *reduced set of parameters*: scalar mass, center of mass 3-vector, and 6 symmetry rotational inertia matrix components. Consequently we conclude that we need **10 parameters** for a link full inertia matrix parametrizations, deriving from constraints equations applicable coming from inertia of a rigid body properties. Of course all these properties do not last anymore when dealing with inertia matrix of a composite complex system of rigid bodies. Alternatively, in this case one can consider base parameter determination method and assume we can use them both as number of independent parameters and as parametrization for inertia matrix. Details can be found in [Jain].

Inertia matrix structure: considerations

Generic structure of mass matrix is not in the end so hard to understand, but requires a careful analysis from different perspectives. Inertia matrix definition is based on kinetic energy but the use of alternative view proposed afore is worthy. Just considering role of entries and meaning inside equations of motion for each row column term, we can get through their physical meaning. Let's call τ^I_k inertial load on axis k and τ^I_{kj} component resulting from generalized coordinate acceleration \ddot{q}_j . By definition each element of mass matrix M_{ij} is the contribution due to generalized acceleration $\ddot{q}_j = 1$ to inertial generalized force τ_{kj}^I .

$$\tau_i = M_{ij} \ddot{q}_j$$

This introduce one of direct way for evaluating inertia matrix once is possible to set properly kinematic conditions (inverse dynamics) in the model. M do not depend on coordinate speeds and accelerations, but just on system geometry and configuration depending on generalized

coordinates q . This property was exploited by some authors ([Walker 82], [Safeea]) for delineating some specific direct and indirect algorithms.

Exploiting sparsity

inertia matrix definition deriving from general properties and equations structure implies that some elements of M will automatically be zero. This happens if there are branches in the kinematic tree, not affecting some links, leading to a tendentially *sparse* matrix. This effect is named *branch induced sparsity*, was studied by Featherstone in [Featherstone 10]. An example of this effect is shown in figure following. Observe nearly half of the elements are zero. It is possible to exploit this sparsity using the factorization algorithms proposed by Featherstone and other authors. Depending on the amount of branching in the tree, the sparse algorithms can run many times faster than the standard algorithms. In serial chain robots sparsity can depend on joint nature and their axis alignment.

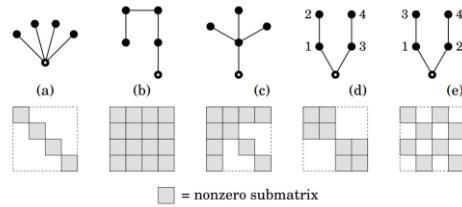


Figure 6.2: Examples of branch-induced sparsity

Recursivity

Mass matrix has a recursive nature and structure. That's easy to see considering that for each joint, equivalent inertia seen should account for all descendant sub-trees and paths masses and inertia. If joints are of the same type and aligned, also components of wrench acting shares some contributes. Even in case of not aligned joints is possible to have some shared terms. Inertia matrix complete layout is just the last step of a recursive calculation starting from tip to root. The process is cumulative and as we get closer to the root expression complexity increases accounting for more contributes. In a kinematic tree, everytime a bifurcation is met, respective seen inertia is the summation of the inertia of branches departing from them.

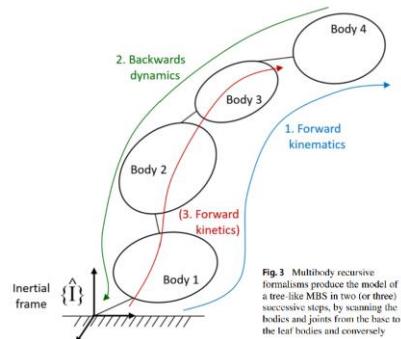


Fig. 3. Multi-step recursive formulations produce the model of a tree-like MBS in two (or three) successive steps, by scanning the bodies and joints from the base to the leaf bodies and conversely

Mass matrix reordering

Since coordinates order choice is free and general, and is part of transformations allowed, we can freely reorder and reorganize coordinates vector and generalized forces vectors in most convenient way, without affecting their validity. Obviously permutation of rows and columns shall concurrently affect all matrices row and columns. There are different reasons for that, from highlighting underlying block structure, exploiting factorisations, or just collect similar terms. Sometimes more complex transformations allow for more evident factorisations as illustrated later.

System of concentrated mass reduction

Observe given a system of concentrated masses and inertia tensor in fixed relative configuration one respect the other, it is possible to reduce all contributions to system dynamics and energy into a single mass and inertia tensor equivalent term without loss of generality. In case mapping back to a mass distribution is not trivial and requires invertibility conditions subsistence, like basing on geometrical assumptions of mass values and configurations of distances. A constraint on inertia equivalence equations shall be applied.

Mass matrix decompositions and factorization

For our purpose we can introduce some other decomposition of the mass matrix in order to make it easier and more clear its understanding. First we can separate terms just depending on system links topology and orientation but not on distances and configuration, from parts that depend on distances and configuration. Moreover we can separate translational speed coordinates from rotational speed of coordinates, since they often do not couple (like in a sort of extension of Konig theorem). In addition we can split parts related to coordinate dependent parts from constant parts.

$$M = M_0 + M(q)$$

Again we can imagine to separate all components related to some link inertial properties from all the others, and split respectively on *mass-related* and *inertia tensor related* treating them independently since linear superposition principle apply, in spite of terms nonlinear dependence on configuration.

$$M = \sum_{k=1}^n M_{m_k} + \sum_{k=1}^n M_{I_k}$$

Matrix linearity also allows for decomposition and separation by coordinate acceleration dependency, making each joint contribution evident. The exact expression of each term of the sum will become clear when treating the Jacobian derivation of the inertia matrix. These decompositions could be derived both in symbolic or numeric or mixed way from algorithms described in [1]. An interesting possible decomposition is using different combinations of trigonometric terms seen in section, factoring common dependency from configuration. This study is less common to find in literature. Decomposition into additive terms is most intuitive and easy to carry out; other factorisations have been considered in literature, with very

prominent results, specifically using cholesky LTL and LDLT factorisation (see Featherstone [Featherstone 05]).

A carnot theorem generalization

The Carnot theorem has some implications for our work, although usually not considered in most of literature. From two vectors Carnot theorem gives:

$$c = (a + b)$$

$$c^2 = (a + b) \cdot (a + b) = a^2 + b^2 + 2 a \cdot b = |a|^2 + |b|^2 + 2 |a| |b| \cos(\angle ab)$$

We can now extend this expression for multiple n-vector sums, avoiding a lot of trigonometric calculations by a simple vectorial algebra consideration. Considering then a generic summation of vectors v_k

$$\begin{aligned} c^2 &= \left(\sum_{k=1}^n v_k \right)^2 = \left(\sum_{i=1}^n v_i \right) \cdot \left(\sum_{h=1}^n v_h \right) = \sum_{i=1}^n \sum_{h=1}^n v_i \cdot v_h = \sum_{k=1}^n v_k^2 + \sum_{i \neq h} v_i \cdot v_h \\ &c^2 = \sum_{k=1}^n |v_k|^2 + \sum_{i \neq h} |v_i| |v_h| \cos(\angle v_i v_k) \end{aligned}$$

where expressions do not depend on the projection frame. The importance of this lemma on inertia matrix calculation comes from the fact that explicit calculation of these expressions results in long trigonometric expressions that can be avoided with opportune formalism. Indeed, the relative position of a center of mass with respect to a joint can be written as a sum of link joint to joint vectors and a last joint to CoM vector, falling in our case. We can also state the more links are in the path, the worse the expression. Furthermore any term lying on a path will share part of these terms. Usually conventional algorithms will just compute all trigonometric terms of joints position, not recognizing common geometry underlying, resulting in complex and not meaningful expressions.

Inertia matrix calculation method: direct and indirect

Inertia matrices can be calculated exploiting analytical and system properties in different ways. We can distinguish between direct methods and indirect methods. Direct methods derive directly the inertia matrix from system kinematics and link inertial properties. Indirect methods exploit equations of motion and inverse dynamics or energy expression for deriving an inertia matrix.

Indirect methods

In literature different indirect methods are used in literature for the calculation of mass matrix. Indirect methods are less efficient and slower than rarely used in applications. A first method consists of just collecting acceleration related terms in symbolic equations for generalized forces, or taking Jacobian respect for them. Usually this way is inefficient even for a few

degrees of freedom. Remark that we do not *need* to account for a full set of equations of motion, since additional terms deriving from potential and/or other forces (friction and external forces) do not impact on mass matrix definition, even if they do not depend on accelerations or speeds, but would increase computation burden. Inverse dynamics can be calculated using different methods (Newton Euler, Eulero-Lagrange, or any other). The use of a customized version of any method for a more straightforward calculation of the inertia matrix is possible, excluding some terms of complete version. Using kinetic energy expression results may be more efficient but still time taking, since collection or jacobian over speeds have to be taken twice. A second consideration, able to speed up calculations and make results more reusable, is related to the fact that kinetic energy structure as well as deriving equations of motions have a separable nature as sum of contributions coming from different terms. Kinetic energy is easier part to understand since is given by the sum of kinetic energy of each single link of the system and also inertial and mass contributes can be split

$$K = \sum_{h=1}^n K_h = \sum_{h=1}^n (K_{mh} + K_{lh})$$

Consequently also restricted equations of motion are separable and could be written as sum of contributes

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{q}_i} \right) - \frac{\partial K}{\partial q_i} = \sum_{h=1}^n \left(\frac{d}{dt} \left(\frac{\partial K_h}{\partial \dot{q}_i} \right) - \frac{\partial K_h}{\partial q_i} \right) = \tau_i$$

This is not without relevance since for high values of degree of freedom computational effort for extraction of a mass matrix from a full set of equations, with hundreds or thousands of terms could become easily and rapidly prohibitive even for high performance computing . Inertia matrix can be decomposed as sum of partial contributes associated to single terms of kinetic energy or equivalently pieces of equations of motions, with lower effort, summing all terms only at the end of the process

$$M_{ijh} = \frac{\partial^2 K_h}{\partial \dot{q}_i \partial \dot{q}_j}$$

$$M_{ij} = \frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j} = \frac{\partial^2}{\partial \dot{q}_i \partial \dot{q}_j} \sum_{h=1}^n K_h = \sum_{h=1}^n \frac{\partial^2 K_h}{\partial \dot{q}_i \partial \dot{q}_j} = \sum_{h=1}^n M_{ijh}$$

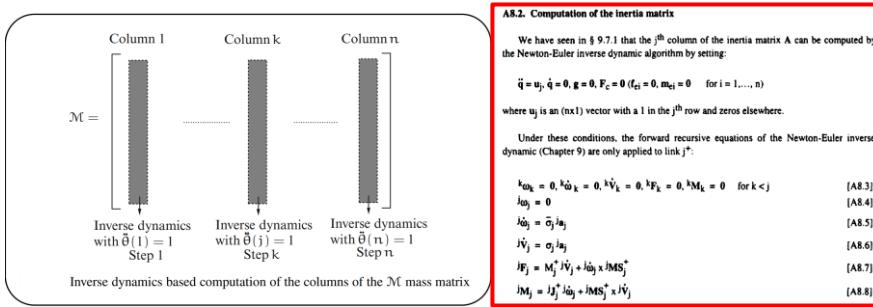
This is valid for a generic mechanical system even if not open chain, including closed chains, and generic complex systems and composite systems of unconnected mechanisms.

Finally considering inertia matrix as joints torque resulting from imposing one joint unitary acceleration and null joint speed, for a given configuration, inverse dynamics gives:

$$M_{ij}(q) = \tau_i \left| \begin{cases} \ddot{q}_{j=1} & i = j \\ \ddot{q}_{j=1} & i \neq j \end{cases} \right.$$

$$M_{ij}(q_k) = \tau_i^I(q_k, \dot{q}_k, \ddot{q}_k) \left| \begin{array}{ll} \ddot{q}_{k=1} & k=j \\ \ddot{q}_{k=0} & k \neq j \\ \dot{q}_k = 0 & \forall k = 1..n \end{array} \right.$$

Assume we are using only inertial forces leaving out all other terms (including gravitational and frictional ones). Based on this definition, an inertia matrix is easily numerically or symbolically calculated using any dynamics derivation method, Euler Lagrange, Newton Euler, or any other. This method is due to Walker, is schematically represented in figure:



Unfortunately it requires multiple runs for determining column by column resulting matrix.

Direct methods

Inertia matrix direct calculation using jacobians

A commonly used direct way to calculate inertia matrix, is applying directly jacobian properties to the definition (based on kinetic energy). For a system of N_B rigid bodies (or links, with different nomenclature), where $N_p(k)$ material points could exist, having pure inertia I_C around joint axis, total system kinetic energy can be written as

$$K = \sum_{k=1}^{N_B} \left(\sum_{j=1}^{N_p(k)} \left(\frac{1}{2} v_{p_{kj}}^T m_{p_{kj}} v_{p_{kj}} \right) + \frac{1}{2} \omega_k^T I_{C_k} \omega_k \right)$$

where we used usual notation for points speed and angular speed. Introducing translational (for every material point) and rotational jacobian (for whole body), it's easy to put all into dependance of generalized speed vector $\dot{\mathbf{q}}$.

$$\begin{aligned} v_{p_{kj}} &= J_{Tr_{kj}} \dot{\mathbf{q}} \\ \omega_k &= J_{R_k} \dot{\mathbf{q}} \end{aligned}$$

$$K = \sum_{k=1}^{N_B} \left(\sum_{j=1}^{N_p(k)} \left(\frac{1}{2} \dot{q}^T J_{Tr,p_{kj}}^T m_{p_{kj}} J_{Tr,p_{kj}} \dot{q} \right) + \frac{1}{2} \dot{q}^T J_{Rk}^T I_{C_k} J_{Rk} \dot{q} \right)$$

Please remark this is valid for an arbitrary system of rigid bodies and material points independently from their connection topology. They don't even need to be connected, then resulting equations are very general and last for open as well as closed as well as unconnected systems and subsystems. Now just collecting out generalized speeds vector we get a straight definition of inertia matrix as

$$K = \frac{1}{2} \dot{q}^T M \dot{q} = \frac{1}{2} \dot{q}^T \sum_{k=1}^{N_B} \left(\sum_{j=1}^{N_p(k)} \left(J_{Tr,p_{kj}}^T m_{p_{kj}} J_{Tr,p_{kj}} \right) + J_{Rk}^T I_{C_k} J_{Rk} \right) \dot{q}$$

leading to explicit definition for inertia matrix components, for single mass (barycentric)

$$M_{ij} = \sum_{k=1}^{N_B} \left(J_{Tr,p_{Ck}}^T m_{p_{Ck}} J_{Tr,p_{Ck}} + J_{Rk}^T I_{C_k} J_{Rk} \right)$$

or using twist notation of 6 vectors for inertia and speeds

$$M_{ij} = \sum_{k=1}^{N_B} \left(J_{p_{Ck}}^T I_{C_k} J_{p_{Ck}} \right)$$

The use of 6-spatial vector notation allows an even more compact formalism. Then there's a need to find Jacobians and just calculate all products summing out different terms. From this expression we don't have immediate evidence of the presence of a specific structure inside an inertia matrix, but it looks like all matrix entries could be a function of all masses and inertia potentially involved in the system. Reason is that indeed it does not require any assumption about connections of system inertial elements, and consequently any coupling could effectively be real. On the other side one can have a look at the Jacobian matrix and infer some deduction, although algebraic proof is more complex. Obviously if any point speed is not causally related with some generalized coordinates, corresponding columns of jacobian are null and will cancel out all products with inertial terms. In a serial chain this happens for all coordinates that correspond to following joints in the chain respect considered point or body. Resulting null submatrix of the jacobian matrix introduces the specific structure into the inertia matrix we studied in other sections.

We shall now remember that Jacobian can be obtained analytically, numerically or *geometrically*. Last option puts in evidence the fundamentally geometric nature of inertia matrix expression, after performing all calculations, results into a geometry dependent terms combination.

$$J_{C_k} = \begin{pmatrix} \sigma_1 z_1 + \bar{\sigma}_1 (z_1 \wedge r_{1C_k}) & \dots & \sigma_n z_n + \bar{\sigma}_n (z_n \wedge r_{nC_k}) \\ \bar{\sigma}_1 z_1 & \dots & \bar{\sigma}_n z_n \end{pmatrix}$$

Consequently an opportune choice of system involving geometric terms allows a direct proof, based on the general structure of the inertia matrix, of a statement already asserted by another author approaching the same problem on a different perspective and method, underlying once more fundamental equivalence of all methods presented. Advantage of geometric jacobian is clear definition of fundamental geometric elements, in other methods could be less apparent and easy to see. These methods, opportunely exploited, also allow a further and more efficient calculation even with respect to the CRBA method. An alternative notation makes use of spatial vector formalism yielding a more compact and intuitive notation, in the end equivalent in terms of process.

CRBA Algorithm

The Composite-Rigid-Body Algorithm (CRBA) is an efficient algorithm for calculating the joint-space inertia matrix (JSIM) of a robot mechanism having the connectivity of a kinematic tree, or of the spanning tree of a robot mechanism containing kinematic loops. Do not matter types of joints involved. Composite rigid body algorithm exploit recursivity properties of kinematic tree for its calculation. Advantage respect other approaches consist in

- Direct calculation of inertia matrix where other methods first calculate equations of motion
- More efficient and fast respect equations of motion based methods
- In spite of fact that RNEA is faster for calculating dynamics, further evaluations required for inertia matrix calculation take longer

Composite rigid body key idea regards to consider for each joint all follower links connected paths as rigid structure (with a fixed configuration determined by generalized coordinates) assuming no other joint motion occurs downstream, and calculating total inertia contribute as seen from joint in order to get inertia matrix component associated projecting to relevant row joint as results of summing up inertia coming from every single rigid body contribution. Consequently massless and inertialess links do not contribute at all.

CRBA algorithm is very simple and reported below. Different implementations are possible basing on the choice of kinematics representation used. Most compact notation exploits spatial vector formalism. We have chosen spatial notation since is most compact and easy to read. Essential structure of CRBA algorithm is the suitable for tree like robots and is based on forward and backward recursion over bodies tree. Algorithm loops over all rigid bodies of the system and for each body

Algorithm 4.2 Recursive computation of the mass matrix

```

 $\mathcal{R}(0) = \mathbf{0}$ 
for  $k = 1 \dots n$ 
   $\mathcal{R}(k) = \phi(k, k-1)\mathcal{R}(k-1)\phi^*(k, k-1) + M(k)$ 
   $X(k) = \mathcal{R}(k)H^*(k), \quad M(k, k) = H(k)X(k)$ 
  for  $j = (k+1) \dots n$ 
     $X(j) = \phi(j, j-1)X(j-1)$ 
     $M(j, k) = M^*(k, j) = H(j)X(j)$ 
  end loop
end loop

```

Algorithm 3.5 Composite-rigid-body algorithm for calculating the ISM

input: model, RNEA partial results
output: H
model data : $N_p, p(i), I_i$
RNEA data : $\Phi_i, X_{p(i)}$

$H = \mathbf{0}$
for $i = 1$ to N_p **do**
 $I_i^T = I_i$
end for
for $i = N_p$ to 1 **do**
 $F = I_i^T \Phi_i$
 $H_i = \Phi_i^T F$
if $p(i) \neq 0$ **then**
 $I_{p(i)}^T = I_{p(i)} + X_{p(i)}^T I_i^T X_{p(i)}$
endif
 $j = i$
while $p(j) \neq 0$ **do**
 $F = (X_{p(j)}^T F$
 $j = p(j)$
 $H_j = I_j^T F$
 $H_j = I_j^T \Phi_j$
end while
end for

See for more details [Featherstone], [Walker and Orin]. Matlab® code for CRBA algorithm is provided in the code-algorithms section.

Composite Rigid Body inertia structure

Inertia matrix generic structure could be put in simpler form using composite rigid body inertia. In fact, I_i^C is the inertia of the composite rigid body formed by the rigid assembly of all the links in child path of i , $c^*(i)$.

$$I_i^C = \sum_{j \in c^*(i)} I_j$$

It represents the inertia of the subpath associated to joint j (all follower links and followers of followers). Resulting expression for inertia matrix is

$$H_{ij} = \begin{cases} \Phi_i^T I_i^C \Phi_j & \text{if } i \in c^*(j) \\ \Phi_j^T I_j^C \Phi_i & \text{if } j \in c^*(i) \\ 0 & \text{otherwise} \end{cases}$$

or for unbranched tree

$$I_i^C = I_i + I_{i-1}^C$$

$$H_{ij} = \Phi_i^T I_{\max(i,j)}^C \Phi_i$$

In this way structure of inertia matrix is *recursively* clear and rather easy to understand. Exemplificative picture of the concept of rigid body inertia is illustrated in figure

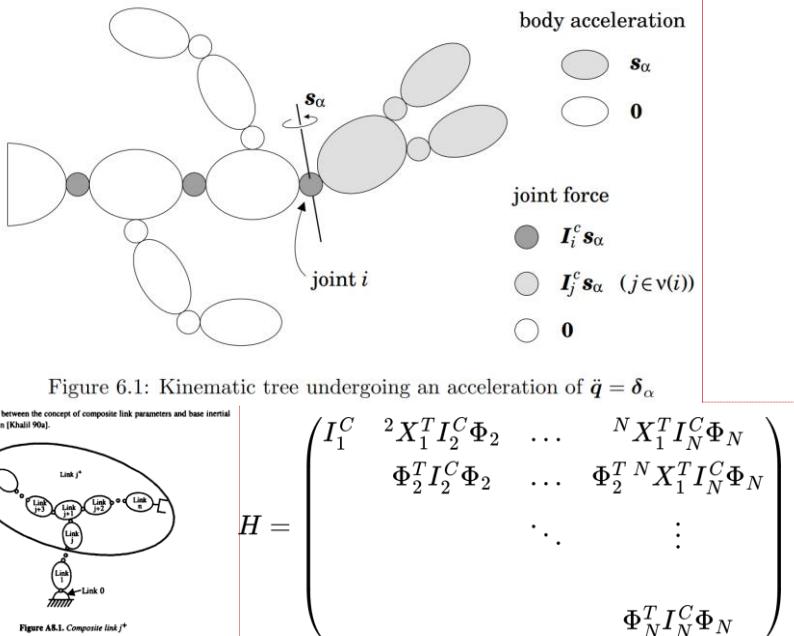


Figure 6.1: Kinematic tree undergoing an acceleration of $\dot{q} = \delta_\alpha$

$$H = \begin{pmatrix} I_1^C & X_1^T I_2^C \Phi_2 & \dots & X_1^T I_N^C \Phi_N \\ \Phi_2^T I_2^C \Phi_2 & \dots & \Phi_2^T X_1^T I_N^C \Phi_N \\ \vdots & & \ddots & \vdots \\ \Phi_N^T I_N^C \Phi_N & & & \end{pmatrix}$$

Commented [8]: Cancel or keep

Using space vector notation we can write, both in case of floating base or fixed base the expression for inertia matrix can be written as above.

Customized Eulero Lagrange method for efficient dynamic model derivation

Khalil in its work for robot identification derived a customized version of Eulero-Lagrange method for dynamics modeling, whose set up exploits considerably internal system geometry of links, centers of mass and inertia. Resulting equations were derivative free and essentially algebraic, although rather complex. Using opportune parametrization of geometrical terms a lot of computational power and symbolic workload and complexity could be saved. For brevity we report directly to the original article [Khalil et al. 96]. Clearly as part of dynamics equation set, inertia matrix set is included as $M\dot{q}$, then easily extracted from.

GDAHJ: a geometrically based inertia matrix method

Authors [Safeea et al. 18] provided proof of a purely geometrically based algorithm for inertia matrix calculator from links inertial properties. Method is based on the evaluation of effect of a unitary joint acceleration \ddot{q}_j on all links (center of mass and inertia) reprojected back onto joint i. Figure depict briefly this concept

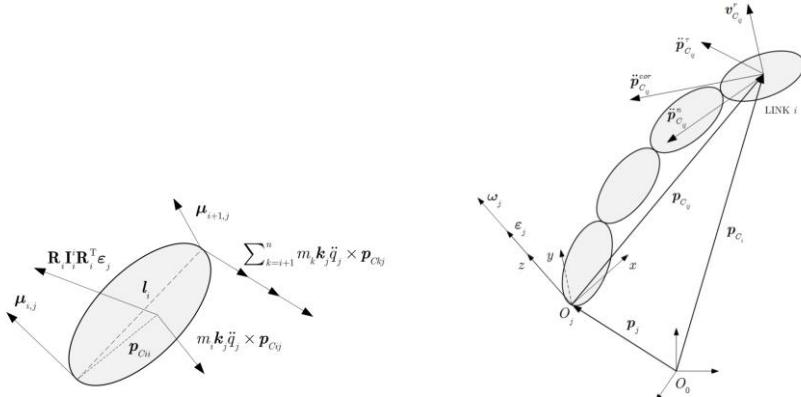


Figure 3. Inertial forces and moments acting on link i due to angular acceleration of joint j .

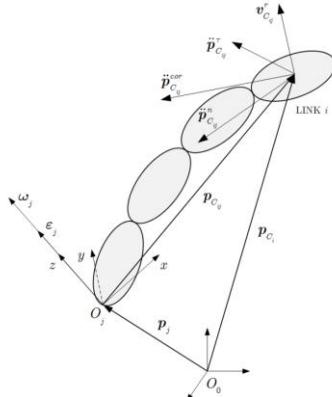


Figure 2. Tangential acceleration of centre of mass of link i transferred by frame j

Resulting algorithm proved to be faster and more efficient than CRBA and easier to understand. In spite of geometrically inspired method, the calculation carried out do not properly account for geometrical element, exploding all terms as in other methods, resulting in substantially similar expressions. This algorithm with opportune modifications would be easier to adapt to the introduction of geometrical variables. Authors derived algorithm only for revolute joint serial robots, but its extension to arbitrary kinematic tree and arbitrary joints is straightforward. For more details see [\[Safeea et al. 18\]](#).

Inertia matrix inversion

Inversion of inertia matrix is not a simple operations, and is explicitly involved in forward dynamics problems. When dealing with symbolic derivation of mass matrix, inverse calculation becomes prohibitive yet for few degree of freedom, and consequently no explicit derivation of forward dynamic is carried out usually in this way. Traditional problem wraparound solution involves numerical evaluation of mass matrix and numerical inversion at every step using current coordinates values (then losing parametrization on inertial parameters and coordinates that generalize expression use, since even few parameters make problem untreatable even in hybrid numerico-symbolic form). Numerical inversion is $O(N^3)$. In truth a mixed inversion is possible but only keeping a limited number of symbolic parameters, and results would be anyway numerical errors affected. Nonetheless alternatives exist from advanced study of spatial operator algebra formulation. Recurring to some works of [Rodriguez-Kreutz], a simplified analytical formulation of direct dynamics for open chain mechanism is possible, by adoption of spatial operators. Results are valid also for general multibody systems as kinematic trees.

Exploiting spatial Kalman Filtering analogy, and nilpotent properties of staked inertia matrix, one can get out with an inversion relations putting in close relations inertia matrix with Kalman gain. General case regards nilpotent matrices.

$$(I + H\Phi G)^{-1} = I - H\Psi G$$

$$M^{-1} = (I - H\Psi G)^T D^{-1} (I - H\Psi G)$$

Consequently direct dynamics problem can be reformulated as

$$\ddot{q} = (I - H\Psi G)^T D^{-1} (I - H\Psi G) T'$$

Same author provide also a schematics for efficient recursive algorithm workflow design.

[\[Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix 1988\]](#)

[\[Recursive Mass Matrix e. Factorization and Inversion 1988\]](#)

Commented [9]: Use notation for bibliography

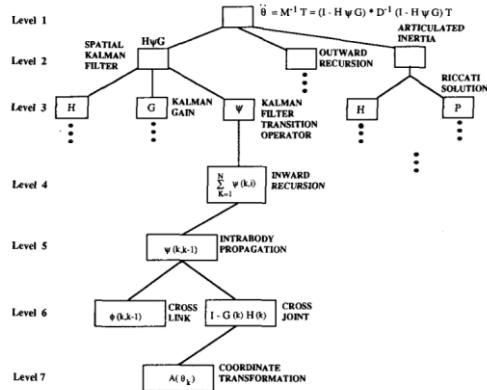


Fig. 3. Spatial operators map to efficient recursive algorithms.

Inertial matrix examples and special case

Joint type notation

Although in most cases revolute joints are used this is not the most general case, since in some situations translational joins are present and used (for example SCARA robots) and helical joints would be possible (1dof). To manage different situations conveniently different notations have been conceived. See [Modeling, Identification and Control of Robots] like the introduction of an auxiliary variable σ_k for each joint that accounts for joint type being 1 when prismatic and zero when revolute, and its complementary barred. In this way we can manage joints twist in a general way:

$$\begin{pmatrix} \omega_k \\ v_k \end{pmatrix} = \dot{q}_k \begin{pmatrix} \bar{\sigma}_k \hat{z}_k \\ \sigma_k \hat{z}_k \end{pmatrix} \quad \begin{cases} \sigma_k = 1 & \text{prismatic joint} \\ \sigma_k = 0 & \text{revolute joint} \\ \bar{\sigma}_k = 1 - \sigma_k \end{cases}$$

$$\begin{array}{c} \sigma = 1 \quad \quad \quad \sigma = 0 \\ \bar{\sigma} = 0 \quad \quad \quad \bar{\sigma} = 1 \end{array}$$

or even more generally introducing screw motion with coefficients α_k and β_k accounting also for a scaling:

$$\begin{pmatrix} \omega_k \\ v_k \end{pmatrix} = \dot{q}_k \begin{pmatrix} \alpha_k \hat{z}_k \\ \beta_k \hat{z}_k \end{pmatrix} \quad \begin{cases} \alpha_k = 1, \beta_k = 0 & \text{prismatic joint} \\ \alpha_k = 0, \beta_k = 1 & \text{revolute joint} \\ \alpha_k \neq 0, \beta_k \neq 0 & \text{screw joint} \end{cases}$$

Pure inertia revolute joints only robot

Considering revolute joint-only robot, whose inertial properties are purely characterized by links inertia tensor (for instance when CoM is coincident with joint axis), we get a simplified version of the inertia matrix. Each link inertia is rotated (depending on configuration consequent to q) from body to joint frame. Summon all contributes of links on successor path of joint i. This is extendable to kinematic tree as is.

$$M_{ij} = \sum_{h=\text{succ}(j)} \hat{z}_i \cdot ({}^j R_h(q) I_h {}^h R_j(q)) \cdot \hat{z}_j$$

Notably contributes from orthogonal axis (body h axis to joint i z axis but also joint j z axis on i z joint axis) disappears. For a planar robot all contributes excepted Inertia respect plane normal disappears as seen above. In that case inertia matrix is constant and do not depend on configuration. This is indeed as applying jacobian rotational part and project results on interest frames. Complexity in expression, related to robot configuration, being inertia constant on body frame, is introduced by joint axis alignment determined by rotation matrix. Any simplification in kinematics relations lead to simpler mass matrix expressions.

Prismatic joints

Prismatic or translational joints are particularly simple to account because each joint of this type, tanks to mass isotropy (independence from axis of translation) and summability of effects. On each joint then a contribute exactly equal to the sum of all masses belonging to link successors (including then, every tree branch) is given. That's for the effect of joint acceleration or speed over itself. Similarly every other joint belonging to successor branches could exert a contribute calculated in a similar way. On joint j will act a force equal to

$$M_{ij} = \sum_{k \in \text{succ}(j)} m_k \hat{z}_j \cdot \hat{z}_{ij}$$

$$M_{ij} = \hat{z}_i \cdot f_i = \hat{z}_i \cdot \left(\sum_{j=1}^n m_j \hat{z}_j \varsigma_{ij} \right) = \sum_{j=1}^n m_j \varsigma_{ij} (\hat{z}_i \cdot \hat{z}_j)$$

$$M_{ij} = \hat{z}_i \cdot f_i = \hat{z}_i \cdot \left(\sum_{j=1}^n m_j \hat{z}_j \varsigma_{ij} \right) = \sum_{j=1}^n m_j \varsigma_{ij} C_{ij}(q)$$

projected on interest joint axis, because of backward transmission of acting force. Clearly if axis are orthogonal there's no contribute. For example for a cartesian robot we've a diagonal matrix:

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 & 0 & 0 \\ 0 & m_2 + m_3 & 0 \\ 0 & 0 & m_3 \end{bmatrix}$$

or in presence of redundant axis, or even in case of tree structure we can generally write for orthogonal axis

$$M = \begin{bmatrix} \sum_{k=\text{succ}(1)} m_k & 0 & \dots & \dots & 0 \\ 0 & \ddots & \dots & \dots & \vdots \\ \vdots & \vdots & \sum_{k=\text{succ}(i)} m_k & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \dots & \dots & 0 & m_n \end{bmatrix}$$

More general case of a skew prismatic robot will lead to a full matrix:

$$M_{ij} = \sum_{k=\text{succ}(j)} m_k (\hat{z}_i \cdot \hat{z}_j) = \begin{bmatrix} \sum_{k=\text{succ}(1)} m_k (\hat{z}_1 \cdot \hat{z}_1) & \sum_{k=\text{succ}(2)} m_k (\hat{z}_1 \cdot \hat{z}_j) & \dots & \sum_{k=\text{succ}(n)} m_k (\hat{z}_1 \cdot \hat{z}_n) \\ \vdots & \sum_{k=\text{succ}(j)} m_k (\hat{z}_j \cdot \hat{z}_j) & & \vdots \\ \sum_{k=\text{succ}(n)} m_k (\hat{z}_n \cdot \hat{z}_1) & \dots & m_n & \sum_{k=\text{succ}(n)} m_k (\hat{z}_n \cdot \hat{z}_n) \end{bmatrix}$$

Sub inertia matrix restricted to translational coordinates results to be diagonal by nature in virtue of its isotropy of mass inertial effects. Remarks all inertia matrices in that case do not depend on configuration coordinates, but only on relative joint orientation. Each term is then a constant. Furthermore, presence of link inertia does not affect at all joint actuation. Finally, a similar reasonament apply in presence of revolute joints, if we include joint type σ (only translational part of overall inertia matrix)

$$M_{ij} = \sum_{k=\text{succ}(j)} m_k (\hat{z}_i \cdot \hat{z}_j) = \begin{bmatrix} \sum_{k=\text{succ}(1)} m_k (\sigma_1 \hat{z}_1 \cdot \sigma_1 \hat{z}_1) & \sum_{k=\text{succ}(2)} m_k (\sigma_1 \hat{z}_1 \cdot \sigma_2 \hat{z}_2) & \dots & \sum_{k=\text{succ}(n)} m_k (\sigma_1 \hat{z}_1 \cdot \sigma_n \hat{z}_n) \\ \vdots & \sum_{k=\text{succ}(j)} m_k (\sigma_j \hat{z}_j \cdot \sigma_j \hat{z}_j) & & \vdots \\ \sum_{k=\text{succ}(n)} m_k (\sigma_n \hat{z}_n \cdot \sigma_1 \hat{z}_1) & \dots & m_n & \sum_{k=\text{succ}(n)} m_k (\sigma_n \hat{z}_n \cdot \sigma_n \hat{z}_n) \end{bmatrix}$$

In spite of similarity with, shall not be confused with the inertia matrix for mass particles of which is a special case, because we're assuming a constrained motion in one direction for each mass. We can reconduct to free particle motion only considering a set of independent directions (preferably orthogonal), where intermediate links are massless.

3R Articulated robot case

Under simplification assumptions on center of mass position we can write inertia matrix as

$$\begin{pmatrix} m_{11}(q_2, q_3) & 0 & 0 \\ 0 & m_{22}(q_3) & m_{23}(q_3) \\ 0 & m_{23}(q_3) & m_{33} \end{pmatrix}$$

decoupling of first joint occurs only in case offset of second joint is zero. Murray provide explicitly using jacobians method

$$\begin{pmatrix} I_{2_{yy}} s_2^2 + I_{3_{yy}} s_{23}^2 + I_{1_{zz}} + I_{2_{zz}} c_2^2 + & & & \\ I_{3_{zz}} c_{23}^2 + m_3 r_2^2 + m_3 l_1 r_2 c_3 + & 0 & & 0 \\ m_2 r_1^2 c_2^2 + m_3 (l_1 c_2 + r_2 c_{23})^2 & & & \\ 0 & I_{3_{xx}} + m_3 r_2^2 + m_3 l_1 r_2 c_3 & I_{3_{xx}} + m_3 r_2^2 + m_3 l_1 r_2 c_3 & \\ 0 & I_{3_{xx}} + m_3 r_2^2 + m_3 l_1 r_2 c_3 & I_{3_{xx}} + m_3 r_2^2 & \end{pmatrix}$$

easily we can derive a short parametrization of a basic 3dof manipulator inertia matrix:

$$\begin{pmatrix} \beta + \alpha_1 c_2^2 + \alpha_2 c_{23}^2 + & 0 & 0 \\ \alpha_3 c_3 + \alpha_4 c_2 c_{23} & & \\ 0 & \gamma + \delta c_3 & \gamma + \delta c_3 \\ 0 & \gamma + \delta c_3 & \gamma \end{pmatrix}$$

$$\begin{aligned} \alpha_1 &= I_{2_{zz}} - I_{2_{yy}} + m_3 l_1^2 + m_2 r_1^2 \\ \alpha_2 &= I_{3_{zz}} - I_{3_{yy}} + m_3 r_2^2 \\ \alpha_3 &= m_3 l_1 r_2 \\ \alpha_4 &= 2m_3 l_1 r_2 \\ \beta &= I_{1_{zz}} + m_3 r_2^2 + I_{2_{yy}} + I_{3_{yy}} \\ \gamma &= I_{3_{xx}} + m_3 r_2^2 \\ \delta &= m_3 l_1 r_2 c_3 \end{aligned}$$

A more complete formulation of a 3dof articulated manipulate gives:

$$\begin{pmatrix} a_1 c_2^2 + a_5 c_2 s_2 + a_4 c_2 s_{23} + a_3 c_{23}^2 + a_6 c_{23} s_{23} + a_2 s_2^2 + a_7 s_{23}^2 + a_8 & a_9 c_2 + a_{11} c_{23} + a_{10} s_2 + a_{12} s_{23} & a_{11} c_{23} + a_{12} s_{23} \\ a_9 c_2 + a_{11} c_{23} + a_{10} s_2 + a_{12} s_{23} & a_1 + a_7 + a_{13} + a_{14} + 2a_4 s_3 & a_7 + a_{14} - \frac{a_4 s_3}{2} \\ a_{11} c_{23} + a_{12} s_{23} & a_7 + a_{14} - \frac{a_4 s_3}{2} & a_7 + a_{14} \end{pmatrix}$$

Commented [10]: Matrix size

Revolute joints

Also in case of revolute joints, considering only mass effects (inertia contributes to inertia matrix has already been examined previously) we can derive some simple closed form expression helping us in building up inertia matrix components. Basing on the concept that inertia matrix is the torque projection contribute given to axis i (row) from inertial torque arising from angular acceleration on axis j. In this calculation all masses shall be considered fixed as a composite rigid body, whose configuration depend on values of generalized coordinates. We can study it then as a pure rotational motion. Considering a couple of rotational joints:

$$M_{ij}\ddot{q}_j = \sum_{h \in \text{succ}(j)} -\left(\hat{z}_i \cdot \left(l_{j_{ch}}(q) \times (l_{j_{ch}}(q) \times \hat{z}_j)\right)m_h\right)\ddot{q}_j$$

$$M_{ij} = \sum_{h \in \text{succ}(j)} \underbrace{\hat{z}_i \cdot l_{j_{ch}}(q) \times \underbrace{\left(\ddot{q}_j \hat{z}_j \times l_{j_{ch}}(q)\right)m_h}_{\substack{\text{C}_h \text{ acceleration } j \\ \text{linear inertia force}}} }_{\text{angular inertial force moment}}$$

This formula highlights directly the role of mass distribution geometry without specifying its dependence on generalized coordinates, that depend on coordinate choice. Indeed often the use of explicit vector components and coordinate exploitation lead to useless long expressions whose simplification is immediate using geometrical expressions.

Mixed effects prismatic to revolute and vice versa

So far we've considered revolute-revolute and prismatic-prismatic joint interactions, and for most cases it's enough. But for completing our case panorama, we need to define interactions arising from prismatic-revolute and vice versa (inertia matrix symmetry assure as effects are equivalent). Consider the force experienced on a prismatic joint due to a revolute joint j motion (acceleration). As seen, pure inertia do not give effects and only mass contribute.

$$M_{ij}\ddot{q}_j = \sum_{h \in \text{succ}(j)} -\left(\hat{z}_i \cdot \left(l_{j_{ch}}(q) \times (l_{j_{ch}}(q) \times \hat{z}_j)\right)m_h\right)\ddot{q}_j$$

Use of trigonometric expression variables

It's common use in dynamic study of applied mechanics and robotics to use a short notation for trigonometric terms that has different advantages. First is the reduction of expression length improving readability. Second, offering a way to identify intermediate variables usable to reduce computational burden. Third, make easier the identification of terms that are common across different terms, offering insight for collecting similar terms and common behaviors dependency on configuration.

A special case: planar robot

Planar revolute joints robots is another special case where it is possible to find simplified solutions easy to manage even manually in spite of the number of joints. Simplifications come from having all revolute joints parallel one another. Consequently each link angular speed is easily written as the scalar sum of joint angular speed, and each joint angle sums up to all others in a cumulative coordinate, that is irrespective to rotation respect inertial frame.

$$\omega_k = \sum_{h=1}^k \dot{q}_h \quad q'_k = \sum_{h=1}^k q_h$$

Inertial contributes to dynamics arising from rotations depend only on I_{zz} , as evident from rotational kinetic energy:

$$K_{r_k} = \frac{1}{2} (0 \quad 0 \quad \omega_k) \begin{pmatrix} I_{k_{xx}} & I_{k_{xy}} & I_{k_{xz}} \\ I_{k_{xy}} & I_{k_{yy}} & I_{k_{yz}} \\ I_{k_{xz}} & I_{k_{yz}} & I_{k_{zz}} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \omega_k \end{pmatrix} = \frac{1}{2} (0 \quad 0 \quad \omega_k) \begin{pmatrix} I_{k_{xz}} \\ I_{k_{yz}} \\ I_{k_{zz}} \end{pmatrix} \omega_k = \frac{1}{2} I_{k_{zz}} \omega_k^2$$

Resulting kinetic energy expression results to be simplified and easy to write, where most complex contributions come from center of mass speed.

$$v_k = \sum_{h=1}^k (\omega_h \wedge l_h) = \sum_{h=1}^k \left(\sum_{j=1}^k \dot{q}_j \hat{z} \wedge l_h \right) = \hat{z} \wedge \sum_{h=1}^k \sum_{j=1}^k \dot{q}_j l_h$$

$$v_k = \sum_{h=1}^k (\omega_h \wedge l_h) = \sum_{h=1}^k \left(\sum_{j=1}^k \dot{q}_j \hat{z} \wedge l_h \right) = \hat{z} \wedge \sum_{h=1}^k l_h \sum_{j=1}^k \dot{q}_j$$

where geometrical term arise naturally.

We can explicitly write the shape of mass matrix for a planar robot like

$$M_{ij} = M_{0ij} + \sum_{k=1}^1 \sum_{k=1}^1 \cos \left(\sum q \right)$$

this also last for every segment of a serial chain that's coplanar

If mass matrix overall structure is known, then also dynamics and equations of motion structure are determined.

In addition, as consequence of what mentioned before, we can use mass matrix minimal or nearly minimal set of parameters for inertial parameters identification.

It should be underlined that considerations made for planar robots extend to a generic robot where a subset of links have aligned parallel axis

Generalized inertia matrix for multiple degree of freedom kinematic tree

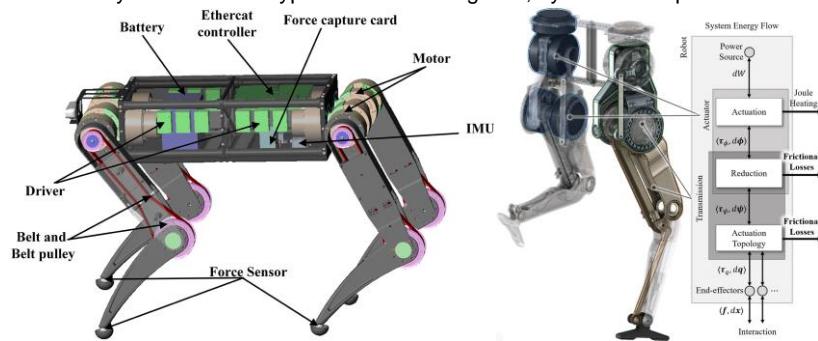
Adopting Denavit Hartenberg convention symbolically is possible to define a kinematics tree with joint parameters $(\alpha, a, d, \theta)_k$ fully symbolic. In this way results are more general and the same model of inertia matrix (then of dynamics) could be reused to describe an arbitrary robot, independently from effective axis orientation (and type), just having the same number of degrees of freedom. Theoretically it would be possible also to use more degrees of freedom and fix some of them, reducing some parameters to zero, simplifying the model. The problem is that the resulting symbolic model is by far more complex and comprising longer expressions, and even for a few degrees of freedom (usually less than 10) could become impractical for automatic computing overcoming time or memory limits. Nonetheless the method has been proved to be applicable for robots up to 8 dof, compatible with most used robots in practical applications. Results are not reported in this text for sake of brevity.

Empirical study of inertia matrix structure

An empirical study has been conducted on the composition of terms of inertia matrix for different numbers of degrees of freedom. Results are not reported for brevity but can be found in [Empirical study of inertia matrix](#).

Actuation, actuators and control

Actuators supply the motive power for robots. Most robot actuators are commercially available components, which are adapted or modified, as necessary, for a specific robot application. Three commonly used actuator types are electromagnetic, hydraulic and pneumatic



The most common types of actuators in robots today are electromagnetic actuators. Specifically joint actuation in robots usually occurs by means of servo motors.

Servomotors

Servomotors are designed to accurately follow the desired position, velocity and torque which change frequently and sometimes abruptly. They have structures similar to ordinary electric motors, but with low inertia and large torque capable for high accelerations. Typical servo motors used for robotic applications are permanent magnet (PM) DC motors and brushless DC ([BLDC](#)) motors. PM DC motors are widely used as a servo motor because of high torque, speed controllability over a wide range, well-behaved torque-speed characteristics, and adaptability to various types of control methods. The DC motor converts electrical energy into rotational or linear mechanical energy. It comes in many different types and configurations. The lowest-cost PM motors use ceramic (ferrite) magnets and robot toys and hobby robots often use this type of motor. A PM motor with a rare-earth (neodymium-iron-boron), [NEO](#) magnet stator produces the most torque and power for its size. Brushless motors, also called AC servomotors or brushless DC motors, are widely used in industrial robots. They substitute magnetic or optical sensors and electronic switching circuitry for the graphite brushes and copper bar commutator used in brush-type DC motors, thus eliminating the friction, sparking, and wear of commutating parts. Brushless motors generally have good performance at *low cost* because of the decreased complexity of the motor. However, the controllers for these motors are more complex and expensive than brush-type motor controllers. A passive multi-pole neodymium magnet rotor and a wire-wound iron stator of a brushless motor provide good heat dissipation and excellent reliability. Linear brushless motors function like unrolled rotary motors. They typically have a long, heavy, multiple magnet passive stator and a short, lightweight, electronically commutated wire-wound forcer (slider).

By the way other actuators exist and are applied for some specific applications where servo motor dimensions are not applicable because of encumbrance, like in case of robot hands and grippers where pulley or string based actuation is preferable.

Other actuators from electromechanical are

- Pneumatics
- Hydraulics
- Pulley
- Piezoelectric
- Shape Memory alloys
- Electroactive polymers
- Magnetically driven
- Heat driven
- Artificial muscles

Actuator type	Functionality	Used by	Hardware
Electric motors	- Locomotion; - Movement; Grabbing & moving objects; Pose control;	Operators	- Robotic joints (rotary, prismatic); - Stepper motors; - Linear motors; - Etc.
Artificial muscles	- Precise limb movement.		- Collision sensors, - Force sensing resistors (FSRs), contact sensors.
Pneumatic	- Used in industry for diverse purposes, but not used for mobile robotics.		- Capacitive proximity sensors;
Hydraulic			- photo-electric sensors. - Humidity, temperature, pressure.
Shape memory alloys	- Small movements.		- Artificial muscles
Electro-active Polymers (EAPs)	- Biological muscle behavior emulation.		

Type of actuator are results of design choice and applications of use. For instance historically actuation of robotic hands and grippers was conditioned by small dimensions and relatively low torques, leading to unconventional actuation methods respect most traditional one. Specifically the use of pulley is still not uncommon, and thermo actuated and memory shaped alloy was also studied.

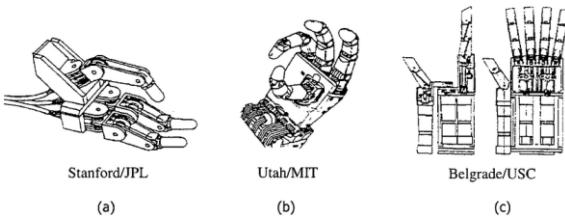
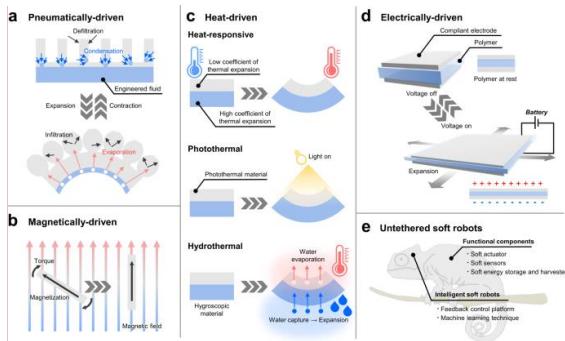


Table I. Characteristics of some robot hands.

Characteristic/ Robot hand	Type of Actuator	Number of finger	Joint of thumb	Total Degree of freedom	Motion Transmission
UTAH/MIT	Pneumatic	4	4	16	Pulley-cord
Belgrade/USC model II	Electrical	5	2	16	Special mechanism
JPL Standford	Electrical	3	3	9	Pulley-cord
METUHAND	Electrical	3	3	9	Pulley-cord
MAT/METU	Pneumatic	4	2	4	Pulley-cord
SMA Hand*	Electrical	3	Effect of shape memory	Special Structure	Heating/cooling

* The robot hand developed in this study.

As well as new robotics trend including soft robots and continuous robots are leading to alternative types of actuations not considered before.



Commented [11]: Superfluous

Design criteria for sizing and selection of actuators are:

- Power
- Maximum torque
- Range of actuation
- Maximum speed
- Supply
- Size
- Weight
- Robustness
- Reliability
- Fault tolerance

These criteria are not only relevant for design and selection, but becomes important also when developing a digital twin willing to respect a real physical layout, from different perspectives, and in order to provide model *requirements* to associate to each model component. Each joint has specific requirements and demands to satisfy, potentially leading to different actuation choices.

Transmission

The purpose of a transmission or drive mechanism is to transfer mechanical power from a source to a load. The design and selection of a robot drive mechanism requires consideration of motion, load, and power requirements and the placement of the actuator with respect to the joint. The primary considerations in transmission design are stiffness, efficiency, and cost. Backlash and windup impact drive stiffness especially in robot applications where motion is constantly reversing and loading is highly variable. High transmission stiffness and low or no backlash result in increased friction losses. Most robot transmission elements have good efficiencies when they are operating at or near their rated power levels but not necessarily when lightly loaded. Larger than necessary drives add weight, inertia and friction loss to the system. Underdesigned drives have lower stiffness, can wear rapidly in continuous or in high duty cycle operation or fail due to accidental overloads. Joint actuation in robots is generally performed by drive mechanisms which interface the actuator (mechanical work source) to the robot links through the joints in an energy-efficient manner. A variety of drive mechanisms are incorporated in practical robots. The transmission ratio of the drive mechanism sets the torque, speed, and inertia relationship of the actuator to the link. Proper placement, sizing, and design of the drive mechanisms set the stiffness, mass, and overall operational performance of the robot. Most modern robots incorporate efficient, overload damage resistant, back-driveable drives.

Drives types

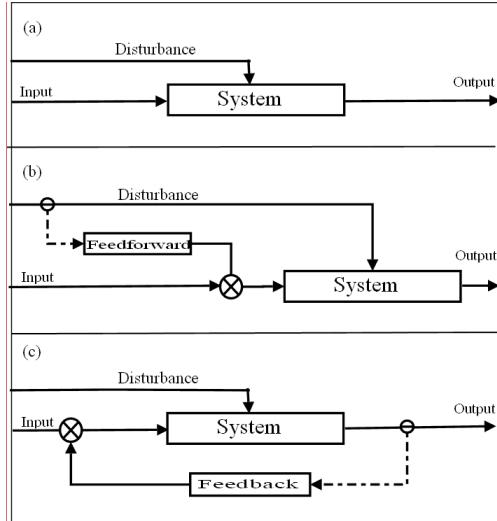
Typically used drives consist in:

- Direct drives
- Band drives
- Belt drives
- Gear drives
- Worm Gear drives
- Proprietary drivers
- Linear drives
- Ball screw drives
- Rack-pinion drives

Joint actuation and control

Typical robot joint control works based on one or multiple feedback loops, working most of the time on input position. In many cases, in order to increase control accuracy, [feedforward](#)

contributes are added based on an inverse dynamic model, requiring a good estimate of dynamic parameters.



Commented [12]: Not strictly necessary

Motion control can occur in joint space or in task space. typical control law scheme used PID control (simpler feedback based scheme) or computed torque law based (feed forward based control)

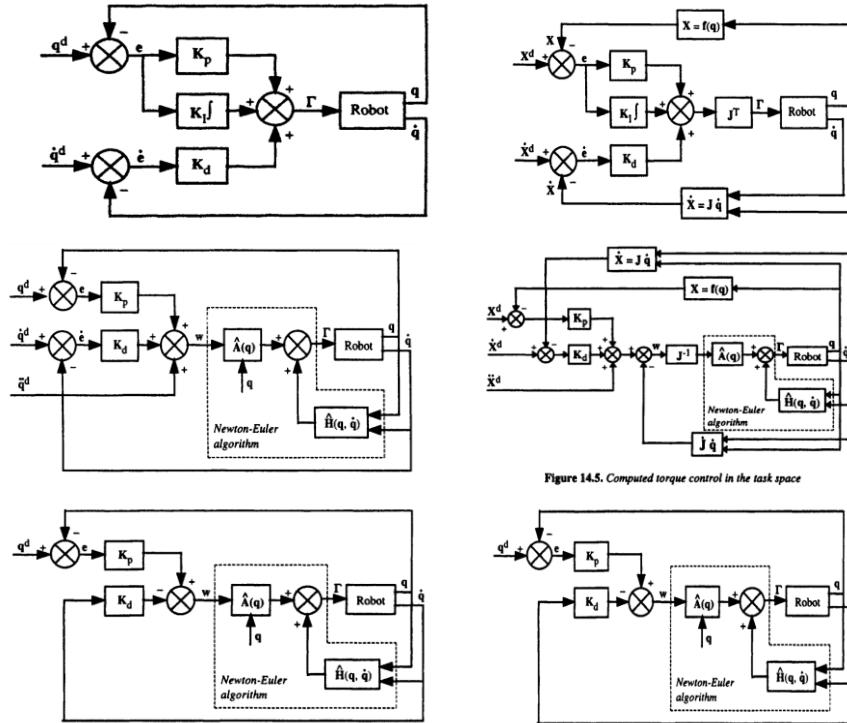
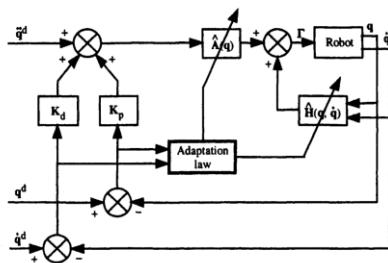


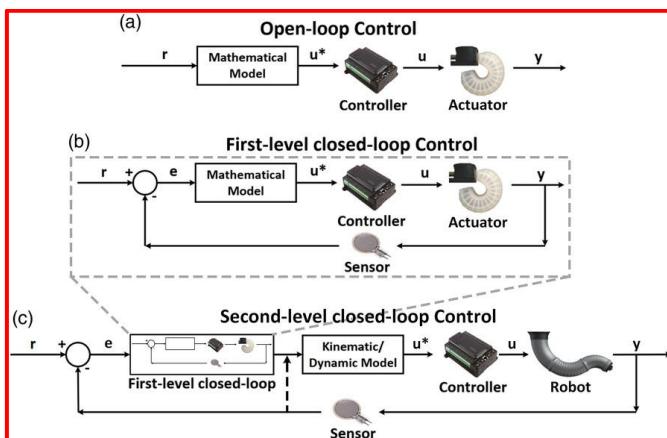
Figure 14.5. Computed torque control in the task space

Tracked desired motion speed and acceleration can be given in input or limit to position computing a simpler error law. In computed torque law Adaptive control schemes are adopted when in computed torque law dynamic parameters are associated with an on-line identification law providing parameters estimate \hat{J} based on current observations.



Multiple control loops

In control applications it is not uncommon to have multiple loops of control in order to assure feedback on more than one control variable sensed, while adoption of open loop control is virtually not adopted, because it would imply a perfect model of system behavior concretely never fully accessible (pure feed forward control), to be assured in any condition. In spite of its simplicity it would expose to a risky operation. Including at least one feedback loop assure a better control on both performance and system response. Most of simpler control schemes seen so far are at first level loop closure, but the use of multiple levels of closure with inner and outer loops is very common in applicative fields.



In spite of the fact that most control loops for traditional robot control are based on motion control imposed (position and derivatives) there's also a rich thread of literature studying force control mode.

Force control

A fundamental requirement for the success of a manipulation task is the capability to handle the physical contact between a robot and the environment. Pure motion control turns out to be inadequate because the unavoidable modeling errors and uncertainties may cause a rise of the contact force, ultimately leading to an unstable behavior during the interaction, especially in the presence of rigid environments. Force feedback and force control becomes mandatory to achieve a *robust* and *versatile* behavior of a robotic system in poorly structured environments as well as safe and dependable operation in the *presence of humans*. Strategy consists in the analysis of *indirect force control*, conceived to keep the contact forces *limited* by ensuring a suitable compliant behavior to the end effector, without requiring an accurate model of the environment. **The second problem consisting in the interaction tasks modeling in the case of a rigid environment and in the case of a compliant environment. For the specification of an interaction task, natural constraints set by the task geometry and artificial constraints set by the control strategy are established, with respect to suitable task frames.**

This formulation is the essential premise to the synthesis of hybrid force/motion control schemes.

From Indirect Force Control to Hybrid Force/Motion Control

Active interaction control strategies can be grouped into two categories: those performing *indirect force control* and those performing *direct force control*. The main difference between the two categories is that the former achieve force control via motion control, without explicit closure of a force feedback loop; the latter instead offer the possibility of controlling the contact force and moment to a desired value, thanks to the closure of a force feedback loop.

To the first category belongs *impedance control* (or *admittance control*), where the deviation of the end-effector motion from the desired motion due to the interaction with the environment is related to the contact force through a mechanical impedance/admittance with adjustable parameters. A robot manipulator under impedance (or admittance) control is described by an equivalent mass–spring–damper system with adjustable parameters. This relationship is an impedance if the robot control reacts to the motion deviation by generating forces, while it corresponds to an admittance if the robot control reacts to interaction forces by imposing a deviation from the desired motion

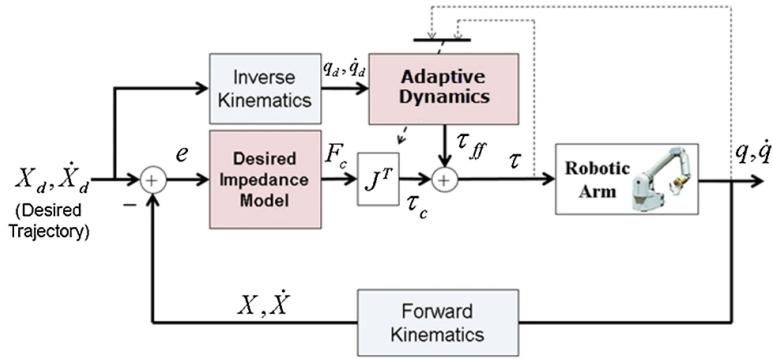
If an accurate model of the environment is not available, the force control action and the motion control action can be superimposed, resulting in a *parallel force/position* control scheme. In this approach, the force controller is designed so as to dominate the motion controller; hence, a position error would be tolerated along the constrained task directions in order to ensure force regulation [9.19].

Stiffness control

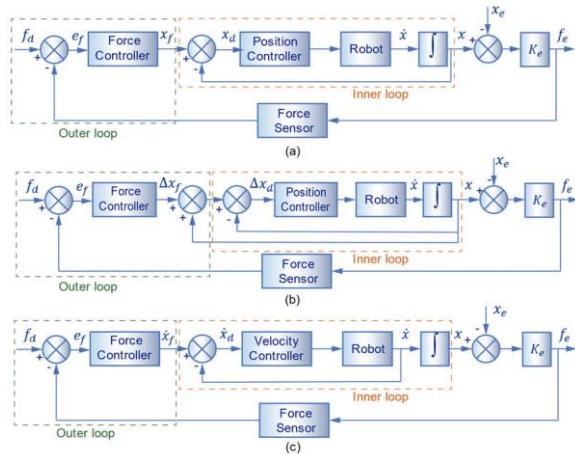
Approach consisting of assigning a desired position and orientation and a suitable *static relationship* between the deviation of the end-effector position and orientation from the desired motion and the force exerted on the environment, is known as *stiffness control*.

Impedance control

Stiffness control is designed to achieve a desired static behavior of the interaction. In fact, the dynamics of the controlled system depends on that of the robot manipulator, which is nonlinear and coupled. A more demanding objective may be that of achieving a desired dynamic behavior for the end-effector, e.g., that of a second-order mechanical system with six degrees of freedom, characterized by a given mass, damping, and stiffness, known as *mechanical impedance*



Objective of direct force control can be achieved including then a second outer loop on motion control loop (absolute or incremental or velocity driven), deputed to force control



Typical robot actuator structure

Typical actuators include a motor, transmission gears (planetary, harmonic or else), measurement encoder/position sensor, mechanical joint. Presenting a typical legged robot and articulated arm actuator we have some sketches from the web:

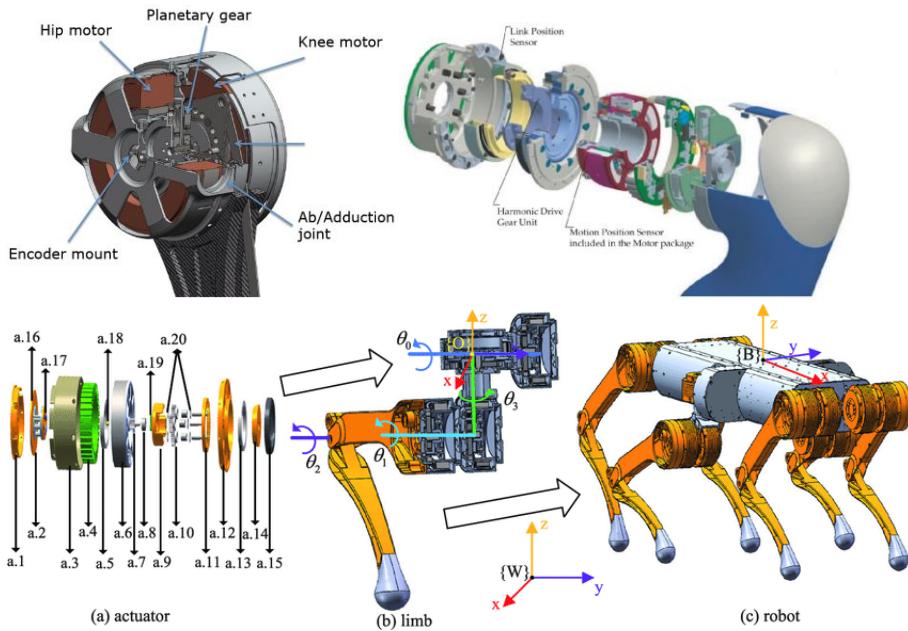
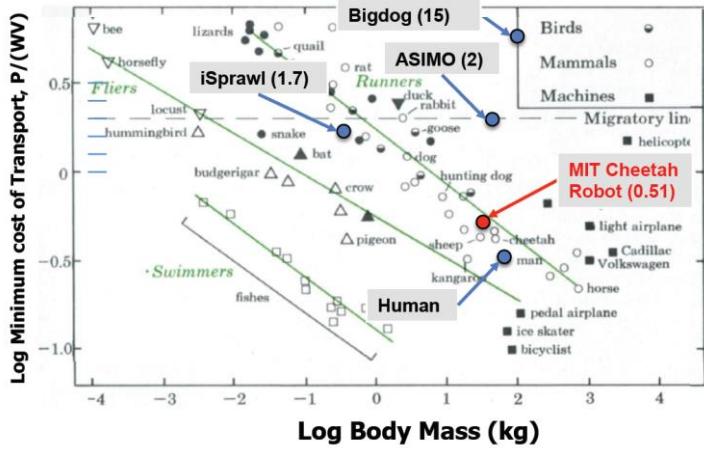


Diagram of the robot from actuator, limb to the whole machine structure. (a) Actuator is consist of (a.1) end cover, (a.2) driver board mounting frame, (a.3) stator support, (a.4) stator coil, (a.5) bearing, (a.7) sun gear, (a.8) center bearing, (a.9) planet carrier, (a.10) planet gear, (a.11) ring gear, (a.12) motor side hood, (a.13) cross ball bearing, (a.14) planet carrier, (a.15) bearing end plate, (a.16) driver board, (a.17) hall sensor, (a.18) encoder magnet, (a.19) hall magnet, (a.20) gasket. (b) The designed 4 DOFs middle limb. (c) The whole robot. The novel design characterise the absolute position actuator based on the mixed encoder and halls. The initial position of the actuator is positioned within 60 degrees by collecting six uniformly distributed hall sensors, and then the absolute initial position is accurate to the absolute initial position through the encoder. The local limb coordinate system is plot in the middle figure. The workspace of the middle limbs are large enough to locomotion as legs and manipulation as arms at the same time.

Beyond classic architectures, we shall remark that key of actuators design consist of desired performances in terms of, torque, speed, accuracy, mass, volume, weight, especially for collaborative and moving robots. A comparison on two is provided in chart following:



Actuators friction

The Lagrangian and Newton–Euler dynamics do not account for friction at the joints, but the friction forces and torques in gearheads and bearings may be significant. Friction is a complex phenomenon that is the subject of considerable current research; any friction model is a gross attempt to capture the average behavior of the micromechanics of contact. Friction models often include a *static friction* term and a velocity-dependent *viscous friction* term. The presence of a static friction term means that a nonzero torque is required to cause the joint to begin to move. The viscous friction term indicates that the amount of friction torque increases with increasing velocity of the joint. See figure for some examples of velocity-dependent friction models. Other factors may contribute to the friction at a joint, including the loading of the joint bearings, the time the joint has been at rest, the temperature, etc. The friction in a gearhead often increases as the gear ratio increases.

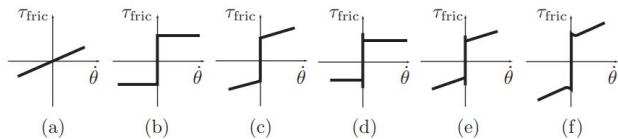


Figure 8.13: Examples of velocity-dependent friction models. (a) Viscous friction, $\tau_{\text{fric}} = b_{\text{viscous}}\dot{\theta}$. (b) Coulomb friction, $\tau_{\text{fric}} = b_{\text{static}} \operatorname{sgn}(\dot{\theta})$; τ_{fric} can take any value in $[-b_{\text{static}}, b_{\text{static}}]$ at zero velocity. (c) Static plus viscous friction, $\tau_{\text{fric}} = b_{\text{static}} \operatorname{sgn}(\dot{\theta}) + b_{\text{viscous}}\dot{\theta}$. (d) Static and kinetic friction, requiring $\tau_{\text{fric}} \geq |b_{\text{static}}|$ to initiate motion and then $\tau_{\text{fric}} = b_{\text{kinetic}} \operatorname{sgn}(\dot{\theta})$ during motion, where $b_{\text{static}} > b_{\text{kinetic}}$. (e) Static, kinetic, and viscous friction. (f) A friction law exhibiting the Stribeck effect – at low velocities, the friction decreases as the velocity increases.

Stribeck friction and Coulomb friction

Friction models present in literature are diverse for type and complexity. A short resume of most commonly used and applied in robotics [Fabris].

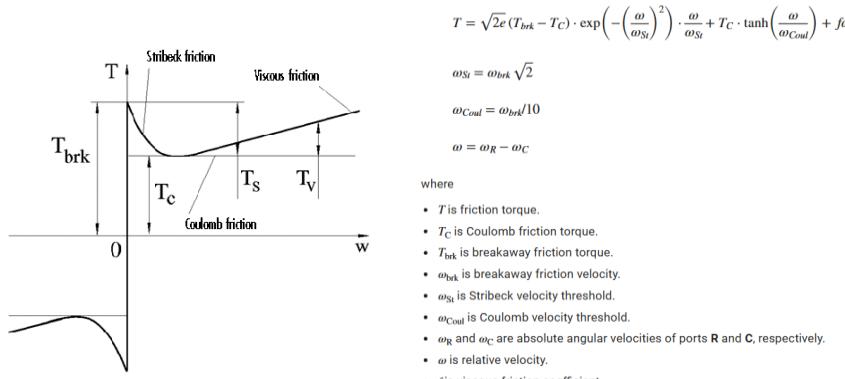
Table 1: Overview on the friction models considered in this paper.

Model	Year	Ref.	Formulation
Bittencourt	2012	[9]	$\tau_{f,th} = \left(a_1 + a_2 e^{-\left(\frac{\dot{q}}{a_3}\right)^{a_4}} \right) \text{sign}(\dot{q}) + a_5 \dot{q}$
Coulomb	1773	[7]	$\tau_{f,th} = a_1 \dot{q} + a_2 \text{sign}(\dot{q})$
Dong	2021	[10]	$\tau_{f,th} = a_1 \dot{q} + a_2 \text{sign}(\dot{q})(1 - e^{-a_3 \text{sign}(\dot{q}) \dot{q}}) + a_4 \text{sign}(\dot{q}) e^{-a_5 \text{sign}(\dot{q}) \dot{q}}$
Gaz (A)	2018	[13]	$\tau_{f,th} = a_1 \dot{q} + a_2 + \frac{a_3}{1 + e^{-a_4(\dot{q} + a_5)}}$
Gaz (B)	2019	[3]	$\tau_{f,th} = \frac{a_1}{1 + e^{-a_2(\dot{q} + a_3)}} - \frac{a_1}{1 + e^{-a_2 \dot{q}}}$
Grotjahn (A)	2001	[8]	$\tau_{f,th} = a_1 \dot{q} + a_2 \text{sign}(\dot{q}) + a_3 \dot{q}^{1/3}$
Grotjahn (B)	2001	[8]	$\tau_{f,th} = a_1 \dot{q} + a_2 \text{sign}(\dot{q}) + a_3 \arctan(a_4 \dot{q})$
Hao	2021	[6]	$\tau_{f,th} = a_1 \text{sign}(\dot{q}) + a_2 \dot{q} + a_3 \dot{q}^2 \text{sign}(\dot{q}) + a_4 \dot{q}^3$
Indri	2013	[12]	$\tau_{f,th} = a_1 S_0 + a_2 \frac{2}{\pi} \arctan(a_3 \dot{q}) + a_4 \dot{q} + (a_5 \dot{q}^2) S_0$ with $S_0 = \frac{2}{\pi} \arctan(a_6 \dot{q})$
Makkar	2005	[11]	$\tau_{f,th} = a_1 (\tanh(a_2 \dot{q}) - \tanh(a_3 \dot{q})) + a_4 \tanh(a_5 \dot{q}) + a_6 \dot{q}$

Usually models of actuator frictions employed in robotic modeling consider a combination of viscous of first and higher order, coulomb friction. These kind of model allow to keep linearity of model of friction parameters suitable for a reliable identification from experimental data.

Besides is not uncommon the adoption of nonlinear models, more close to real physics.

The *Stribeck friction*, is the negatively sloped characteristics taking place at low velocities [1]. The *Coulomb friction*, T_C , results in a constant torque at any velocity. The *viscous friction*, T_V , opposes motion with the torque directly proportional to the relative velocity. The sum of the Coulomb and Stribeck frictions at the vicinity of zero velocity is often referred to as the *breakaway friction*, T_{brk} . The friction is approximated with the following equations:



The exponential function used in the Stribeck portion of the force equation is continuous and decays at velocity magnitudes greater than the breakaway friction velocity. The hyperbolic tangent function used in the Coulomb portion of the force equation ensures that the equation is smooth and continuous through $\omega = 0$, but quickly reaches its full value at nonzero velocities. The block positive direction is from port **R** to port **C**. This means that if the port **R** velocity is greater than that of port **C**, the block transmits torque from **R** to **C**.

Joint and Link Flexibility

In practice, a robot's joints and links are likely to exhibit some flexibility. For example, the flexspline element of a harmonic drive gearhead achieves essentially zero backlash by being somewhat flexible. A model of a joint with harmonic drive gearing, then, could include a relatively stiff torsional spring between the motor's rotor and the link to which the gearhead is attached. Similarly, links themselves are not infinitely stiff. Their finite stiffness is exhibited as vibrations along the link. Flexible joints and links introduce *extra states* to the dynamics of the robot, significantly complicating the dynamics and control. While many robots are designed to be stiff, in order to minimize these complexities, in some cases this is impractical owing to the extra link mass required to create the stiffness.

Actuator dynamic model

Figure shows a brushed DC motor with an encoder and a gearhead. The torque τ , measured in newton-meters (N m), created by a DC motor is governed by the equation

$$\tau = k_t I$$

where I , measured in amps (A), is the current through the windings. The constant k_t , measured in newton-meters per amp (N m/A), is called the torque constant.

The power dissipated as heat by the windings, measured in watts (W), is governed by

$$P_{heat} = R I^2$$

where R is the resistance of the windings in ohms (Ω). To keep the motor windings from overheating, the continuous current flowing through the motor must be limited. Accordingly, in continuous operation, the motor torque must be kept below a continuous-torque limit τ_{cont} determined by the thermal properties of the motor. A

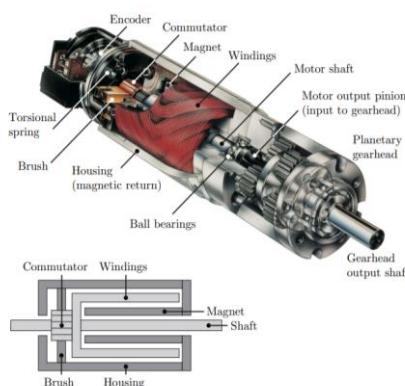


Figure 8.9: (Top) A cutaway view of a Maxon brushed DC motor with an encoder and gearhead. (Cutaway image courtesy of Maxon Precision Motors, Inc., maxonmotorsusa.com.) The motor's rotor consists of the windings, commutator ring, and shaft. Each of the several windings connects different segments of the commutator, and as the motor rotates, the two brushes slide over the commutator ring and make contact with different segments, sending current through one or more windings. One end of the motor shaft turns the encoder, and the other end is input to the gearhead. (Bottom) A simplified cross-section of the motor only, showing the stator (brushes, housing, and magnets) in dark gray and the rotor (windings, commutator, and shaft) in light gray.

simplified model of a DC motor, where all units are in the SI system, can be derived by equating the electrical power consumed by the motor $P_{elec} = V \cdot I$ in watts (W) to the mechanical power $P_{mech} = \tau\omega$ (also in W) and other power produced by the motor,

$$V \cdot I = \tau\omega + R \cdot I^2 + L \cdot I \frac{dI}{dt} + \text{friction + other power-loss terms}$$

and other power-loss terms, where V is the voltage applied to the motor in volts (V), w is the angular speed of the motor in radians per second (1/s), and L is the inductance due to the windings in henries (H). The terms on the right-hand side are the mechanical power produced by the motor, the power lost to heating the windings due to the resistance of the wires, the power consumed or produced by energizing or de-energizing the inductance of the windings (since the energy stored in an inductor is $\frac{1}{2}LI^2$, and power is the time derivative of energy), and the power lost to friction in bearings, etc. Dropping this last term, replacing terms, and dividing by I, we get the voltage equation

$$V = k_t\omega + R \cdot I + L \frac{dI}{dt}$$

Often written with the electrical constant k_e (with units of V s) instead of the torque constant k_t , but in SI units (V s or N m/A) the numerical values of the two are identical; they represent the same constant property of the motor. So we prefer to use k_t . The voltage first term is called the *back electromotive force* or *back-emf* for short, and it is what differentiates a motor from being simply a resistor and inductor in series. It also allows a motor, which we usually think of as converting electrical power to mechanical, to be run as a generator, converting mechanical power to electrical. If the motor's electrical inputs are disconnected (so no current can flow) and the shaft is forced to turn by some external torque, you can measure the back-emf voltage across the motor's inputs.

The operating region of the motor in the torque–speed plane is as shown in figure. Note that the signs of τ and ω are opposite in the second and fourth quadrants of this plane, and therefore the product $\tau\omega$ is negative. When the motor operates in these quadrants, it is actually consuming mechanical power, not producing mechanical power. The motor is acting like a damper. Focusing on the first quadrant ($\tau \geq 0, \omega \geq 0, \tau\omega \geq 0$), the boundary of the operating region is called the speed–torque curve. The no-load speed $\omega_0 = V_{max}/k_t$ at one end of the speed–torque curve is the speed at which the motor spins when it is powered by V_{max} but is providing no torque. In this operating condition, the back-emf $k_t\omega$ is equal to the applied voltage, so there is no voltage remaining to create current (or torque). The stall torque $\tau_{stall} = k_t V_{max}/R$ at the other end of the speed–torque curve is achieved when the shaft is blocked from spinning, so there is no back-emf. Figure also indicates the continuous operating region where $|\tau| < \tau_{cont}$. The motor may be operated intermittently outside the continuous operating region, but extended operation outside the continuous operating region raises the possibility that the motor will overheat. The motor's rated mechanical power is $P_{rated} = \tau_{cont}\omega_{cont}$. Even if the motor's rated power is sufficient for a particular application, the torque generated by a DC motor is typically too low to be useful. As mentioned earlier, gearing is therefore used to

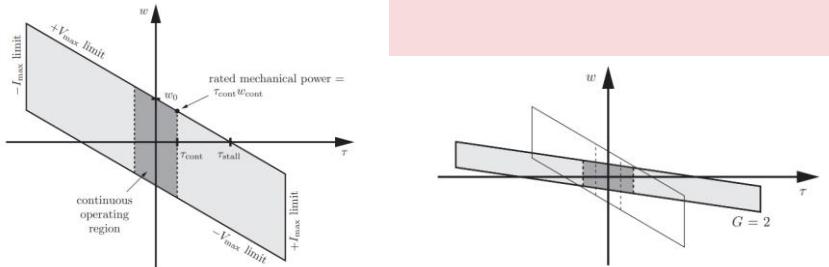
increase the torque while also decreasing the speed. For an ideal gearhead, no power is lost in the torque conversion. Power balance gives

$$\begin{aligned}\omega_{\text{gear}} G &= \omega_{\text{motor}} \\ \tau_{\text{gear}} &= G \tau_{\text{motor}}\end{aligned}$$

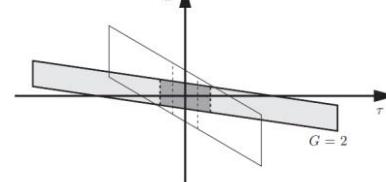
In practice, some mechanical power is lost due to friction and impacts between gear teeth, bearings, etc., so

$$\tau_{\text{gear}} = \eta G \tau_{\text{motor}}$$

where $\eta \leq 1$ is the efficiency of the gearhead.



Commented [13]: Need to be commented?



The motor's stator is attached to one link and the rotor is attached to another link, possibly through a gearhead. Therefore, when calculating the contribution of a motor to the masses and inertias of the links, the mass and inertia of the stator must be assigned to one link and the mass and inertia of the rotor must be assigned to the other link. Consider a stationary link 0 with the stator of the joint-1 gearmotor attached to it. If you were to grab link 1 and rotate it manually, the inertia contributed by the rotor would feel as if it were a factor G^2 larger than its actual inertia, owing to the gearhead. This is the *apparent inertia* (often called the *reflected inertia*) of the rotor about the axis. One consequence as the gear ratio becomes large is that the inertia seen by joint i becomes increasingly dominated by the apparent inertia of the rotor, the robot's mass matrix becomes more diagonal, the dynamics of the robot are decoupled – the dynamics at one joint has no dependence on the configuration or motion of the other joints.

Speed Ratios

The speed ratios of a robot relate the velocity F_p of a point F_p in the end-effector to the joint rates

$$\dot{q} = (\dot{q}_1, \dot{q}_2, \dots, \dot{q}_m)^T$$

that is

$$F_p \dot{v} = v + \omega \times (F_p - d)$$

$F_p \dot{v} \propto C! F_p \dot{d}$; (4.4) where d and v are the position and velocity of a reference point, respectively, and $!$ is the angular velocity of the end-effector. The vectors v and $!$ depend on the joint rates qP_j through the formula

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{\partial v}{\partial q_1} & \frac{\partial v}{\partial q_2} & \cdots & \frac{\partial v}{\partial q_m} \\ \frac{\partial \omega}{\partial q_1} & \frac{\partial \omega}{\partial q_2} & \cdots & \frac{\partial \omega}{\partial q_m} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_m \end{pmatrix}$$

or

$$v = J(q)\dot{q}$$

The coefficient matrix J in this equation is called the Jacobian and is a matrix of speed ratios relating the velocity of the tool to the input joint rotation rates.

Speed Ratios

A six-axis robot has a 6x6 Jacobian J obtained from [4.5] that is an array of speed ratios relating the components of the velocity v of the wrist center and the angular velocity of the end-effector to each of the joint velocities. Equation [4.9] shows that this Jacobian defines the force-torque vector f exerted at the wrist center in terms of the torque applied by each of the actuators. The link parameters of the robot can be selected to provide a Jacobian J with specific properties. The sum of the squares of the actuator torques of a robot is often used as a measure of effort [4.22, 23]. From equations we have

Commented [14]: correct equations

$$\tau^T \tau = f^T J J^T f$$

The matrix $J \cdot J^T$ is square and positive definite. Therefore, it can be viewed as defining a hyperellipsoid in six-dimensional space [4.24]. The lengths of the semi diameters of this ellipsoid are the inverse of the absolute value of the eigenvalues of the Jacobian J . These eigenvalues may be viewed as modal speed ratios that define the amplification associated with each joint velocity. Their reciprocals are the associated modal mechanical advantages, so the shape of this ellipsoid illustrates the force amplification properties of the robot. The ratio of the largest of these eigenvalues to the smallest, called the condition number, gives a measure of the anisotropy or out-of-roundness of the ellipsoid. A sphere has a condition number of one and is termed isotropic. When the end-effector of a robot is in a position with an isotropic Jacobian there is no amplification of the speed ratios or mechanical advantage. This is considered to provide high-fidelity coupling between the input and output because errors are not amplified [4.25, 26]. Thus, the condition number is used as a criterion in a robot design [4.27]. In this case, it is assumed that the basic design of the robot provides a workspace that includes the task space. Parameter optimization finds the internal link parameters that yield the desired properties for the Jacobian. As in minimizing the distance to a desired workspace, optimization based on the Jacobian depends on a careful formulation to avoid coordinate dependency.

Parameters identification and filtering

Generalities on estimation and identification theory

In this part we're going to explore a topic with a significant relevance in robotics literature and beyond it, adding also a considerable interdisciplinary value to this work and to many other publications. Different branches of research and academic publications are active on these subjects. Topics extend by far from robotics and involve many different subjects from different engineeristic fields.

Identification, estimate, filtering and learning process are intimately connected. They're different aspects of the same problem: acquiring information (learning) about a system (model or system and parametric identification) , using statistical information (estimate) from observations (signals or measurements) , removing unwanted components (filtering). All of them regard the same process but focus on different perspectives and characteristics. Keeping it in mind makes it easier to find interdisciplinary connections in different fields, from engineeristic applications, statics and measurements, information theory, machine learning and last but not least artificial intelligence. Remarkably we're going to talk about model based identification here (white box or gray box), but nonetheless black box identification (black box), also known as data driven identification, is gaining significant interest from researchers in the field nowadays, as a potential breakthrough. Indeed model knowledge assumptions is not always such a reliable start point in real applications, especially where system and environment are not fully controlled, isolated, or predefined.

Identification process is based on acquiring observations for deriving statistical estimates of system parameters, to be used for other purposes, when they're not accessible directly as observations, for instance by measurements.

Since early years of robotics identification of robot dynamic parameters together with control have been a key topic of advancement, still live today because of its difficulties

Identification in general can regard many topics, but in classic robotics generally attention is focused on dynamic and geometric parameters. First are used as input for fine tuning of feedforward model based control laws. for high speed and precision applications otherwise not achievable. Target parameters are then named dynamic or inertial parameters because related to inertial properties of robot links or dynamic loads acting on joints from inertial wrenches. For the same purpose the identification of friction parameters (part of dynamics parameters) on joint is similarly relevant. Finally, estimated payload inertial parameters like mass and inertia are fundamental for achieving both high movement and speed precision, as well for estimating joint torque loads.

In some parts of this work we adopt nomenclature from one author and somewhere else from others, but substantially nothing changes. We hope readers will be able to understand from context.

Parameters identification and estimation (learning)

Types of identification

There are different types of identification found in literature. Most common identification problems found in engineeristic applications regard *parametric identification*.

Parametric identification is used when a *model* is known or assumed as a function of a set of parameters and its final model identification results from finding parameters best estimate sets that minimize error between observations and expected values (prediction). Notably this process is essentially a key process used in any machine learning method.

Identification can be seen as a process associating a final estimate to a set of inputs could be of different nature, like a priori information, raw direct measurements, indirect measurements, pre-processed observations, simulations, and others.

A greatly impacting element of classification of identification is the nature of estimated quantities or parameters:

- discrete
- continuous
- interval based
- constrained values
- Mixtures of all above and others

The second element is the model used that could be or not continuous and/or differentiable on parameters type mentioned.

A special care is deserved for parameter consistency. Indeed most algorithms, especially LSQ, do not consider any constraint on parameters signs. If all is good this shouldn't be an issue. Nonetheless a consistency condition, at least a posteriori should be considered in order to validate the solution. Some parameters for instance should be strictly positive, like masses and principal inertia. Other kinds of constraints could arise from geometrical considerations.

Machine learning algorithms are usually classified into regression and classification algorithms. But generally speaking this is a little forcing. For instance Neural Networks that are able to approximate most of these algorithms functions just using proper activation functions, open the way to mixed situations where output is neither a pure regression nor a pure classification but something midway, even in case we're not using a pure NN based method. Thus estimated parameters seen from a more general point of view could be a first effective application of the mixture concept presented.

In association with this mixed method we could also enframe some fuzzy method where a logical or discrete output or function could be replaced by a real value function.

Especially solution method adopted is fundamentally affected. For instance LSQ methods are applicable only for models differentiable on parameters interval considered, or otherwise modified techniques shall be considered.

In case of presence of discrete parameters is required the use of some ILS, or MILS methods, in combination with eventual required constraints. Also machine learning methods could be applicable

Is not uncommon for the case where the range of parameters is even unbounded, but a priori knowledge drive our estimate to consider more likely values surrounding a priori known. Indeed estimate essentially is a statistical process affected by uncertainty on input as output as well. Statistical properties of used observations greatly affect statistical properties of inputs.

Indeed assumptions regarding statistical properties of observations are of fundamental importance to final estimate quality.

Another issue regards the uniqueness of the solution, because in case multiple solutions exist we should provide additional criteria of selection or keep in mind ambiguity associated with the solution.

Last but not least residuals of solutions provide further insight on fitness, since there may be reasons to discard models if residuals provide evidence of information content.

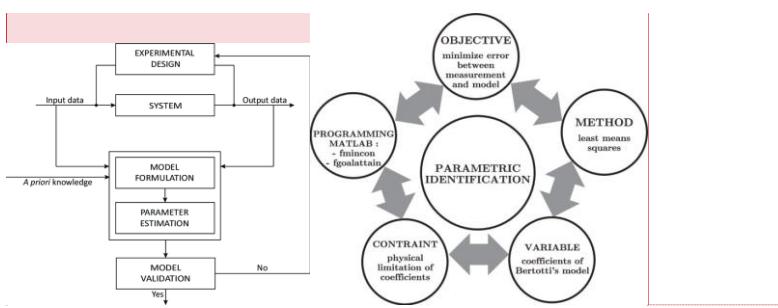
Sometimes what we need is indeed not parameter identification but model identification. In some specially simple cases like those of linear systems it is effectively possible to model identification by means of fourier analysis of frequency domain transfer function. This method can be extended to slightly nonlinear and perturbed systems with some cautions (because nonlinearities introduce effects like frequency multiplications and limit cycles could significantly change model interpretation). A more general model identification approach, which recall and extend previous method makes use of a general basis of functions (usually eigenfunctions of some differential operator whether applicable at least as approximation of real system) and develop a generic response as linear combination of base vector elements (even in converging series in case). These methods quite general applies also numerically for finding best approximation of a fitting model for a given system.

Model identification requires also understanding of the nature of the model we're considering, for instance if it is a derivative involving a model or not, which are dependent and independent variables. Models could indeed consist on a PDE, or on a ODE, or in a simple functional relation, not necessarily bijective, but could also be a non monodrome function or an hysteresis cycle. Different types of problems require different approaches.

When nonlinearities are present the problem indeed becomes by far more hard to solve. Fortunately in most identification problems we deal with linearity in parameters is an assumption that holds true because of a priori knowledge or assumptions better about the system.

Conversely non parametric identifications rely on non parametric statistics where we relax assumptions on statistical distributions applicable. This approach is quite different and exploit other methods that literature applicative mainstream do not consider, privileging model based parametric estimation for its reliability, since models can be carried out with limited uncertainty.

Commented [15]: Keep or leave?



Commented [16]: Eliminate

Robot parameters identification

When identification is applied to robotic field, most common subject of study are

- Kinematic parameters defining system kinematics
- Inertial parameters (identifiable links mass, center of mass position, and inertia tensor components)
- Standard inertia parameters (full set of inertial parameters characterizing each link)
- Friction parameters
- Thermal parameters (currently object of interest because affecting friction parameters and operative performances)
- Dynamic parameters (combination of inertial and friction parameters), named like this because determining robot dynamics
- Geometrical parameters, affecting inertial parameters
- Electrical parameters (actuators related)
- Loads estimation and trajectories optimization

Potentially other parameters could be included into the identification model, but usually these are most used for following reasons. Robot dynamics is often occurring in a controlled environment where minimal disturbances are desired in order to get good performances.

Consider moreover that linearity on estimated parameters could be a primary rationale for parameter selection for identification, since inclusion of non linearly dependent parameters would lead to a change in nature of the problem yielding to the need for more complex solutions methods, potentially less accurate with respect to the linear case. A good example are geometrical parameters whose coupling with other parameters lead to nonlinear problems whose convergence is not assured. Although not impossible (non linear identification algorithms exists and are used as described in following paragraphs), this topic is beyond purpose of this work, and where for linear model solution is unique and granted, this do not last usually for non linear models increasing unreliability of method and results., or imply larger computational burdens. Consequently a good estimation strategy can consist in separating estimation problems of different parameters types into different processes optimized independently. Geometrical terms can be partially identified using a kinematic identification.

Friction and inertial parameters are subject of interest first one because they are involved in control scheme optimization of planned trajectories and control torques calculation in closed loop.

Second one is of relevance for control purpose but also for the optimization of performances and health state monitoring. Third, the use of inertial characteristics allow for identification of loads and optimization of controls and trajectory planning

We shall first distinguish between different identification process that could be

- *On line identification*, working while operating (still not strictly requiring to be real time or synchronous, since near real time with delay or with lower rate respect task frame rate may be an acceptable compromise)
- *Off line identification*, that uses non in-operations observations for identification purpose

In literature often is made reference to off line identification because allows better results using specifically designed trajectories, while working online operative constraints could limit capabilities of identification. A lot of literature was produced basing on this approach. Other subclass of identification methods have been considered by researchers, as illustrated in the following.

The process include imposing a known *excitation trajectory* imposed to the robot through its internal position-speed control loop (indeed from practical point of view robots are controlled in position with prescribed motion trajectories, since this is natural reference approach for designing a task, where applied joint torques are just a means for. At same time it doesn't make sense to give a torque input since effective load required is a consequence of control loop and shall adapt to errors arising for any reason along control process). Nonetheless force control exist as topic in literature but is related to environmental interaction.

Once defined trajectory is decided and optimized (torque loads, speed and position limits respected, appropriate values for identification purposes) and provided it's effectively and properly followed by the robot, we can use calculated/measured control law torques (as measured indirectly from resulting motor currents, usually provided as output) as input for identification process, using the so called identification model expressing dependence of torques on identification parameters. Usually taking some samples of time history we get enough data and equations to solve on OLS or WLS problems.

Shall be kept in mind that samples quality and selection could play a primary role in determining final results quality, then should be performed properly in dependence of given trajectory.

Once identified, parameters could be used in a validation process using a validation trajectory different from initial one for checking results and evaluate a posteriori quality of regression process as suggested in machine learning methods.

Off-line identification

There are **three** main methods for *estimating* the inertial parameters of a robot:

physical experiments: if we could disassemble the robot to isolate each link, the following parameters could be obtained by physical experiment [Armstrong 86]:

- the mass could be *weighed directly*
- the coordinates of the center-of-mass could be estimated by determining *counterbalanced points* of the link;
- the diagonal elements of the inertia tensor could be *obtained by pendular motions*. This method is very tedious and should be realized by the manufacturer before assembling the robot;

In physical experiments, the special measuring devices are necessary and the joint characteristic is neglected. The identification accuracy depends on the accuracy of measuring devices. Based on physical experiments, some other methods are developed: [7], [8], [9], [10], [11]

Frequency response function: the vibration response of the links is used to determine the inertial parameters.

Modal model method: the modal model of the link is utilized to determine the inertial parameters.

Inertia restricted method: the method is developed based on the residual inertia of the rigid body, which is computed from the measuring value of frequency response experiment.

Direct system identified method: the inertial parameters are identified to minimize the error between the measuring value and the theoretic value of frequency response.

Computer aided design (CAD) techniques: all robotics [CAD/CAM](#) packages provide tools to calculate the inertia parameters from 3D models. This method is prone to errors due to the fact that the geometry of the links is complicated to define precisely, and that certain parts such as bearings, bolts, nuts, and washers are generally neglected; these approaches find the required dynamic parameters of a link by using its geometric and material characteristics. It is easy to obtain the independent parameter value. In the design phase of a robot, the dynamic performance and model-based control performance of the robot can be investigated based on the estimated dynamic parameters, and the performance analysis can be used further to improve the design. However, the precision of the link model in CAD system determines the accuracy of the estimated parameter. Due to the manufacturing error of the link, the CAD model is not identical with the real part and the accuracy of the estimated parameter values is affected. Moreover, estimates of friction parameters are not provided by the manufacturers and are not predictable from CAD drawing.

Identification: this approach is based on the analysis of the “input/output” behavior of the robot on some planned motion and on estimating the parameter values by minimizing the difference between a function of the real variables and its mathematical model. This method has been used extensively and was found to be the best in terms of ease of experimentation and precision of the obtained values [12]. Using this method, Guegan et al. [13] identified the 43 base dynamic parameters of Orthoglide parallel kinematic machine. Vivas et al. [14] identified the base dynamic parameters of H4 parallel robot, which are necessary for the dynamic model, and pointed out the use of the accelerometer and rotation sensor was not very necessary. Compared to the above two methods, the identification approach can obtain good accuracy of identification, and the measurement is relatively easy. This method gives better results than the physical experiment and CAD technique.

In this paragraph, we consider off-line identification methods, for which we collect all the input-output data prior to analysis. The on-line identification will be treated later, and notions of application when presenting the adaptive control techniques will be given.

Several schemes have been proposed in the literature to identify the dynamic parameters [Ferreira 84], [Mayeda 84], [An 85], [Khosla 85], [Atkeson 86], [Gautier 86], [Olsen 86], [Aldon 86], [Kawasaki 88], [Bouzouia 89], [Raudent 90], Identification of the dynamic parameters [Aubin 91], [PrUfer 94], [Gautier 95], [Restrepo 96].

These methods present the following common features:

- the use of a *linear model* in the dynamic parameters (dynamic model, energy model, power model, model of the wrench exerted on the base of the robot);
- the construction of an overdetermined linear system of equations by applying the identification model at a sufficient number of points along some trajectories of the robot. In general, a constant sampling rate is used between the different points;
- the estimation of the parameter values using *linear regression techniques* (ordinary least-squares solution or any other alternative method).

All the identification models can be written in the following general form:

$$y(\tau, \dot{q}) = W(q, \dot{q}, \ddot{q}) \chi$$

Applying the identification model at a sufficient number of points on some trajectories, we construct the following overdetermined linear system of equations in χ

$$y(\tau, \dot{q}) = W(q, \dot{q}, \ddot{q}) \chi + \rho$$

where W is an $(r \times c)$ observation matrix, or regressor (or design matrix), r is the total number of equations, c is the number of parameters such that $r \gg c$, and ρ is the residual error vector.

The identification handbooks provide a large variety of deterministic and stochastic methods to estimate χ from the previous system of equations. The use of ordinary least-squares solution of linear overdetermined system of equations, such as those based on the [SVD](#) or [QR](#) decomposition, gives good results if some care is taken in processing the data measured and the elements of the matrices Y and W as we will show in this chapter. Note that the use

Commented [17]: References

of scientific software packages such as Matlab® or Mathematica® facilitates the application of the proposed data processing. Consequently, the estimated values $\hat{\chi}$ can be obtained from equation

$$\begin{aligned}\hat{\chi} &= \min_{\chi} (\|\rho\|^2) \\ \hat{\chi} &= \arg \left(\min_{\chi} (\|\rho\|^2) \right)\end{aligned}$$

Commented [18]: Correct version ? Khalil

(see [optimization problems](#) and [constrained optimization](#)) that can be solved as a unconstrained minimization problem. If W is of full rank, the explicit solution of this problem is given by

$$\hat{\chi} = (W^T W)^{-1} W^T Y = W^+ Y$$

where W^+ is the pseudo inverse matrix of W. It should be noted that this least-squares estimation is biased because the observation matrix W is random, and because W and p are realizations of random and correlated variables [Mendel 73], [Eykhoff 74], [de Larminat 77], [Gautier 86]. Furthermore, the elements of the matrix W are nonlinear functions in q and \dot{q} , which leads one to assume some statistical properties of the noise in order to calculate the quality of the estimation process (bias and standard deviation) [Armstrong 89], [Raudent 90]. Consequently, it is important to verify the accuracy of the values obtained using appropriate validation procedures.

Commented [19]: report bibliography

Least square method

Detailed informations on [least square \(OLS\)](#) and [weighted LS \(WLS\)](#) method applied in robot dynamic identification problems, compared toward other methods are found in literature [Poignet et al. 00], [Gautier 01], [Numerical Methods for Least Squares Problems]. OLS and WLS are applied usually offline, but can be used also on line. However in on line application the adoption of different (but from theoretical point of view connected) schemes is more usual, like [RLS](#), [KF](#) and [EKF](#), that are characterized by sequentiality and iterative process. This allow from practical perspective to save memory and computational burden for embedded implementations, but from theoretical standpoint present current estimate as a predictor corrector process whose estimate is a bayesian conditioned estimate of current state given previous estimate using current observations batch.

A consumptive treatise about OLS and Kalman filtering with EKF extension method and implications for multibody dynamics are given in following section.

[Least squares and other topics \(Kalman filtering\)](#)

Method election and comparative analysis in literature

In the class of estimation algorithms, the weighted *least squares estimation* method takes a particular place for robot manipulator identification [Khalil et al. 07]. Based on the use of the inverse model linear in relation to the parameters, it allows to estimate the base inertial parameters providing measurement or estimation of the joint torques and the joint positions. The *weighted least square* method, which is a *noniterative* method that finds the parameter estimates in a *single step* using the singular value decomposition, optimizes the root-mean square residual error of the model under the assumption that the measurement errors are

negligible. However, one problem for the parameter estimates of the weighted least square method is the *sensitivity to measurement noise* [Modelling and Identification in Robotics]. Noise will limit the accuracy of parameters obtained by the least squares, and will limit the convergence rate of the *recursive least squares algorithm*. To overcome this problem, one can generate the so-called exciting trajectories and/or use data filtering. By using such “input data improvement”, a lot of excellent parameter estimations using the least squares method have been obtained [Ha et al. 89], [Kawasaki et al. 88]. The comparison shows that estimations of the parameters are very close for both methods, but *Extended Kalman filtering* algorithm is very sensitive to the *initial conditions* and the convergence speed is slower. Thus, the weighted least squares method with the inverse dynamic model appears to be *better* than the extended Kalman filtering approach for *off-line identification*.

From Least squares to recursive filtering

Which is the relationship between least squares and filtering?

Although two concepts arise independently in literature, there's a connection between them. It is possible to prove that manipulating LSQ expressions for a batch of observations data increments sequentially, that application of normal equations and associated solution do not require mandatorily to repeat the whole process for the full dataset of observations but an iterative and sequential process is possible based on previous solution and new observations batch.

Mathematical passages just need to have some knowledge of theorems involved in matrix inversion:

$$(A^T P^{-1} A)x = A^T P^{-1} u$$

There are different advantages, lower computational effort, that especially for large datasets and long observations spans could lead to unmanageable workloads, identification of affecting observations sets in case there are significant changes in parameters sets, ease of application in an online estimator vs an offline estimator.

Reasons why its considered a filter rely on the fact prediction made with the model could be assimilated to filtering process where noises and even deterministic parts of the signal that do not interest are cut off, leaving only relevant informations, provided model and signal are matching enough.

Commented [20]: Eliminate or just keep formula

Parameters pre estimate and constrained estimate

Parametric estimation as presented so far does not consider any a priori knowledge about parameters themselves, like range boundaries, positive definiteness, approximated values deriving from other considerations (like CAD estimate, geometrical approximations, geometrical solid constraints etc). Ideally one could include in optimization problems some constraint and/or a priori estimated value, estimating value itself or a delta respect reference value as new parameter unknown to estimate, without losing problem linearity. A common precaution in these case is to impose a lower variance on parameter to reduce deviation tendency from pre estimation, should be calibrated basing on uncertainty associated to pre-

estimated values, using a weight covariance matrix or a damping matrix, using damped least square method, with damping factor λ .

$$\hat{x} = (A^T A + \lambda^2 I)^{-1} A^T y$$

A priori or pre estimate comes from a manufacturer's specifications or in the case of a recalibration, or from CAD models or geometrical model pre evaluations. Constraints come from total mass, geometrical or dynamical boundaries or other applicable constraints that could consist of equalities or inequalities. Implementation of constraints can be done using the constrained least squares method (basing on Lagrange multipliers) if least squares are used, but also other global optimization methods are possible. In order to discard infeasible solutions, we consider upper and lower bounds on the single components of parameters as well as on the total sum of the link masses.

On-line identification

On-line identification is a classical and well studied problem: finding values of the parameters in the mathematical model of a system from on-line measured data such that the predicted dynamic response coincides with that of the real system [Serban et al. 01]:

Potential advantages and use of online parameter estimation are:

- Real time parameters estimate for adaptive control
- Near-Real time control law ret-uning (especially for friction and temperature dependence)
- On line faults, deviations and anomalies detection and preventive or interventive corrective actions
- Forecasting of malfunction or performance deterioration

Indeed if during trajectory (that could be a designed trajectory, but also an operative trajectory if suitable for the purpose, at least partially) one have access to coordinates, speeds and accelerations of joints, measured with suitable accuracy, and measured currents and/or associated relative torques (by calculation of conversion in case), it is possible to derive online inertial and friction parameters of robot and possibly payloads. For online identification purposes one can use the same analytical regressor model used for off line regression.

Adaptive control algorithms: the method has been investigated extensively as an interesting approach to estimate or adjust on-line the dynamic parameter values used in the control. For parallel robots, the workspace constraints and the complexity of the dynamic equations make an independent excitation of different parameters extremely difficult. If also friction in passive joints of the structure is taken into account, the parameter space will be much larger. These problems are reduced with adaptive control algorithms, since a sufficient excitation is not as crucial as for identification approaches. Based on the Jacobian matrix, Burdet and Codourey [16] projected the force and torque of each moving part onto the Cartesian space. This is the base for the nonlinear adaptive control algorithm to be used for the identification of dynamic

Commented [21]: E Burdet, A. Codourey
Evaluation of parametric and nonparametric nonlinear adaptive controllers
Robotica, 16 (1) (1998), pp. 59-73

parameters. Honegger et al. [17] used a nonlinear adaptive control algorithm to identify the base dynamic parameters of the Hexaglide parallel kinematic machine. In this algorithm, however, parameter convergence cannot be guaranteed.

Commented [22]: Honegger M, Codourey A, Burdet E. Adaptive control of the Hexaglide, a 6 dof parallel manipulator. In: Proceedings of the IEEE international conference on robotics and automation, Albuquerque, NM, USA, 1997. p. 543–8.

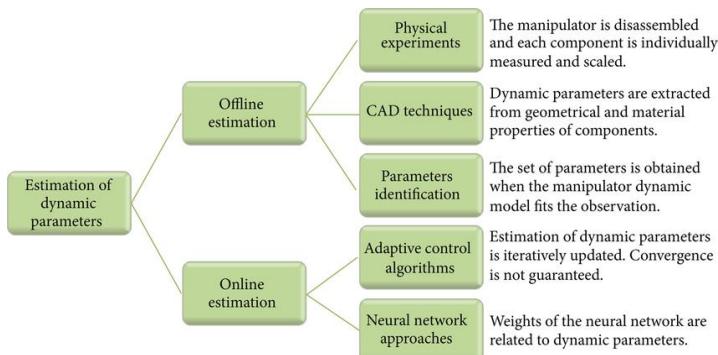
On-line approach based on neural network: neural networks have received considerable attention in the *control* and *identification* fields. It has been demonstrated that neural networks can be used effectively for the identification of nonlinear systems [18]. The identified parameters are regarded as the *weights of the network*, and the weights approach the actual values of the identified parameters by training the weights. Neural network can realize the on-line identification since the sampled real-time data are input to the network to obtain the real-time parameter value. Jiang et al. [19] proposed an identification approach with neural network based compensation of uncertain dynamics, and the dynamic parameter identification process is divided into two steps.

Commented [23]: KS Narendra, K. Parthasarathy Identification and control of dynamical systems using neural network IEEE Transaction on Neural Network, 1 (1) (1990), pp. 5-27

Although dynamic parameter values of a robot can be obtained by off-line and on-line methods, all of them involve approximations and errors. This means that the dynamic parameters are only known with some uncertainties. In these parameters, static moments and diagonal terms of the inertia matrices should be estimated with a good accuracy, and the other terms are not so important and could be estimated with a lower accuracy.

Commented [24]: Jiang ZH, Ishida T, Sunawada M. Neural network aided dynamic parameter identification of robot manipulators. In: Proceedings of the IEEE international conference on systems, man and cybernetics, ,Taipei, Taiwan, 2007. p. 3298–303.

Identification methods can be briefly resumed in following picture:



Observations and measurements: subtle differences

When dealing with data used for identification, filtering, fitting or any other signal or measurement processing we shall distinguish different types of data

- Data coming from direct measurements
- Data coming from processed direct measurements
 - Data coming from indirect measurements (could be considered a special case of processed type)

- Data coming from combinations of direct measurements (linear or nonlinear, like differences, sums, averages, weighted averages, other linear combinations...)
- Data coming from combinations of direct measurements and processed measurements
- Data coming from combinations of processed measurements
- Data coming from combinations of measurements and assumptions
- Data coming from assumption or a priori knowledge (like assumed or known a priori parameters)
- Data coming from simulations used alone or together with other types of measured or processed data
- Other kinds of data

Parameters identifiability

The dynamic parameters can be classified into three groups: *fully identifiable*, *identifiable* in *linear combinations*, and consequently *unidentifiable*. The observation matrix W corresponding to the set of parameters χ is rank deficient (some columns of W are linearly dependent whatever the values of q , \dot{q} and \ddot{q}). In order to obtain a unique solution, we have to determine a set of independent identifiable parameters, which are also called base *dynamic parameters* or *minimum dynamic parameters*. We will show how to calculate the base inertial parameters using *symbolic methods* or *numerical methods*. It is easy to demonstrate that the columns corresponding to the Coulomb and viscous friction parameters are independent. The determination of the base parameters is a prerequisite for the identification algorithms. It should be noted that the grouping equations do not need to be computed since the identification will give directly the grouped values. To simplify the notations, we assume in the following that χ represents the base inertial parameters and the friction parameters, and that W is composed of the corresponding columns.

Dynamic parameters in robot identifications: reasons and utility

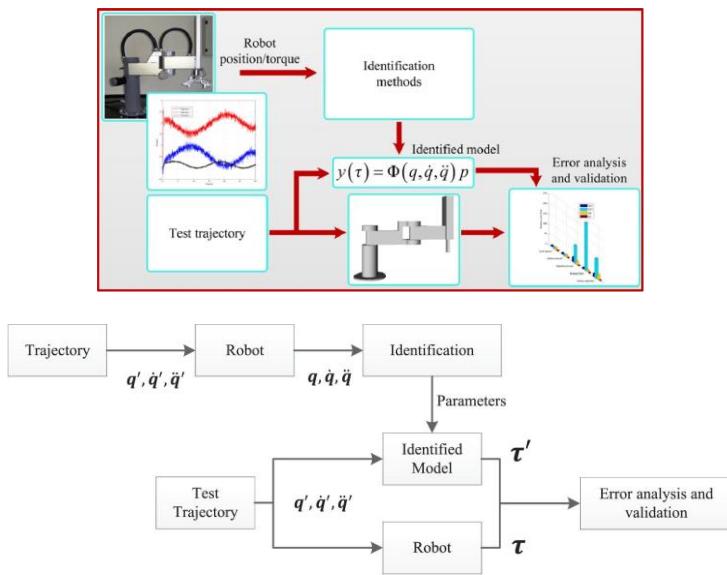
Identification for robot control

The use of model and parametric identification is quite consolidated in the literature of robotics as means for determination of dynamic coefficients. Dynamic parameters are used for describing its behavior, then can be used for digital twins, but are also used in embedded control design for designing feed forward control laws like [computed torque law](#), providing better performances (precision, accuracy, efficiency, stability, and robustness of these schemes depend, to a large extent, on the accuracy of the parameters that describe the dynamic model) respect traditionally used [PID](#) controllers diffused in industrial applications. Identification process, and dynamic parameters can then be carried out off line when the robot is not operative or while operating in parallel with controller operative. In this last case we define it as adaptive control. See also [Applied Nonlinear Control], [Modeling, Identification and Control of Robots], [Slotine-Li 87]. Adaptive and robust schemes can tolerate some errors in the dynamic parameters, while other schemes aimed at achieving perfect feedback linearization, such as the computed torque technique, assume precise knowledge of the dynamic parameters. In view of this, a priori precise determination of the dynamic parameters is useful to most schemes and is crucial to some others. Furthermore, these values are

necessary to simulate the dynamic equations. Accurate values of the dynamic parameters are typically unknown, even to the robot manufacturers.

The dynamic parameters (inertial and friction parameters) that describe the dynamic model are important not only for the advanced *model-based control* algorithms, but also for the *validation* of the *simulation* results and the accurate *path planning* algorithms [Robot calibration]. However, the dynamic model of the robot contains *uncertainties* in many parameters and many control methods are *sensitive* to their values. Sensitivity to parameter uncertainty is especially severe in *high speed* operations. On the other side, if the control problem was at the origin of dynamic identification, modern robotics more oriented toward collaboration and interaction with humans, or sharing environments, lead to lower dynamic requests where the issue of dynamic loads becomes less relevant and stringent.

In general, a standard robot identification procedure consists of *modeling*, *experiment design*, *data acquisition*, *signal processing*, *parameter estimation* and *model validation* [Dynamic model identification for industrial robots]. The last step of the identification procedure is model validation, where the user verifies that the model satisfies the accuracy specifications. If the obtained model does not pass the validation tests, one or several steps of the procedure are repeated and some of the choices are reconsidered. The identification of dynamic parameters has attracted considerable attention from numerous researchers. Atkeson et al. [Atkeson et al 86] proposed the estimation of inertial parameters. Gautier et al. [Gautier et al. 08] proposed a dynamic identification method from *only the torque data*. Grotjahn et al. [Grotjahn et al. 01] used the two-step approach to perform the friction and rigid body identification of robot dynamics. Typical scheme for (off-line) identification is given in following figure:



The accuracy of the dynamic model depends both on the *geometrical parameters* (calibration problem, see [Handbook of robotics] and [Modeling, Identification and Control of Robots]) and dynamic parameters (then current and joint position, speed and acceleration measurements). High-precision geometrical parameters can be obtained by *kinematic calibration*, and the dynamic parameters should be estimated by identification methods.

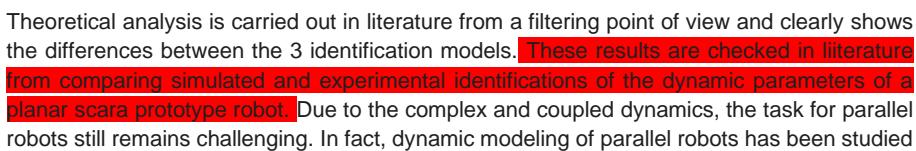
Joint positions are assumed to be known with sufficient accuracy. Many different methods, which can be categorized as on-line identification and off-line identification methods, are proposed to determine the dynamic parameter values. In the off-line procedure, all the input-output data can be collected prior to analysis and do not impose any limits on the computation time. In contrast, the on-line procedure deals with real-time updates of the parameters to be identified during robot operations.

Model for identification in robotics:dynamic and power model

The model to be identified can be classified into three forms: *explicit dynamic model*, *implicit dynamic model* and *energy model*. Gautier [Gautier 96], and Gautier and Khalil [Gautier-Khalil 94] compared the 3 models and the results show that the 3 models give close estimations and accuracy, and the filtered power model is very attractive for its simplicity and accuracy. The other two dynamic models give the same results, then there is no advantage in using the dynamic model with implicit acceleration because its expression is more complicated than that of the dynamic model with explicit acceleration.

Three models which can be used to identify the minimum dynamic parameters of robots :

- ❖ a *dynamic model* which depends on the *joint acceleration* and needs an explicit derivation of the velocity, it will be named the *explicit dynamic model*.
- ❖ a *dynamic model* which *avoids using explicit acceleration* but needs the derivation of a function of the velocity, it will be named the *implicit dynamic model*.
- ❖ the *energy model*, which doesn't need neither acceleration nor implicit velocity derivation. A *power model*, which is the differential expression of the energy model, is introduced to enlighten the comparison between dynamic and energy models and to improve the filtering of the energy model.

Theoretical analysis is carried out in literature from a filtering point of view and clearly shows the differences between the 3 identification models.  These results are checked in literature from comparing simulated and experimental identifications of the dynamic parameters of a planar scara prototype robot. Due to the complex and coupled dynamics, the task for parallel robots still remains challenging. In fact, dynamic modeling of parallel robots has been studied extensively by using general approaches, however, it is difficult to rewrite the dynamic model into the linear form.

Power model based identification

In order to avoid the calculation of the joint accelerations, a model based on energy theorem has been proposed [Gautier 88]. In addition, this model has the following advantages:

- it is linear in the dynamic parameters, and the corresponding base dynamic parameters are the same as those of the dynamic model;

Commented [25]: M Gautier, W. Khalil
Direct calculation of minimum inertial parameters of serial robots
IEEE Transaction on Robotics and Automation, 6 (3) (1990), pp. 368-373

- the planning of an exciting trajectory is easier than for the dynamic model;
- The computation of the observation matrix is easier than that using the dynamic model.

In spite of these advantages, often torques based models are still mostly used.

The energy (or integral) nKxiel is based on the energy theorem, which states that the total mechanical energy applied to the robot is equal to the sum of potential and kinetic energy contained in the system. Let the total energy of the system, also termed Hamiltonian, be denoted by $H = T + U$. From the energy theorem, we can write:

$$dH = J T' dq \quad [12.38] \quad dH = T' q dt \quad [12.39] \quad \text{where: } T = r - \text{diag}(\text{sign}(q)) F_c - \text{diag}(q) F_v$$

The integrator appearing in the energy model is an infinite-gain filter at zero frequency. This means that small low-frequency errors such as offsets can produce large errors. To overcome this problem, we can make use of the differential equation [12.39] instead of the integral one [Gautier 96]. This leads to the power model, which is written as:

Sequential identification

The most widespread approaches propose to use a set of *different trajectories* where each trajectory excites some parameters. For instance, we can move some joints while locking some others [Mayeda 84], [Olsen 86], [Atkeson 86], [Ha 89], [Aubin 91], [Gaudin 92]. This technique *simplifies* the identification equations. However, an accumulation of errors may occur since the values of some estimated parameters will be assumed to be known in subsequent identification. Vandajan [Vandajan 95] has proposed to avoid this drawback by generating four different trajectories to excite four different physical phenomena, which are: *inertial effect*, *centrifugal coupling*, *inertial coupling* and *gravity effect*. The trajectories are periodic between two points (except for gravity). During an experiment, a limited number of joints move while the others are locked. The experiments are designed in order to ensure optimal condition number of the observation matrix. These trajectories are then combined in a global identification system of equations.

Multiple trajectories identification

Since inertial parameters should not change over time (hopefully), multiple trajectories from different experiments or from the same experiment could be used for identification problems in the same optimization solution just stacking observations relative model equations using corresppective kinematic quantities. Usually this is not performed because one experiment is often enough to carry out most part of identifiable parameters related to given planned motion. Under these assumptions there's no significant advantage in solving a multiple trajectory problem with respect to sequential identification. It can be used as parameter cross validation or in case doubt exists on single step identification validity. Moreover there's no need to use all observations relative to a single or multiple trajectory , but samples can be taken from a full set of all trajectories in order to get a better coverage of configurations or getting better conditioning of the regressor resulting in a more reliable estimate. Another use of multiple trajectories is the use of repeated acquisitions on the same trajectory, sometimes performed for sake of reducing observation matrix ill conditioning.

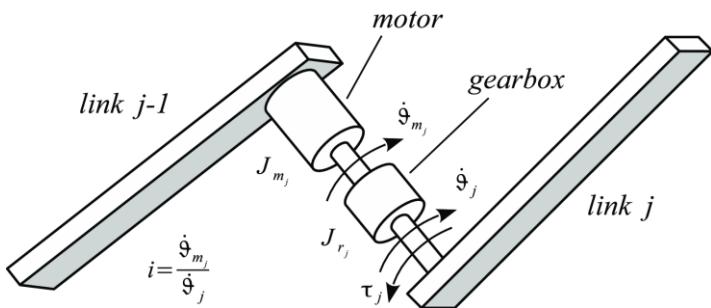
Friction model identification

Using the classical friction model at non-zero velocity, which is represented by viscous and Coulomb frictions, we can write the friction torque on joint j as

$$\Gamma_{f_j} = F_{c_j} \text{sign}(\dot{\theta}_j) + F_{v_j} \dot{\theta}_j$$

Two approaches can be used to identify the joint friction parameters:

- *global identification* of the inertial and friction parameters: in this approach, we consider the estimation of the inertial parameters together with the friction parameters;
- *separate identification* of the friction parameters: in this approach, we identify at first the friction parameters using constant velocity motion on an axis-by-axis basis [Spetch 88], [Held 88]. These parameters are then considered to be known for the identification of the inertial parameters. This simple method induces the risk of error accumulation between the two steps.

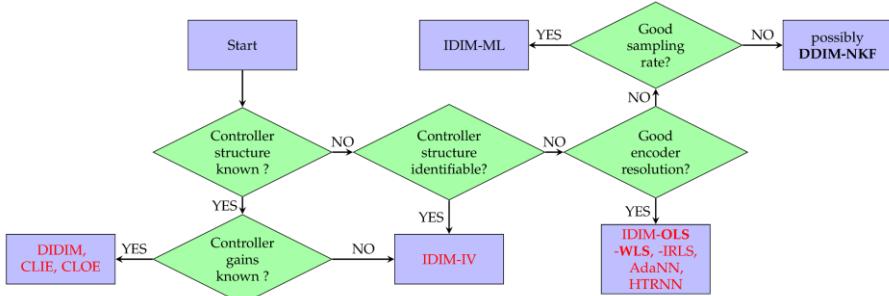


Actuators and transmission ratio

In addition to link inertial parameters, a special mention is deserved for actuators inertia, that's usually accounted together with link parameters, but since a transmission ratio is involved, they do not appear alone.

Physical consistency of inertial parameters

One key issue when dealing with identification is the reconstruction of link parameters and geometry from identified parameters. Without any further information this is not a problem with a solution, since it is not possible to establish a bijective map between geometry, link inertial parameters, and base parameters, that's the reason why we recur to columns elimination and grouping. By the way, making some additional assumptions make it possible not only to establish an invertible map, but also checking physical consistency of determined parameters. Literature works explore this point providing some suggestions on how to apply it practically.



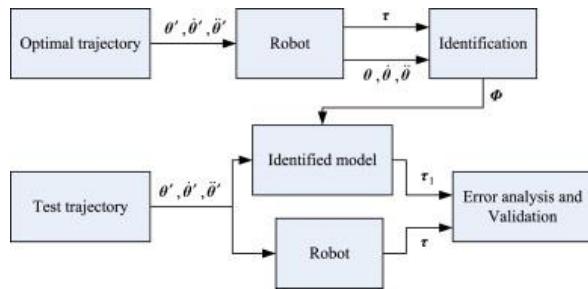
Validation and application

The aim of the model validation is to obtain confidence in the estimated robot model in view of its intended application. Obviously, the most appropriate validation test is to use the model in the application and evaluate its success. The validation scheme is shown in figure. The validation of the estimated values can be carried out using the following methods:

- carrying out *another experiment* with a different validation trajectory by comparing the measured torques and an estimate of these torques based on the model and the measured position data;
- carrying out the *identification process using the energy model* and then by using the *dynamic model* and *comparing* the obtained values;
- direct validation on the identification trajectory by calculating the *error vector* and
- reidentify the parameters of the robot while it is *loaded by a known load*. In this case the values of some base parameters will be changed as a function of the load parameters. The values calculated from the base parameters identified without load plus the load effect are the same as those of the identified values with the load. For example,

Gautierin [Gautier et al. 95] used two methods to verify the identification results:

- carrying out the identification process using the energy model and then by using the dynamic model and comparing the obtained values and
- reidentify the parameters of the robot while it is loaded by a known load. All these tests show very good results.



Identification sensing forces and motion in robotics

Identification process in robotics, for dynamics inertial parameters of links, is performed usually by means of joint torques measurements (or through currents measurements and later converted), relative to a known robot imposed trajectory.

Shall be remarked in the present chapter as in most of literature, it will be assumed that observations are joint actuation generalized forces (mainly torques), although more extensive data acquisitions are possible theoretically, but in the industrial field this is still the dominant paradigm. For instance in humanoid robots dynamic identification gyros and accelerometers, together with marker acquisition and force sensors at the base has been extensively used. In manipulator robotics additional sensors are also commonly employed for calibration. In any case we assume joint position (at least) is known, by reading or assuming imposed position commands do not differ significantly from effective joint position.

Then knowing the dynamic model of the robot, is possible by means of ordinary least squares technique (OLS), weighted if required (WLS) to obtain an estimate of parameters vector. Care shall be taken for preliminarily checking and assuring regressor and observation matrix shall be full rank in order to have a well conditioned normal equations set for solution, and anyway a check of matrix conditioning number is opportune in purpose to avoid numerical instability of the solution to the estimation problem, leading to numerical sensitivity to data errors. Another cause of rank deficiency comes from inclusion of not alone- identifiable parameters, or in case of suboptimal excitation trajectory. In case the matrix is not full rank, a reduction of parameters vector is required usually by means of QR decomposition or SVD decomposition, allowing to identify a subset of base parameters. Rank deficiency occurs when columns of regressor are zero (independence condition) or when there are linearly dependent columns (leading to aggregations of parameters).

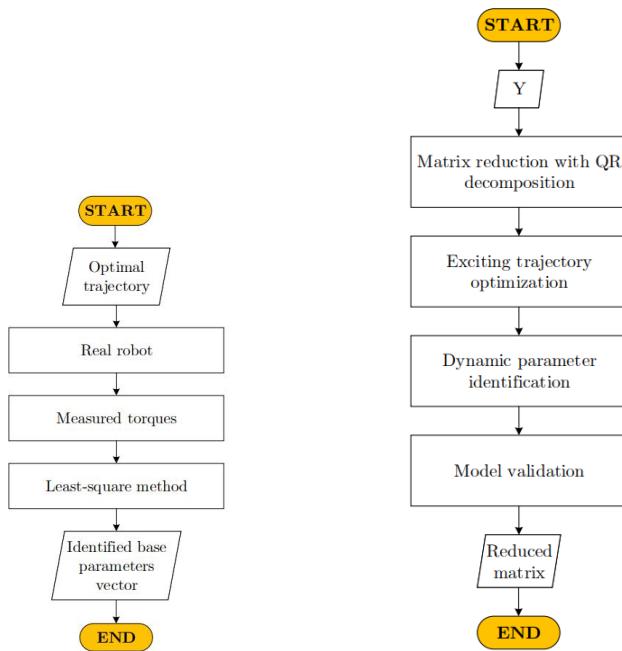


Figure 3.5: Parameters identification using least-squares algorithm

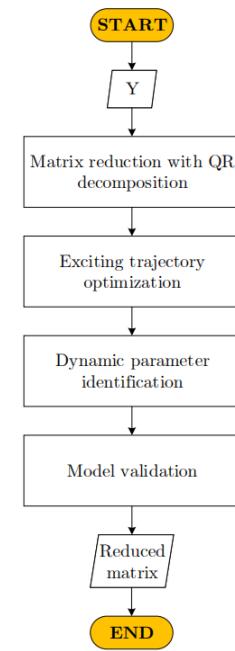


Figure 3.3: QR procedure

Modeling robot dynamics as function of inertial parameters

We've already seen how it is possible to get the dynamic model of a generic tree like a robot. Once we want to take our dynamic model and consider it as a function of some parameters, we get some interesting conclusions. In many cases dynamic equations structure shows an additive form where a group of parameters other than traditional generalized coordinates act as coefficients to coordinate dependent terms. That allows us to write them in a linear or quasi linear form. This is not true just for final robot actuation torques model, but derives from almost general dependance of each link model as a rigid body dynamics from geometrical and inertial parameters. We've seen Euler equations for a rigid body as a coupled set of equations. Indeed we can rewrite links dynamics using Euler equations collecting all inertial parameters of a rigid body as (see [\[An et al. 85\]](#)):

$$\begin{pmatrix} f_i \\ n_i \end{pmatrix} = \begin{bmatrix} \dot{v}_{oi} + \tilde{\omega}_i v_{oi} - g & \dot{\omega}_o + \tilde{\omega}_{oi} \tilde{\omega}_{oi} & 0_{3,6} \\ 0_{3,1} & -(v_{oi} + \tilde{\omega}_{oi} v_{oi} - g) \cdot \dot{\Omega}_i + \tilde{\omega}_{oi} \Omega_i \end{bmatrix} \begin{pmatrix} m_i \\ m_i c_{x_i} \\ m_i c_{y_i} \\ m_i c_{z_i} \\ I_{xx_i} \\ I_{xy_i} \\ I_{xz_i} \\ I_{yy_i} \\ I_{yz_i} \\ I_{zz_i} \end{pmatrix}$$

Whose *standard inertial parameter vector* (not in the physical sense, but algebraic one, underlying linear relationship existing, indeed can be reordered at pleasure) is 10x1 size, is enough to give out all forces and torques acting on it given linear speed of reference point and angular speed in body frame, independently from joints geometry possibly occurring between them. Same equation can be written in shorter form, using the wrench concept.

$$w_{ii} = A_i \phi_i$$

Each link wrench (a 6 vector of forces and torques action on a link or generic rigid body) can be written as a linear map on a vector of link's standard inertial parameters ϕ then. Every link is subject to the wrench caused by its motion itself and to the wrench transmitted from other bodies of the system through joints existing between them. Using a connection matrix ζ that's 1 or 0 whether corps are connected

$$w_i = \sum_{j=1}^N \zeta_{ij} w_{ij}$$

Remarkably these equations imply some recursion since each wrench transmitted depends on others wrench transmitted between other bodies indirectly.

For a chain it simplifies to the sum of all wrenches from tip end of chain backward to body i. Resulting connection matrix is indeed upper diagonal. Namely:

$$w_i = \sum_{j=1}^N w_{ij}$$

Unfortunately this construction is not straightforward since every wrench is designed in body frame and needs to be rotated into a relevant frame. using a rotation matrix one can write

Henceforth we can put together equations for all bodies of a system of rigid bodies, using a standard inertial parameter vector given by concatenations of single bodies parameters vectors.

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} & U_{13} & \dots & U_{1n-1} & U_{1n} \\ U_{21} & U_{22} & U_{23} & \dots & U_{2n-1} & U_{2n} \\ U_{31} & U_{32} & U_{33} & \dots & U_{3n-1} & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ U_{n-11} & U_{n-12} & U_{n-13} & \dots & U_{n-1n-1} & U_{n-1n} \\ U_{n1} & U_{n2} & U_{n3} & \dots & U_{nn-1} & U_{nn} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \Phi_{n-1} \\ \Phi_n \end{pmatrix}$$

Thus as seen above, for serial open chain mechanisms we have an upper triangular matrix:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} & U_{13} & \dots & U_{1n-1} & U_{1n} \\ 0 & U_{22} & U_{23} & \dots & U_{2n-1} & U_{2n} \\ 0 & 0 & U_{33} & \dots & U_{3n-1} & U_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & U_{n-1n-1} & U_{n-1n} \\ 0 & 0 & 0 & \dots & U_{nn-1} & U_{nn} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \Phi_{n-1} \\ \Phi_n \end{pmatrix}$$

Now what we've is a wrench vector for all bodies of the system, but when dealing with joints connecting one body to another one, we've that what one is interested in is no more total wrench but torque or/and force acting along or around actuation axis (depending on joint nature), if only one axis of actuation is involved, or torques or/and forces acting along axis involved for more complex joints. In simple words we need to project wrench vector on axis of actuation in order to get a reduced system of equations like:

$$\begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \Gamma_4 \\ \vdots \\ \Gamma_n \end{pmatrix} = \begin{pmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} & \Phi_{14} & \dots & \Phi_{1n} \\ 0 & \Phi_{22} & & & & \\ & 0 & \Phi_{33} & & & \\ & & 0 & \ddots & & \\ \vdots & & & 0 & & \vdots \\ 0 & & \dots & & 0 & \Phi_{nn} \end{pmatrix} \begin{pmatrix} \chi_1 \\ \chi_2 \\ \chi_3 \\ \chi_4 \\ \vdots \\ \chi_n \end{pmatrix}$$

Remaining wrench vector elements disappeared becomes vincular reactions and are determined from solutions of equations for motion realized once defined by direct or inverse dynamics problems with actuations inputs (if any). Here we changed notation following [Modeling, Identification and Control of Robots]. Those inertial parameters are effectively involved into equations are included in vector named χ (since not all of them are necessarily involved) and resulting actuations vector is named Γ . As remarked the fact that not all inertial parameters do not necessarily play a role, imply existence of not identifiable parameters (in structural sense).

More details are found in relevant work of [An et al. 85], or in [Neubauer,Gatringer,Bremer] and books [Modeling, Identification and Control of Robots] for identification purposes, or also in modern textbooks like [Featherstone] [Rigid body dynamics algorithms]. Method presented is indeed associated to projective method used in dynamic equations writing, although its generality extend beyond that, and comes out to be equivalent to alternative dynamics equations method like Newton Euler or Euler Lagrange, just putting in evidence inertial parameters dependance, that indeed would be a possible variant to original formulation. We're going to see indeed in next paragraphs that an elegant energy based reformulation using inertial parameters vector in evidence has been developed by Khalil.

In the end we've proven the equations of dynamics and consequent regressor structure are linear on inertial parameters vector at single link level as well at system level. Obviously we can extend reasoning beyond interior parameters, and put together any vector of parameters linearly dependent for each single link and build up a concatenated vector for all them. Including but not limited to joints friction parameters often considered in identification problems in robotics.

Regresso structure

When converting an inverse dynamics model for a given trajectory into a matrix form dependent linearly on parameters, we get a typical structure (making an opportune choice of variables order). Indeed, depending on your order choice you can get an upper triangular block structure, although even visual inspection of matrix terms should make evident the presence of a triangular structure due to *serial open chain* topological order, making every row depending only on subsequent index columns (from root to tip). Thus, choosing proper order we get a typical triangular block-like structure making more simple the model for identification

and check of matrix rank. In case matrix column reduction is required, after the regrouping process (for instance using QR method), the result would preserve triangular structure.

$$N \times \begin{bmatrix} n_c \\ n_c \\ n_c \\ n_c \\ n_c \end{bmatrix} \begin{bmatrix} Y_1(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_1(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \\ Y_2(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_2(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \\ \vdots \\ Y_N(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y_N(q(t_{n_c}), \dot{q}(t_{n_c}), \ddot{q}(t_{n_c})) \end{bmatrix} a = \begin{bmatrix} u_1(t_1) \\ \vdots \\ u_1(t_{n_c}) \\ u_2(t_1) \\ \vdots \\ u_2(t_{n_c}) \\ \vdots \\ u_N(t_1) \\ \vdots \\ u_N(t_{n_c}) \end{bmatrix} \quad \bar{Y}a = \bar{u}$$

▪ numerical check of full column rank is more robust \Leftrightarrow rank = p (# of col's)

Following [Sciavicco] symbology

$$\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} y_{11}^T & y_{12}^T & \dots & y_{1n}^T \\ 0^T & y_{22}^T & \dots & y_{2n}^T \\ \vdots & \vdots & \ddots & \vdots \\ 0^T & 0^T & \dots & y_{nn}^T \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_n \end{bmatrix}$$

Extensions of regressor to general linear model

As seen in the previous paragraph, the same approach is applicable to ANY torque contribution that's linear in parameters vector, in case extending parameter vector itself. In principle the method is valid whatever observable linear in parameter set is used for regression (although not necessarily all observations have a triangular block regressor submatrix). The method is identical, and only concerns could arise from keeping regressor matrix rank suitable for regression, or in practical terms, observations are relative to samples where they contribute effectively weight on torque (in some sense are properly excited). A trivial example found in literature is the inclusion of simple linear models of friction (disregarding model order) and actuators simplified dynamics. Actuator inertia aggregate with joint inertia, where friction can be proved is fully decoupled and each parameter have a specific regressor independent column. Obviously introduction of nonlinear friction model could break this conclusion.

Slotine-Li regressor

An alternative formulation of regressor due to [Gabicci et al. 09] work where explicit formulation of the matrices used in the linear-regressor form of the dynamics of general n-dof serial manipulators is derived. First, a closed formula for the classical regressor is derived directly from the Lagrange equations, in terms of the Denavit-Hartenberg Jacobians, the configuration vector q and its derivatives. Then, a specialized version is obtained, which is suitable for application of the Slotine-Li adaptive control scheme [Slotine-Li 87].

Final formulation gives:

$$\begin{aligned}
Y_0^{(i)} &= \dot{X}_0^{(i)} - W_0^{(i)} + Z_0^{(i)} \in \mathbb{R}^{n \times 1} \\
Y_1^{(i)} &= \dot{X}_1^{(i)} - W_1^{(i)} + Z_1^{(i)} \in \mathbb{R}^{n \times 3} \\
Y_2^{(i)} &= \dot{X}_2^{(i)} - W_2^{(i)} \in \mathbb{R}^{n \times 6}.
\end{aligned}$$

$$\begin{aligned}
\dot{X}_{0_r}^{(i)} &= (J_{v_i}^T J_{v_i}) \ddot{q}_r + \frac{1}{2} \left\{ \left[\frac{\partial(J_{v_i}^T J_{v_i})}{\partial q} + \left(\frac{\partial(J_{v_i}^T J_{v_i})}{\partial q} \right)^{T_{132}} \right] \dot{q} \right\} \dot{q}_r, \\
\dot{X}_{1_r}^{(i)} &= X_1^{(i)} \ddot{q}_r + \frac{1}{2} \left\{ \left[\frac{\partial X_1^{(i)}}{\partial q} + \left(\frac{\partial X_1^{(i)}}{\partial q} \right)^{T_{1324}} \right] \dot{q} \right\} \dot{q}_r, \\
\dot{X}_{2_r}^{(i)} &= X_2^{(i)} \ddot{q}_r + \frac{1}{2} \left\{ \left[\frac{\partial X_2^{(i)}}{\partial q} + \left(\frac{\partial X_2^{(i)}}{\partial q} \right)^{T_{1324}} \right] \dot{q} \right\} \dot{q}_r, \\
\left[\frac{\partial T^{(i)}}{\partial q} \right] &= \frac{1}{2} \left\{ \dot{q}^T \left[\frac{\partial}{\partial q} (J_{v_i}^T J_{v_i}) \right] \dot{q} \right\}^T m_i + \\
&+ \frac{1}{2} \left\{ \dot{q}^T \left[\frac{\partial}{\partial q} (J_{v_i}^T S(J_{\omega_i} \dot{q})^0 R_i - J_{\omega_i}^T (J_{v_i} \dot{q})^0 R_i) \right] \right\}^T m_i p_{iG_i} + \\
&+ \frac{1}{2} \left\{ \dot{q}^T \left[\frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i^i I_i^0 R_i^T J_{\omega_i}) \right] \dot{q} \right\}^T \\
\left\{ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i^i I_i^0 R_i^T J_{\omega_i}) \right\}^T &= \begin{bmatrix} \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_1^0 R_i^T J_{\omega_i}) \\ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_2^0 R_i^T J_{\omega_i}) \\ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_3^0 R_i^T J_{\omega_i}) \\ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_4^0 R_i^T J_{\omega_i}) \\ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_5^0 R_i^T J_{\omega_i}) \\ \frac{\partial}{\partial q} (J_{\omega_i}^{T0} R_i E_6^0 R_i^T J_{\omega_i}) \end{bmatrix} \bar{J}_i
\end{aligned}$$

$$\begin{aligned}
X_1^{(i)} &= J_{\omega_i}^{T0} R_i [Q_1 | Q_2 | Q_3]^0 R_i^T J_{v_i} - J_{v_i}^{T0} R_i [Q_1 | Q_2 | Q_3]^0 R_i^T J_{\omega_i} \in \mathbb{R}^{n \times n \times 3}, \\
X_2^{(i)} &= J_{\omega_i}^{T0} R_i [E_1 | E_2 | \dots | E_6]^0 R_i^T J_{\omega_i} \in \mathbb{R}^{n \times n \times 6},
\end{aligned}$$

$$\begin{aligned}
W_{0_r}^{(i)} &= \frac{1}{2} \dot{q}_r^T \begin{bmatrix} \frac{\partial}{\partial q_1} (J_{v_i}^T J_{v_i}) \\ \vdots \\ \frac{\partial}{\partial q_n} (J_{v_i}^T J_{v_i}) \end{bmatrix} \dot{q} \\
W_{1_r}^{(i)} &= \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial q_1} [{}^0 R_i^T S^T (J_{\omega_i} \dot{q}) J_{v_i} \dot{q}_r - {}^0 R_i^T S^T (J_{v_i} \dot{q}) J_{\omega_i} \dot{q}_r] \\ \vdots \\ \frac{\partial}{\partial q_n} [{}^0 R_i^T S^T (J_{\omega_i} \dot{q}) J_{v_i} \dot{q}_r - {}^0 R_i^T S^T (J_{v_i} \dot{q}) J_{\omega_i} \dot{q}_r] \end{bmatrix} \\
W_{2_r}^{(i)} &= \frac{1}{2} \dot{q}_r^T \begin{bmatrix} \frac{\partial}{\partial q_1} (J_{\omega_i}^{T0} R_i E^0 R_i^T J_{\omega_i}) \\ \vdots \\ \frac{\partial}{\partial q_n} (J_{\omega_i}^{T0} R_i E^0 R_i^T J_{\omega_i}) \end{bmatrix} \dot{q}.
\end{aligned}$$

This method has been recently adopted to build regressors for identification or articulated manipulators like UR5 model of Universal Robotics.

Kinematic tree regressor structure

As remarked, the triangular block structure of the regressor is connected with chain sequence relationships of wrenches on a chain of bodies. Consequently when dealing with a kinematic tree regressor triangular structure is lost because of different connections between links through joints.. Linearity on dynamic parameters is instead preserved. As noted from Featherstone et al. [Featherstone 04], [Featherstone 05], [Featherstone 10], [Rigid body dynamics algorithms], [Robot and Multibody Dynamics: Analysis and Algorithms], Inertia matrix and consequently (for same reasons eventually) regressor matrix assume a sparse structure. Remarks, with opportune ordering and links indexing, is possible to regroup branches entries, such that for each branch a triangular submatrix exists, and with opportune choice a clustering can be obtained. For details see [Rigid body dynamics algorithms] and [Robot and Multibody Dynamics: Analysis and Algorithms]

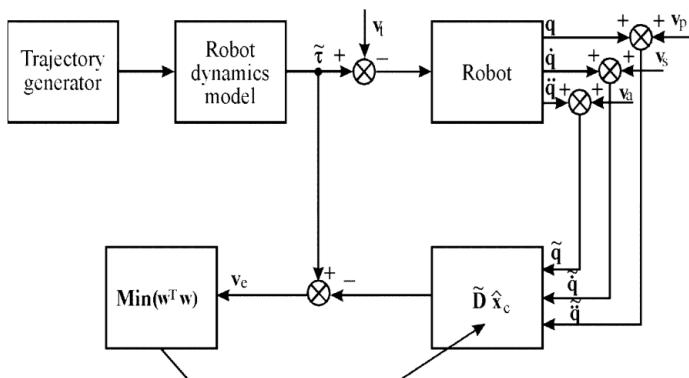


Fig. 4.1. The identification scheme for the differential model

Naive (numerical) regressor calculation method

An easy, although simple conceptually way to compute system regressor numerically, where a model exist, is just imposing all sets of inertial parameters to zero but one chosen at time and calculate output torques for all joints

$$\tau_i = Y_{ij}\beta_j \implies Y_{ij} = \tau_i \mid \beta_k = 1, \beta_j = 0 \quad k \neq j$$

This approach is similar to one used to carry out an inertia matrix from motion equations, and in principle is closely related to. As said the method is *not efficient* but easy to implement and avoid symbolic differentiation issues, in purely symbolic models, allowing mixed symbolic-numerical approach also, and could be used even in presence of purely numerical models, if any available, that allows to manipulate inertial parameters. It can be used for a check, although requiring multiple runs, exactly N_{prms} of evaluation, not much convenient in many

cases. In some numerical cases we need to consider also specifically the consistency of the trajectory path with considered parameter. This mode on the other way is suitable for a digital twin for instance, in a black box manner. Just setting inertial parameters of the model to zero except one considered at time set to one, model response to a n input trajectory supposed to excite it, lead to a joint actuation equal to the regressor column at each instant sample of evaluation. one can then select desired samples and all runs (one for each column of regressor) and assemble it, in a purely numerical manner. The risk is to have singularities in simulation due to null inertial parameters, but the use of small values should overcome it together with a threshold on joints output. See also [Wensing -RPNA - Algorithms](#).

Gravity based identification and quasi static identification

Using linear relations with gravity terms depending only on system geometrical configurations and not on derivative, an independent and separated gravity term identification is possible.

$$\mathbf{g}(\mathbf{q}) = \mathbf{Y}_g(\mathbf{q})\boldsymbol{\pi}_g$$

Quasi static models identification in presence of gravity or base acceleration

One possible application of digital twin developed is for static modeling. In these conditions all inertial loads are neglected and just some contributes due to gravitational terms, making model much more simple. Within rigid-body dynamics algorithms, effects of gravity are often addressed by fictitiously accelerating the base opposite gravity. This trick is applied in the Recursive Newton-Euler algorithm [Luh et al. 80] for inverse dynamics and the articulated-body algorithm for forward dynamics [Featherstone 08]. The same approach also works to address gravitational effects for identifiability

In previous paragraphs we've seen how it is possible to recur to a reduced order model including only static terms equivalent to equilibrium torques of a configuration in absence of motion, that from a practical perspective is anyway a possible case study of limited scope. Remarks, such conditions include but do not limit to gravitational loads, since any other static load should be included into modeling. Self-weight gravity induced loads is just the most common situation. By the way, a static hypothesis is not strictly required, when dealing with low dynamics, since inertial loads are lower order of magnitude with respect other static loads under such assumption.

$$\frac{\tau}{\tau_s} \sim 1 \iff \tau_I \ll \tau_s \approx \tau_g$$

When dealing with situations where motion is really slow, or anyhow negligible or secondary for our study level of details, we can still use this quasi static model as reduced order model or approximation of real model, with applicative results still useful, enough to identify some of dynamic parameters (not all, for instance pure inertia contributes are not identifiable under this condition) because of low values loads, and measure uncertainty. Conversely one should observe identifiability of parameters depending also on dynamic conditions the system is operated on, since in low speed operations (like in collaborative robotics as well others kind

applications) they effectively don't affect loads or dynamics, making them ignorable. In such a sense identification only of a system is meaningful only under its nominal operative range conditions.

Trajectory definition

Trajectory selection, generation and optimization

A robotic motion task is specified by defining a path along which the robot must move. A *path* is a sequence of *points* defined either in task coordinates (end-effector coordinates) or in joint coordinates. The issue of identification trajectory generation is to compute for the control system the desired reference joint or end-effector variables as functions of time such that the robot tracks the desired path. Thus, a *trajectory* refers to a path and a *time history* along the path. *Identification trajectory* suitable and optima for its purpose is named *persistent excitation trajectory* in literature. The trajectories of a robot can be classified as follows:

- trajectory between two points with free path between them;
- trajectory between two points via a sequence of desired intermediate points, also called *via points* or *waypoints*, with free paths between via points;
- trajectory between two points with *constrained path* between the points (straight line segment for instance);
- trajectory between two points via intermediate points with constrained paths between the via points.

In the first two classes, the trajectory is generally generated in the *joint space*. In the last two classes, it is better to generate the trajectory in the *task space*.

Trajectory selection

In order to improve the *convergence rate* and the *noise immunity* of the least squares estimation, the trajectory used in the identification must be carefully selected. Such a trajectory is known as a *persistently exciting trajectory*. To obtain an exciting trajectory, two schemes are generally used:

- calculation of a trajectory satisfying some *optimization criteria*;
- use of *sequential* sets of special test motions, where each motion will excite some dynamic parameters. As the number of the parameters to be identified is reduced with respect to the global problem, it is easier in this case to find an exciting trajectory.

Applicable optimization criteria are [Handobook] numerous. Some of the most prominent are the following:

- *A-optimality* minimizes the trace of $(A^T A)^{-1}$ to obtain regression designs.
- *D-optimality* maximizes the determinant of $(A^T A)^{-1}$.
- *E-optimality* maximizes the *minimum singular value* of $(A^T A)^{-1}$.
- *G-optimality* minimizes the *maximum prediction variance*, and does not have a simple expression in terms of singular values.

[Borm et al. 89] proposed an observability index (here termed O_1 and numbered consecutively below) that maximizes the product of all of the singular values

$$O_1 = \frac{\sqrt{\mu_1 \dots \mu_r}}{\sqrt{P}} \quad \text{where also} \quad \sqrt{\det(A^T A)} = \mu_1 \dots \mu_r$$

Optimal Experimental Design

An observability index is typically used to decide how many data points to collect. As one begins to add data points, the observability increases, then *plateaus*. Adding *additional data* does not then improve the quality of the estimates. For kinematic calibration, data may be added through *random selection*, or optimal design methods can be employed that can drastically reduce the amount of data required. Optimal experimental designs work by measuring the effect of adding or exchanging data points [Sun et al. 08].

Trajectory optimization

When designing an identification experiment for a system, it is necessary to consider the *sufficiency* of excitation [Pukelsheim 06]. It is shown that the *convergence* rate and *noise immunity* of a parameter identification experiment depend directly upon the *condition number* of the *persistent excitation matrix* computed from the inverse dynamic model. It is important to emphasize that the *configurations* for which measurements are taken must correspond to a *well-conditioned reduced observation matrix* since the condition number represents an upper limit for *input/output error transmissibility* [Gautier-Khalil 92]. In the literature, other criteria have also been used to define the exciting condition [Lu et al. 93]:

- 1) the sum of the condition number of observation matrix with a parameter equilibrating the values of the elements of observation matrix:

$$C = \text{cond}(\Phi) + \frac{\max |\phi_{ij}|}{\min |\phi_{ij}|}$$

$$\min |\phi_{ij}| \neq 0$$

where ϕ_{ij} is the (i,j) element of ϕ .

- 2) The sum of the condition number of observation matrix and the inverse of the smallest singular value of observation matrix:

$$C = \text{cond}(\Phi) + k_1 \frac{1}{\sigma_{\min}}$$

where $k_1 > 0$ is a weighting scalar parameter and σ_{\min} is the mini singular value of ϕ .

- 3) The condition number of a weighted observation matrix:

$$C = \text{cond}(\Phi \text{ diag}(Z))$$

where Z is a weighting vector.

For more details see also [Wu et al. 10].

The use of the condition number of the observation matrix is subject to statistical assumptions as was pointed out by Presse and Gautier [Presse - Gautier 93]. It is known that such a condition ensures that the parameters to be identified are well-excited and have a balanced effect on the external forces. Consequently, Armstrong [Armstrong 89] and Presse and Gautier [Presse - Gautier 93] dealt with these configurations directly, at the generalized accelerations

level in the first and the generalized velocities and positions in the second, proposing the use of an optimization process with the aim of minimizing the condition number of the observation matrix. However, this process has some disadvantages. To overcome these disadvantages, Swevers et al. [Swevers et al. 97] suggested considering the configurations governed by a single global trajectory. This trajectory is parameterized using a finite Fourier series function to guarantee periodic excitation. Periodic excitation enables time-domain data averaging and estimation of the characteristics of the measurement noise, which is useful for maximum-likelihood parameter estimation. This method has received more attention [Abdellatif et al. 04] and Park [Park 06] has proposed the use of parameterization based on a combined *Fourier series and polynomial* functions to overcome the disadvantages of the Fourier series parameterization.

The relative standard deviation can be used as a criterion to measure the quality of the identification value for each parameter. It is obtained as:

$$\sigma_{\hat{\chi}_j} \% = \frac{\sigma_{\hat{\chi}_j}}{|\hat{\chi}_j|}$$

For example, if the relative standard deviation of a parameter is greater than ten times the minimum relative standard deviation value, this parameter can be considered as poorly identified.

Scaling and parameters scaling

The numerical conditioning of parameter estimation can be improved by scaling both the output measurements (task variable scaling) and the parameters. Scaling of parameters is important for proper convergence in nonlinear optimization and for singular value decomposition. If parameters have vastly different magnitudes, then the singular values are not directly comparable. Also, parameter scaling can improve the conditioning of the regressor A and help to avoid invertibility problems. Whereas left multiplication of A in equations results in task variable scaling, right multiplication of A by a weighting matrix H results in parameter scaling.

$$\Delta y = (AH)(H^{-1}\Delta x) \equiv \bar{A}\Delta \bar{x}$$

The least-squares solution is not changed by parameter scaling, whereas it is by task variable scaling. The most common approach towards parameter weighting is column scaling, which does not require a priori statistical information. Define a diagonal matrix $H = \text{diag}(h_1 \dots h_p)$

$$h_j = \begin{cases} \|a_j\| & \|c_j\| \neq 0 \\ 1 & \|c_j\| = 0 \text{ where } c_j \text{ are matrix A columns.} \end{cases}$$

Trajectory adjustment

Using analytical models is possible to adjust prescribed trajectory in order to optimize system response for a given inertial parameter. Indeed taking partial derivatives of torques model,

respect desired inertial parameter, generally a function of motions, we can modify trajectory in order to amplify the relative contribute, whether compatible with robot dynamics and control loop limits.

$$\frac{\partial \tau}{\partial \chi} = \Gamma_k(q, \dot{q}, \ddot{q}) \Rightarrow \text{Trajectory}(q, \dot{q}, \ddot{q})$$

Resulting motions demand could be part of an existing trajectory implying its adjust and modification, or alternatively, used trajectory may be accommodate in a unique general sequence, or once more, could be an independent excitation trajectory. The use of digital twins is helpful in defining and checking effectiveness of prescribed trajectories in providing response actuation useful for identifications.

Examples of input trajectories

There are some examples of simple analytical expression of input trajectory really useful to easily manipulate, with relevant equations

- Polynomial
- Harmonic trajectory
- Arctan or sigmoid based trajectory (configuration to configuration or smoothed steplike profiles)

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \dots$$

Or in pithy form

$$q(t) = \sum_{k=0}^n a_k t^k$$

Another useful method allowing for scanning efficiently coordinates space is based on the use of harmonic functions

$$q(t) = \sum_{k=0}^n a_k \sin(\omega_k t + \phi_k) + b_k \cos(\omega_k t + \phi_k)$$

Atan based profiles usually one is enough to give a configuration to configuration profile, but combinations of more than one is possible.

$$q(t) = \sum_{k=0}^n a_k \arctan(\omega_k t + \phi_k) + b_k$$

Alternatively one can use a sigmoid function. Other functions could be used alternatively, but most diffused versions are those presented.

In many of them we can define a set of parameters for

- Phase shifting ϕ
- Offsetting b
- Scaling a
- Shaping ω

Leading to a general form, once decided a base profile function or a set of ξ_i

$$q(t) = \sum_{i=0}^{n_f} \sum_{k=0}^n a_k \xi_i(\omega_k, \phi_k, t) + b_k$$

Usually in motion planning a polynomial profile is used because it can be proved that given boundary conditions of a motion path from configuration to configuration considering initial and final configurations as rest, could be solved in closed form and determined analytically

Scale issues: Dynamic Scaling of Trajectories

The existence of dynamic constraints to be taken into account for trajectory generation has been mentioned. In practice, with reference to the given trajectory time or path shape (segments with high curvature), the trajectories that can be obtained with any of the illustrated methods may impose too severe *dynamic* performance for the manipulator. A typical case is that when the required torques to generate the motion are larger than the maximum torques the actuators can supply. In this case, an infeasible trajectory has to be suitably *time-scaled*. Suppose a trajectory has been generated for all the manipulator joints as $q(t)$, for $t \in [0, t_f]$. Computing inverse dynamics allows the evaluation of the time history of the torques $\tau(t)$ required for the execution of the given motion. By comparing the obtained torques with the *torque limits* available at the actuators, it is easy to check whether or not the trajectory is actually executable. The problem is then to seek an automatic trajectory *dynamic scaling* technique avoiding inverse dynamics recomputation so that the manipulator can execute the motion on the specified path with a proper timing law without exceeding the torque limits. Considering a modified trajectory $q(r(t))$ where $r(t)$ is a strictly increasing scalar function of time with $r(0) = 0$ and $r(t_f) = t_f$ leads to torques expression

$$\tau_s(t) = \dot{r}\tau(r) + \ddot{r}M(q(r))q'(r)$$

The expression gives the relationship between the torque contributions depending on velocities and accelerations required by the manipulator when this is subject to motions having the same path but different timing laws, obtained through a time scaling of joint variables. Gravitational torques have not been considered, since they are a function of the joint positions only, and thus their contribution is not influenced by time scaling. The simplest choice for the scaling function $r(t)$ is certainly the linear function:

$$r(t) = ct$$

$$\tau_s(t) = c^2\tau_s(ct)$$

With the use of a recursive algorithm for inverse dynamics computation, it is possible to check whether the torques exceed the allowed limits during trajectory execution; obviously, limit violation should *not* be caused by the *sole gravity torques*. It is necessary to find the joint for

which the torque has exceeded the limit more than the others, and to compute the torque contribution subject to scaling, which in turn determines the factor c^2 . It is then possible to compute the time-scaled trajectory as a function of the new time variable $r = ct$ which no longer exceeds torque limits. It should be pointed out, however, that with this kind of linear scaling the *entire trajectory* may be penalized, even when a torque limit on a single joint is exceeded only for a short interval of time.

Trajectory dependant base inertial parameters

Afore discussions shall make clear that effective rank of regressor is dependent on assumed trajectory and speeds (for real case) or randomly assigned values (for numerical evaluations, more likely to cover full rank of matrix in fact). If random choice does not matter, because extremely probable, it matters a lot for *prescribed* trajectories, thus it should be optimized.

On the other hand one can imagine to reduce its analysis for sake of simplicity or for any another reason, to adopt a trajectory that lead to a *not full rank matrix*, reflecting the point that observation matrix structure depend on trajectory, or better, directly on set of motion variables (q, \dot{q}, \ddot{q}) and sample point selected. Consequently could happen that a trajectory touches or depend (or better shall *significatively* depend) only part of base inertial parameters. These are not essential parameters, but are like a sort of special case of essential parameters for specific pair trajectory and sample time. We propose a term for this concept as trajectory base parameters. Use of idea like this could potentially find application for instance, splitting identification problem on more than one steps of identification taken as subproblems, could eventually be later assembled into a larger normal equations set for a full identification. In this sense such approach would be a sort of trajectory induced problem reduction or decoupling, acting as a sort of projection of model into a submodel de facto. On the other side this imply the need for eliminating from bse vector not affecting terms. If in some cases this is straightforward, that's not necessarily always true like is not necessarily true this approach lead to a sequential identification that could be considered a sort of special case of here presented idea.

Filtering of noisy observations

It is critically important to note that the OLS estimates in general, will be *unbiased* if and only if the observation matrix W is *uncorrelated* with the *error* term ϵ , or in other words, that the following equality holds

$$E [W^T \epsilon] = 0$$

Unfortunately, the presence of uncorrelated random components within the observation matrix does not allow this hypothesis to be validated in practice. Noise sensitivity is especially problematic in the context of robotic systems as the joint accelerations, obtained by *double time differentiation* of the *noisy encoder data*, often exhibit poor *signal-to-noise ratio* with multiple outliers. A workaround to this issue is to filter the joint measurement signals as suggested by [Gautier 97] where a pragmatic and tailor-made data-filtering is proposed.

It is worth noting that in the case of *periodic excitation trajectory* with a known characteristic frequency, it is possible to use Fourier analysis tools to perform frequency domain filtering as suggested in [Olsen 02], [Swevers et al. 07]. The other alternative is to use identification methods that are robust against a violation of uncorrelation condition.

Practical issues of identification

Identification procedure expect to impose a chosen trajectory to the robot and record resulting torques or motor currents (and relative noise and unmodeled contributes). Nonetheless to make procedure applicable and valid it shall be assured that prescribed motions is reasonably matching with real one, with limited numerical or measurement noise. Since motion is imposed by means of a control loop, this control loop shall be assured *working effectively* keeping the path along prescribed trajectory. Alternatively if *real trajectory* can be *measured*, we can use directly real trajectory joint positions for deriving effective values to insert into regressor, caring of the presence of *errors* in measured positions, especially for speeds and accelerations, when calculated numerically, or adopting some smoothing or interpolation technique allowing for error reduction. Indeed is quite normal performing numerical differentiation to encounter larger errors as degree of derivatives calculated increase, as can be easily seen using a laplace domain approach, for large frequency values. This process is usually off-line and using non *causal filters*.

When using real data there are different precautions to take, like

- *outliers pre-elimination*
- select *data batches* most likely to *match your model* (if dynamics, gravitational or friction, payload mass or any other effect is neglected, select data batches more likely to match with your *assumptions*)
- when dealing with friction free model, *neglect data close to zero speed*

More details are found in

Identification of other parameters

In addition to classic inertial parameters, torques or current measured are affected by other elements like friction parameters and transmission ratio. Transmission ration is usually declared from constructor, but friction parameters depend on operative conditions like lubrication state, temperature, joint speed. Usually they're modeled using a combination of static friction and joint speed dependant terms (linear, quadratic, cubic, or even nonlinear).

What is relevant to note is that all friction torques have a linear dependance on assumed model frictions parameters unknown, except maybe when dealing with non linear model., when parameter is embedded in nonlinear expression and not as coefficient. Henceforth, they affect regressor structure in same way as all other inertial or linear contributes to torque, and respective parameters can be added to vector to be estimated with little effort and concerning. Using same LSQ technique one can get best estimate of, provided assumptions on randomic part respect distribution assumptions on residuals.

Recommendations for experimental application

For experimental application of the identification algorithms, the following recommendations and remarks should be taken into account:

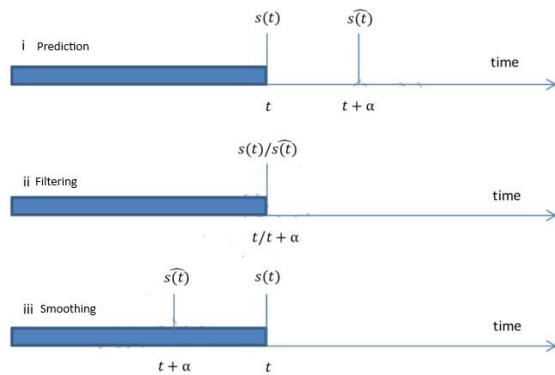
- the dynamic model consists of as many equations as the number of joints, while the energy (or power) model is composed of a single one. Thus, the dynamic model is

basically more exciting and the energy model is more sensitive to the use of exciting trajectories;

- the energy identification method is robust with respect to high frequency perturbation, thus less sensitive to the filtering parameters (cut-off frequency, order of the filter). On the contrary, the dynamic identification model is more sensitive to the filtering parameters and to the quality of estimating the joint velocities and accelerations;
- after filtering the joint positions, the joint velocities and accelerations should be obtained with a central difference algorithm to avoid phase lag;
- when using the dynamic identification model, we must filter the vector of measured torques in order to reject the high frequency ripples. The same filter must be applied to the columns of the observation matrix. This process is called parallel filtering. We note that parallel filtering is automatically incorporated in the filtered dynamic model (reduced order dynamic model);
- it may appear that the filtered dynamic model has the advantages of the dynamic model (number of equations equal to the number of joints) and that of the energy model (no joint accelerations are needed). However, if the dynamic model is correctly processed, it gives equivalent results to the filtered dynamic model [Restrepo 96];
- in the case of robots with several degrees of freedom, six for instance, it is recommended to carry out the identification sequentially in two steps: first, identify the parameters of the wrist links, then, identify the parameters of the shoulder links while locking the wrist joints and assuming that the wrist parameters are known. This procedure is especially efficient because the dimensions of the wrist links are generally not in the same order of magnitude as those of the shoulder;
- the number of equations must be at least 500 times the number of parameters to identify;
- the filtering parameters (cut-off frequency, order of the filter, ...) can be determined by simulating the identification method;
- the relative standard deviation can be used as a criterion to measure the identification quality of a parameter. If the relative standard deviation of a parameter is greater than ten times the minimum relative standard deviation value, this parameter can be considered as poorly identified. Thus, either this parameter has not been sufficiently excited or its contribution to the model is negligible. If the same result is obtained with different trajectories, and if the value of this parameter is relatively small with respect to the other parameters, we can cancel this parameter and define a new set of essential parameters that can be better identified [Pham 91b];
- in order to validate the results obtained, the following tests can be carried out:
- direct validation on the identification trajectory, by calculating the error vector;
- cross validation on a new trajectory that has not been used for the identification;
- verify that the inertia matrix of the robot computed with the estimated parameters is positive definite [Yoshida 00];
- identification of the dynamic parameters twice: without load, and with a known load, to verify if the load parameter values can be correctly estimated;
- carrying out the identification using different methods - dynamic model, filtered dynamic model and power model - to compare the results;
- realizing a simulator of the robot using the identified parameters and comparing the response of the real robot with that of the simulator.

Filtering, smoothing, fitting identification and interpolation: common perspective

We want to give a general insight on duality of different common types of problems in applied engineering and sciences, in the end could be managed on same theoretical basis of statistics. Consider fundamental issue to find, from a set of given observations points P_k a set of parameters that is able to provide respective set of predicted points \hat{P}_k that in some way represent a best estimate of provided points. This can be achieved in different ways, under different assumptions. When we assume that given data are error free or that their error do not significantly impact on results, we can directly use a model pointwise and ask for fulfillment of descending relations for all points. Solution can be find in closed form sometimes, using analytical expression, where possible, or recurring to some linearization or numerical technique. In this subclass of problems we impose a residual on assigned points strictly equal to zero. We can call them *interpolation* problems and could be interpreted as a *limit case of regression*, where not null residuals presence imply *model inadequacy*. Please remark model on parameters are not required necessarily to be continuous or differentiable, or parameters set could be not suitable for differentiation like in case of discrete, rational, non everywhere continuous sets. Alternative methods shall be used then.



When we assume errors or *discrepancies from model are allowed and present*, we cannot demand zero residuals, and we consequently just impose minimization of residuals. For avoiding sign compensation absolute value (robust regression), square or other even powers of residuals are used. Proceeding with one can find best approximating model to given data solving

$$\hat{\chi} = \arg \left(\min_{\chi} \| P_k - f(\chi) \| \right)$$

As we well know from machine learning we need to keep in mind that generalization error shall be considered and evaluate with some cross validation technique possible overfitting and best tradeoff between accuracy and bias (model error).

A further aspect to consider is the number of points provided. Some methods like interpolation could work even with few points, just minimum required to resolve for model parameters.

Fitting method on the other side require many more points to reduce proneness toward data. The less the data, the less the significance of resulted estimate, especially when they're not representative of whole population could lead to data (as in most of ML methods). This holds also for interpolations on the other hand.

Another approach is the use of filtering. Traditionally they're considered different fields, but we want to underline possible analogy between them

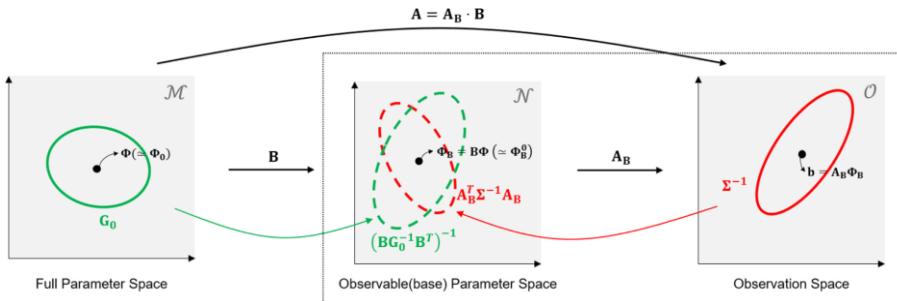
Major difference is the absence in filtering of an explicit model, making them more generally applicable, relying just on some feature of the signal, like frequency bandwidth in LP, HP, BP filters.

We could say that fitting is a kind of filtering that respects model informations content, where traditional filtering makes different assumptions on desired and acceptable signal properties.

Base parameters

In this section, we study the concept of *base inertial parameters* or minimum **identifiable parameters** and other related concepts useful for better understanding their role in robot identification. Illustrated process will clarify the convenience to start from links *standard inertial parameters* in order to obtain in a well formalized and automated way for selecting independent identifiable parameters set as general linear combinations of them.

We need to map from full parameter space into observable parameters space first and finally, considering remapping on effective observations space where estimate occur. These spaces usually do not coincide and their covariance ellipsoid are different, leading to different uncertainty values, a good estimate process should account for. Process is depicted in the following:



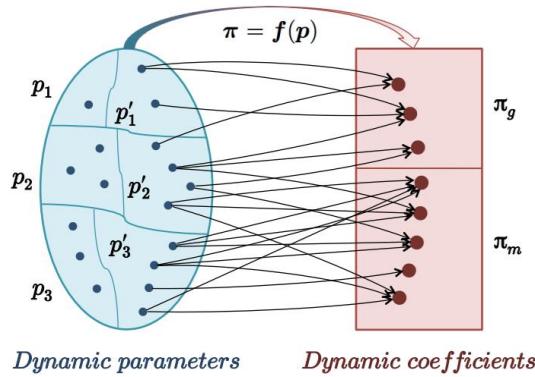
Geometric distortion view of the proposed excitation criterion. The metrics characterized with (green and red) bold ellipses are predefined respectively on the full parameter space and observation space. The (green and red) dashed ellipses represent respectively the pushforward and pullback of the metrics to the observable (base) parameter space under the linear mappings B and A_B . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Commented [26]: cancel or modify

Dynamic coefficients and robot parameters

Identification of the dynamic model of a robot is a long standing problem addressed with a variety of estimation techniques ([Handbook of robotics] chapter 6). All of them exploit the fundamental property of *linear dependence* of the robot dynamic equations in terms of a set of p *dynamic coefficients*, also known as *base parameters* ([Modeling, Identification and Control of Robots]), denoted here by $\pi \in \mathbb{R}^p$. These coefficients are suitable combinations of *geometric* and *inertial* data of single robot bodies, namely of the 10 dynamic parameters specifying the mass, the position of the center of mass, and the symmetric inertia matrix tensor for each robot link. For a robot with N links, there are $10N$ such parameters, denoted here by $p \in \mathbb{R}^{10N}$. Unfortunately standard inertial parameters do not enter all linearly in kinetic energy expressions by themselves. The dynamic parameters p can be *redefined* (keeping their total number constant) so that they appear *linearly* in the kinetic and potential energy of the robot, and thus also in its dynamic equations (the standard parameters mentioned in [Modeling,

Identification and Control of Robots]). The regrouping of the 10N parameters into p dynamic coefficients, typically with $p < 10N$, can be expressed as defining a vectorial relation $\pi = f(p)$, and is a necessary step for guaranteeing a *full rank regressor matrix* in the identification problem ([Dynamic model identification for industrial robots]). When deriving the Lagrangian model of a robot in symbolic form, the regrouping of dynamic parameters can be performed in different ways [Gautier-Khalil 94][Gaz et al. 14], so as to yield a (possibly) minimal number of dynamic coefficients, the quantities that can be identified with motion experiments (on sufficiently exciting trajectories) and that really matter in the robot dynamics. However, only a subset $p' \subseteq p$ of the original dynamic parameters will appear in the equations, whereas the remaining $p \setminus p'$ cannot be observed through any motion experiments. Moreover, the majority of the dynamic parameters p' can be identified only in groups. Only a very small number of them will appear as singletons in the coefficients π , being thus *separately identifiable*. In many cases, the experimental identification (or an a priori knowledge) of a complete set of dynamic coefficients π will be sufficient for robot control and planning purposes. However, being able to extract from the identified dynamic coefficients some *feasible* numerical values for the *original dynamic parameters* p is a relevant objective, at least in some operative situations.



The problem of recovering a complete set of values for the original robot parameters, starting from the identified dynamic coefficients is a *nonlinear problem with infinitely many solutions*, but not all having a *physical significance*. In order to discard such unfeasible solutions, we can consider upper and lower bounds on the single components of p as well as on the total sum of the link masses. [] a more extensive treatment of the subject is found in [\[Gaz et al. 16\]](#).

Standard inertial parameters

As stated above, each link of a robot is characterized by a set of 10 independent standard inertial parameters, that can be collected in a single vector together with eventual actuator inertia parameters, any inertial parameter used in identification is a subset or a linear combination of. The sequence order is not strictly relevant but affects resulting regressor columns order and structure, as well for any other parameter.

$$(m_i, m_i r_{c_{xi}}, m_i r_{c_{yi}}, m_i r_{c_{zi}}, I_{xx_i}, I_{xy_i}, I_{xz_i}, I_{yy_i}, I_{xy_i}, I_{zz_i}, I_{a_i})$$

or following notation and order (not too relevant) of [Modeling, Identification and Control of Robots]:

$$(XX_j \ XY_j \ XZ_j \ YY_j \ YZ_j \ ZZ_j \ MX_j \ MY_j \ MZ_j \ M_j)$$

Regressor rank: structural identifiability, data identifiability, base parameters, and essential parameters

As explained in [Gautier 97], and in other sections of this work, the rank of the sampled observation system is mainly function of two factors, namely the *nature of the collected data samples*, and the *internal structure of the regressor matrix Y_X* . **Data rank deficiency** of the observation system is a direct consequence of *noisy* or *improper data samples* that do *not adequately excite* the model parameters (i.e., that do not sufficiently reveal their influence on the system dynamics). This issue can most of the time be solved by having the robot track a *trajectory* that is specifically designed to excite the model parameters [Gautier-Khalil 92] [38]. The other aspect of the problem lies in the fact that the *intrinsic geometrical properties* of robot manipulators naturally imply that some of the dynamic parameters within the χ vector simply cannot be identified, for having *rigorously no influence* on the actual robot motions *whatever* the followed trajectory. In the same manner, some parameters can only be identified *jointly* because of their combined influence on the system. Remarkably also this definition depend on what *observed or measured quantities* should be. As mentioned we assumed to base our analysis on *joint torques as observables*. From a formal point of view, these issues can be characterized as a **structural rank deficiency** of the regression matrix Y_X . To solve this issue, it is necessary to perform a set of *column rearrangements* within Y_X , eventually leading to the set of $b \leq p$ **base parameters** that are actually *linear combinations* of the *standard inertial parameters* of robot links χ . *Base inertial parameters* are defined by [Gautier-Khalil 94] [Mayeda et al. 88] [39,40] as the set of dynamic parameters which is *sufficient* to completely describe the intrinsically constrained-dynamics of a robot mechanism. As exposed in [Modeling, Identification and Control of Robots] [37], the base parameter vector $\beta \in \mathbb{R}^b$ can be obtained by performing a set of rearrangements in the symbolic expressions of **regressor equations** (note that in this context, some of the performed symbolic simplifications are only made possible using proximal DH convention). In [41] [Beschi et al. 15], an alternative computation method was proposed based on Fourier series decomposition of the robot dynamic equations. Dedicated numerical methods based, for example, on QR decomposition can also be used, or alternatively to SVD (Singular Value Decomposition). Consider the observation matrix $W_X(q, \dot{q}, \ddot{q}) \in \mathbb{R}^{nN \times p}$ $W_X(q, \dot{q}, \ddot{q}) \in \mathbb{R}^{nN \times p}$ constructed by stacking the samples of Y_X obtained from a randomly generated set of $N >> 1$ distinct joint values:

The problem of robot identification can hence be formulated as estimating the value of β such that the dynamic behavior of the model matches that of the actual robot while it is tracking a permanently exciting trajectory. It is worth noting that as some of the base parameters may only have a **minor influence** on the robot dynamics, they can be neglected in practice. Therefore, a reduced set of parameters, referred to as "**essential**" parameters can be identified but could not be considered. [Gautier-Pham 91]. Advantages are relative to identification model simplification and increased robustness and immunity of LS method to

noise and errors, together with reduced algorithms and computational complexity and burden. Clearly also optimal trajectory choice plays a fundamental role in essential parameters selection.

A subtle observation regards base inertial parameters for identifications as conceptually different from base inertial parameters for dynamics. Base inertial parameters for identification is the subset of parameters that could be distinguished in the identification process. On the other hand, the base parameter for dynamics definition is the set of inertial parameters that are sufficient for defining the dynamics of the system. In the end, this reasoning is invertible, and substantially equivalent, since a set of parameters that do not affect dynamics is by definition not identifiable by means of sole torques (or generalized forces), and vice versa any dynamic affecting parameter can, at least in principle be identified.

For a deep overview on the subject we suggest some papers [[Leboutet et al. 21](#)] and [[Wu et al. 10](#)].

Type I and Type II identifiability

From work [[Minimum dynamics parameters of tree structure robot models](#)] [[Kawasaki et al. 91](#)] identifiability of parameters shall be distinguished in

- Type I identifiability,
- Type II identifiability;

This work introduce identification in kinematic tree like robots.

Process for obtaining base parameters

To get base parameters different alternative methods have been elaborated in literature, from direct *formal* (or symbolic) methods [[Gautier-Khalil 94](#)] to purely *numerical* methods [Gautier 91]. An extensive treatment is found in [Modeling, Identification and Control of Robots]. Numerical methods are nowaday more widely applied because easy to implement in program code instructions, although results often hide partially physical insight implied, is more evident using formal methods.

Straightforward **closed-form method** to determine base parameters set from standard inertial parameters problem was studied and developed and published from Khalil et al. in 90'. These parameters constitute the minimum set of inertial parameters that are needed to compute the dynamic model of a robot [[Mayeda 90](#)]. In addition The use of the base inertial parameters in a customized Newton-Euler algorithm reduces its computational cost [[Khalil 86c](#)], [[Khalil 87b](#)].

The base inertial parameters can be obtained from reduction of standard inertial parameters set by *eliminating* those that have no effect on the dynamic model and by *grouping* in a linear combination some others not linearly independent.

Symbolic methods bases on properties of robot kinematics exploiting clustering of parallel axis joints vs orthogonal or not aligned axis joints

Basing on these properties and associated theorems for revolute and prismatic joints is possible to define a predefined regrouping rule for joints

Direct calculation of minimal set of identifiable dynamic parameters

Shall be underlined that here we're still keeping the assumption of using only *torque data* (or similarly actuators currents we use for deriving them) or power models for the identification, with a rest base. In addition we assume required measurements or quantities, joint position, speed and accelerations, are already *known* as given as input with imposed trajectory. Torques are unknown *a priori* since depending on *unknown* parameters we're willing to identify. In a more general case we could consider a moving (imposed or floating or flexible) base, where some motion is prescribed, or let free under robot base dynamics. The use of base motion make possible to extend identification parameters set, taking advantage of base motion where possible to enhance structural or numerical observability of parameters, in some cases occur. In a more general approach to identification process we could (and should) use or consider different measurements and information sources in addition to torques or current data. An example of alternative approaches is given in section [Moving base robots](#). Use of different data potentially widen the range of identifiable parameters and could mitigate common errors and bias induced by measurement common factors.

We've defined (standard) link parameters as the set of 10 parameters (or more if we include actuators and transmissions elements inertial properties) characterizing link inertial properties. We've defined minimum inertial parameters as the set of parameters we use in identification problems, estimated using least squares on dynamic model response. They're usually a linear combination of link parameters coupled with geometrical terms configuration dependant. Y matrix could not depend on some inertial parameters, then corresponding column will be zero and do not affect dynamics, consequently cannot be identified. Some others could appear only in fixed combination with others, then associated columns are linearly dependant.

Following Khalil-Dombre [Modeling, Identification and Control of Robots], [Robotcs Handbook] notation we use D in place of Y but the same.

Leading to a set of grouped parameters called **dynamic coefficients** or following Khalil **base parameters**, that are the set of inertial parameters that affects dynamics. The number of these parameters p shall also be evaluated numerically. p shall be minimal.

Shall be remarked that grouping of base parameters is not unique, and consequently there is a free choice for dynamic coefficients, and consequently also for Y matrix or regressor.

Hence we have different possible methods for choosing and grouping inertial parameters.

- Heuristic method based on matrix inspection
- following a general procedure

In order to explain the direct formal method we need to introduce some notions used in dynamic model treatment specifically in purpose. We start from standard inertial parameters set seen afore, ad define system energy as function of. Since the kinetic energy and the potential energy are linear in the standard inertial parameters, we deduce that the dynamic model is also linear in these parameters. It can be written as:

$$\Gamma = \sum_{j=1}^{N_p} D^j K_j = DK$$

In this section we'll use K as vector of standard inertial parameters of dimension $11N=N$, and D is a matrix $N \times N_p$ function of (q, \dot{q}, \ddot{q}) and geometric parameters. If parameter K_j has no effect on dynamics then $D^j = 0$. We can set freely K_j without changing resulting value Γ . Then K_j can be eliminated form model. Secondly, if columns of D are linearly dependent for a set of constant coefficients t_{jk} :

$$D^j = t_{j1} D^{j1} + \cdots + t_{jr} D^{jr}$$

parameter K_j can be grouped with some other parameters KR_{j1}, \dots, KR_{jr} . In that case, the column D^j and the parameter K_j can be eliminated while the parameters K_{j1}, \dots, K_{jr} will be replaced by KR_{j1}, \dots, KR_{jr} where $KR_{jp} = K_{jp} + t_{jp} K_j$ $KR_{jp} = K_{jp} + t_{jp} K_j$, for $p = 1, \dots, r$. This operation will be repeated until the elimination of all the parameters with dependent columns. The selection of the parameters to be eliminated is not unique. We choose to eliminate those with the highest subscript in K . The search for dependent columns of D starts with the last column and moves backwards toward the first one.

Generally speaking, mathematical proof of, one can find up to $p < 10N$ independent groups of parameters, associated to independent columns of D . Then D can be partitioned in two blocks, D_1 and D_2 , where first contain independent columns and second dependent columns, so we can write:

$$\Gamma = [D_1 \quad D_2] \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}$$

then

$$\Gamma = [D_1 (K_1 + \beta K_2)] = D_1 K_B$$

where K_B is the vector of base parameters. In this way is rather easy to account for effects between links and group terms opportunely using linear combination factor β resulting from numerical process or analytical methods.

$$K_G = K_1 + \beta K_2$$

Now we need to go in detail on the derivation of dynamic model as function of standard inertia parameters used for defining matrix D . We can define system energy H

$$H_j = E_j + U_j = h_j K_j = (e_j + u_j)^j K_j$$

where

$$h_j = [h_{xx_j} \ h_{xy_j} \ h_{xz_j} \ h_{yy_j} \ h_{xy_j} \ h_{zz_j} \ h_{mx_j} \ h_{my_j} \ h_{mz_j} \ h_{mj}]$$

$${}^j K_j = [XX_j \ XY_j \ XZ_j \ YY_j \ YZ_j \ ZZ_j \ MX_j \ MY_j \ MZ_j \ M_j]$$

leading to expressions for h elements:

$$\begin{cases} h_{xx_j} = \frac{1}{2}\omega_{1,j}\omega_{1,j} \\ h_{xy_j} = \frac{1}{2}\omega_{1,j}\omega_{2,j} \\ h_{xz_j} = \frac{1}{2}\omega_{1,j}\omega_{3,j} \\ h_{yy_j} = \frac{1}{2}\omega_{2,j}\omega_{2,j} \\ h_{yz_j} = \frac{1}{2}\omega_{2,j}\omega_{3,j} \\ h_{zz_j} = \frac{1}{2}\omega_{3,j}\omega_{3,j} \\ h_{Mx_j} = \frac{1}{2}\omega_{3,j}V_{2,j} - \frac{1}{2}\omega_{2,j}V_{3,j} - g_0^T {}^0 s_j \\ h_{My_j} = \frac{1}{2}\omega_{1,j}V_{3,j} - \frac{1}{2}\omega_{3,j}V_{1,j} - g_0^T {}^0 n_j \\ h_{Mz_j} = \frac{1}{2}\omega_{2,j}V_{1,j} - \frac{1}{2}\omega_{1,j}V_{2,j} - g_0^T {}^0 a_j \\ h_{M_j} = \frac{1}{2} {}^k V_j {}^k V_j - g_0^T {}^0 P_j \end{cases}$$

we can infer a recursive expression for energy function of link j and j-1

$$h_j = h_{j-1} {}^{j-1} \lambda_j + \dot{q}_j \eta_j$$

where η_j is a kinematic function of link linear and angular speeds and joint type:

$$\begin{aligned} \eta_j &= \bar{\sigma}_j [0 \ 0 \ \omega_{1,j} \ 0 \ \omega_{2,j} \ (\omega_{3,j} - \frac{1}{2}\dot{q}_j) \ V_{2,j} \ -V_{1,j} \ 0 \ 0] + \\ &\quad \sigma_j [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ -\omega_{2,j} \ \omega_{1,j} \ 0 \ (V_{3,j} - \frac{1}{2}\dot{q}_j)] \end{aligned}$$

and where λ is function of geometric parameters of frame j.

$$\left[\begin{array}{cccccccccc} cc\theta & -2cs\theta & 0 & ss\theta & 0 & 0 & 0 & 0 & 2r & r^2 \\ cs\theta c\alpha & (cc\theta - ss\theta)c\alpha & -c\theta s\alpha & -c\theta c\alpha & s\theta s\alpha & 0 & -ds\theta c\alpha + rc\theta s\alpha & -dc\theta c\alpha - rs\theta s\alpha & ds\alpha & drs\alpha \\ cs\theta s\alpha & (cc\theta - ss\theta)s\alpha & c\theta c\alpha & -c\theta s\alpha & -s\theta c\alpha & 0 & -ds\theta s\alpha + rc\theta c\alpha & -dc\theta s\alpha + rs\theta c\alpha & -dc\alpha & -dr\alpha \\ ss\theta c\alpha & 2cs\theta c\alpha & -2s\theta c\alpha & cc\theta c\alpha & -2c\theta c\alpha & ss\alpha & 2(dc\theta + rs\theta c\alpha) & 2(-ds\theta + rc\theta c\alpha) & 2rc\alpha & d^2 + r^2 c\alpha \\ ss\theta c\alpha & 2cs\theta c\alpha & s\theta(cc\alpha - ss\alpha) & cc\theta c\alpha & c\theta(cc\alpha - ss\alpha) & -cs\alpha & rs\theta(ss\alpha - cc\theta) & rc\theta(ss\alpha - cc\theta) & 2rcs\alpha & r^2 c\alpha \\ ss\theta s\alpha & 2cs\theta s\alpha & 2s\theta c\alpha & cc\theta s\alpha & 2c\theta c\alpha & c\alpha & 2(dc\theta - rs\theta c\alpha) & 2(-ds\theta - rc\theta c\alpha) & 2rss\alpha & d^2 + r^2 s\alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & c\theta & -s\theta & 0 & d \\ 0 & 0 & 0 & 0 & 0 & 0 & s\theta c\alpha & c\theta c\alpha & -s\alpha & -rs\alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & s\theta s\alpha & c\theta s\alpha & c\alpha & rc\alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

The matrix transposed can be rewritten in the simpler form (as from article [Gautier-Khalil 94])

$$\lambda^j = \begin{pmatrix} DJ & 0_{6,3} & 0_{6,1} \\ DS & {}^j A_{j-1} & 0_{3,1} \\ Dm & {}^{j-1} P_j^T & 1 \end{pmatrix}$$

where each submatrix is

$$DJ = \begin{pmatrix} cc\theta & cs\theta c\alpha & cs\theta s\alpha & ss\theta ccc\alpha & ss\theta csc\alpha & ss\theta ssc\alpha \\ -2cs\theta & (cc\theta - ss\theta)c\alpha & (cc\theta - ss\theta)s\alpha & 2cs\theta ccc\alpha & 2cs\theta csc\alpha & 2cs\theta ssc\alpha \\ 0 & -c\theta s\alpha & c\theta c\alpha & -2s\theta c\alpha & s\theta(cc\alpha - ss\alpha) & 2s\theta c\alpha \\ ss\theta & cs\theta c\alpha & -cs\theta s\alpha & cc\theta ccc\alpha & cc\theta csc\alpha & cc\theta ssc\alpha \\ 0 & s\theta s\alpha & -s\theta c\alpha & -2c\theta c\alpha & c\theta(cc\alpha - ss\alpha) & 2c\theta c\alpha \\ 0 & 0 & 0 & ss\alpha & -cs\alpha & ccc\alpha \end{pmatrix}$$

$$DS^T = \begin{pmatrix} 0 & 0 & 2r \\ -ds\theta c\alpha + rc\theta s\alpha & -dc\theta c\alpha - rs\theta s\alpha & ds\alpha \\ -ds\theta s\alpha - rc\theta c\alpha & -dc\theta s\alpha + rs\theta c\alpha & -dc\alpha \\ 2(dc\theta + rs\theta c\alpha) & 2(-ds\theta + rc\theta c\alpha) & 2rcc\alpha \\ rs\theta(ss\alpha - ccc\alpha) & rc\theta(ss\alpha - ccc\alpha) & 2rcs\alpha \\ 2(dc\theta - rs\theta c\alpha) & 2(-ds\theta - rc\theta c\alpha) & 2rss\alpha \end{pmatrix}$$

$$Dm = (r^2 \ drs\alpha \ -drcc\alpha \ d^2 + r^2 ccc\alpha \ r^2 csc\alpha \ d^2 + r^2 ssc\alpha)$$

$${}^{j-1} P_j^T = (d \ -rs\alpha \ rcc\alpha)$$

$${}^j A_{j-1} = \begin{pmatrix} c\theta & c\alpha s\theta & s\alpha s\theta \\ -s\theta & c\alpha c\theta & s\alpha c\theta \\ 0 & -s\alpha & c\alpha \end{pmatrix}$$

The matrix ${}^{j-1} \lambda_j$ represents the transformation matrix of the inertial parameters of a link from frame j into frame j-1 such that:

$${}^{j-1} K_j = {}^{j-1} \lambda_j {}^j K_j$$

From equations above, we deduce that an inertial parameter K_j has no effect on the dynamic model if the corresponding energy function h_j is constant;

Direct method for base parameter calculation

Specifically for revolute joints we've that grouping could occur for three parameters, chosen to be YY_j , MZ_j , M_j , leading to following grouping equation set (not unique):

First we need to identify from root first revolute joint (call it r_1) and following sequence of joints up to next not aligned revolute joint (call it r_2). We've a set of rules that depend on joint nature (prismatic or revolute), using following rules, for all links from tip to root.

From velocity relations we get:

$${}^j\omega_j = {}^jR_{j-1}{}^{j-1}\omega_{j-1} + \bar{\sigma}_j \dot{q}_j {}^j\hat{z}_j = {}^j\omega_{j-1} + \bar{\sigma}_j \dot{q}_j {}^j\hat{z}_j$$

$${}^jV_j = {}^jR_{j-1}({}^{j-1}V_{j-1} + {}^{j-1}\omega_{j-1} \times {}^{j-1}P_j) + \sigma_j \dot{q}_j {}^j\hat{z}_j$$

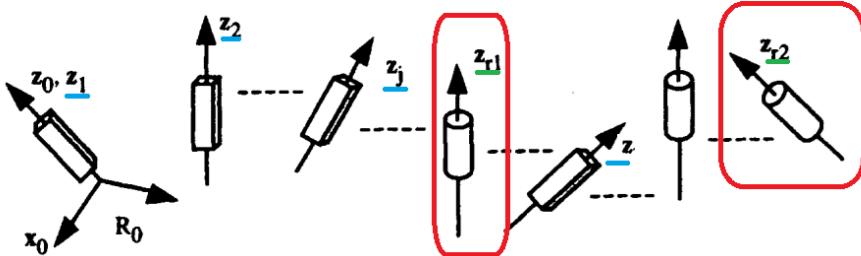


Figure 9.4. Definition of joints r_1 and r_2

Using the recursive relation for the energy functions, we can find general energy functions that satisfy equation in the following. These functions depend on the type of joint.

$$h_j = \sum_{p=1}^r t_{jp} h_{jp} + \text{const}$$

Consequently we can apply rules of grouping for involved parameters. Let us first consider the case where joint j is *revolute*. The following three relations always hold:

$$\begin{aligned} h_{xx_j} + h_{yy_j} &= h_{j-1}({}^{j-1}\lambda_j^1 + {}^{j-1}\lambda_j^4) \\ h_{mz_j} &= h_{j-1}{}^{j-1}\lambda_j^9 \\ h_{m_j} &= h_{j-1}{}^{j-1}\lambda_j^{10} \end{aligned}$$

Consequently, **three** inertial parameters can be grouped with the others. The choice of these parameters is not unique. By choosing to group the parameters YY_j , MZ_j and M_j we obtain:

$$XX_{R_j} = XX_j - YY_j$$

$$\mathbf{KR}_{j-1} = \mathbf{K}_{j-1} + YY_j(j^{-1}\lambda_j^1, j^{-1}\lambda_j^4) + MZ_j j^{-1}\lambda_j^9 + M_j j^{-1}\lambda_j^{10} \quad [9.52]$$

Substituting for $j^{-1}\lambda_j^1, j^{-1}\lambda_j^4, j^{-1}\lambda_j^9, j^{-1}\lambda_j^{10}$ from Table 9.1 into equations [9.51] and [9.52] gives the following theorem:

Theorem 9.1. If joint j is revolute, the parameters YY_j, MZ_j and M_j can be grouped with the parameters of link j and link $j-1$. The resulting grouped parameters are:

$$\begin{aligned} XXR_j &= XX_j - YY_j \\ XXR_{j-1} &= XX_{j-1} + YY_j + 2r_j MZ_j + r_j^2 M_j \\ XYR_{j-1} &= XY_{j-1} + d_j S\alpha_j MZ_j + d_j r_j S\alpha_j M_j \\ XZR_{j-1} &= XZ_{j-1} - d_j C\alpha_j MZ_j - d_j r_j C\alpha_j M_j \\ YYR_{j-1} &= YY_{j-1} + CC\alpha_j YY_j + 2r_j CC\alpha_j MZ_j + (d_j^2 + r_j^2 CC\alpha_j) M_j \\ YZR_{j-1} &= YZ_{j-1} + CS\alpha_j YY_j + 2r_j CS\alpha_j MZ_j + r_j^2 CS\alpha_j M_j \\ ZZR_{j-1} &= ZZ_{j-1} + SS\alpha_j YY_j + 2r_j SS\alpha_j MZ_j + (d_j^2 + r_j^2 SS\alpha_j) M_j \\ MXR_{j-1} &= MX_{j-1} + d_j M_j \\ MYR_{j-1} &= MY_{j-1} - S\alpha_j MZ_j - r_j S\alpha_j M_j \\ MZR_{j-1} &= MZ_{j-1} + C\alpha_j MZ_j + r_j C\alpha_j M_j \\ MR_{j-1} &= M_{j-1} + M_j \end{aligned} \quad [9.53]$$

where $SS(*) = S(*) S(*)$, $CC(*) = C(*) C(*)$ and $CS(*) = C(*) S(*)$.

$$K_{R_{j-1}} = K_{j-1} + YY_j (j^{-1}\lambda_j^1 + j^{-1}\lambda_j^4) + MZ_j j^{-1}\lambda_j^9 + M_j j^{-1}\lambda_j^{10}$$

after substituting columns 1,4,9,10, corresponding to $j^{-1}\lambda_j^1, j^{-1}\lambda_j^4, j^{-1}\lambda_j^9, j^{-1}\lambda_j^{10}$ grouping relations results to be:

$$\begin{aligned} XX_{R_j} &= XX_j - YY_j \\ XX_{R_{j-1}} &= XX_{j-1} + YY_j + 2r_j MZ_j + r_j^2 M_j \\ XY_{R_{j-1}} &= XY_{j-1} + d_j S\alpha_j MZ_j + d_j r_j S\alpha_j M_j \\ XZ_{R_{j-1}} &= XZ_{j-1} - d_j C\alpha_j MZ_j - d_j r_j C\alpha_j M_j \\ YY_{R_{j-1}} &= YY_{j-1} + CC\alpha_j YY_j + 2r_j CC\alpha_j MZ_j + (d_j^2 + r_j^2 CC\alpha_j) M_j \\ YZ_{R_{j-1}} &= YZ_{j-1} + CS\alpha_j YY_j + 2r_j CS\alpha_j MZ_j + r_j^2 CS\alpha_j M_j \\ ZZ_{R_{j-1}} &= ZZ_{j-1} + SS\alpha_j YY_j + 2r_j SS\alpha_j MZ_j + (d_j^2 + r_j^2 SS\alpha_j) M_j \\ MX_{R_{j-1}} &= MX_{j-1} + d_j M_j \\ MY_{R_{j-1}} &= MY_{j-1} - S\alpha_j MZ_j - r_j S\alpha_j M_j \\ MZ_{R_{j-1}} &= MZ_{j-1} + C\alpha_j MZ_j + r_j C\alpha_j M_j \\ M_{R_{j-1}} &= M_{j-1} + M_j \end{aligned}$$

that for $\alpha = 0$ (parallel axis) and $r = 0$ and $d = 0$ (no offsets) simplifies to

$$\begin{aligned}
XX_{R_j} &= XX_j - YY_j \\
XX_{R_{j-1}} &= XX_{j-1} + YY_j + 2r_j MZ_j + r_j^2 M_j \\
XY_{R_{j-1}} &= XY_{j-1} \\
XZ_{R_{j-1}} &= XZ_{j-1} - d_j MZ_j - d_j r_j M_j \\
YY_{R_{j-1}} &= YY_{j-1} + YY_j + 2r_j MZ_j + (d_j^2 + r_j^2) M_j \\
YZ_{R_{j-1}} &= YZ_{j-1} \\
ZZ_{R_{j-1}} &= ZZ_{j-1} + MZ_j + (d_j^2) M_j \\
MX_{R_{j-1}} &= MX_{j-1} + d_j M_j \\
MY_{R_{j-1}} &= MY_{j-1} \\
MZ_{R_{j-1}} &= MZ_{j-1} + MZ_j + r_j M_j \\
M_{R_{j-1}} &= M_{j-1} + M_j
\end{aligned}$$

$$\begin{array}{ll}
XX_{R_j} = XX_j - YY_j & XX_{R_j} = XX_j - YY_j \\
XX_{R_{j-1}} = XX_{j-1} + YY_j & XX_{R_{j-1}} = XX_{j-1} + YY_j \\
XY_{R_{j-1}} = XY_{j-1} & XY_{R_{j-1}} = XY_{j-1} \\
XZ_{R_{j-1}} = XZ_{j-1} - d_j MZ_j & XZ_{R_{j-1}} = XZ_{j-1} \\
YY_{R_{j-1}} = YY_{j-1} + YY_j + (d_j^2) M_j & YY_{R_{j-1}} = YY_{j-1} + YY_j \\
YZ_{R_{j-1}} = YZ_{j-1} & YZ_{R_{j-1}} = YZ_{j-1} \\
ZZ_{R_{j-1}} = ZZ_{j-1} + MZ_j + (d_j^2) M_j & ZZ_{R_{j-1}} = ZZ_{j-1} + MZ_j \\
MX_{R_{j-1}} = MX_{j-1} + d_j M_j & MX_{R_{j-1}} = MX_{j-1} \\
MY_{R_{j-1}} = MY_{j-1} & MY_{R_{j-1}} = MY_{j-1} \\
MZ_{R_{j-1}} = MZ_{j-1} + MZ_j & MZ_{R_{j-1}} = MZ_{j-1} + MZ_j \\
M_{R_{j-1}} = M_{j-1} + M_j & M_{R_{j-1}} = M_{j-1} + M_j
\end{array}$$

Rules for practical base parameters determination

The following algorithm can be used to determine all the parameters that can be eliminated or grouped. The remaining parameters constitute the set of base inertial parameters of the links. The grouped relations make use of closed-form symbolic expressions.

For $j = n, \dots, 1$:

- 1) use the general grouping relations above (revolute joint and prismatic joint) to group:
 - a) YY_j , YY_j , MZ_j , MZ_j and M_j , M_j if joint j is revolute ($\sigma_j = 0$);
 - b) XX_j , XY_j , XZ_j , YY_j , YZ_j and ZZ_j if joint j is prismatic ($\sigma_j = 1$);
- 2) if joint j is prismatic and z_j , a_j is parallel to af_i , for $r_i < j < r_2$, then eliminate MZ_j , MZ_j and group MX_j , MX_j and MY_j , MY_j using relation [9.60];
- 3) if joint j is prismatic and a_j is not parallel to z_j , a_j , for $r_1 < j < r_2$, then group or eliminate one of the parameters MX_j , MX_j , MY_j , MY_j , MX_j , MZ_j using Table 9.2;

- 4) if joint j is revolute, for $r_i < j < r_2$, then eliminate $XX_j, XX_j, XY_j, XZ_j, XZ_j$ and XZ_j, YZ_j . Note the axis of this joint is parallel to the axis of joint r_i , and that the parameter YY_j, YY_j has been eliminated by rule 1;
- 5) if joint j is revolute, for $r_i < j < r_2$, and the z_j axis is along z_j axis, and if z_j is parallel to a^i and to gravity g, for all $i < j$, then eliminate the parameters MX_j, MX_j, MY_j . Note that MZ_j is eliminated by rule 1;
- 6) if joint j is prismatic and $j < r_i$, then eliminate the parameters $MX_j, MX_j, MY_j, MY_j, MZ_j, MZ_j$.

Number of base parameters

From [Khalil 90], [Modeling, Identification and Control of Robots] and [1] we get a set of conditions applicable to identifiable inertial parameters.

Theorem 7: From theorems 3, 4, 5, 6, we can deduce that the *number of minimum inertial parameters* is equal or less than:

$$b_m \leq 7n_r + 4n_t - 3 - \bar{\sigma}_1 - 2n_{g0}$$

where:

- n_r = number of revolute joints $\sum \bar{\sigma}_k$
- n_t = number of prismatic or translational joints $\sum \sigma_k$
- $n_{g0} = 1$ if the first joint is revolute and parallel to gravity, $n_{g0} = 0$ otherwise.

The proposed algorithm of the detection of the set of minimum parameters will be carried out as follows.

From this algorithm, we deduce that the number of minimum inertial parameters of the links for a general serial robot (without considering the inertia of the rotors) is given by:

$$bm \leq 7nr + 4np - 3 - \bar{\sigma}_1 - 2ng_0 \quad [9.61]$$

with:

- nr : number of revolute joints = $Z \Delta_j^*$
- np : number of prismatic joints = $J C_j$
- $n_{g0} = 1$ if the first joint is revolute and parallel to gravity, $n_{g0} = 0$ otherwise.

This equation gives in most cases the exact number of base inertial parameters. Nonetheless one shall consider in real case many or most of parameters included in the model are negligible and computing results overestimated. For instance inertia products are often considered negligible and barycenter positions are often considered to be axis aligned or nearly. Small distances are sometimes neglected to avoid complex expressions. This formula does not account for actuator inertia.

Considerations for applicative case

Is remarkable that for most modern manipulators the first joint is revolute with a vertical axis respect base and the second joint is revolute with axis horizontal. This configuration allows great flexibility (is half a wrist) with few joints.

- Consequently $r_1=1$ and $r_2=2$.
- Most of manipulators have only revolute joints only (for enhancing operative space reachability)
- essentially we need of rules 1.a) and correctly apply to root proper elimination
- Base parameter and grouped parameters are rather similar for most of robots

See results of some examples in tables.

General procedure is easily implemented in Matlab® using general matrix λ specialized for a chosen set of parameters to regroup, looping through all links backward until the final table is done.

	XX	XY	XZ	YY	YZ	ZZ	MX	MY	MZ	M	Ia	
1	0	0	0	0	0	ZZR_1	0	0	0	0		
2	XXR_2	XY_2	XZ_2	0	XY_2	ZZR_2	MX_2	MY_2	MY_2	M_2		
3	XXR_3	XY_3	XZ_3	0	XY_3	ZZR_3	MX_3	MY_3	MY_3	M_3		
4	XXR_4	XY_4	XZ_4	0	XY_4	ZZR_4	MX_4	MY_4	MY_4	M_4		
5	XXR_5	XY_5	XZ_5	0	XY_5	ZZR_5	MX_5	MY_5	MY_5	M_5		
6	XXR_6	XY_6	XZ_6	0	XY_6	ZZR_6	MX_6	MY_6	MY_6	M_6		

Example of table for a planar robot RRR...

Using rules we can state that all clusters of joints are parallel and standard parameters XX,XY,XZ,YZ are eliminated. In addition since joints are revolute we can group together M,MZ and YY.

Assuming a negligible contribute of MY we can further simplify leading to a simpler model where each link contribute for at least one

When dealing with base link we can eliminate all but ZZ1 have effect on dynamics. Eventual actuators inertia could be grouped with, and each actuator inertia could be grouped with respective inertia ZZR

Generally speaking we have then three parameters for each link.

	XX	XY	XZ	YY	YZ	ZZ	MX	MY	MZ	M	Ia	
1	0	0	0	0	0	ZZR_1	0	0	0	0		
2	XXR_2	0	0	0	0	ZZR_2	0	0	0	0		
3	XXR_3	0	0	0	0	ZZR_3	0	0	0	0		
4	XXR_4	0	0	0	0	ZZR_4	0	0	0	0		
5	XXR_5	0	0	0	0	ZZR_5	0	0	0	0		
6	XXR_6	0	0	0	0	ZZR_6	0	0	0	0		

Extension to tree-like robots

Although the method was developed for serial robots it could be extended easily to tree-like robots following the same rules. Indeed at each branch node, we've just to introduce energy contributions coming from both branches in place of just a single branch of a serial robot. That implies that respective inertia parameters shall be regrouped accordingly through usual rules.

In that case rotation matrix shall be expressed opportunely between a link and its predecessor
 λ

Numerical determination of base parameters

In previous sections we've seen how to compute regressor matrix associated to dynamic model of a robot. Once it's known, we can use standard inertial parameter vector together with numerical methods in order to find the base parameter of the system. Using multiple random sampling on regressor variables, it is possible to get a sample instance of a stacked regressor matrix that can be used for numerical determination of base parameters, by means of QR decomposition or SVD decomposition. Indeed in most cases these methods are not applicable in the general case of symbolic regression, because of computational difficulties even with not large matrices and not excessively complex models for observations. The practicality of using numerical methods is essentially the ease of application using a general algorithm, differently from closed form methods that requires evaluation of specific properties of the kinematic chain of robot joints. On the other hand, numerical methods directly consider regressor matrix rank, where formal methods obtain full rank only indirectly. The study of the minimum inertial parameters carried out by studying the dependence of the symbolic expressions of the elements of the columns $Y_{:,j}$ or of the functions h_j is then limited. Numerically approach is equivalent to study the space span by the columns of a observation matrix W which can be taken as D or D_o , or H . The use of D or D_o , or H is mathematically equivalent, but this doesn't implies the numerical equivalence because of different scalings of these 3 matrices. The proprieties to study are:

- the rank b of W which is the dimension of the space and the number of base parameters,
- the choice of $c-b$ columns to be deleted and of b independent columns constituting a base of the space and which define the base parameters,

- the determination of c-b linear relations between the base columns and the deleted ones,
- the determination of the values of the b base parameters from those of the standard parameters. We propose to use a QR and a SVD decomposition of W to solve these problems.

We now illustrate synthetically the procedure adopted for getting numerically a set of base parameters, as descriptive in Gautier work [Gautier 91]. First a set of random values are assigned to the regression model Y in order to get a random sample of observation matrix W, which size depend on number of samples selected as procedure parameter. Since there is no assigned trajectory, there's freedom of choice on values of coordinates, joint speed and accelerations, although it would be better to respect some realistic constraint, but method is general and does not require it. Once a sample observation matrix W is obtained in numerical form (that means any additional parameter contained in the regressor model shall be instantiated with reasonable values, although theoretically they should not appear and be embedded into inertial parameters). Hence is possible to decompose the matrix in a QR factorization such that

$$Q^T W = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Rank evaluation of W could lead to a deficiency, meaning that matrix is likely to have multiple possible decompositions, or collinear columns, but a permutation exists such that could be put in a form like the following, that's unique.

$$Q^T W \Pi = \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}$$

where Q is orthogonal, R1 is a bxb upper triangular and regular matrix, and R2 is a bx(c-b) matrix. Once again, as in the symbolic approach, the choice of columns (corresponding to parameters) to be deleted is not unique. A choice usually is taking the largest index column to be regrouped with lower index columns. Those lasting will become base parameters. Adoption of a permutation matrix is possible to define linear dependence of parameters to regroup. It is given by decomposition algorithm

$$P^T X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$WP = [W_1 \quad W_2] = [Q_1 \quad Q_2] \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix} = [W_1 R_1 \quad W_1 R_2]$$

Since we need to define regressor structure and associate parameter vector as a linear combination, it results

$$W_2 = W_1 R_1^{-1} R_2$$

Evaluation of Kd matrix (at right), allow to redefine base parameters as a linear combination of standard inertial parameters:

$$XR_1 = X_1 + R_1^{-1}R_2(X_2 - XR_2)$$

Again, we have infinite possible values for XR_2 , and consequently of the resulting base parameter, but standard choice just eliminates them.

A further consideration worthy to note is that these methods are not specifically relying on any assumption about the nature of the problem or regressor, so they're general enough to be applied to a larger class of problems of OLS or even NLS type (possibly, have to be justified).

Singular Value Decomposition for numerical evaluation of base parameters

Another approach for numerical solution of base inertial parameters set definition is by use of SVD. Starting from random observation matrix W , we get a decomposition like

$$\begin{aligned} XR_1 &= X_1 + R_1^{-1}R_2X_2 \\ U^T W &= SV^T \quad S = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} \end{aligned}$$

Σ is a cxc diagonal matrix whose elements σ_i are in nonincreasing order $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_c \geq 0$. The elements σ_i are the singular values of W . Adopting a tolerance t , we get singular values and consequently columns to delete from regressor matrix.

Imposing linear relation for regrouping we get from decomposition, similar relations to QR case:

$$XR = X + V_2X_a$$

$$\begin{bmatrix} XR_1 \\ XR_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} V_{21} \\ V_{22} \end{bmatrix} X_a$$

$$XR_1 = X_1 - V_{11}V_{22}^{-1}X_2$$

A further consideration worthy to note is that these methods are not specifically relying on any assumption about the nature of the problem or regressor, so they're general enough to be applied to a larger class of problems of OLS or even NLS type (possibly, have to be justified).

Rule of change for additional loads and payload presence

Only dynamic parameters of the link where a load is added will change (typically added to the last one as payload).

- last link dynamic parameter m_n mass and center of mass $c_n = (c_{nx} \ c_{ny} \ c_{nz})^T$ and inertia tensor I_n in frame n
- payload dynamic parameters mass m_L , center of mass $c_L = (c_{Lx} \ c_{Ly} \ c_{Lz})^T$ and inertia tensor I_L in frame n

Rules consist of

- mass $m_n \rightarrow m_n + m_L$
- center of mass
$$c_{ni}m_n \rightarrow \underbrace{\frac{c_n m_n + c_L i m_L}{m_n + m_L}}_{\text{weighted average}} (m_n + m_L) = c_{ni}m_n + c_{Li}m_L \quad \text{where } i = x, y, z$$
- Inertia tensor $I_n \rightarrow I_n + I_L$ expressed in same reference frame
- linear parametrization is preserved with any kinematic convention (the parameters of the last link will always appear in the form shown above)

More details on payload accounting and estimation are found in [\[Gaz et al. 17\]](#) and [\[Linear parametrization and identification\]](#).

A third way: inertia matrix inspection and parametrization

Although is clear so far different methods exist in order to assure regressor full rank as function of the choice of base inertial parameters, as consequence of linear dependence on observation matrix columns driven by regressor model definition, we propose a further definition based on inertia matrix structure. As seen in literature base inertial parameter is a set of independent linear combinations of inertial parameters allowing a complete definition of the dynamics. Equivalently we know from our studies on dynamic equations of robot that once inertia matrix is known this is essentially determined regarding part relative to inertia parameters (then other parameters to identify like friction parameters are no affecting results). Thus finding a minimal parametrization of mass matrix elements or coefficients, is enough to define a set of base inertial parameters. Indeed:

- Inertia matrix elements already contains all and only standard inertial parameters affecting dynamics
- There are less element to evaluate and we do not need to declare or define all standard inertial parameters as in classic method
- For large number of degree of freedom method could lead to consider less standard parameters before being eliminated (**TBC**)
- Computational effort could be lower, since we do not need to get full torque expressions and take partial derivative (as in numerical case) for selecting parameters. Base set can be chosen prior of equations calculation and maybe reduced a priori
- Analysis of inertial parameters effect on dynamics and consequently identifiability (or observability) of inertial parameters becomes straightforward or anyway simpler
- Inertia matrix elements are function of inertial parameters and coordinates
- Inertia matrix is straightly associated to kinetic energy, then there's a clear direct connection with Khalil approach of sequential energy definition
- These elements are linear combinations of standard inertial parameters
- Since base parameter choice is arbitrary, selection of a minimal representation of mass matrix is enough to assure to have identified a set of base parameters
- Inertia matrix elements are already grouped by their influence on output and affecting joints coordinates (with a physical meaning) without relying on arbitrary selections
- Inertia matrix structure reflect recursive nature of composite rigid body inertia (for serial robots)
- Inertia matrix study allows direct considerations on relevance and relative weight of each inertial parameter, allowing easier identification of essential parameters and leading to possible approximations
- Usually by inspection we shall easily see a close analogy between base inertial parameters and inertia matrix elements
- Joints nature do not provide special complexity or complication on the process

On the other side:

- Differently from base inertial parameters, there is not, up to now, a direct process for defining them (but further studies could lead to)
- Parametrization is not unique, so there are freedom relying choices

Use of base parameter for mass matrix parametrization

As seen afore, base inertial parameters is the set of inertial parameters wrote as combination of standard inertial parameters that allow full definition of dynamics. Consequently they appear naturally into mass matrix as components, and can be used for reparametrization. Vice-versa we can use any minimal re-parametrization of mass matrix as base inertial parameter set. It is indeed easy to prove that any *bijection linear mapping* between two minimal set is equivalent for both identification and dynamics definition purposes, as consequence of application of invertible linear transformations between observation or mass matrix, that are binded by equations of motion. Indeed as inertia matrix is sufficient to define dynamics equations, and consequently observation matrix on base parameter, converse is true too. Any redefinition of base parameters allows for a re-expression of equations of motion and inertia matrix.

As further observation we can say that base inertial parameters as well as standard inertial parameters do not depend on generalized coordinates or neither derivative, although depending on system geometry, differently from mass matrix elements that could depend also on coordinates, but any parametrizations in purpose of identification should exclude this dependency separating it from intrinsic links geometry dependent part. Indeed in purpose of identification we cannot admit parameters depending on variables entity in order to preserve linearity of regressor. Easiest way to apply this fundamental idea is using standard inertial parameter or base parameters defined after, for instance numerical determination on regressor, and replace in inertia matrix all these terms.

Other formal identification methods

Recently other geometry and velocity span null space based methods have been proposed in literature, exploiting the concept of inertia transport. These approaches propose original and significant insight regarding the connection between base inertial parameters, controllability and observability matrices used in classic control theory, and physical meaning associated, leading a very original and effective method for base parameters definition.

Details are provided in the following.

[Identifiability conditions in relation to observability and controllability: Null space based approach](#)

Simscape modelling

Why simscape

Simscape® is one of most modern and rich tools for multiphysics modelling actually available. Based on block diagrams and graphical layout and interfaces, it makes easier modelling using systemist language, intuitive but powerful. Notably there are few or any as powerful similar graphical language available for free. Although not universally adopted, especially in some specialististic applicative fields, Simulink is one of most commonly adopted in industrial as well as academic environments. Its power derives from not being an isolated suite, but exploiting its embedding in Matlab® environment. Simulink initially born as standalone was later acquired and fully integrated in Matlab® environment. Operative experience of this solution

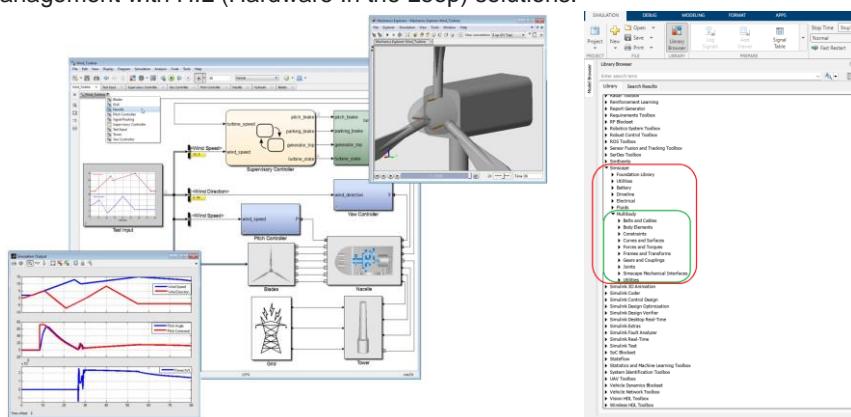
Simscape® allows an easy but effective way for modelling complex multibody dynamics

Notable characteristics of Simulink

Simulink have many powerful features that usually make it outstanding most of its competitors, in spite of maybe purpose specificity. Nonetheless presence of a rich set of application aimed libraries, and of community of developers, make it a very attractive solution. Simulink allows:

- Simulink Verification and Validation enables systematic verification and validation of models through modelling style checking, [requirements traceability](#) and model coverage analysis. Simulink Design Verifier uses [formal methods](#) to identify design errors, and generates test case scenarios for [model checking](#) within the Simulink environment.
- Simulink can [automatically generate C source code](#) for [real-time](#) implementation of systems.
- Simulink Real-Time), together with x86-based real-time systems, is an environment for simulating and testing Simulink and Stateflow models in real-time on the physical system.

As seen Simulink provides all features required for a complete digital twin realisation, from modelling to deployment, including embedded system interfacing and management with HIL (Hardware In the Loop) solutions.



What is true for Simulink is also true for most of Simscape® models and design. Indeed Simscape® modelling occur in Simulink environment, proposed as a set of libraries (shown in picture). In the following we present fundamental steps required for modelling a robot in Simscape® environment. What we present here is just part of whole set of blocks and functionalities we do not pretend to cover here under limited scope.

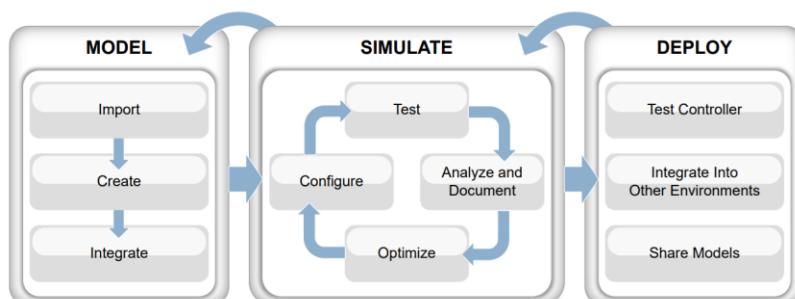
These are nowadays considered professional tools actively used both in industrial and academic field for both research and development sake, and specific use case may be found on private sources as well in community development [Matlab Central](#) and [File Exchange](#) as well on larger internet communities and sources, not rarely also not Matlab® specific (or more Python or C/C++ oriented) like [Github](#), often used also by researchers and scholars. We do not judge here, or suggest a best way, but just clue Matlab®/Simulink® environment allows for integration of different languages, and DLL use.

For more comprehensive explanations we redirect to [Simscape](#) and [Simulink](#) product pages.

Simscape/Simulink modeling process

Modeling process generalities

Modeling of a complex mechanical system could take few minutes using Slmulink®/Simscape® suite, making it a perfect tool for agile and fast development of a mechanical system digital twin. On the other side the issue of debugging models and optimize them for efficient running and correcting mistakes could be sometimes a long and hard task to achieve. In literature some examples of models for deployment process are used like one depicted hereafter. In most case, unfortunately, especially for beginners, all start in a very crafty way and multiple times stepback and redesigns are required due to limited experience and lack of confidence and knowledge used tools and their potentiality and limits. Nonetheless the advantage of agile tools like Simulink® and simscape rely on the fact even in case of low expertise usually desired target is achieved, even if maybe not in best way. The knowledge of some basis in management, process design and project management are useful assets to exploit, in particular when dealing complex projects. Having in mind some of these layouts of workflow is probably a positive point.



You can approach model development using different workflows. Some are simpler than those illustrated already, and usually match user spontaneous experience, like those following.

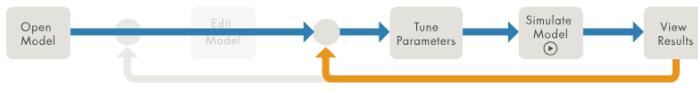
The edit-update-repeat workflow.



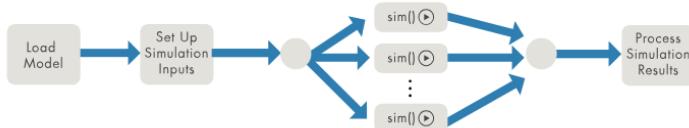
The edit-sim-repeat workflow.



The tune-sim-repeat workflow.



The multiple sim workflow.



Different workflows have different technical characteristics. Depending on your specific simulation workflow and the model characteristics (initialization time versus execution time, etc.), you could quickly refer to Table 1 below to select and try the right techniques.

Techniques	Workflow			
	Edit-Update-Repeat	Edit-Sim-Repeat	Tune-Sim-Repeat	Multiple Sim
Simulation Mode		x	x	x
Fast Restart			x	x
Simulation Cache	x	x	x	x
Model Reference - Parallel Build	x	x		
Model Reference - Incremental Loading and Rebuilding	x	x		
Simulink Profiler	x	x	x	x
Solver Profiler		x	x	x
Modify Your Models		x	x	x
Parallel Simulation				x

Table 1. Typical techniques to improve simulation performance.

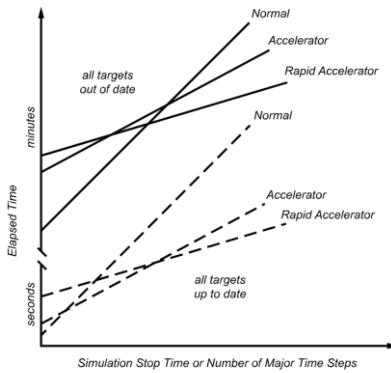
Simulink®, and consequently Simscape® models can run in different modes: *Normal*, *Accelerated*, *Rapid acceleration* mode.

In **normal mode**, Simulink® interprets your model during each simulation run. If you change your model frequently, this is generally the preferred mode to use because it requires no separate compilation step as do accelerator and rapid accelerator mode.

In **accelerator mode**, Simulink® compiles a model into an execution engine in memory, eliminating the block-to-block overhead of an interpreted simulation in normal mode. Accelerator mode supports the debugger and profiler but only a limited set of runtime diagnostics. Accelerator is often used when the simulation time is much longer than the compilation time.

In **rapid accelerator** mode, Simulink® compiles a standalone executable for the model, which can run on a separate process. You can use rapid accelerator mode only when the full model is capable of generating code. This mode restricts interaction with the model during simulations. For example, rapid accelerator mode does not support debugging. As with accelerator mode, it's best to use rapid accelerator mode when your simulations take much longer than the one-time compilation.

Which mode you should choose for your workflow can be inferred from figures that shows the performance of a hypothetical model simulation in normal, accelerator, and rapid accelerator modes.



Typically, normal mode is recommended when you are in a “development” workflow where you modify the model often and execute *update diagram* or *short simulation* between modifications. Conversely, use rapid accelerator when you want to run multiple simulations without making structural changes to the models, such as adding or removing blocks.

Simscape modelling

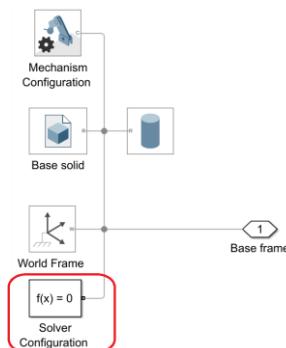
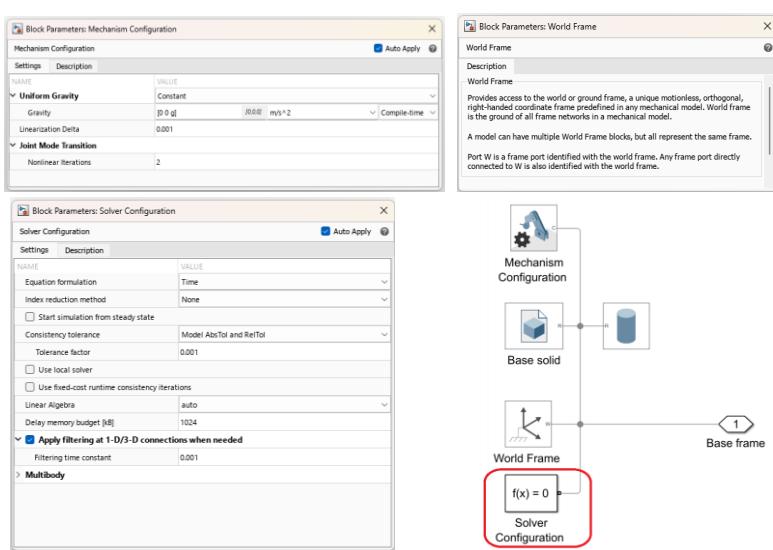
In this work we used prominently features coming from [Simscape/Multibody](#) product library, but invite to keep in mind the use and knowledge of most of Simulink® and Simscape, as well of a good fundamentals of Matlab® is required to manage and use at best proposed tools.

Simscape modelling using Simulink® environment workflow shares many different passages of the same process. In addition the use of Matlab® scripts for managing models run, setup, or in place of ordinary Simulink® blocks is quite common and is a suggested practice when dialoguing with different users and specificity. In our case for instance, the use of symbolic models is integrable into Simulink®/Simscape™ models by means of first translation in matlab code/function or scripts and then implementation of blocks, when it is used in association (on line) model execution in place of a not synchronous off-line batch execution.

Hereafter we go illustrating some of the fundamental elements used for modelling in Simscape.

Physical network concept

Every block should be part, in Simscape, part of a physical network and be connected to a *solver block*. If any Simscape block is not connected to a physical network (a solver block), an error signal will be raised. Since usual setup requires also the definition of a *configuration block* defining gravity vector, linearization tolerance, and iterations. A world frame is also necessary for defining a first initial frame, assumed inertial.



Solver block setup allow for defining which are suitable values for our simulation.

Equations can be formulated in time domain or in time and frequency domain, index reduction method, projective or derivative replacement. Solver configuration allows also to define type of solver to use, local or less. User can also set up *time constant* to use in filtering. Selection of sparse, full or auto linear algebra.

Usually all these parameter of configuration for a simscape model should be evaluated and set properly. But in truth in most of case the default setup is reliable enough to get desired results and no need for modification is required.

Inertia blocks

Beyond solid related custom or computed inertial properties (that are limited in some sense, to geometric frames and homogeneity of mass distribution when calculated) we can and prefer usually to define independently inertial and mass properties of a rigidly connected set of multibody elements by means of a specific inertia mass element with fixed inertial properties.



Also varying mass/inertial properties blocks exist, but their use is considered out of scope for this work, since theory of robotics is usually developed under assumptions of rigid body links, accounting for some flexibility if required, but almost never considering varying mass. These assumptions are rather realistic, but we can't exclude some theoretical developments and practical case of application also for this kind of modelling blocks.

Blocks allow attribution of a dedicated frame and specific inertia vector and mass in specified frame.

Joints

As said a joint connect a base with a follower.

Joints in simscape are characterised by different fields:

- Primitives
 - State target: position or velocity
 - Internal mechanics
 - Spring Stiffness
 - Damping coefficient
 - Equilibrium position
- Limits
 - Upper limit of joint position (bound, spring stiffness, damping coefficient, transition region)
 - Lower limit of joint position (bound, spring stiffness, damping coefficient, transition region)
- Actuation
 - Torque/Force (Automatically computes/Provided by input/None)
 - Motion (Automatically computes/Provided by input/None)
- Sensing
 - Position
 - Velocity
 - Acceleration
 - Actuator torque
 - Lower limit torque
 - Upper limit torque

These quantities are defined for each primitive, where for whole joint we define:

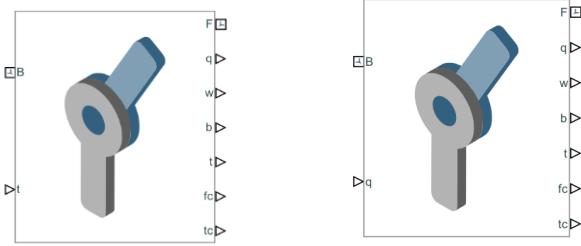
- Mode Configuration (working principal of joint)
 - Disengaged
 - Provided by input
 - Normal
- Composite Force/Torques Sensing
 - Direction
 - Resolution frame: Base or follower
 - Constraint forces
 - Constraint torques
 - Total force
 - Total torque

In the following we present a synoptic of used joints in Simscape with their primitives:

Joint Block	Joint Primitives							
6-DOF Joint	Px	Py	Pz	.	.	.	S	.
Bearing Joint	.	.	Pz	Rx	Ry	Rz	.	.
Bushing Joint	Px	Py	Pz	Rx	Ry	Rz	.	.
Cartesian Joint	Px	Py	Pz
Constant Velocity Joint	CV	.
Cylindrical Joint	.	.	Pz	.	.	Rz	.	.
Gimbal Joint	.	.	.	Rx	Ry	Rz	.	.
Leadscrew Joint	Lsz	.
Pin Slot Joint	Px	Rz	.	.
Planar Joint	Px	Py	.	.	.	Rz	.	.
Prismatic Joint	.	.	Pz
Rectangular Joint	Px	Py
Revolute Joint	Rz	.	.
Spherical Joint	S	.	.
Telescoping Joint	.	.	Pz	.	.	S	.	.
Universal Joint	.	.	.	Rx	Ry	.	.	.
Weld Joint

Joint Block	Translational DoFs	Rotational DoFs	Total DoFs
6-DOF Joint	3	3	6
Bearing Joint	1	3	4
Bushing Joint	3	3	6
Cartesian Joint	3	0	3
Constant Velocity Joint	0	2	2
Cylindrical Joint	1	1	2
Gimbal Joint	0	3	3
Leadscrew Joint	1 (coupled translational-rotational)	1	1
Pin Slot Joint	1	1	2
Planar Joint	2	1	3
Prismatic Joint	1	0	1
Rectangular Joint	2	0	2
Revolute Joint	0	1	1
Spherical Joint	0	3	3
Telescoping Joint	1	3	4
Universal Joint	0	2	2
Weld Joint	0	0	0

Most interesting joints in use for robotics applications (using robotic conventions) are revolute and prismatic joints



Ports names are conventional. Care shall be used in adopting signal compliant units, since not always checked for.

Torques driven block can be used in joint-direct dynamics (computing joint motion, then used also in hybrid case) where position driven block can be used in joint-inverse dynamics (computing joint torque, then used also in hybrid case). Imposing no torque we get a joint suitable for free motion and passive joint models.

Usual syntax and naming are used for joints.

Joint blocks have by default two frame ports. It also has optional physical signal ports for sensing dynamical variables such as forces, torques, and joint position, speed and acceleration. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Sensing Ports

The **leadscrew** joint primitive provides the following sensing ports:

- q — Angular position
- w — Angular velocity

- b — Angular acceleration
- p — Linear position
- v — Linear velocity
- a — Linear acceleration

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to 0, the joint behaves normally. If the input is equal to -1, the joint behaves as disengaged.

Valid for most of joints

Frames modelling

In general sense most delicate and error prone operation n defining Simscape modelling is the creation of a suitable set of frames. Indications for good modelling suggest:

- Use less frames possible to simplify management
- Use frames having significative meaning and placement (based on barycenter, geometric centre, joints, or other key model points) with suitable orientations (for instance z axis parallel to joint axis)
- Use distinct frames for geometric objects, joints, and inertial properties.
- Use intuitive transformations simplifying understanding and debugging.

This lead a natural need for a trade off between number of frames and pragmatism, one shall decide about. The use of some purpose specific and standard derived block may be useful like using Denavit Hartenberg method or other user defined. Anyway by author experience this has been one of most delicate (surprisingly) parts in development, maybe because a graphical based and body based method would be more intuitive and straightforward. Non seldomly the use of intermediate frames is useful, but reduce model readability.

Signal modelling

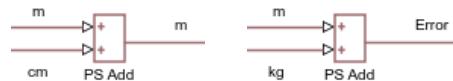
Another peculiarity of the Smsscape environment is the use of physical signals in place of traditional Simulink® signals, that are not excluded but shall be converted.

Simscape physical signals are much different from other signals used in Simulink®

- They're not monodirectional but bidirectional
- Usually unit of measure are, or should be specified.
- They can't be logged directly as other signals
- Can't be used in place of a Simulink® signal as input or output of a block

- Standard Simulink signals can't be used as input or output of a Simscape block without conversion and vice versa

Indeed for some fundamental operations some compatible replica are provided from Simscape library, together with specific sources physical signals

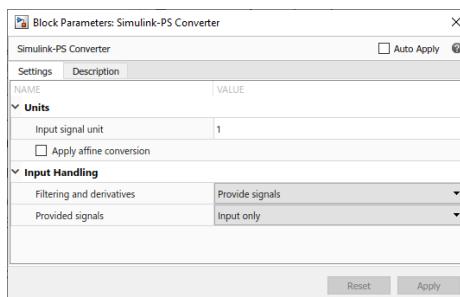


In addition physical signal are designed to manage signal derivatives, offering different options:

- Only signal
- Signal and first derivative
- Signal and first and second derivative
- Signal with first derivative computed numerically
- Signal with first and second derivative computed numerically

The use of numerical derivative can be misleading since potentially resulting in steplike shape.

We referred to defined a signal deriving from analytical differentiation and input to the model in order to avoid these spurious numerical effects potentially leading to artefacts in numerical models.



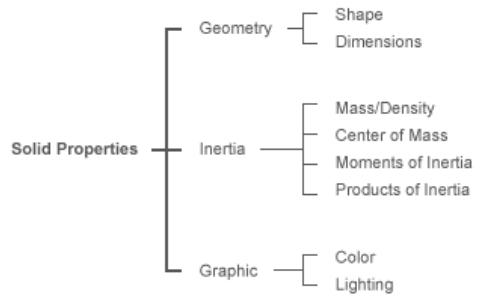
The need of using Simulink®-Simscape™ conversion is a serious drawback of this modelling approach, making models uselessly more complex.

Rigid body modeling

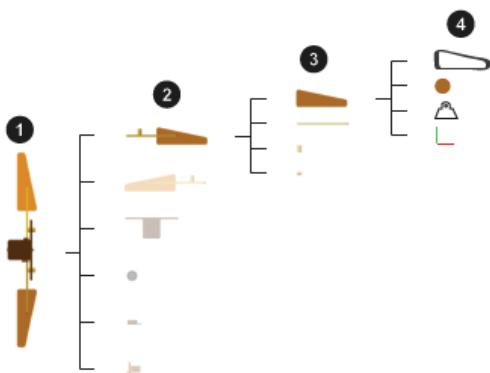
Solid properties

Each solid included in the model is characterized by a:

- Geometry
- Inertia
- Graphic



Geometric modelling and assemblies



Forces modelling

Simscape environment provide means for managing external and interbody forces and torques (wrenches)

Friction models

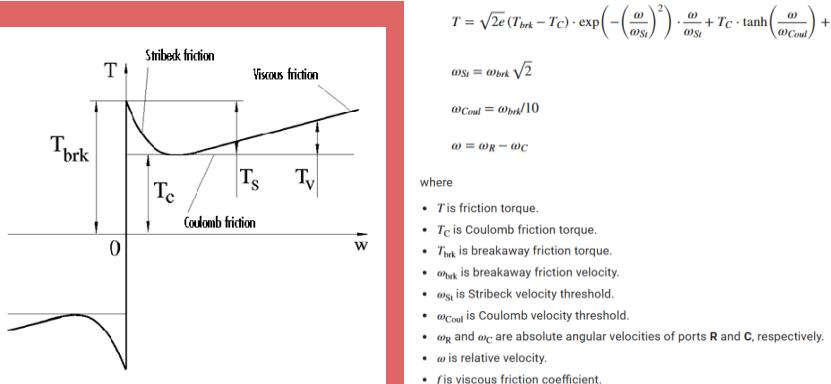
Unfortunately is not included in joint blocks models, that is rather common in all robotics and mechanical applications, the inclusion of friction terms for modelling viscous and coulomb

friction and other higher order terms if required, shall be done outside. Friction modelling could be done attaching additional Simscape blocks to joint block, allowing for more freedom in choice, but making the whole process more complex and cumbersome in terms of design.

Blocks used



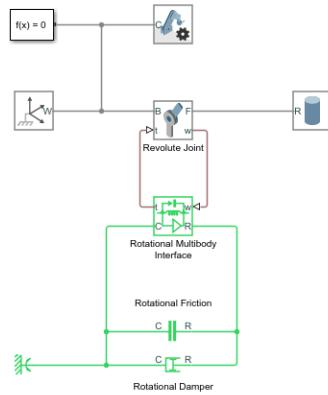
The Stribeck friction, T_S , is the negatively sloped characteristics taking place at low velocities. The Coulomb friction, T_C , results in a constant torque at any velocity. The viscous friction, T_V , opposes motion with the torque directly proportional to the relative velocity. The sum of the Coulomb and Stribeck frictions at the vicinity of zero velocity is often referred to as the breakaway friction, T_{brk} . The friction is approximated with the following equations:



The exponential function used in the Stribeck portion of the force equation is continuous and decays at velocity magnitudes greater than the breakaway friction velocity. The hyperbolic tangent function used in the Coulomb portion of the force equation ensures that the equation is smooth and continuous through $\omega = 0$, but quickly reaches its full value at nonzero velocities. The block positive direction is from port R to port C. This means that if the port R velocity is greater than that of port C, the block transmits torque from R to C.

To couple fraction model blocks to joint a specific interface block shall be added and correctly connected with joint input output ports. An example of connection of a joint with friction models is depicted in figure. Is possible also to add an actuator inertia to simulate





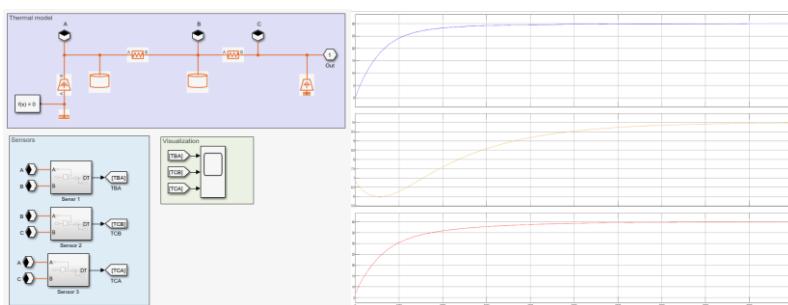
Usually temperature effects are not included in the model, but many works are focusing attention on time dependence of friction coefficients, which could be easily accommodated in generic Simscape models both in closed form solution or as dynamic subsystem (in Laplace domain). The model is provided with two time constant:

$$\frac{T(s)}{W_f(s)} = \frac{c_1}{s + \tau_1^{-1}} + \frac{c_2}{s + \tau_2^{-1}}$$

Same dynamic system can be provided using Simulink blocks or Simscape physical networks. Some authors [Hao et al. 2021] used:

$$\tau_f = \left[1 + \alpha W_{f,rms} \left(c_1 \tau_1 \left(1 - e^{-\frac{t}{\tau_1}} \right) + c_2 \tau_2 \left(1 - e^{-\frac{t}{\tau_2}} \right) \right) \right] \tau_0$$

Example of thermal models in simscape:



Mechanical systems simulation and visualization

Simscape model running and results analysis

Simscape results explorer

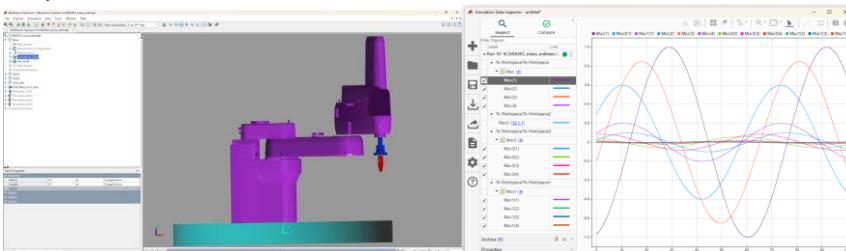
Simscape provides specific functionalities for enhanced results exploration, by means of a dedicated graphical interface, easing the process of signal survey, compare and analysis.

Although it is not a suitable tool for all kinds of analysis for a base exploration of resulting signals from one or more simulations (is possible to save and recall past data simulation) it's probably the best method. In case further analysis and data elaboration are required the use of prompt or script based commands combinations is usually more useful.

The use of results explorer requires some preliminary activity before running for predispose signal logging, since process automation passes through this step, discussed also elsewhere.

Mechanic explorer

Simscape provide additional simulation results analysis tools in multibody simulations, namely Mechanics explore, a friendly graphical interface providing animation of your model simulation results, using graphical elements specified in relevant blocks, either in case of basic geometries or complex cad models, using show/hide options useful for navigating solid model. This is very helpful for managing and investigating model response otherwise it would not be so easy by means of bare signals output. Unfortunately this tool do not provide graphical representation of some additional very useful quantities and vectors like accelerations, forces, torques, angular speeds, linear speeds, displacements, would be very useful for a deeper analysis.



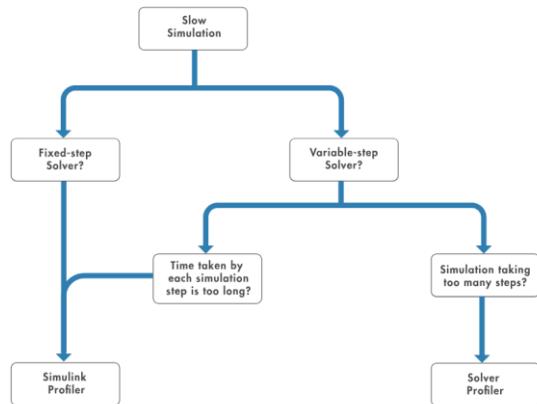
In addition the tool provides graphical representation of every model defined frame in show/hide mode, and centre of mass representation, sometimes helpful in understanding dynamics of the model.

The tool also provides means for recording some video animation for presentation purposes and a camera setting for improving animation meaningfulness. Obviously also standard set of view options like zoom in and out, point of view rotation, ans pan zoom, together with planes standard view are present.

Simulink profiler

Profiling is a standard technique in coding, programming and modelling used in order to optimize efficiency and run time. There are two types of profilers you can use in Simulink® to profile your model for *slow simulations*. If you are not sure which one to use, refer to Figure

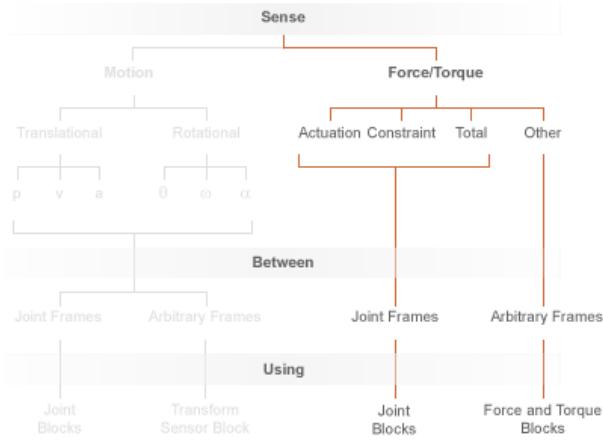
for a quick start. Simulink Profiler is for determining which blocks take the most simulation time; and Solve Profiler analyzes why variable-step solvers take certain steps. Profiler helps users in identifying the best way to improve model performances and reduce or optimize his simulation.



Simscape profiling is somewhat different and can also include Jacobians inspection and analysis.

Generalisation and automation of modelling

Conventional body frames



Structure of a link representative block

Usually when interested in representing a link or general rigid body maybe we need more than one block to give all representative information required. There are different reasons, one first is the need for multiple frames associated to a same object, especially when dealing with robotics links, we usually have at least:

- Body geometrical and/or mass center frame (with principal axes preferably)
- A joint frame with origin located on joint of predecessor (if concept is applicable) and z axis aligned with joint displacement axis
- A second joint frame for child joint (if applicable)
- Any other joint frame if more than two are present
- Others auxiliary frames used for other purposes

In literature a distinction between inboard frames and joint frames is done, where first locates joint position respect parent frame (usually located on parent joint) and the frame used to define movement of child respect to parent link. This last also for trees obviously.

In this regard we can see both [Handbook of robotics] or [Rigid body dynamics algorithms] of Featherstone.

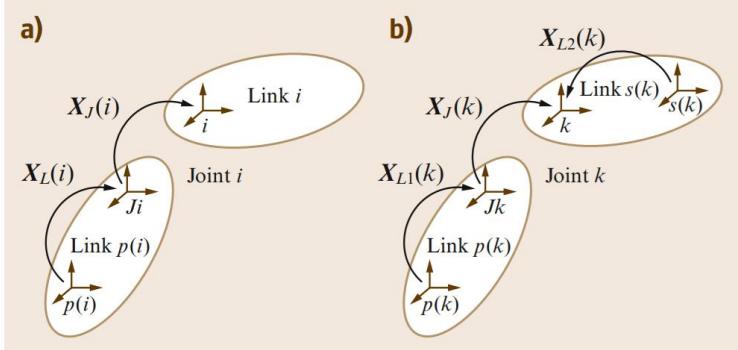


Fig.3.4a,b Coordinate frames and transforms associated with (a) a tree joint and (b) a loop-closing joint

Inertial properties of an object

For instance is good practice to separate geometrical representation of an object respect its inertial properties in order to avoid unwanted mistakes, unless there's significative chance to deduce them from geometry, an assumption to take with care because often geometry is not enough for defining correctly mass distribution of an object and consequent inertia and center of mass position.

This functionality is very helpful, especially when we want to avoid specific calculation of inertial properties where not given, usually an automatic derivation from geometry could be a suitable choice, but generally is not so reliable to take it for granted.

We can separate inertial terms from geometrical using one or more solid blocks for visualization and an inertial block for inertial properties. Indeed one advantage is the possibility to use multiple visualizations of the object (from file, with simple geometries) without caring of the impact on inertial properties of the link, and manipulate them (like geometrical dimensions and units) without impact on inertial properties and system dynamics. For instance we can change and move frames of geometrical objects for better accommodation of representation without caring about consequences.

Proper system definition requires anyway careful and correct choice of frames and where to attach objects to.

Geometrical modeling, CAD importing and inertial estimate

As presented in previous sections, the use of CAD modeling as well of any other geometrical modeling tool is one of ways researchers and industrial engineers have for estimating robot linked inertial properties.

Simscape offers similar functionalities and opportunities in almost simple, automatic and straightforward ways.

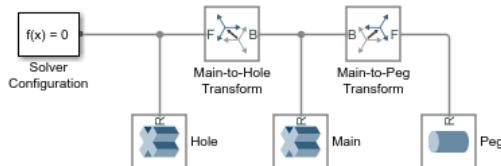
Simscape offers tools for simple geometrical modeling of tri dimensional objects like extrusions, cylinders, blocks, whose inertia properties and geometries are easily predictable,

But well aware of strong limitations offered from this primitive modeling tool, a more powerful and useful tool for importing complex geometries from common export files used in CAD modelling, like step, stl, and other formats, allow for automatically importing these geometries in Simulink®/Simscape environment and use them in our models and represent them in a virtual environment like MechanicsExplorer

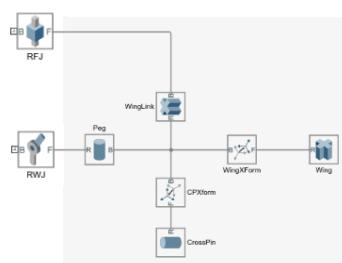
In addition a further useful functionality is given by the ability to assign to all these geometrical object an average density (assuming it's homogeneous) and derived by calculation the relevant object inertial properties:

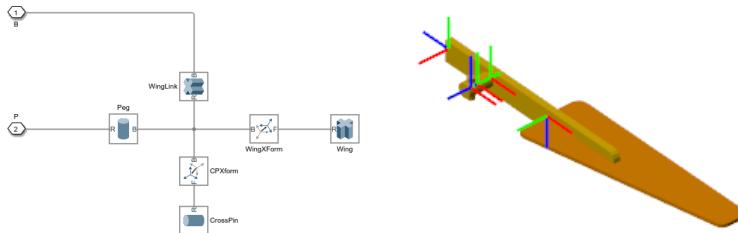
- Center of mass
- Total mass
- Inertia tensor

Please note neither a single object neither a group of compound geometrical object rigidly connected would define final inertial properties of the whole ensemble, since even pure inertial objects contribute to.



This function is fundamental for representing complex bodies





Our suggestion is to use a direct inertia measurement block in properly defined and selected frames for giving correct total mass, centre of mass, and inertia tensor, in order to avoid calculation efforts and errors threat.

Nonetheless, especially for very simple objects and configurations, this is a powerful tool for getting pretty automatically all inertial properties of even complex objects, like robotic arms of modern shape.

Nonetheless the basic assumption of uniform density is by far too restrictive to be realistic, since real robot mass distribution is highly concentrated (for example in correspondence of actuators and joints) and does not resemble uniform at all.

Nevermore, In some works researchers tried to improve approximation of a priori estimate of links inertial properties using similar techniques, for example using simplified link shape.

In Simscape three different types of inertia for an object are available:
Once we refer to so far is calculated from geometry.

Derived Values		Update
Mass	+9.916786e-01	kg
Center of Mass	[+1.469190e-02, +2.734326e-02, +1.270000e-01]	m
Moments of Iner...	[+5.871301e-03, +5.518708e-03, +7.268190e-03]	kg*m^2
Products of Iner...	[+8.673617e-19, -2.168404e-19, +1.756705e-04]	kg*m^2

Custom allows to specify arbitrary values of moments and products of inertia, center of mass and scalar mass.

Inertia	
Type	Custom
Mass	Calculate from Geometry
Center of Mass	Point Mass
Moments of Iner...	Custom
Products of Inertia	[1 1 1] kg*m^2

Inertia tensor definition is based on following:

$$\begin{aligned}
I_{xx} &= \int_V (y^2 + z^2) \rho dV \\
I_{yy} &= \int_V (x^2 + z^2) \rho dV \\
I_{zz} &= \int_V (x^2 + y^2) \rho dV \\
I_{xy} &= - \int_V xy \rho dV \\
I_{yz} &= - \int_V yz \rho dV \\
I_{zx} &= - \int_V zx \rho dV
\end{aligned}$$

Consequently the inertia tensor is symmetric. Its inertia nature is easy to prove from the above equations.

$$\begin{aligned}
I &= \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \\
&\quad \begin{pmatrix} \int_V (y^2 + z^2) \rho dV & - \int_V xy \rho dV & - \int_V xz \rho dV \\ - \int_V yx \rho dV & \int_V (x^2 + z^2) \rho dV & - \int_V yz \rho dV \\ - \int_V zx \rho dV & - \int_V zy \rho dV & \int_V (x^2 + y^2) \rho dV \end{pmatrix}
\end{aligned}$$

Also in presence of discrete or concentrated mass distributions it is possible to derive an inertia tensor, in similar way, applying density as dirac distribution (could also be added to a continuous).

$$\begin{pmatrix} \sum_k^N m_k (y_k^2 + z_k^2) & - \sum_k^N m_k (x_k y_k) & - \sum_k^N m_k (x_k z_k) \\ - \sum_k^N m_k (y_k x_k) & \sum_k^N m_k (x_k^2 + z_k^2) & - \sum_k^N m_k (y_k z_k) \\ - \sum_k^N m_k (z_k x_k) & - \sum_k^N m_k (z_k y_k) & \sum_k^N m_k (y_k^2 + x_k^2) \end{pmatrix}$$

This formulation is especially relevant not only when effectively pointlike masses and concentrated masses are present, but also as approximations of a continuous mass distribution, in order to simplify its mathematical treatment. This method is applied for instance in [Uiker] as well as in more recent works in order to simplify calculation of mass centers [Muscolo et al. 17].

For simple shapes inertia tensor can be easily derived by direct integration of geometric envelop functions, when density is constant.

[List of moments of inertia - Wikipedia](#)

Body	Figure	Mass Center	Moments of Inertia
Rectangular Paralleliped		-	$I_{xx} = \frac{1}{12}m(a^2 + l^2)$ $I_{yy} = \frac{1}{12}m(b^2 + l^2)$ $I_{zz} = \frac{1}{12}m(a^2 + b^2)$ $I_{y,y_1} = \frac{1}{12}mb^2 + \frac{1}{3}ml^2$
Circular Cylinder		-	$I_{xx} = \frac{1}{2}mr^2 + \frac{1}{12}ml^2$ $I_{x_1x_1} = \frac{1}{4}mr^2 + \frac{1}{3}ml^2$ $I_{zz} = \frac{1}{2}mr^2$
Semicylinder		$\bar{x} = \frac{4r}{3\pi}$	$I_{xx} = I_{yy} = \frac{1}{2}mr^2 + \frac{1}{12}ml^2$ $I_{x_1x_1} = I_{y,y_1} = \frac{1}{4}mr^2 + \frac{1}{3}ml^2$ $I_{zz} = \frac{1}{2}mr^2$
Circular Cylindrical Shell		-	$I_{xx} = \frac{1}{2}mr^2 + \frac{1}{12}ml^2$ $I_{x_1x_1} = \frac{1}{2}mr^2 + \frac{1}{3}ml^2$ $I_{zz} = mr^2$
Half Cylindrical Shell		$\bar{x} = \frac{2r}{\pi}$	$I_{xx} = I_{yy} = \frac{1}{2}mr^2 + \frac{1}{12}ml^2$ $I_{x_1x_1} = I_{y,y_1} = \frac{1}{2}mr^2 + \frac{1}{3}ml^2$ $I_{zz} = mr^2$
Sphere		-	$I_{zz} = \frac{2}{5}mr^2$
Spherical Shell		-	$I_{zz} = \frac{2}{3}mr^2$
Hemisphere		$\bar{x} = \frac{3r}{8}$	$I_{xx} = I_{yy} = I_{zz} = \frac{2}{5}mr^2$
Hemispherical Shell		$\bar{x} = \frac{r}{2}$	$I_{xx} = I_{yy} = I_{zz} = \frac{2}{3}mr^2$
Uniform Slender Rod		-	$I_{yy} = \frac{1}{12}ml^2$ $I_{y,y_1} = \frac{1}{3}ml^2$

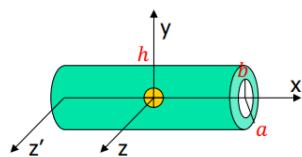
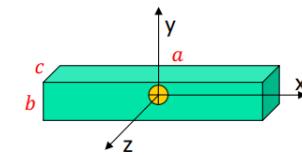
Using a general revolute profile around z axis, given by $g(z)$ we get from Pappus-Guldino's theorem, for a unitary density shape

$$J_{zz} = \int_V (x^2 + y^2) \rho dV = \iiint_V r_z^2 dx dy dz = 2\pi R_{Gz} A_r = 2\pi R_{Gz} \int_z g(z) dz$$

$$R_{Gz} = \int_z z g(z) dz$$

$$M = \int_V \rho dV = \iiint_V \rho dV = \rho V \underbrace{V}_{\rho=1}$$

That for simple homogeneous bars model lead to (using dimension conventions)



$$J_0 = \begin{pmatrix} \frac{1}{12} M (b^2 + c^2) & 0 & 0 \\ 0 & \frac{1}{12} M (a^2 + c^2) & 0 \\ 0 & 0 & \frac{1}{12} M (a^2 + b^2) \end{pmatrix}$$

$$J_0 = \begin{pmatrix} \frac{1}{12} M (a^2 + b^2) & 0 & 0 \\ 0 & \frac{1}{12} M (3(a^2 + c^2) + h^2) & 0 \\ 0 & 0 & J_{0zz} \end{pmatrix}$$

Specializing for a constant density and section (almost) straight rod with main axis aligned with x axis we get a simpler and enough general approximated expression

$$J_{0zz} \approx J_{0yy} = \int_{-\frac{L}{2}}^{\frac{L}{2}} \rho(x) A(x) \left(x^2 + o(x^2) \right) dx \underset{\rho=\text{const}(x)}{=} \rho \int_{-\frac{L}{2}}^{\frac{L}{2}} A(x) x^2 dx \underset{A=\text{const}(x)}{=} \rho A \int_{-\frac{L}{2}}^{\frac{L}{2}} x^2 dx = \rho A \left[\frac{x^3}{3} \right]_{-L/2}^{L/2}$$

$$J_{0yy} = J_{0zz} = \rho A \int_{-\frac{L}{2}}^{\frac{L}{2}} x^2 dx = \rho A \left[\frac{x^3}{3} \right]_{-L/2}^{L/2} = \rho A \cdot 2 \frac{L^3}{3 \cdot 8} = \rho A L \frac{L^2}{12} = \frac{ML^2}{12}$$

when axis are aligned with principal axis tensor results to be

$$J_0 = \begin{pmatrix} J_{0xx} & 0 & 0 \\ 0 & \frac{ML^2}{12} & 0 \\ 0 & 0 & \frac{ML^2}{12} \end{pmatrix}$$

Using symbolic approach and a distribution of section and density a more general expression is possible

$$J_0 = \begin{pmatrix} J_{0xx} & 0 & 0 \\ 0 & \int_{-\frac{L}{2}}^{\frac{L}{2}} \rho(x) A(x) x^2 dx & 0 \\ 0 & 0 & \int_{-\frac{L}{2}}^{\frac{L}{2}} \rho(x) A(x) x^2 dx \end{pmatrix}$$

That's a useful result since in most practical cases we can simplify link inertia supposing it is like a rod with joint to joint axis aligned with rod x axis.

Can be used to derive inertia around any axis given shape. immediate modifications for known density distributions is easily possible. Common formulae require to use total mass M. Same results comes out using polar coordinates or using Fubini (or Tonelli's) theorem or polar or spherical coordinates integrations with opportune axis alignment of figure (more suitable for non revolution solids) and exploiting where present geometrical and mass distribution symmetries. Advantage of this formulation is the possibility to implement it automatically using a symbolic approach.

Additionally one should always remember Huygen-Steiner theorem of inertia translation for parallel axes. Generally using coordinate free form first, then in coordinates and specialized form:

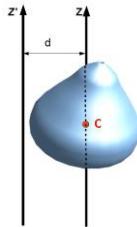
$$J = I + m[(R \cdot R)E_3 - R \otimes R]$$

$$J_{ij} = I_{ij} + m(|R|^2 \delta_{ij} + R_i R_j)$$

Or versus one axis inertia

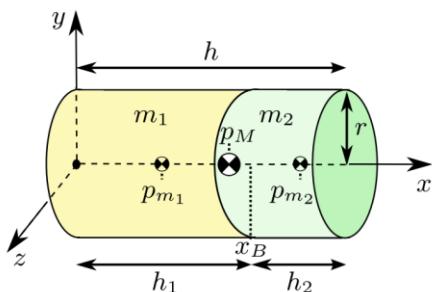
$$J' = J_0 + mR^2$$

Where R is displacement vector



Evidently under assumptions of homogeneity of density of mass, and geometrical assumptions, a parametrization of inertia tensor as function of geometrical parameters is possible. This allows a translation of an inertial parameters identification problem into a geometrical identification problem, easier to solve and check, by means of direct measures. Unfortunately assumptions used are mostly not realistic and corrections are required. Alternatively one can use an approximated model using a discretization of the robot into multiple simpler geometries. Conceptually the approach is the same.

A similar approach has been used by some authors for the sake of simplifying mapping of inertial parameters to model geometry, for instance on dynamics simulation or control applications. Indeed, unfortunately at present to simulate a free or controlled dynamics we need to fix some numerical value for dynamic model parameters, since a symbolic integration, exception done for closed form analytical solutions, is not available. A further advantage of this modeling is indeed the possibility to keep parametric and symbolic formulations, without specifying numerical values for geometrical and mass/density values. Furthermore this formulation is intrinsically consistent in the measure geometric approximation is with real geometry and mass distributions, assuring at least partially consistency with physical values.



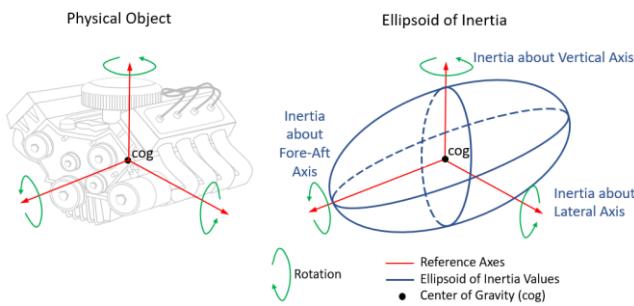
From this kind of geometrical modeling we can also infer some conditions coming from convexity of geometry on centre of mass position, inertia matrix and mass scalar value limits.

Unfortunately Simscape does not provide any constraint on convexity conditions on set parameters, thus this is the user's responsibility to keep them consistent. In addition, as said in the previous paragraph, being Simscape®, a tool mainly aimed at simulation, cannot

manage symbolic parameters, although block numerical parameters are managed both at runtime and at compile time.

As well known from the fundamentals of mechanics, diagonal elements, moment of inertia measure the spreading of mass from reference point, where products of inertia measure the asymmetry of mass distribution.

Presence of not null product of inertia reflects the fact the used frame is not aligned with principal axes. Principal axes are eigenvectors of the inertia tensor and represent directions of axis of ellipsoid of inertia.

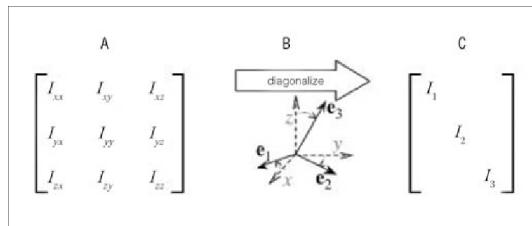


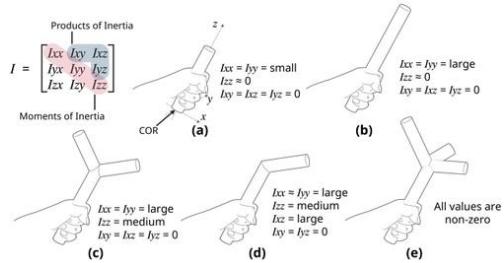
As every tensor of rank two in three dimensions, the inertia tensor has three principal directions and eigenvalues given by the solution of the eigenproblem.

$$(J_0 - \lambda I)v = 0$$

Inertia matrix is a 2-tensor, then it transforms under rotations by well known rule

$$J = R J_0 R^T$$



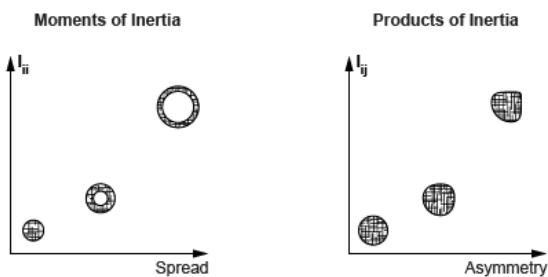


To any 2 tensor A correspond a set of invariants, trace, determinant and

$$I_1 = \text{Tr}(J)$$

$$I_2 = \frac{1}{2} \text{Tr}^2(J) - \text{Tr}(J^2)$$

$$I_3 = \det(J)$$



Consequently, the full description of inertial properties of a rigid system of masses (even compound) in a given frame is represented by a 10 components vector (disregarding order of terms), corresponding to inertia standard parameters.

$$(m \quad mr_{cx} \quad mr_{cy} \quad mr_{cz} \quad I_{xx} \quad I_{yy} \quad I_{zz} \quad I_{xy} \quad I_{xz} \quad I_{yz})^T$$

This vector transform under frame change leaving unchanged scalar total mass, where all other components changes as follow

$$m' = m$$

$$mr'_c = R \cdot mr_c$$

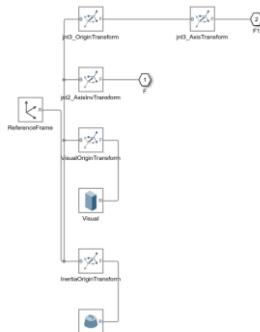
$$J'_p = R \cdot \text{skew}(-J_{xy}, J_{xz}, -J_{yz})$$

$$J'_d = R \cdot \text{diag}(J_{xx}, J_{yy}, J_{zz})$$

Definition Of library blocks

Geometry parameters, specified as a numeric vector, numeric matrix, or string scalar. The type input determines the format of this value.

Transformation block parameters



In picture a typical automatically generated link block diagram with associated references.

Automatic layout can be considerably improved using some ad hoc strategies illustrated later.

Simscape model management programmatically

The use of graphical interfaces in Simscape models can be rather slow and cumbersome, needing access to masks and windows by clicking step by step. These kinds of operations are not impossible to automate but impractical.

An alternative approach warmly suggested by expert users, allowing much more flexibility, consists in setting model properties programmatically using commands from prompt window or from script. Using this strategy is possible to perform complex operations and manage your model in a preconfigured and standardized way. Almost any block parameter is accessible by command using command

```
set param(object, param1, value1, param2,value2.....paramN,valueN)
```

and every block property can be read from model using

```
get param(object.param)
```

These commands can also be used at runtime, but update of parameters, unless explicitly commanded, occur only at model compile time.

In addition users can use these commands in [block callback](#) or in [model callback](#) in order to customize model behavior using current configuration variables value. An extensive use was made of this technique for automated model generation workflow, using standard block coloring and automatic instrumentation of the model providing input/output interface block for transferring data from script where trajectory is designed to model where dynamics is simulated..

Other useful commands include addition of blocks

[add_block\(source,dest\)](#)

Every block present many fields can be modified programmatically, some belonging specifically to block type, other general belonging to all blocks. In order to know which parameters a block have and value assumed one can use commands like

[get\(handle\)](#), providing also relative values associated to handle block (you can get using `get_param(handle/blockname,'Handle')`), or [get\(handle,'ObjectParameters'\)](#), providing a complete list of object fields.

More specifically for joint blocks we can manage properties programmatically using commands like:

```
set_param(handle,'MotionActuationMode','InputMotion');
set_param(handle,'TorqueActuationMode','ComputedTorque');
```

For position driven joints and for force driven joints:

```
set_param(get_param(h{k},'Handle'),'MotionActuationMode','ComputedMotion');
set_param(get_param(h{k},'Handle'),'TorqueActuationMode','InputTorque');
```

We can also set internal mechanics of the joint (internal spring stiffness, damping coefficient and equilibrium position), to be given as string. Further suggestion is to use arrays of values for easing access by indexing, or structures (access by field, `structname.(fieldname)`) allowing you to use a strings vectors as alphanumeric index)

```
set_param(handle,'SpringStiffness',sprintf('Ki(%d)',k));
set_param(handle,'DampingCoefficient',sprintf('Kd(%d)',k));
set_param(handle,'Handle'), 'EquilibriumPosition', sprintf('Eq(%d)',k));
```

For searching blocks to manipulate there are some ready to use functions like

```
BlockList==find_system(gcs,'LookUnderMasks','all','ClassName','Inertia');
```

option LookUndeMask is precious to find all blocks even when hidden by a masked block (for instance using a library block user defined). You can also comment programmatically blocks without modifying your model using:

```
set_param(handle,'Commented','on');
```

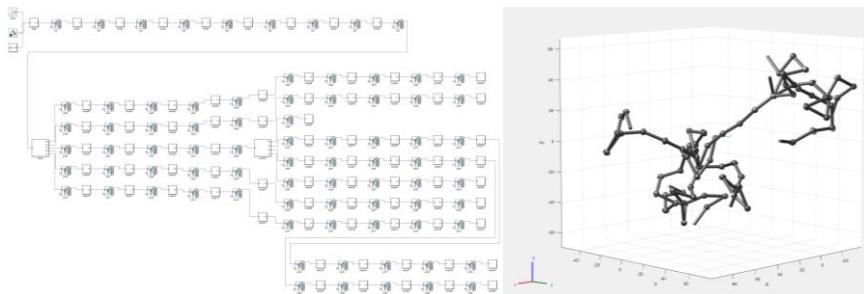
Using some scripts it is possible to manage smartly even large models with few commands, keeping configurations. For running simulations one can set simulation options (overriding model setup)

```
simIn = Simulink.SimulationInput(mdl);
simIn = setModelParameter(simIn,"Solver","daessc",...
    "StopTime","50");
out=sim(simIn);
```

At the end of simulation the object out contains all simulation data in a structure usable for post processing model outputs, using scripts commands. One can also convert easily kinematic trees built using robotic toolbox into a Simscape model using

```
[ModelName,dataFileName] = smimport(robot,'DataFileName',prmRobotFile);
```

Providing accessory files for definitions.



We want to warn reader, in spite of the easing of programmatic modeling, it cannot substitute the human part of model understanding and design required for

Simulation and solver setting

On the other side, although using solvers (numerical integrators like ode45,ode23,ode15s and many others) used traditionally with Simulink® (and Matlab®) for ODE integration, some specific for DAE systems are provided and often used (like

Solver characteristics

Main characteristics of a numerical solver are:

- Solver to be used
- Time step, fixes or variable (adaptative). Usually variable step solvers perform better
- Absolute tolerance (error of integration)
- Relative tolerance (on integration error)
- Norm error control
- Max/min step size
- Initial step

Many others are available for a finer control of solver behavior. We refer to documentation pages for more details. All solver setup options can be set using programmatically [odeset](#) function, or through graphical interface.

Solver initial conditions

The solver computes the initial conditions by finding *initial values* for all the system variables that exactly satisfy all the model equations. You can affect the initial conditions computation by block-level variable initialization, that is, by specifying the priority and target initial values in the Initial Targets section of the block dialog box. You can also initialize variables for a whole model from a saved operating point.

The values you specify during variable initialization are not the actual values of the respective variables, but rather their target values at the beginning of simulation ($t = 0$). Depending on the results of the solve, some of these targets may or may not be satisfied. The solver tries to satisfy the high-priority targets first, then the low-priority ones:

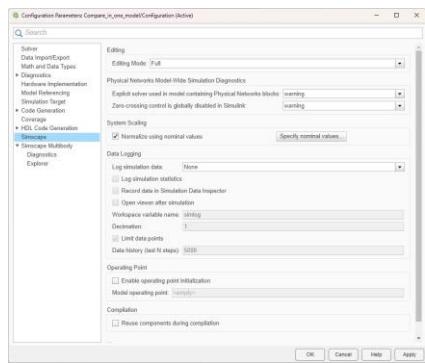
At first, the solver tries to find a solution where all the high-priority variable targets are met exactly, and the low-priority targets are approximated as closely as possible. If the solution is found during this stage, it satisfies all the high-priority targets. Some of the low-priority targets might also be met exactly, the others are approximated.

If the solver cannot find a solution that exactly satisfies all the high-priority targets, it issues a warning and enters the second stage, where High priority is relaxed to Low. That is, the solver tries to find a solution by approximating both the high-priority and the low-priority targets as closely as possible.

After you initialize the variables and prior to simulating the model, you can open the Variable Viewer to see which of the variable targets have been satisfied. For more information on block-level variable initialization, see Variable Initialization.

Simscape model setting

In Simscape model setting in addition to aforementioned options others are required/offered for a better results control (sometimes unavoidable). Useful for controlling data logging and automatising data collected at each simulation.



One useful option is the automatic launch of *Mechanics Explorer* on *model update* or *simulation*.

DAE equations

Differential algebraic equations are a type of differential equation where one or more derivatives of dependent variables are *not present* in the equations. Variables that appear in the equations without their derivative are called *algebraic*, and the presence of algebraic variables means that you cannot write down the equations in the explicit form $y'=f(t,y)$ used by ODE solvers. These equations effectively work as constraint equations for the system. In that case instead of ODE solvers, you can use DAE solvers.

$$\begin{aligned}y' &= f(t, y, z) \\0 &= g(t, y, z)\end{aligned}$$

In this form, the presence of algebraic variables leads to a singular mass matrix, since there are one or more zeros on the main diagonal.

$$My' = \begin{pmatrix} y'_1 & 0 & \dots & 0 \\ 0 & y'_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix}$$

In the fully implicit form, the presence of algebraic variables leads to a singular Jacobian matrix.

$$f(t, y, y') = 0$$

$$J = \frac{\partial f}{\partial y'} = \begin{pmatrix} \frac{\partial f_1}{\partial y'_1} & \cdots & \frac{\partial f_1}{\partial y'_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y'_1} & \cdots & \frac{\partial f_m}{\partial y'_n} \end{pmatrix}$$

This is because at least one of the columns in the matrix is guaranteed to contain all zeros, since the derivative of that variable does not appear in the equations. By default, solvers automatically test the singularity of the mass matrix to detect DAE systems. If you know about singularity ahead of time then you can set the MassSingular option of `odeset` to 'yes'. With DAEs, you can also provide the solver with a guess of the initial conditions for

Using the `InitialSlope` property of `odeset`. This is in addition to specifying the usual initial conditions for y_0 in the call to the solver.

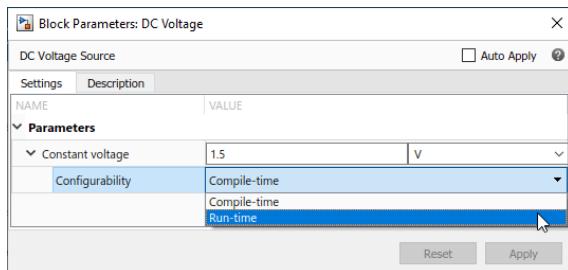
DAEs are characterized by their [differential index](#), which is a measure of their singularity. By differentiating equations you can eliminate algebraic variables, and if you do this enough times then the equations take the form of a system of explicit ODEs. The differential index of a system of DAEs is the number of derivatives you must take to express the system as an equivalent system of explicit ODEs. Thus, ODEs have a differential index of 0.

Run time and compile time parameters

Simscape as well Simulink allows for parameters get values from numerical variables (symbolic variables are not allowed) from matlab workspace or from model workspace, updated at compile time (then when model is compiled they're load and don't change anymore during simulation time) or at runtime (they can change while simulating). These options are very useful for advanced simulations when parameters changes at runtime are required for different reasons:

- User interaction
- Describe complex dynamic behaviors
- Any other

For setting block parameters as run time (default are all compile time), one should select parameter type on block mask interface. In Simscape most of block parameters allow runtime update.



Simulations with simscape models: trade off time accuracy

Sometimes simscape models (multiphysics but also multibody only) are hard to simulate because of complex dynamic interactions requiring a fine setting.

Simscape models usually are suggested to be used with dedicated solvers

Other suggestion include

- Use of variable step options to adapt solver step size to equations numerical stiffness
- Use reasonable trade off between absolute tolerance, relative tolerance to avoid excessive timespan of simulation but tight enough to assure sufficient results accuracy

Any other options exist for tuning model numerical behavior like

- Zero crossing
- Transition rate
- Others

In addition many other options exist relative logging and results data management with results explorer or by command line or script. Author suggestion is to exploit as much as possible provided functionality of data logging allowing to collect simulation data without instrumenting and modifying it manually that's often undue work better to avoid whenever possible.

Unfortunately not all data could be logged in this way and some model adjustment shall be used (even in case of debugging often!). We're going to discuss this point in the following section.

Model instrumenting

Author developed an automatic model instrumenting of simscape model in order to

- Avoid unrequired model reworking
- Simplify operations on model when necessary but somewhat mechanic and invaluable
- Automatize operations reducing human error

This work is not fully complete but is almost working and usable, automatizing input/output model set up.

In this way a model preconfigured for motion input and one for torques input could be automatically generated and used almost straightforwardly and directly simulated with any desired input profile.

Two study case considered are then

- *Imposed motion* (Inverse dynamics) where at each time a generalized coordinate, speed and accelerations are set
 - Control on consistency is left (and assured) to input generation, since three components are set independently, and no automatic filtered derivatives are used for in order to avoid numerical issues experienced during development
- *Imposed torque/forces* (Direct dynamics): at each time generalized forces applied to joints (robotic systems or mechanical system multibody case) is given in input and model define response trajectory (generalized coordinates, speed and accelerations)

- *Free motion* (Direct dynamics without input torques). Is a special interesting case useful to analyze natural system dynamics (null space of equations of motion)
- *Hybrid case, one joint*. Simscape allows for mixed situations where in a joint (usually *composite joints*) some inputs are torque/force type and others are motion, also known as hybrid dynamics (see [Rigid body dynamics algorithms]). There's a constraint in the degree of freedom *balance* to fulfill. Since this is a more complex case it is not treated in this work and left for future studies.
- *Hybrid case, many joints*: There is also a chance to select some joints to be driven by input motion and let others to be subject to torques/forces or free as a special case of null force). This is known as hybrid dynamics (see [Rigid body dynamics algorithms]) This case is of practical interest since it could be used to make some evaluation in special cases like emulating an imposed base motion for a branch and looking for a response.

Input generation

Inputs expected to be used are essentially

- Motion driven: all degree of freedom of the system are set and automatically generated and run using desired trajectory in configuration space
- Torque driven
- Also mixed situations are possible and could be generated if desired but for aforementioned

Input generation is once again based on symbolic modelling of forcing function depending on a desired set of parameters (allowing for definition of families of input using the same input structure).

Furthermore since symbolic definition is a preliminary step leading to a second step of numerical translation on a set of timepoints, numerical definition is compatible with the same workflow, loading them from other sources like:

- External datasets as file (disregarding their origin)
- External input models (expressed in mathematical or numerical form)
- Previous simulations
- Otherwise processed profiles (simplified models, reduced order models, numerical processing...)
- Datasets from experimental data
- Potentially also real time data (not implemented neither tried yet because of lack of experimental setups)
- Post Processing of real or simulated data (by filtering, fitting or interpolations)

In this way different use cases are embraced in a unified processing scheme.

For our development we preferred symbolic input definition in order to facilitate the management of complex profiles, that potentially could include fourier series few degree approximations, smoothed transitions between two configurations, polynomial trajectories (like those used in motion planning) or any other desired.

The use of analytical models is also an ease and support when dealing with numerical data time series since allow to manage used models for real data could be translated into analytical expression avoiding classic numerical issues like:

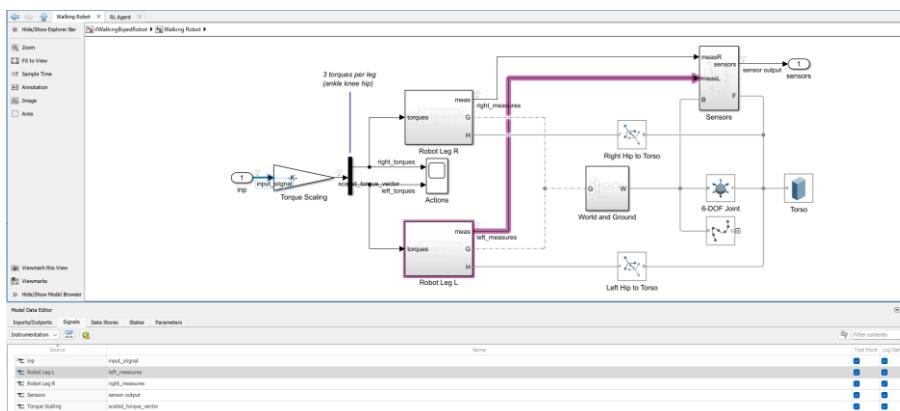
- Numerical differentiation
- Real and numeric noise
- **Windowing**
- Function manipulation

Since at some point of processing we may need to apply some analytical operation to our signals, we've better to use mathematical expression in place of numerical sequences, provided they're enough effectively representative, with better results even respect traditional filtering frequency based or digital filtering, when signal structure and shape is in someway already known a priori and we just need to match proper values.

Requirements and test points

In Simscape™ and Simulink® environment suggestion of using signals and datalogging is warmly suggested and supported. Using signals, data stores, states and data logging users can set up models for purpose and customized use also using setup scripts programmatically or by command line, or by model menu of signal manager.

In this way one can decide signal naming and renaming, select them for data logging and testing without navigating throughout the whole model, a sincerely frustrating experience when model is large and your machine a little bit not up to date.



Another major point is the use of *requirements* for models. Adding requirements to signals allows you to perform model testing and validation rigorously, structurally and easily, without excessive workload from the user side, saving time and energy. Most of the process is already designed for a structured and friendly use, providing requirements tables downloadable and uploadable as files. Requirements allows you to specify your model desired values, and using opportune formalization is possible to automatise some testing tasks , avoid user defined tests design, work demanding and error prone.

Requirements are a subtle and complex topic in model development. Some typical situations could occur is to have

- Conflicting requirements
- Incomplete requirements sets
- Wrong requirements
- Requirements-model misalignment (what you expect is not what model does)

Requirements also involve considering different styles of designing and modeling, each one depending on your target and experience.

Assumptions also are of most importance.

Not all requirements are verified by simulations. Formal verification methods can be used.

Validation of requirements is a complex task because usually work demanding, and a lot of workload reduction is reachable by smartly selecting testing cases.

In this work no explicit use of model requirements was made, but its use in Simscape modeling and digital twin building and development is warmly suggested because it is a structured and professional way of development making products ascribable for effective use in industry. By the way also in academic and research environments there can be a way to produce well specified and formed models for exchange purposes.

Sensing measurements from frames and body relative position and speed

Simscape provide [transformation sensor blocks](#) able to provide measurements of different types:

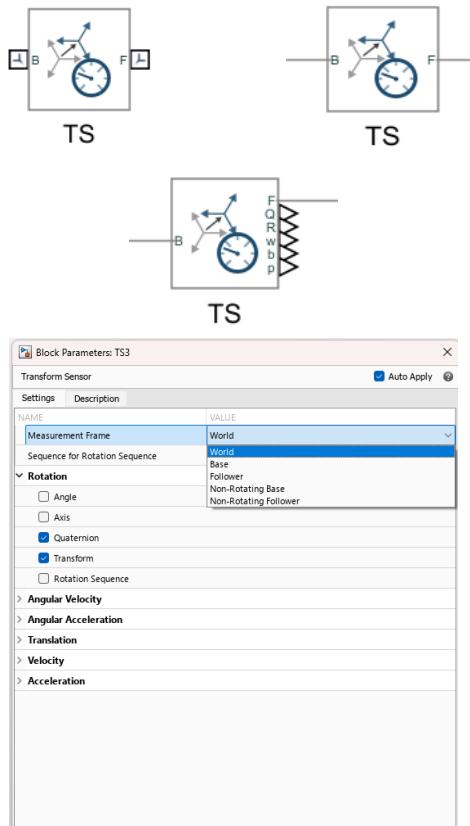
- Frame relative orientations (as angle, axis, quaternions, transform matrix, rotation sequence)
- Angular velocity (as components or vector or as transformation, quaternion or sequence)
- Angular acceleration (as components or vector or as transformation, quaternion or sequence)
- Translation (as components or vector or radius, azimuth, distance or inclination)
- Translational velocity (as components or vector or radius, azimuth, distance or inclination)
- Translational acceleration (as components or vector or radius, azimuth, distance or inclination)

Measure are relative to

- World frame
- Base frame
- Follower frame
- Non rotating base frames
- Non rotating follower frame

Notions of *base* and *follower* frames come out from block connection as depicted in figures.

Sensed measures selected are provided as output ports to the block.



This block is focused on the measurement of kinematic quantities, for measuring other dynamics related quantities like torques and forces we need to use a different approach.

On the other side one shall consider that some quantities are measurable only by direct use of relevant blocks for use purpose. We refer to joint blocks providing capability of logging of relevant dynamic and kinematic quantities under [joint sensing](#) tabs(see also [measurement frame selection](#), [translational measures](#), [rotational measures](#))

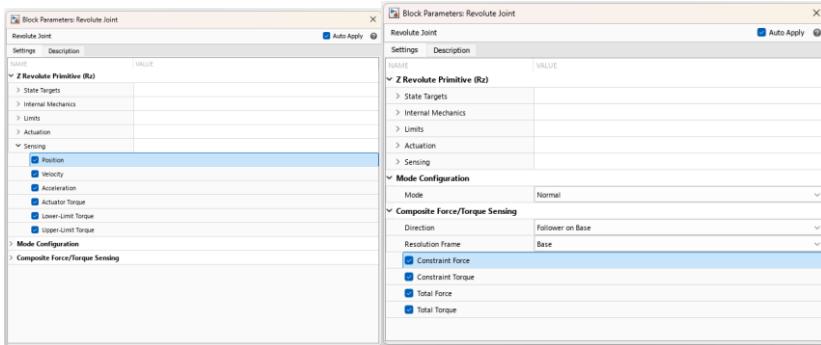
- Joint position
- Joint velocity
- Joint acceleration
- Actuator torque/force
- Lower limit torque/force
- Upper limit torque/force

Under section composite [force/torque sensing](#) (but also [constraint forces](#) and [joint acting forces/torques](#))we've other relevant and useful dynamic quantities:

- Constraint force

- Constraint torque
- Total force
- Total torque

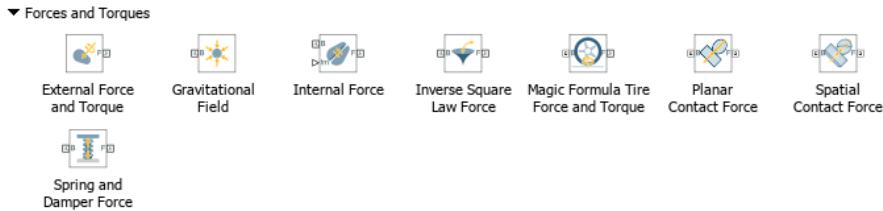
In this case we need also to specify which is the reference frame used for reaction wrench resolution among base or follower, and relevant direction of action base on follower or follower on base.



Forces and torques

Simscape libraries provide different forces source defined as

- *External* forces/torques (gravitational, generic)
- *Interaction* or *internal* forces between frames or bodies (inverse square law, spring damper, generic interaction, spatial contact, planar contact)



Connecting relevant frames allow to introduce into the model forces and torques coming from non inertial neither controlled sources, but simulating some kinds of environmental and interaction deriving forces and torques. A classic and useful example include contact forces/torques and square law forces.

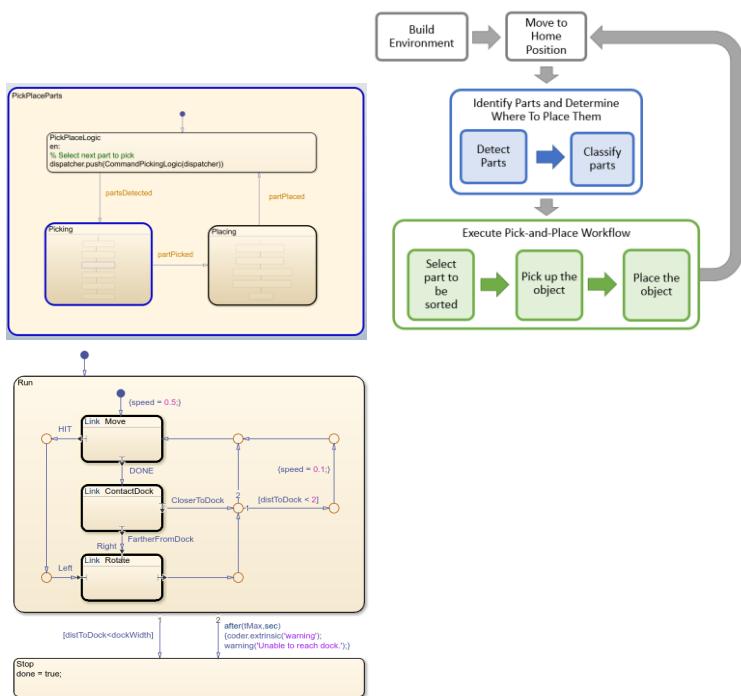
Stateflow blocks

Inside a model Simscape users are allowed to insert some [Stateflow](#) blocks in order to create state machines able to govern and manage system state providing different operative modes.

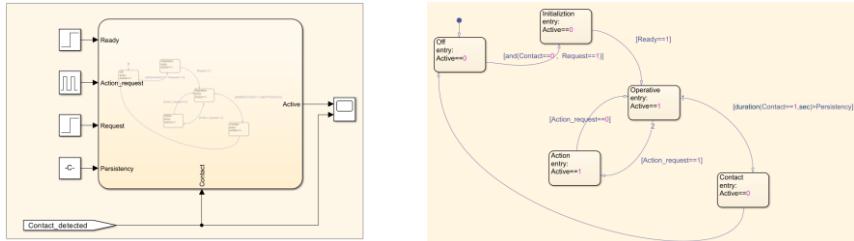
Stateflow is a product that provides a graphical language that includes state transition diagrams, flow charts, state transition tables, and truth tables. All these tools allow for managing finite state machines usually adopted for controlling system operative state,

exceptions and operative modes. Although managing state machines, flow charts and state transition is not impossible in other ways, using scripts or Simulink® diagrams and blocks specifically designed, is *highly recommended* the use of a properly and purposefully tough tool like Stateflow, because of its targeted and optimized aimed design. In addition considering automatic code generation, Stateflow is one of tools applicable, finding applications even in major NASA space missions. Custom methods otherwise would lead to, first a considerable effort, second is prone to bugs and errors, as well as not optimal, unless after a long use. On the other side we do not deny sometimes (very unlikely) a custom solution is unavoidable or preferable. In regard of specific application to robotic and digital twin modeling, it can be used for designing or simulation of implemented robot state machines.

Some examples are given in the following:

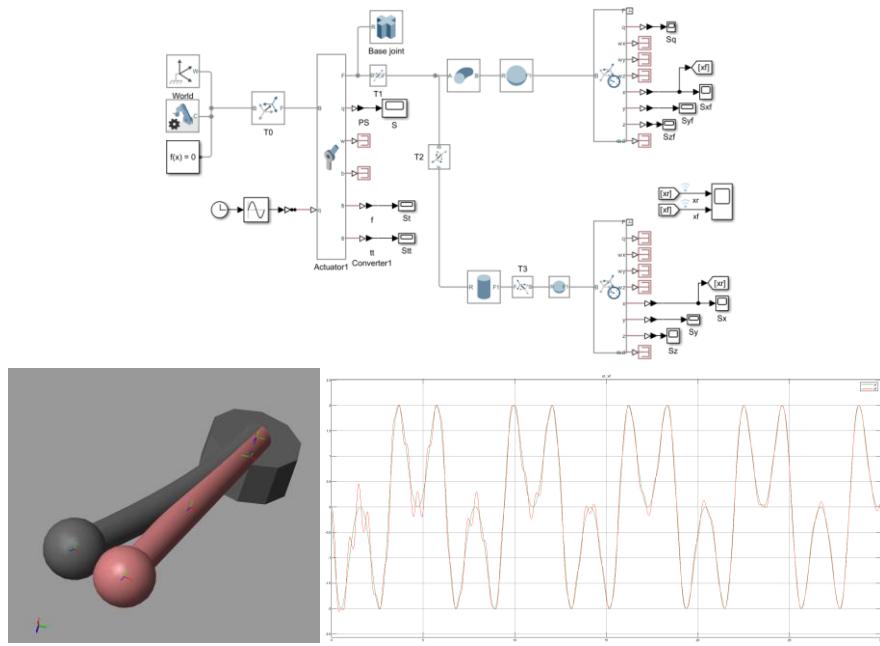


We implemented a simple state machine for managing unexpected collision using a simulated contact sensor, and a simple state machine for start up and health check.



Flexibility effects in simulations

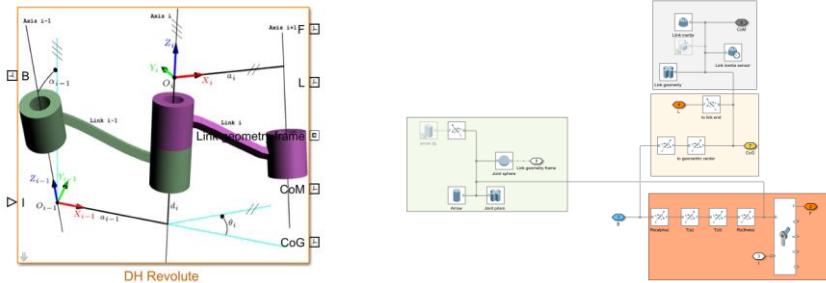
We can present some introductory results on simple models for flexibility effects.



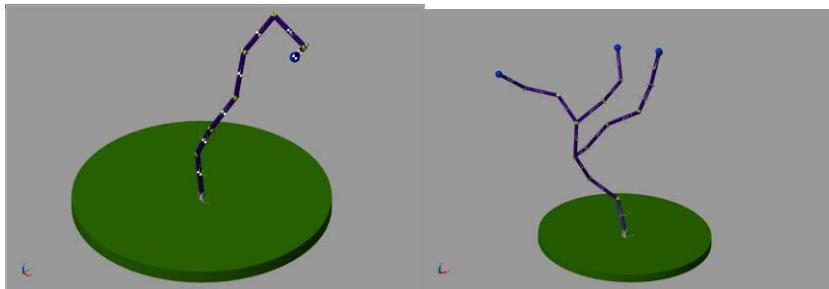
Simscape blocks libraries configured for managing Denavit Hartenberg parameters

Default Simscape blocks do not manage directly DH parameters as would be nice to have.

We just defined a set of library blocks where DH transformations are converted into frame transformations following sequences admitted from respective formalism. Usually DH formalism include actuation joint, but this can lead to issues since is required to select whether a joint is position driven or torque driven.



Blocks allow you to give input I as torque or position (block setting occurs at compile time). Output include joint follower frame F, centre of mass position CoM and geometric centre CoG. Using these base blocks it is easy to build up any serial robot or even a kinematic tree in a few steps. It is also possible to build up some block libraries for arms, legs and so on. Remark writing down an analytical model for a high dof system would be prohibitive in terms of symbolic terms overcoming memory or time of elaboration limits after a few tens of dofs. This architecture allows you to reconfigure links quickly link's connections as you like and reconfigure joints orientation and positions. Note anyway collision check is not automatic, so undesired and unrealistic kinematics configuration could occur.



Using blocks with Denavit Hartenberg parameters also management on multiple joints links can be managed with dedicated blocks or using some block parametrization and customization at update time. In addition further blocks using flexible links and joint flexibility effects are possible, although not implemented in a first attempt. Remark all modelling occurs numerically and no analytical counterpart is provided automatically but would be a nice enhancement to be able to extract it directly from the model.

For properly and generally defining a block suitable for implementation of a DH parametrization Simscape block is opportune to decouple:

- Kinematics frames, directly related to DH parameters
- Link geometry frames
- Joint geometry frames
- Link Inertia definition frame
- Actuator frame (geometry and inertia can be left coupled in case)

Further considerations are required from Simscape specific joint modelling design, since we need to define operative mode of our joint, between

- *Position driven*
- *Torque driven*

In addition we need to choose whether to give some additional properties like initial position and speed, internal mechanics (stiffness and damping).

What presented so far is just a single degree of freedom block, but nothing forbids the adoption of more complex joints, still keeping DH parameterization. Hence balance between inputs and calculated joint parameters shall be consistent and numerically solvable.

Furthermore in the most general situation of tree robots, we can have multiple joints associated to the same link, henceforth link inertial properties remain one set per link, but joint position and orientations are possibly multiple. Thus more than one set of DH parameters (one for each joint associated with the link, is required). It could also occur that some of the parameters are common between all or part of joints, but every set shall differ by at least one (the variable one for instance). In picture illustration of the typical configuration

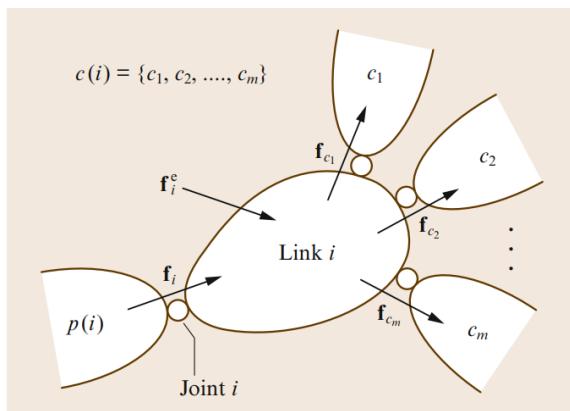


Fig. 3.6 Forces acting on link i

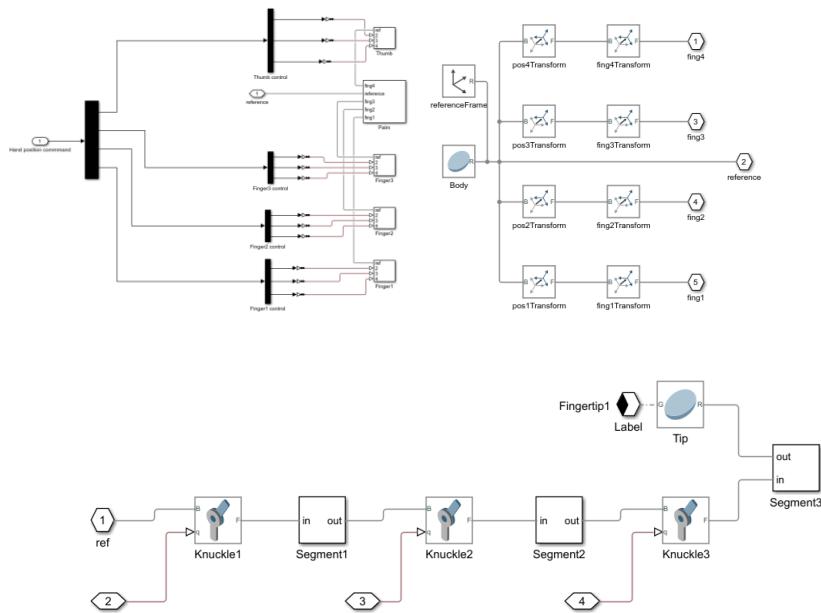
Modelling such a situation is nonetheless difficult to manage generally since requiring structural modifications to our block.

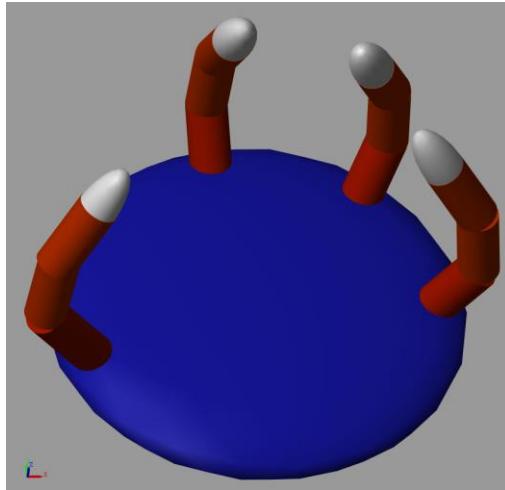
For physical consistency a further set of constraints should be included strictly speaking for managing collision between links elements (even in simple chain models), but usually this is neglected for burden reasons. One can alternatively manage it imposing some limits to variable range.

In addition one can choose to add to the model block some flexible element in order to allow and compare rigid body motion with flexural effects, and their propagation.

Ideally a user achieves the same results by simply using default library blocks, but adopting some specifically designed blocks for robotic modelling designed following usual conventions make the job really straightforward, easy, fast and whole process less prone to errors and contemporarily easier to manage and update leveraging explicit grouping of key variables of interest. Unfortunately at the moment no robot aimed library is available in Simscape, and most of the work is still on designers shoulders. By the way, it is not a big deal, unless we shall face some lacks and divergences, respect, for instance Robotic Toolbox.

Grippers modelling





Analytical model generation from Simscape/Simulink model

As remarked in other sections in spite of the fact code generation is a provided feature for Simulink® and Simscape™ products, there's no way to extract directly analytical model from a Simscape model. On the other side the process is viable, using a custom solution, at least for not complex model, as robotic cae could be. Using rigid body equations for each link (for instance starting from custom defined blocks), one can write Euler equations of the body, then equations can be derived connecting them using joints maps. Similarly kinematic equations can be derived. We left this option for a future development aimed at improving interoperability of analytical models with graphical and numerical models. On the other side we didn't proceed in this way aware of limitations of analytical model may be uneasily managed respect their counterpart (joint limits, contact effects etc.).

Inverse dynamics model (IDM)

Inverse dynamics simulations are easier to carry out both analytically and on a Simscape model, since there are no significant integration steps to perform.

In the analytical case we just insert expressions for coordinated and time derivatives up to second order and insert them on our model to carry out force/torques output.

In the Simscape model the output is just the result of propagation of forces and torques on every frame due to the motion and usually does not take long time neither imply significative numerical errors.

In spite of simplicity, this also the most useful application since

- Allow to retrieve forces and torques associated to a prescribed motion, usually a desired path in operational space
- Is useful for identification purpose and to monitor system (real or digital) response versus expected one
- Could be used for machine learning purpose as a training set.
- **Others?**

Direct of forward dynamics and simulation (FD or DD)

Direct dynamics requires numerical integration of differential equations sometimes also stiff (or in simpler words, equations whose dependence of derivatives on state change abruptly, making integration hard to carry out). When dealing with stiff problem specific integrators are suggested in order to get significative results and avoid numerical issues.

In most of cases numerical integration is anyway an hard task requiring long simulation time and small integration steps not always compatible with real applications and use like co simulation with real systems or twinning with other digital models having different pace, causing bottlenecking issues.

In these situations adoption of lower precision numerical schemes like discrete time (Euler forward integration scheme) may be usable but expose to numerical divergence if not crosschecked, for instance closing a loop of control with some real data measurement

On the other side direct dynamics may be somewhat less important than inverse dynamics on applicative purpose. Indeed where is possible to accurately measure joint position and this is usually done, joint torques are affected by many contributes like internal and external frictions, aerodynamic forces, external loads, not always meaningful simulated response, without having a complete and realistic model of the system. Conversely is not true for inverse dynamics where we can in case use response as a first approximation and add some contributes to joints due diverse factors.

This is related to numerical stability of the problem, sometimes not assured. Indeed is well known that free dynamics of a robotic arm is like the one of a composite pendulum, that are as well known in literature chaotic systems. Consequently small differences in system state could lead to significative differences in final trajectory when there's not a control loop to stabilise them.

Essentially is the difference in complexity between direct and inverse model to drive, together with uncertainty on affecting factors and input, and integrations accumulated error, the factor leading to a different level of effort for the two models.

Consequently is hard to imagine to develop a one to one digital twin that allows open loop torque calculation and imposition for the reproduction of a planned trajectory, but adding some feedback terms on desired position should be rather easy stabilise results and avoid divergence using a sort of feedback linearization of the system.

A special care shall be used with using joints with no inertial properties on follower side (directly connected to joint out frame, since they lead to singularities preventing simulation both in direct and inverse mode or any other. Consequently using joints at least a small negligible inertia or mass shall be imposed to resolve joint motion (whether prismatic or revolute), when forces and torques calculation are required (as we do for default in our models), or vice versa when kinematics is calculated from applied torques.

Differently from analytical approach then, it's no possible to create virtual joints just nullifying masses and inertia, as often done in robotics.

Results are substantially unaffected from this artefact modelling trick, but in design should be considered as a caution to keep in mind.

Hybrid/mixed inverse-direct dynamics with Simscape models

As previously stated is possible using Simscape easily manage a third way that is not thoroughly direct neither thoroughly indirect case. While doing it in other context could be rather hard and awkward to carry out because a mixture of different methodologies should be used simultaneously, without sufficient analytical support, using Simscape results in rather straightforward and natural, just matter of setting joints input properly.

We can see an opportunity to set only part of joints in direct mode and part in inverse mode leading to another version of mixed or hybrid mode dynamics. See Featherstone for theoretical references [Rigid body dynamics algorithms].

Alternatively another hybrid condition regards the use of mixed computed/input joint conditions, since as explained, Simscape™ constraint regards the resolution of the kinematic/dynamic problem, no matter which quantity is input or calculated. This last mostly for complex joints where multiple input exists, respect traditional single degree of freedom joints where only trivial combinations exist. But is anyway an opportunity offered from the model that is not usually contemplated in analytical models, even when dealing with complex joints, although not impossible, for sure not easy.

The use of mixed modality is fundamental for instance in the simulation of underactuated motion, where some joints are active and some others are passive. Underactuated motion has proven to be a prominent field of research on modern robotics, leading to energy efficient and more natural motions.

Gripper modelling

Using SimscapeTM also modelization of complex mechanisms like grippers could become rather affordable without specific knowledge. In this work we provided a rather simple example of automatic design and simulation tool of a gripper system, taking as parameters geometric design dimensions (finger length, palm dimensions, fingertips) and other geometric parameters like number of fingers, number of elements for each finger, angular distance between fingers.

Advantages are different:

- Parameterization
- Ease of modelization
- Integration with simulink
- Ease of assembling

It is indeed direct to assemble a multibody structure in SimscapeTM, just connecting one of the body elements (usually the root, but is not mandatory) to any frame of another multibody model.

In addition, in case one wants to perform simulations neglecting some subset of bodies or groups of, they just need to comment them and uncomment where desired, making model management quite easy even for not specialistic users.

Code of Practices (COP)

Although there are no strict rules of modelling beyond those imposed by modelling language itself, it is good practice to choose a set of criteria for our modelling, because of many well funded reasons.

1. Modelling homogeneity, readability, understandability, compatibility, exchange
2. Use *colours* for highlighting different purpose of model parts (logic blocks and signals, numerical value signals, signals involving different UoM, physical signals)
3. Use *standard blocks* and *library blocks* where possible in place of customised blocks (usually they're optimised and more reliable)
4. Choose signals and bus architecture as leading choice for your connections, where applicable
5. Use *naming conventions* for blocks and signals (easing of programmatically access and finding elements and their management)
6. Choose criteria for using signal connections in place of other routings (Goto/From tags, Dataread/write blocks, Simscape labels...)
7. Use blocks properties as tags and user data in order to improve model manageability
8. Use blocks *callbacks* for configuring blocks at compile time in place of explicitly command based or using manual setting
9. Use *programmatically* simscape design *scripts* for setting up your model properly and coherently with current configuration parameters. This help keeping order in your model and reuse them
10. Use formalised methods for run (*case tables*) and reporting *templates* for your simulations and models (*model reports*)
11. Design an operative *workflow* fixing key points of the process
12. Provide *sandbox* for customisations and trials
13. Provide official outcomes in *folders* conventionally named

The use and compatibility, as well the efficiency of resulting models would be significantly increased by adoption of these technical standardisation in a community of development.

Templates for reporting and simulation

A simple tool for automatic reporting was created in order to make reporting operations easier and more automated.

Reporting tool creates a pdf document with the list of figures generated from work process launched (usually some script).

A further reporting function allows for creating a pdf document with a full documented report of Simulink® model blocks and properties.

A third tool created is for managing simulations orderly. A simple google sheet table is used for launching a set of simulations to run, provided in the form of a string of commands in case are required setup command or script invoking.

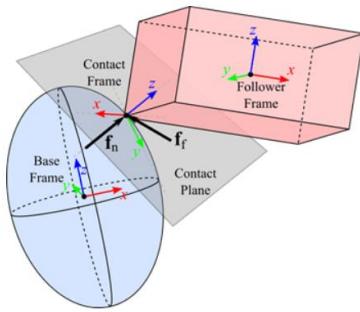
These kinds of tools are not strictly necessary for specific thesis work development, but was considered a helpful tool for debugging trials and organising simulation sessions of explorative or presentation purpose. Similarly reporting allows to reduce work effort in both presentation and analysis of simulation sessions.

Modelling contact forces

Contact interactions are some of the most troubling and difficult to manage problems in mechanics because of its dependence on geometry and discontinuous nature of contact forces. Also at simulation level introduction of contact conditions make simulation more difficult and time demanding. Analytical modelling of contact forces is usually avoided for these reasons, as well in custom numerical modelling with tools like simulink or matlab, or other simulation libraries. But is also a most important topic in robotics especially when dealing with objects manipulation and other physical interactions occurring in ordinary robot operations. For instance modelling of a gripper would require as minimum this kind of physical modelling.

In Simscape™ modelling contact between objects is rather easy and straightforward because:

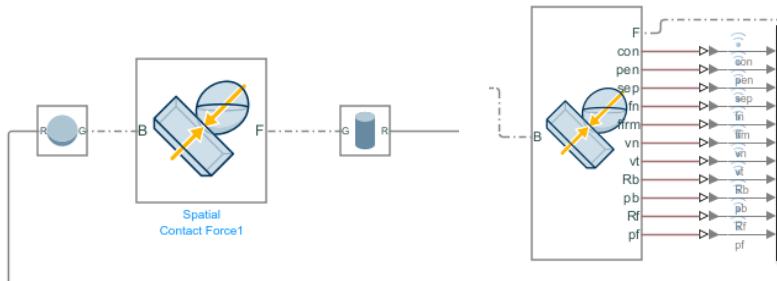
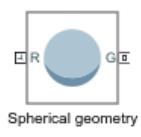
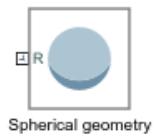
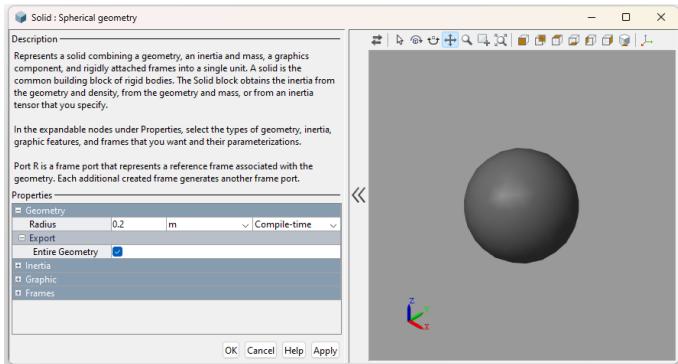
- Geometric modelling is implicitly assumed as part system description and modelling
- Ready to use blocks are available providing all required inputs and outputs
- Simulation of the model includes these options like force calculator or others



Simscape™ provide some blocks for modelling contact forces, mainly spatial contact, and planar contact force blocks.



For modelling contact we need to specify geometrical surfaces of contact. This requires to modify associated solid block to output envelop geometry for export, providing a further output port to be connected to contact block, at least, and in case setting up further outputs to log for any purpose.



When using Simscape™ one shall specify contact parameters. Key parameters to be defined includes:

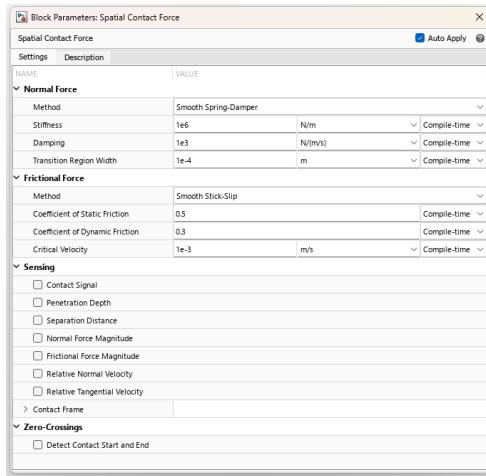
- Method of contact physical modelling of normal
 - Smoothed spring damper model (simple and faster, mostly used)
 - Input given
- Frictional forces modelling
- Sensed quantities including
 - Contact signal
 - Penetration depth
 - Separation distance
 - Normal forces magnitude
 - Frictional forces magnitude
 - Relative normal velocity
 - Relative tangential velocity

- Zero crossing: contact start and end

When defining contact as spring damper model we need to specify

- Stiffness with UoM (usually [N/m])
- Damping with UoM (usually [N/ms])
- Transition region depth with UoM (usually [m])

As usual all these parameters can be set at compile time or at runtime ([tunable parameters](#)).



Contact normal force model for smooth spring damper gives

$$f_n = s(d, w) \cdot (k \cdot d + b \cdot d)$$

where:

- f_n is the normal force applied in equal-and-opposite fashion to each contacting geometry.
- d is the penetration depth between two contacting geometries.
- w is the transition region width specified in the block.
- d is the first time derivative of the penetration depth.
- k is the normal-force stiffness specified in the block.
- b is the normal-force damping specified in the block.
- $s(d, w)$ is the smoothing function.

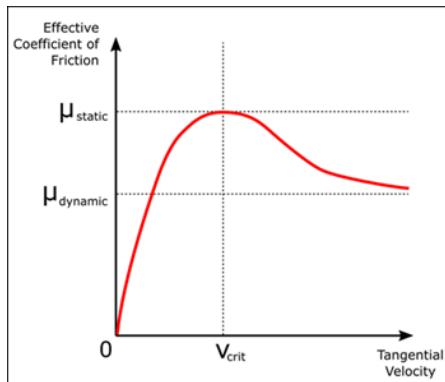
If you select Smooth Stick-Slip, the frictional force is always directly opposed to the direction of the relative velocity at the contact point and is related to the normal force through a coefficient of friction that varies depending on the magnitude of the relative velocity:

$$|f_f| = \mu \cdot |f_n|$$

where:

- f_f is the frictional force.
- f_n is the normal force.
- μ is the effective coefficient of friction.

The effective coefficient of friction is a function of the values of the **Coefficient of Static Friction**, **Coefficient of Dynamic Friction**, and **Critical Velocity** parameters, and the magnitude of the relative tangential velocity. At high relative velocities, the value of the effective coefficient of friction is close to that of the coefficient of dynamic friction. At the critical velocity, the effective coefficient of friction achieves a maximum value that is equal to the coefficient of static friction. The graph shows the basic relationship in the typical case where $\mu_{static} \geq \mu_{dynamic}$. In this case, the model is able to approximate stiction with a higher effective coefficient of friction near small tangential velocities.



In this case, the model is able to approximate stiction with a higher effective coefficient of friction near small tangential velocities. When modelling contacts that involve a point cloud, the Spatial Contact Force block calculates the contact quantities for each point and the output signals have the same order as the points specified in the Point Could block. For the points that are not in contact with the other geometry, the measured values are zero. When using input forces, the size and order of the input signals must match the size and order of the points specified in the Point Could block.

Contact models allow to manage a variety of situations like:

- Obstacles collision
- Floor contact
- Walking simulation
- Rolling on surfaces (wheels, objects)
- Bouncing against walls
- Gripping

Compared to modelling the mechanical relationships in a multibody system, modeling contact presents more choices, and the chosen methods can impact simulation speed, accuracy, and model maintainability. Simscape™ Multibody™ provides basic contact modeling constructs, but there are many ways to use them.

Solver Parameters

The computed normal contact forces are continuous functions of penetration depth and penetration velocity. Simscape™ Multibody™ computes these normal forces by reducing the spring and damper forces when the depth is less than the **Transition Region Width**. Increasing the **Transition Region Width** reduces the sharpness of the contact force, making the system easier for the solver to advance, but allowing greater amounts of penetration. Decreasing the **Transition Region Width** gives the contact forces a sharper profile, closer to idealized rigid-body contact. However, decreasing the **Transition Region Width** may also degrade solver performance.

Setting the maximum step size to 1e-3 seconds or lower improves accuracy in many models. Reducing the relative solver tolerance can produce a similar effect. However, reducing the step size also reduces the simulation speed.

Explicit solvers, such as ode45, are usually better for systems with many rapid collisions or contact changes. Implicit solvers may struggle under such circumstances. If the contact changes are infrequent and more stable, implicit solvers may offer a speed advantage due to their ability to handle the stiffness introduced by the contact forces.

Limit Penetration Depth

The separation distance between two solid blocks is positive when the two geometries are not in contact, zero when they are touching, and negative when there is nontrivial penetration between the geometries. When the separation distance is negative, its magnitude is also known as the *penetration depth*. Penetration depth is a continuous function of the positions and orientations of the geometries.

Simscape™ Multibody™ uses geometry analysis algorithms that are tuned for small contact-modelling applications. Some of the algorithms assume that the penetration depth is small compared to the size of the geometries. If this is not the case, the computed (negative) separation distances may only be approximate, or may exhibit discontinuities, and the resulting contact forces may be erratic. For best results, limit penetration depth in your model.

Flexible bodies

A single Brick Solid, File Solid, or General Flexible Beam block may suffice to completely model a body. More often, several are required. The body is then a composite of simpler body elements fixed to one another. Frame connection lines between the blocks establish the necessary rigid connections between the body elements. Rigid Transform blocks, normally inserted in the connection lines, provide the relative positions and orientations required for proper assembly.

Simscape™ allows furthermore to model flexible bodies using different blocks, mainly flexible beams, representing slender bodies with specified cross-sections and reduced order models

The Reduced Order Flexible Solid block creates a flexible body by using reduced-order model (ROM, see also [MOR](#) concept) data.

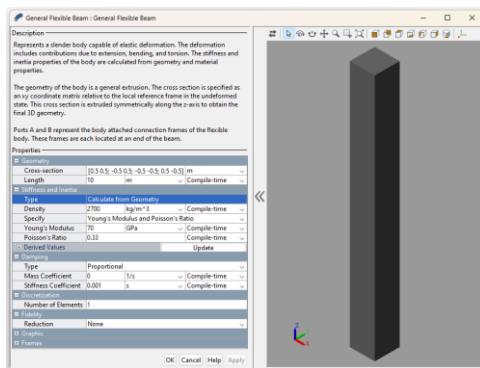
A reduced-order model is a computationally efficient model that characterise the properties of a flexible body that can have linear elastic deformations. The ROM data must include:

- Coordinates and unit quaternions that specify the positions and orientations of all interface frames relative to a [common reference frame](#). See [Interface Frames](#).
- A symmetric stiffness matrix that describes the elastic properties of the flexible body. See [Stiffness Matrix](#).
- A symmetric mass matrix that describes the inertial properties of the flexible body. See [Mass Matrix](#).

To generate ROM data, you can use the [Flexible Body Model Builder](#) app or finite-element analysis tools, such as the Partial Differential Equation Toolbox™. By using the toolbox, you can start with the CAD geometry of the body, generate a finite-element mesh, apply the Craig-Bampton method, and generate the corresponding ROM data. For more information, see [Model an Excavator Dipper Arm as a Flexible Body](#).

The data provided to the Reduced Order Flexible Solid block must refer to a consistent common reference frame. This reference frame defines the x, y, and z directions that specify the relative position of all points in the body. The frame also defines the directions of the elastic degrees of freedom associated with each interface frame.

The Flexible Body Model Builder app generates the reduced-order model (ROM) data for flexible bodies by using the Craig-Bampton method



General cross section flexible beam requires the definition of

- Geometrical cross section
- Stiffness and inertial properties
 - From geometry
 - Or customised
- Damping properties
- Number of element of discretization
- Modal reduction

Flexible beams in Simscape™ Multibody™ are assumed to be made of a *homogeneous, isotropic, and linearly elastic material*. The beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, are automatically calculated by the block using the material and geometry properties that you specify. To see the computed values, in the beam block dialog box, open **Stiffness and Inertia > Derived Values** and click the **Update** button.

Material properties are required for deriving beam stiffness like:

- Material density (usually in [kg/m³])
- Young's modulus (usually in [GPa])
- Poisson ratio (adimensional)

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to Custom. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers. The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{pmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{pmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the x-axis.
- H_y^b is the centroidal bending stiffness about the y-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\bar{M}^c = \begin{pmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x^c & -I_{xy}^c & 0 \\ 0 & 0 & 0 & -I_{xy}^c & I_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z^c \end{pmatrix}$$

where:

- m^c is the mass per unit length.
- I_x^c is the mass moment of inertia density about the x-axis.
- I_y^c is the mass moment of inertia density about the y-axis.
- I_z^c is the polar mass moment of inertia density.
- I_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{pmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{pmatrix} = \bar{S}^b \begin{pmatrix} \bar{\epsilon}_z \\ \bar{k}_x \\ \bar{k}_y \\ \bar{k}_z \end{pmatrix}$$

where:

- \bar{F}_z is the axial force along the beam.
- \bar{M}_x is the bending moment about the x-axis.
- \bar{M}_y is the bending moment about the y-axis.
- \bar{M}_z is the torsional moment about the z-axis.
- $\bar{\epsilon}_z$ is the axial strain along the beam.
- \bar{k}_x is the bending curvature about the x-axis.
- \bar{k}_y is the bending curvature about the y-axis.
- \bar{k}_z is the torsional twist about the z-axis.

Damping can be calculated with proportional or modal model, first is given by

- Mass coefficient (usually in [1/s])
- Stiffness coefficient (usually in [s])

Modal damping requires definition of

- Adimensional damping ratio

The beam blocks support two damping methods: uniform modal damping and proportional damping. The uniform modal damping method applies identical damping ratios to all the vibration modes of the beam. In the proportional damping method, the damping matrix [C] is a linear combination of the mass matrix [M] and the stiffness matrix [K]:

$$[C] = \alpha [M] + \beta [K]$$

where α and β are scalar coefficients.

Discretization

The Number of Elements parameter in the Discretization section of the beam block dialog box specifies the number of finite elements used to discretize the beam. You can select its value to obtain a good compromise between simulation accuracy, which may require more elements, and simulation speed, which requires fewer elements. Use the fewest elements needed to satisfy your accuracy requirements.

For bending deformations, the beam blocks use the cubic Hermite interpolation method to compute the displacement distributions throughout each element. The distributions of axial displacement and torsional rotation are obtained by linear interpolation method.

When using beam blocks in a model, several factors impact the accuracy and speed of the simulation performance. This section discusses the impact of the three most important factors: flexible beam usage, solver selection, and damping settings.

Even though using flexible beams can increase the accuracy of a multibody simulation, the flexible beams tend to slow it down by increasing the numerical stiffness and the number of degrees of freedom of the system. To speed up the simulation, you should use a rigid body whenever the deformation of the body is negligible. Moreover, the *Number of Elements* parameter in the *Discretization* section heavily impacts the performance of the simulation. For more information, see the section.

The solver is critical to the performance of a multibody simulation. The stiff solvers, such as `ode15s`, `ode23t`, or [daessc](#) (and [best practices](#)), tend to work better for systems with flexible beams due to the stiff nature of these systems. Additionally, solver tolerances and maximum order also impact the accuracy and speed of the simulation. For more information, see [Choose a Solver](#).

Note: All the solvers, except `ode23t`, provide some level of numerical dissipation, which can be helpful for modeling flexible multibody systems.

When modeling a flexible beam with little or no damping, undesirable high-frequency modes in the response can slow down the simulation if the solver does not already provide adequate numerical dissipation. In that case, adding a small amount of damping can improve the speed of the simulation without significantly affecting the accuracy of the model.

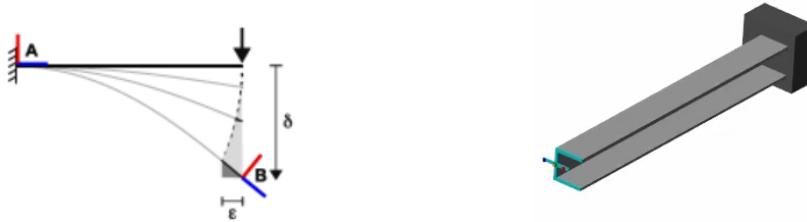
Flexible beams in Simscape™ Multibody™ respond to gravity, but only that specified in the [Mechanism Configuration](#) block. The force due to a [Gravitational Field](#) block is ignored. If the frame network of which the flexible beam block is a part contains a Gravitational Field block, the body behaves as though in zero gravity. Using flexible body and Gravitational Field blocks in the same frame network causes **Diagnostic Viewer** to issue a compilation warning. Modeling gravity with both the Mechanism Configuration and Gravitational Field blocks results in a compilation error.

The use of flexible elements can then be adopted wherever flexibility effects are considered relevant for application. Model shall assume slender bodies with constant cross-sections that can have small and linear elastic deformations. These deformations include extension, bending, and torsion.

The geometry of a beam is an extrusion of its cross-section. The general cross-sections, with or without holes, are supported by the [General Flexible Beam](#) block. Additionally, the beam cross-section can take many standard shapes, such as channel, angle, and hollow cylindrical. For beams with standard cross-sectional shapes, use the library flexible beam blocks.

Each beam has two connection frames labeled **A** and **B**. Each connection frame has a frame port on the block that can connect to another block. The connection frames are located at the ends of the beam and fall on the z-axis of the local reference frame labeled **R**. The reference frame serves merely as an internal reference for the beam and has no frame port.

In Simscape™ Multibody™, all the flexible beams can have elastic bending, axial, and torsional deformations. The beams are assumed to be slender bodies whose length must far exceed its overall cross-sectional dimensions, and all the deformations should be linear and small.



The bending and axial deformations of a beam follow classical (Euler-Bernoulli) beam theory. The bending can be about any axis in the cross-sectional plane (xy -plane) of the beam. Cross-sectional slices are assumed to be rigid in-plane, to stay planar during deformation, and to always be perpendicular to the deformed neutral axis of the beam. The twisting of a beam derives from Saint-Venant torsion theory, and the cross-sectional slices are rigid in-plane but free to warp out-of-plane.

When one or more of these assumptions are not met, the result may be inaccurate. For example, in the figure, a cantilevered beam that is subjected to a transverse point load will get an inaccurate result when the bending deformation, δ , is large. During the bending, the free end of the beam moves downward perpendicularly instead of following the true physical path, which is indicated by the dotted trajectory. The discrepancy, ϵ , increases as the δ increases.

Conversion between models

Conversion from simscape to rigid body tree

Simscape™ is not a tool purposed to robotic design, but it is rather easy to implement some user specific blocks for robotics modeling purpose. On the other side the integration with other robotic modeling tools offered in Matlab® seems not to be optimized. First in Simscape™ we can define an arbitrary mechanism, even a closed chain or a multiphysics model, where rigid

body trees represent only a subset of possible models. Thus in general is not possible a direct conversion from a Simscape model (even if just Multibody) and a rigid body tree model.

Code generation

In Simscape™ one can also provide automatically the deployment of a C code for given model, using predefined App. Simulink® -Coder™ software can be used to generate standalone C or C++ code from your Physical Networks models and enhance simulation *speed* and *portability*. Certain features of Simulink® software also make use of generated or external code. Code versions of Simscape models typically require *fixed-step Simulink® solvers*, discussed in the Simulink documentation. Some features of Simscape™ software are restricted when you translate a model into code. Code generated from Simscape models is intended for rapid prototyping and *hardware-in-the-loop* applications. It *is not intended* for use as production code in *embedded controller* applications. Add-on products based on the Simscape™ platform also support code generation, with some variations and exceptions. In this work we didn't used it, but the feature is of great interest in sight of more applicative works. Notably code generation output is directly a program file, and no way to extract directly from Simscape™ (at least part of) analytical model or pseudo analytical model is given. We've reputed this lack a serious obstacle in a correct workflow management, since relying on an externally and independently derived mathematical model could be origin of errors and misalignments, we'd repute better to avoid. Of course this is partially motivated by commercial considerations, but is considered a potential lack also from customer/user value standpoint.

Simulink/Simscape based identification without analytical models

All methods of identification seen in the topic referring section are based on the assumption of an analytical model or at least a numerical quantification of the observation matrix. Unfortunately when developing a Simscape™ model we do not directly have either an explicit analytical model representation we can manipulate in order to get desired quantities nor a direct numerical evaluation of the regressor matrix associated with our identification problem.

To be honest it would be possible to carry out a sort of numerical regressor in case there's a chance to manipulate model parameters and perform a set of simulations setting zero for all parameters (in case of linear model used) but one. Output would be, as in case of naive regressor, one column of desired matrix, at each operative point condition of a given trajectory. Matters with this approach are numerical computation burden and the rapidly increasing number of simulations to carry out (remember for inertial parameters we start with ten parameters by link) and even more relevant, risk for singularities using null parameter values. In addition we need to evaluate separately base parameters later, for instance using QR or SVD decomposition of regressor matrix. Finally, but not least important, the method relies on model linearity assumptions over parameters as fundamental hypotheses. This way was attempted but not concluded because of lack of time.

Otherwise we fall under the case of modeless identification, where data-driven and machine learning methods apply. Also this path has been avoided because of lack of time required for its development.

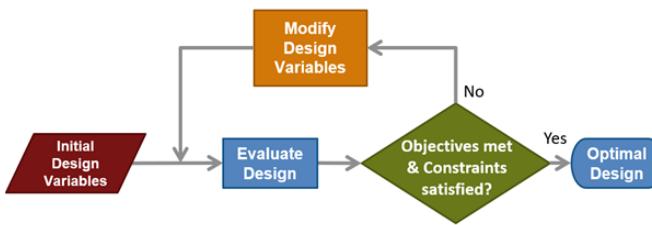
Identification using

We've seen in other sections that in case of linearity in parameters of the model for a vector of observable quantities, under the assumption it is possible to carry out experiments or simulations (that's the case of digital twins) setting parameters values to zero selectively, then a purely numerical version of observability matrix can be got out. Running multiple simulations or experiments where all but one parameter is zero, each observation vector corresponds to a column of regressor (eventually scaled by the value of parameter). Then running a simulation for each parameter (of relevance in case some are already known to be negligible), one can get regressor columns for samples extracting and stacking observations vectors. Since could happen observations are linearly dependent, it is suggested to perform a post stacking QR decomposition and define an estimated parameters vector with corresponding factorisation. Main constraints in using this approach are: need to run multiple simulations (for inertial parameters we need up to $10N$ parameters). Each simulation shall refer to the same conditions to be meaningful, and meanwhile exciting properly the parameters, namely output corresponding shall be not negligible or errors shaded. Each extracted observation shall correspond to the same simulation samples. Requires the capability to set parameters properly, that's not always possible nor easy, especially in experimental cases. On the other side, multiple simulations or experiments, that could become prohibitive or not possible. On the other side, numerical components are more difficult to analyze and understand in case of problem solving needs.

The method is of general applicability and avoids the need of analytical models. Essentially this way was not adopted in this work because less transparent, leaving its exploration for future researches.

Parameter identification using Simulink design optimization app

Another way exists that wins for flexibility, and ease of use. Dropping (potentially) model linearity hypothesis on parameters we just define a reference *expected systems response* over time, given by *experimental data* or *by simulation* or imposing it *a priori*. We then use a specific *optimization tool* (we used *Parameters Estimation of Simulink® Design Optimization™* toolbox) in order to find a best fitting parameters vector minimizing residuals of system output with respect chosen reference. Basic principle of work are found in [Design optimization](#), and depicted in figure.



Notably this approach can be used for identification problems as well for design optimization, response optimization, optimal control, sensitivity analysis and many other applications. Potentially any problem that can be expressed as an optimization problem can exploit this tool.

$$\hat{\chi} = \arg \left(\min_{\chi} (\|S_{ref} - S(\chi, \dots)\|) \right)$$

In addition *no constraints on parameters type* are required, since optimization can occur on continuous as well as discrete parameters space. Search range can also be constrained at pleasure, improving solution physical *consistency*.

There's no need to define any analytical model, nor perform multiple simulations, although the process implicitly performs a set of runs on simulink target to derive resulting residuals and define parameters vector correction to apply. Unfortunately dropping linearity assumptions exposed to risk of non-convergence and to local minima convergence, consequently erroneous estimation are possible. On the other hand the tool does not allow easy detection of linear model dependence unless using *sensitivity analysis*, that in presence of many parameters results to be hard and time taking to carry out. The other issue is no method is provided for the detection of group-identifiable parameters, nor for their identification, since it would require design modifications. On the other hand, knowing *a priori* our model depends linearly on some parameters should assure *fast and global convergence*, unless parameters

estimated belong to *non-identifiable* or *grouped parameters set*. As first suggestion, all parameters belonging to the last two mentioned categories should be left out.

The fundamental advantage of this approach is total *independence* from *analytical models* and relying on design environment resources only, and avoidance of assumptions on model nature and parameters dependance. On the other side we *lose* the possibility to *study* parameters *identifiability* in a *complete way*, or exploit analytical models for getting further inferences. Furthermore, this approach of most generality and allow to identify (theoretically and potentially) any parameter, even geometrical or any other kind, provided affecting system output, even if non linearly related or discrete. Quality of results may be lower when *many* parameters are estimated together, because of the numerical nature of the method and errors could spread over parameters vectors. Use of nonlinear optimization is more prone and sensitive to numerical errors and *modeling coupling* than linear methods. Last but not least this approach is easy to automatize and can be applied to any arbitrary model.

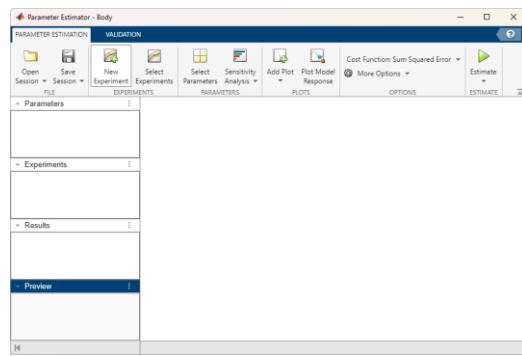
What is worthy of underlining is the lack of need of having a model of the system in the classic analytical sense. Effectively what we do is replace the *analytical model* with another one, using a different “language” but effectively equivalent, the one of a *numerical scheme*, that’s also not transparent to the user, that can treat it as *black or gray box*. This at first could be cumbersome, since the way to manage the two is rather different, although equivalent, and a translation key lacks. Passage from one model to another one is neither straightforward nor transparent, but this is the price to pay for easing modeling and identification and optimization problems, and reducing effort required, abandoning traditional methods, leading to more streamlined and automated workflow. From an applicative perspective the difference matters and is substantial. Indeed as remarked in other sections, dealing with many degrees of freedom systems lead to analytical models explosion quickly. There numerical models manifest *superior performances* with pretty linear cost over degree of freedom, making complex models accessible from a practical point of view.

We're not stating analytical methods are to forget and dump, since they're useful in many ways, but dealing with two languages, or worlds whose translation between is necessarily imperfect.

Process description

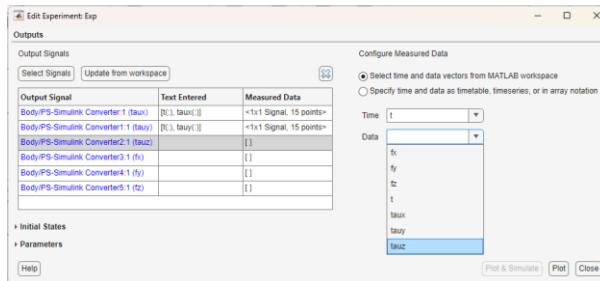
The processes considered make use of [Simulink parameters estimation app](#) (see references for details). A short preliminary description with some sample test case are provided hereafter. App menu presents like figure:



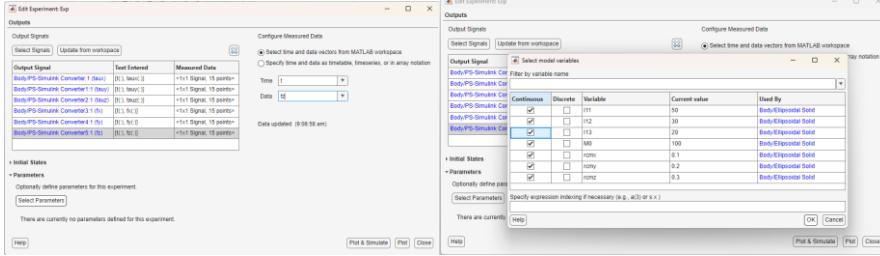


A new experiment shall be created with relative button. Process is largely automated and provide automatic identification of signals from workspace. Just selecting output signals (setup can also be automated using command line or scripts, generated automatically from the app itself or by a custom script). One simple way to make automatically tracking output signals candidates is using data logging on signals.

Then for each signal a mapping shall be established between selected output, from workspace, and *expected response*. Time vector is also required. Remarks that in case you need you can also use a predefined input and give it as input to your model, like in case of playback of an experimental trajectory.



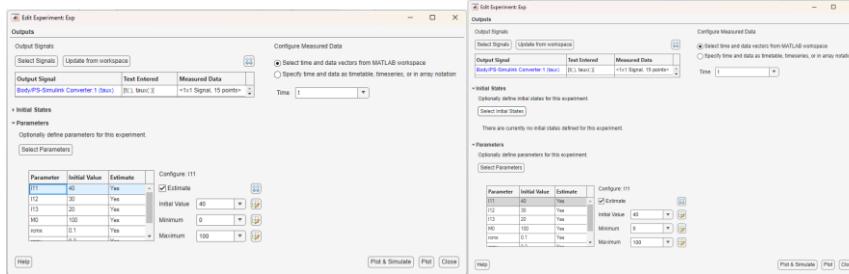
Then you can choose among *model predefined parameters* (essentially variables used to parametrize your model taken from workspace, you have to provide by design) which one to use as parameter for optimization (or estimate/identification in this case), or in case which to use as parameter for your session. You can design multiple experiments and save relative sessions when there's need for attempting different setups.



Remark you can select *any kind of parameter* (inertial, geometrical, or any other) but you need to define its type (*continuous* or *discrete*, options are *mutually exclusive*) in order to perform a proper estimate, as seen in the identification chapter. Once defined desired target estimation parameters we need to give them:

- *Initial value* (default current value), usually a prior estimate is suggested to speed up estimation process
- *Minimum value* (default -inf),
- *Maximum value* (default inf)

search range lower/upper bound helps in assuring consistency and reduce process time and workload. Is warmly suggested to use physically meaningful limits to them to reduce potential erroneous search in forbidden parameters space area, keeping model consistency at least partially, where possible. For instance, mass and principal inertia shall be positive, and their limit values have lower and upper bounds can be imposed a priori from overall total system known or realistic values, even if approximated. In case more complex constraints apply on parameters you can set them up only programmatically. Luckily there's way also to provide these constraints as expression involving multiple parameters, in linear form or non linear form, really be the case for some systems like for imposing distance constraints, convexity conditions, mass-inertia-geometry congruence conditions and many others, involving systems of complex relations between estimated parameters, as usually happens in a constrained optimization problem. See references [Estimate model parameters with parameter constraints code](#) for details.



Remark is also possible, in case your model expects some, to introduce some *initial condition* as estimated target parameters, or as a priori knowledge. That means you can estimate even only initial states, or a mix of parameters and initial conditions. That option may be useful in some cases. Once the experiment is set up, you can run an estimate.

Simulink® optimization toolbox (as well as parameter estimator toolbox) provide also some useful customizations for setting up your optimization problem:

- Choice of objective or cost function among
 - Squared error sum (SSE)
 - Absolute value errors sum (SAE)
 - Raw error
 - Custom cost function
- Choice of optimization method between
 - Lsqnonlin: non linear least squares
 - Gradient search (using fmincon)
 - Simple search (using fminsearch)
 - Pattern search (using patternsearch)

$$F(x) = \sum_{t=0}^{t_N} e(t) \times e(t) \quad F(x) = \sum_{t=0}^{t_N} \begin{vmatrix} e(t) \end{vmatrix} \quad F(x) = \begin{bmatrix} e(0) \\ \vdots \\ e(N) \end{bmatrix}$$

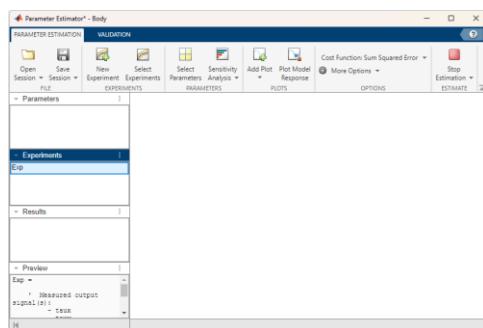
N is the number of samples. N is the number of samples. N is the number of samples.

We need to distinguish between

- Minimization problems
- Feasibility problems
- Mixed feasibility and minimization problems

See [Optimization problem formulation for parameter estimation](#) for a deepening on the subject.

Optimization process steps and iterations are monitored and logged with residuals value, as well as parameters trajectory.

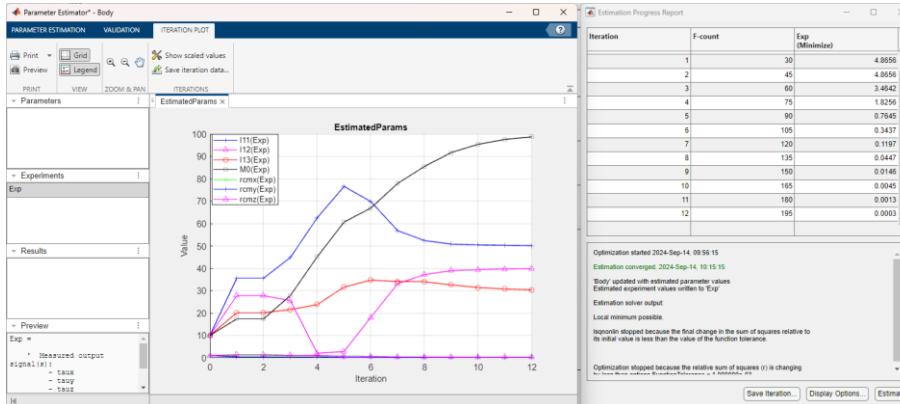


Estimation process, especially for complex models is iterative and time taking, so less performing respects model based and linear estimates, but this is the price to pay for a procedure that's essentially analytical independent. The tool allows a variety of user-friendly

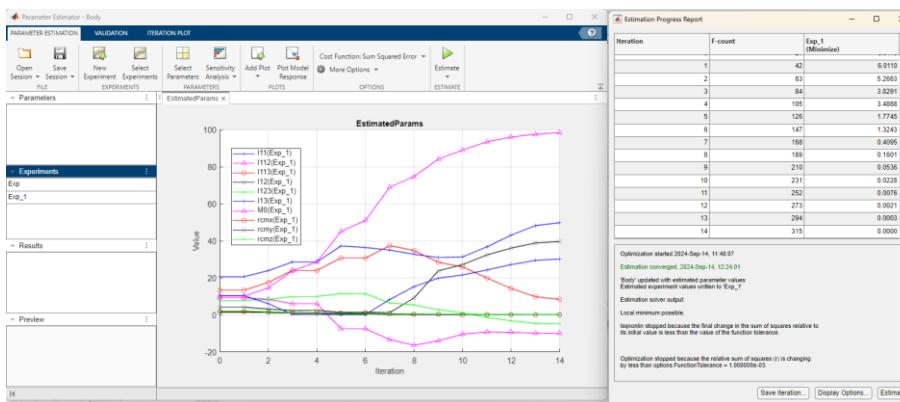
model and parameters tracking management utilities, like saving session, multiple experiment, boundary parameters management, *model response verification*, making it extremely simple and useful, accessible even to people with a minimal knowledge of optimization and identification problems.

Some applicative examples

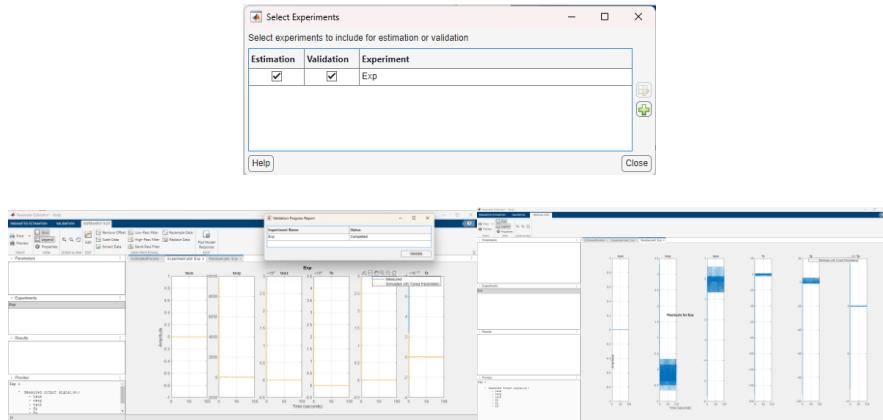
Even a relatively *simple rigid body model* could take twenty minutes to converge to the final solution, and convergence of trajectory is not necessarily that smooth, following oscillating trajectories potentially, where they exist, converging to local minima. By the way there are good chances to get a suitable solution (residual metric allows judging fitting goodness).



Not only principal but also the product of inertia can be estimated with the same method.



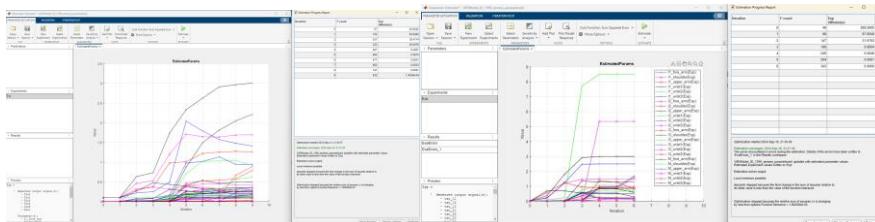
Once a solution is carried out, one can validate it by selecting experiment and launching a model validation, checking compatibility of system response with expected output and residuals.



Coping with more complex mechanism could pose serious challenges because of:

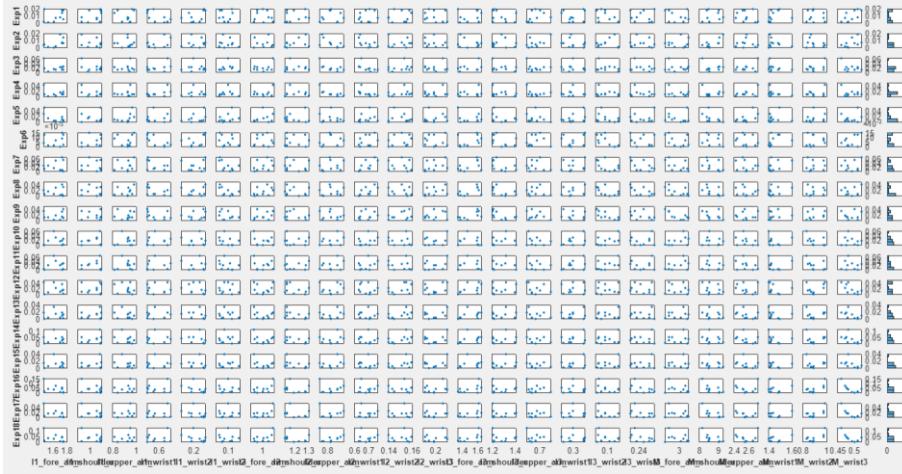
- Parameters unobservability and direct/indirect identifiability
- Parameter grouping effects
- Very long time for convergence (up to hours for just some degree of freedom)
- Parameter correctness and sensitivity

As depicted below parameters convergence is more tricky and difficult and problems of parameters coupling will arise easily as well as error spreading. On those cases we can say traditional estimation techniques results still more effective, efficient and reliable.



Sensitivity analysis

In order to dodge issues of identifiability it is warmly suggested to perform prior to facing the estimation problem, a parameters sensitivity analysis. Picture below illustrates how complex (and long) sensitivity analysis can become when dealing with many parameters. For robot identification the number of standard parameters involved is in the order of $10N$, where N is the number of degrees of freedom.



Avoidance of local minima

Avoidance of local minima convergence problem can be avoided using alternative methods of optimization, maybe slower, but leading to more reliable solutions.

Speed improvement

Since time for estimation becomes a primary issue, when using nonlinear and pattern search methods, improvements are possible, and should be considered, using:

- *Parallelization of computing* options
- *Fast restart* options
- *Model acceleration*

All these options are available under the tool configuration.

Anyway the advantage of using a linear model just taking few seconds to find a solution against numerical models is remarkable. The balance is clear, the more complex and long a model is to run, the more is time required to evaluate each parameter increment leading to an explosion estimation time, increasing convenience of pre evaluated targets, that do not require strictly an analytical model, but just a qualitative understanding of parameters effect on model output.

Sensor fusion and parameters selection

A point for this approach is the ease it allows to perform a perfectly natural sensor fusion using complex models, without going into analytical details of signals models of measured data. Since signals used in the model are arbitrary, one can use whatever one preferred for its sake, and at same time change selected one with extreme ease, as well as one can do in selecting parameters. Keeping the same model, applying minor changes, is possible to afford different use cases and application sessions, in a rather straightforward, flexible, and fast way, reducing error proneness and avoiding all additional workload required to adjust model parts usually

adopted for representing system associated phenomena. Anyway offline preference of the tools make it not adapt for online sensor fusion that's mostly the applicative case for.

Convergence and linearity

As seen above, model convergence and solution correctness are closely related to problem linearity. It can be shown that pure linear problem estimation adopting the same optimization method (default is lsqnonlin) leads to faster solution convergence (in some iterations) and the final solution is likely to be the correct and unique one. Adding non linearities (like coupling with geometrical parameters) lead to longer optimization sequence convergence and potential undesirable wrong local minima convergence. Consequently, wherever possible is suggested to adopt a linear estimation problem, or anyway evaluate carefully parameters set used. Also sequential and iterative multi step method can be adopted for mitigating risks.

Final considerations

In conclusion, although this approach presents different advantages and is still promising and interesting for applicative purposes, further studies are deserved and required for a more stringent evaluation of pros and cons, nonetheless of best practice to adopt for its use. Supporting the study with a priori considerations we believe this tool could effectively provide a suitable alternative method for model estimation and optimisation, in spite of present inferior performances when used at scratch, not only in terms of speed, but also in terms of convergence correctness.

Reinforcement learning

Use of reinforcement learning in robotics is knowing increasing interest especially for control applications where in place of traditional control laws given by explicit laws (like PID controllers, LQR and other model based optima controls), a general black box control is developed by training system over a huge set of *experiments* where a target value function is maximized (reward) or minimized (penalty). In spite of being a deterministic (but can be also of statistical nature) law, usually its internal structure is not fully explicit neither easy to write down. In spite of the limited *explainability* of control mechanism underlying to the algorithm, reinforcement learning has demonstrate unparalleled achievements in complex tasks like walking, passive dynamics, through learning autonomously based on a given environment.

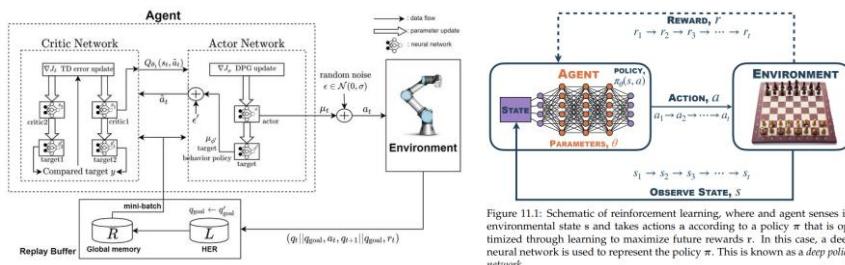


Figure 11.1: Schematic of reinforcement learning, where agent senses its environmental state s and takes actions according to a policy π that is optimized through learning to maximize future rewards r . In this case, a deep neural network is used to represent the policy π . This is known as a *deep policy network*.

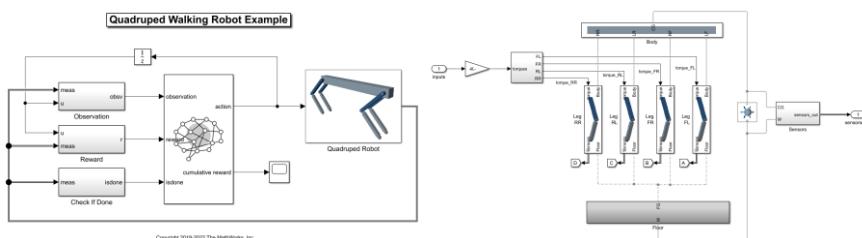
How does reinforcement learning works using simscape

Reinforcement learning using matlab is rather simple, respect using custom solutions. Matlab®, as for many of its toolboxes, provides a specific app for facilitating users in using their tools, though in later phases command based scripts are more efficient, they're a useful guideline for learning basic commands. For more details see [Reinforcement learning toolbox](#). You need just to define your model and introduce your *agent* block and define which is your *environment*. Environment in our specific case is the system we're going to interact with, in order to get some desired behavior by means of a *set of actions* determined from the agent based on a set of predefined *observations*, selected among those which are mostly rewarding. User then have just to define:

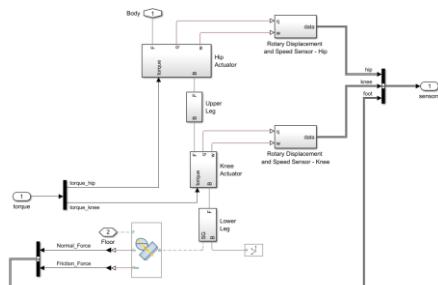
- Environment or target system (could be of any kind, from a matlab script, to a simulink model, or even a simscape model)
- Reward (or penalty) function fundamental for defining final environment response
- Policy (crucial to define training course and response)
- Agent structure and type (actions set and observations set)

Training is performed in order to find best reward actions against system observations of system response. The selection of the right reward function drives the system to select the best course of action. There are strong connections of reinforcement learning with optima control theory, since both rely on the optimization of a cost or reward function. Indeed reinforcement learning is often used for control purposes, where the agent can be regarded as *controller*.

Consequently reward function design is very system and *purpose (task) specific*, but overall architecture is common and does not change much from one problem to another. Is it then possible to define a general architecture of the model and then just select most proper observations and policy blocks for the environment (our target systems) and reward function. The overall architecture in addition requires you to provide a set of *isdone* condition signal for stopping episode evaluation. An example of generic architecture is given below:



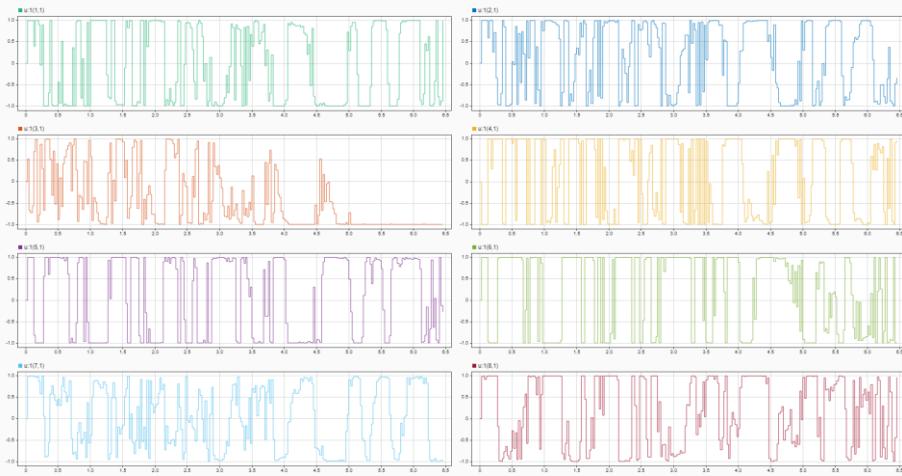
Internal Simscape™ (environment) model, providing dynamic evolution of the system is composed of four legs controlled by *actuation*, and a *sensor* block. Each leg interact with main body, the floor and with actuation motors. The model include contact forces modeling with floor, is then a rather complex case of dynamic model, whose analytical replica would not be straightforward. Model is represented in the following:



Sample simulations output of a trained walk

Main drawbacks of reinforcement learning are the long simulation time required and the computational effort, often prioritizing the use of *parallel computing* over single machine runs, where available. Single training session can last even hours, and convergence is not always assured. For these motivations simulations study performed where limited to few basic case, in spite of strong interest in further case explorations. Trained results are not much intuitive, and not following a precise deterministic law, because of their statistical nature, have a pretty randomic appearance, in spite of existing underlying pattern arising from quasi-periodic action emerging from walking dynamics. It's not a rigorously predetermined motion but a consequence of optimality criteria sought, leading to motion coordination, generating order

from chaos in a counterintuitive and unpredictable way. Consequently, outputs are not self-explicative in appearance and only with careful processing one can extract relevant information from. Since this is not the core topic of this work, we propose to carry out this research as part of future works. Conversely we note, adopted policy, in place of control law, is surprisingly physically meaning and intuitive, leading to a control design in some sense revolutionary in comparison with traditional control theory.



Reward function construction

The reward function is the core of reinforcement learning, since it defines the attitude of solution toward defined properties. Note, differently from the traditional approach, these properties are not given as strict required constraints mandatorily satisfied but as fuzzy condition of preference, can be violated at any episode in some measure, with a penalty associated, but not forbidden. That means provided all other rewards are higher, deviations are accepted, and relative state space explored. Some analogy exist between penalty function with classical mechanics minimum action principle, it defines the base function that drive system dynamics, similarly to kinetic energy and potential functions in variational theory. Building up a reward function conceptually and qualitatively simple, but providing a working one may be much harder, especially because there's a need to set suitable values for coefficients compatible with system dynamics, and providing conditions allowing for a training short and feasible enough for practical purposes. Indeed training is in the end one of the most challenging steps of a reinforcement learning problem resolution.

As an example we can have a look at the policy commonly used to define a walking motion. We give a set of reward (positive contribute) for episodes that

- Keeping motion *forward* (using speed)
- Episodes lasting longer (*duration* reward)

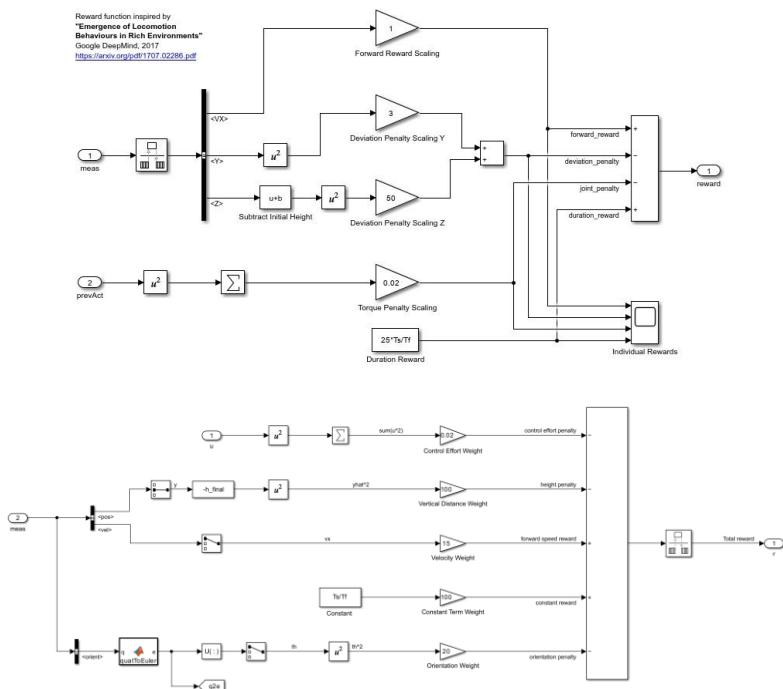
A penalty (negative contribute or cost) is associated to

- Applied control actions (the more *actuation*, the lower the reward)

- Deviations of main body center of mass respect initial *height* (up and down)
- Deviations of main body orientation, from initial *direction*, aligned with proceeding desired direction

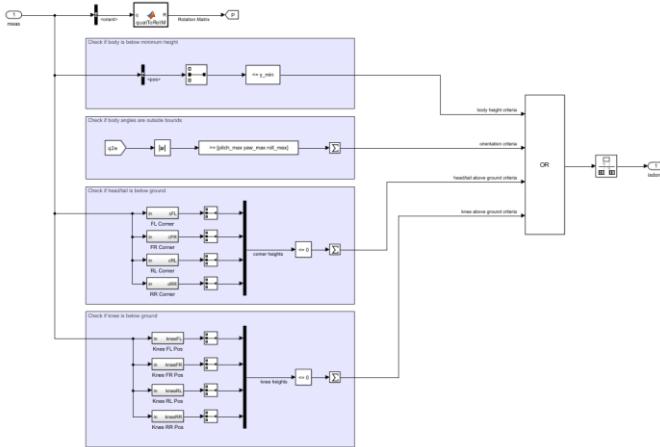
Total cumulative reward of every episode is calculated and used to decide the best episode of training.

As depicted in the figure the reward function is rather simple, but the implications for training are subtle and not always evident or simple.



IsDone condition

In reinforcement deep learning there's need to identify a simulation *termination condition*, defining events giving a conclusion to the observation batch and consequently to the final reward associated with the combination of actions and states considered. Usually we define it by means of a set of conditions reputed to be *unrecoverable* and/or *unacceptable*, giving an end to significant conditions for our system. Some examples are: *excessive deviation* from desired course, *falling down* (given as center of mass height), *excessive control effort demand*, *exceeding body orientation angles*, as depicted in the diagram sketched in figure below. Any (OR condition) of these conditions is considered enough to stop episode evaluation. Without training would last indefinitely or spend too much time on not promising at all configurations and states.



Extending RL applications

We've seen some very simple applications of RL with deep neural networks, able to achieve some traditionally hard objectives like walk control by means of simple rewarding functions. Simplicity of the resulting task (hiding a significant complexity in truth) is not limitative but can be widely extended. For instance just introducing a long term reference control as input to our reward function we can imagine to control our walkaround trajectory as well as any other motion characteristic we want to include into our reward function, once we find a suitable trained solution. In this way we can build up a sort of control law, without specifying expressly all terms of a control law (now replaced by a black or gray box model), but just stating which are desired properties of our resulting motion shall preferably satisfy.

With a little imagination and practical sense one can develop different policies for performing specific tasks, defining criteria making one action set better than others. The potential of this branch of research is considerable, but resources are demanding.

Why reinforcement learning?

Why use a control or problem resolution strategy so complex and time taking as reinforcement learning? It is time consuming, sometimes counterintuitive and unexplainable, so its interpretation is hard, and may be difficult to replicate or generalize to different systems.

In spite of all these drawbacks there are different advantages characterizing reinforcement learning.

- Exploiting emerging behavior
- Ease of managing underactuated and natural motion

RL allows us to see how some complex and highly coordinated movements like walking and other not naturally stable dynamics are surprisingly naturally emerging from trial and error approaches applied to systems natural dynamics, resulting in motions that are energy saving, low actuation requiring, and highly coordinated. Mimic such motions by the use of

mathematical control laws would be extremely difficult as well as keeping coordination between limbs.

Observations

Conventionally observations are separated into multiple channels, each of which carries a group of single elements all belonging to either a numeric (infinite and *continuous*) set or to a finite (*discrete*) set. Each group can be organized according to any number of dimensions (for example a vector or a matrix). Note that only one channel is allowed for the action, while the reward must be a numeric scalar.

Agents

Agents are classified in base to their action (*continuous* or *discrete* or a mix of) and basing on their criteria for learning. Discrete or continuous is the nature of *action space*. Agents are then classified basing on types:

- Policy based agents:
- On policy agents attempt to evaluate or improve used policy
- Off policy agents attempt to evaluate or improve policy that can be different from one under use or used to generate data

Distinction between action space is relevant since not all algorithms are able to work on all types of spaces or both. Only SAC works on hybrid spaces and SARSA, Q-learning agent, and DQN works only on discrete spaces, where DDPG and TD3 works only on continuous spaces.

Agent and Learning algorithms

An agent is something that is autonomous in the sense that it can observe the environment and then, on its own, choose how to act based on those observations. An agent could be something that physically moves, like a robot or a vehicle, or it could be something that doesn't, like a controller in a larger software or logistical system. As long as the entity has the ability to autonomously react to observations, and its actions change the state of the environment, then it's an agent.

The core of reinforcement learning is the learning algorithm, working on one or more parameterized function approximator that learn the policy:

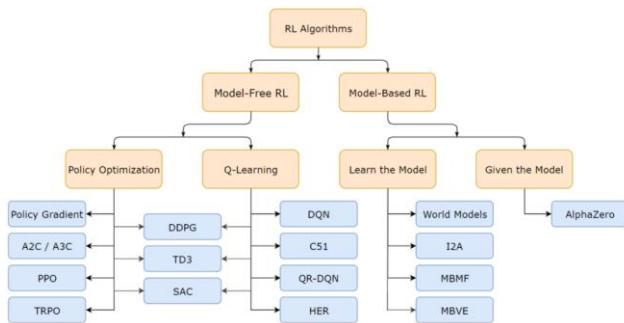
- **Critic:** for a given couple of observations and actions gives an approximation of policy value corrected by a discount factor
- **Actor** giving for a given observation action maximising policy value

Agents that use only critics to select their actions rely on an *indirect policy representation*. These agents are also referred to as *value-based*, and they use an approximator to represent a value function (value as a function of the observation) or Q-value function (value as a function of observation and action). In general, these agents work better with discrete action spaces but can become computationally expensive for continuous action spaces.

Agents that use only actors to select their actions rely on a *direct policy representation*. These agents are also referred to as *policy-based*. The policy can be either deterministic or

stochastic. In general, these agents are simpler and can handle continuous action spaces, though the training algorithm can be sensitive to noisy measurement and can converge on local minima.

Agents that use both an actor and a critic are referred to as *actor-critic* agents. In these agents, during training, the actor learns the best action to take using feedback from the critic (instead of using the reward directly). At the same time, the critic learns the value function from the rewards so that it can properly criticise the actor. In general, these agents can handle both discrete and continuous action spaces.



Policy

The agent contains two components: a policy and a learning algorithm.

The policy is a mapping from the current environment observation to a probability distribution of the actions to be taken. Within an agent, the policy is implemented by a function approximator with tunable parameters and a specific approximation model, such as a deep neural network.

$$\pi(s, a) = \Pr(a = a | s = s)$$

$$\pi(s, a)$$

The policy is essentially analogous to the resulting output of a traditional control law, or of a decision making process.

Reward function give us a metric of goodness of policy application results.

A first distinction shall be made between discrete action and continuous action.

For real physical systems continuous action is most suitable because they're characterized mostly from continuous states.

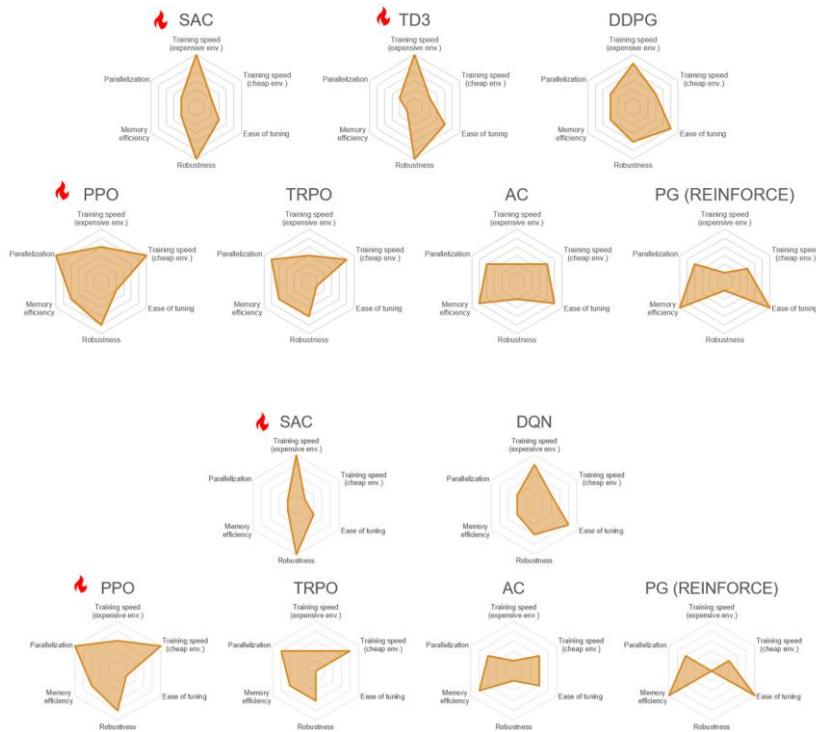
Policy based methods are a branch of used RL training algorithms , although other exists, is one of most widely applied and successful.

Different policies exist:

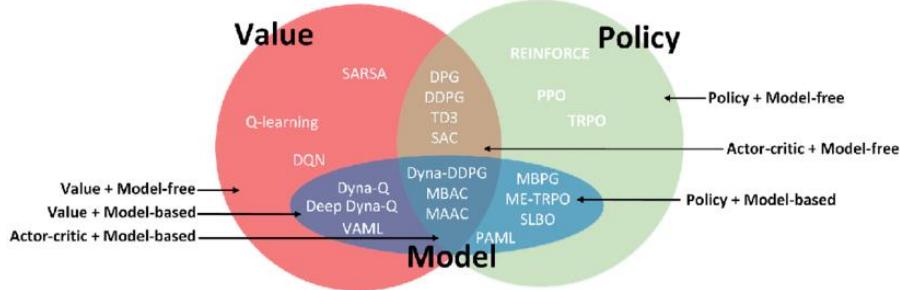
- Policy Gradient

In following picture we depict some of most relevant metrics for reinforcement learning algorithms

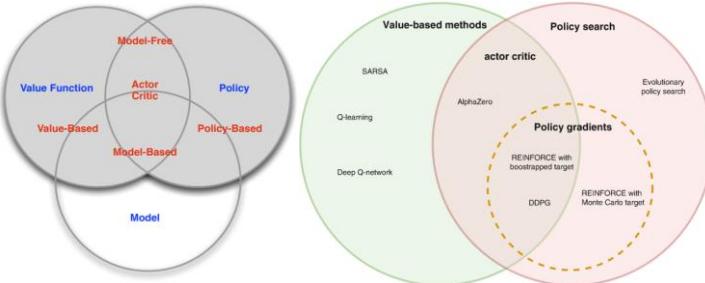
- Training speed
- Parallelization
- Robustness
- Memory efficiency
- Ease of tuning



Hereafter we want to present a synthetic view of key concepts just presented and how different agents types are involved:

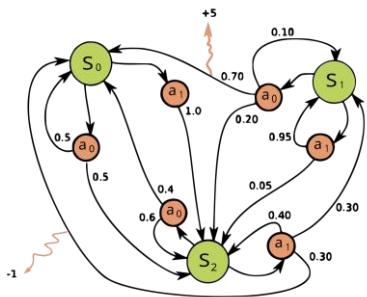


Nature of robotics reinforcement learning problems based on *continuous state and action space*, and in most of case based on a value function optimization, lead to preference for agents and training algorithms like DPG, DDPG, TD3 and SAC as seen previously, beyond performances and training burden considerations. Nevertheless the existence of a model could provide some chances for alternative training. In following pictures we present a couple of more clarifying representation of set partitioning into different typology classes.



Policy as Markov Decision Process (MDP)

Nature of policy is either deterministic or probabilistic, in any case it defined the map between current observed state and action to next action to take. From this point of view it can be studied as a Markov Decision Process ([Markov decision process - Wikipedia](#)), intended as a stochastic dynamic programming or stochastic control problem, modelling a sequential action or decision making process whose outcomes are uncertain. Involved states could be discrete or continuous, as well as time. There's also an interesting connection with stochastic game theory.



Reinforcement learning and identification

From above considerations is clear we can also use reinforcement learning for inverse problems like identification of system parameters by setting reward and penalties such that promotion of parameters sets (even time changing!) allow to better match environment output respect expected one.

Robot Learning

Robot learning from humans relates to situations in which the robot learns from interacting with a human. This must be contrasted to the vast body of work on robot learning where the robot learns on its own, that is, through trial and error and without external guidance. In this chapter, we cover works that combine reinforcement learning (RL) with techniques that use human guidance, e.g., to bootstrap the search in RL. However, we exclude from this survey all works that use purely reinforcement learning, even though one could argue that providing a reward is one form of human guidance. We consider that providing a reward function is akin to providing an objective function and hence refer the reader to the companion chapter on Machine Learning for robotics. We also exclude works where the robot learns implicitly from being in presence of a human, while the human is not actively coaching the robot, as these works are covered in the companion chapter on Social Robotics. We hence focus our survey to all works where the human is actively teaching the robot, by providing demonstrations of how to perform the task. Various terminologies have been used to refer to this body of work. These include *programming by demonstration* (PbD), *learning from human demonstration* (LfD), imitation learning, and apprenticeship learning. All of these refer to a general paradigm for enabling robots to autonomously perform new tasks from observing and learning, therefore, from the observation of humans performing these tasks.

The main principle of robot learning from demonstration is that end-users can teach robots new tasks without programming.

Imitation learning

Teaching force control-task

While most LfD to date work focused on learning the kinematics of motions by recording the position of the end-effector and/or the position of the robot's joints, more recently, some works have investigated transmission of force-based signals through human demonstration [74.56, 73–76].

Inverse Reinforcement Learning (IRL)

While most of the works that combine imitation learning with reinforcement learning assume the reward to be known, inverse reinforcement learning (IRL) offers a framework to determine automatically the reward and the optimal control policy. When using human demonstrations to guide learning, IRL solves jointly the what to imitate and how to imitate problems. Other approaches to estimating the reward or cost function automatically have been proposed, see, for instance, the maximum margin planning technique [74.117] and the automatic extraction of constraints [74.118]. Underlying all IRL works is the assumption of a consistent reward function. When demonstrations are provided by multiple experts, this assumes that all experts optimise the same objectives. This is constraining and does not exploit the variability of ways in which humans may solve the same task. Recent IRL works consider multiple experts and identify multiple different reward functions [74.119, 120]. This allows the robots to learn multiple (albeit suboptimal) ways to perform the same task. The hope is that this multiplicity of policies will make the controller more robust, offering alternative ways to complete the task, when the context no longer allows the robot to perform the task in the optimal way.

Robustness

Safety

The use of not explainable or partially explainable policies and agents lead to unsafe conditions in a pretty obvious way. Not being fully explainable there's no way to predict in any conditions agent action neither systems or environment reaction to them, potentially leading to unexpected states even dangerous for the whole or part of the system. Even in traditional control is not possible to avoid such kind of situation completely, but the adoption of a black box model makes it even more challenging and dangerous, posing serious threats at safety level fr system and for whatever interacts with.

One possible mitigation action is the adoption of monitors and blocking systems outside agent-environment basic scheme in order to avoid occurrence of them with deterministic and well known behavior elements.

Architectural mitigation

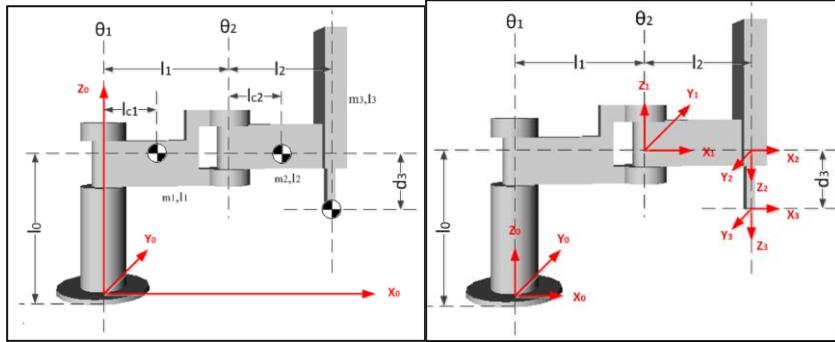
An alternative way to limit potentially hazardous behaviors of a RL system it to modify agent architecture in order to make it less unpredictable and more explainable. One way to achieve

this is by means of traditional deterministic control block as means for acting on environment, driven by a black box RL agent as input. In this way effects on final environment is moderated from traditional control deterministic and explainable action.

For deepening on the subject we redirect to literature: [Heess et al. 17], [Reinforcement learning with matlab and Simulink](#),

Case study I: SCARA robot modeling and identification

As proof of effectiveness of methods illustrated in previous pages, we applied Simscape™ modeling and analysis methods to a simple example of SCARA robot.



In place of using automatic conversion of existing URDF model file we preferred to use this case as example of manual Simscape modeling and design.

Resulting Simscape model is rather simple

[SCARA robot DH parameters table](#)

SCARA kinematic model

SCARA kinematic model for end effector is an example among simplest and easy to manage and check. Thence it's a good case study for application of Lie group and twist formalism (see [Murray] fr details and [Y.Xu-R. Liu]), introducing twist vectors

$$\begin{aligned}\xi_1 &= (0 \quad 0 \quad 0 \quad 0 \quad 1)^T \\ \xi_2 &= (l_1 \quad 0 \quad 0 \quad 0 \quad 1)^T \\ \xi_3 &= (l_1 + l_2 \quad 0 \quad 0 \quad 0 \quad 1)^T \\ \xi_4 &= (0 \quad 0 \quad 1 \quad 0 \quad 0)^T\end{aligned}$$

Whose corresppective matrix exponential gives:

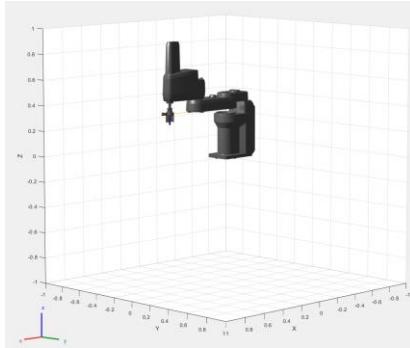
$$e^{\hat{\xi}_1 q_1} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad e^{\hat{\xi}_2 q_2} = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 s_2 \\ s_2 & c_2 & 0 & l_1 (1 - c_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$e^{\hat{\xi}_3 q_3} = \begin{bmatrix} c_3 & -s_3 & 0 & (l_1 + l_2)s_3 \\ s_3 & c_3 & 0 & (l_1 + l_2)(1 - c_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad e^{\hat{\xi}_4 q_4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composing applying PoE method we get:

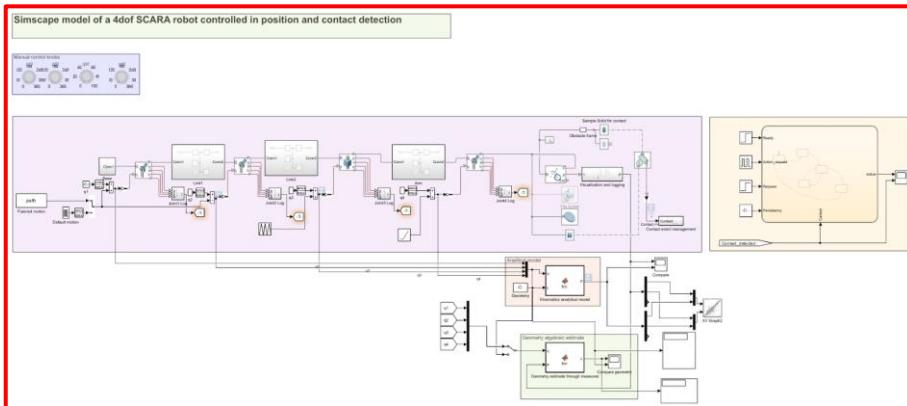
$$g_{ST}(q) = \begin{bmatrix} R(q) & p(q) \\ 0 & 1 \end{bmatrix} = e^{\hat{\xi}_1 q_1} e^{\hat{\xi}_2 q_2} e^{\hat{\xi}_3 q_3} e^{\hat{\xi}_4 q_4} g_{ST}(0) = \begin{bmatrix} c_{123} & -s_{123} & 0 & -l_1 s_1 - l_2 s_{12} \\ s_{123} & c_{123} & 0 & l_1 c_1 + l_2 c_{12} \\ 0 & 0 & 1 & l_0 + q_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The result can be obtained also using classic kinematic chain equations, where we can observe the motion coplanarity effect on first three joints, resulting in natural use of a sum of joint position coordinates making resulting kinematic equations by far more simple than simple composition of trigonometric functions. Differential kinematic equations can be easily obtained by differentiation at first and second order.

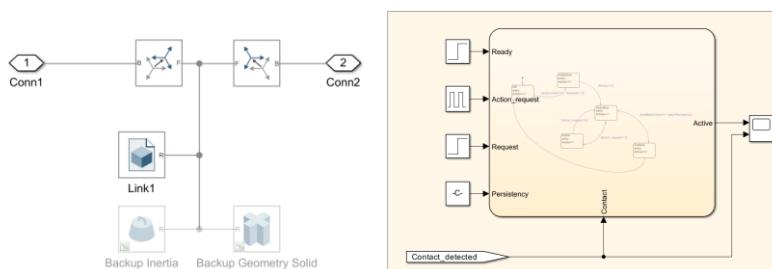


SCARA robot Simscape model

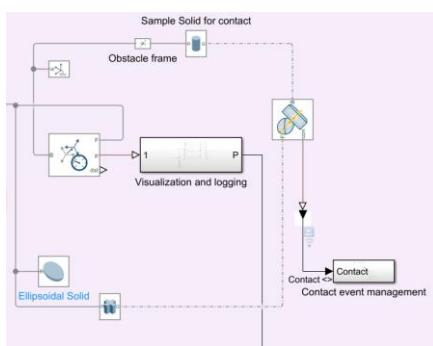
Using a Simscape™ multibody model is possible to create a digital twin of robot dynamic and kinematics, as well other multiphysics factors can be introduced in our model and simulated.



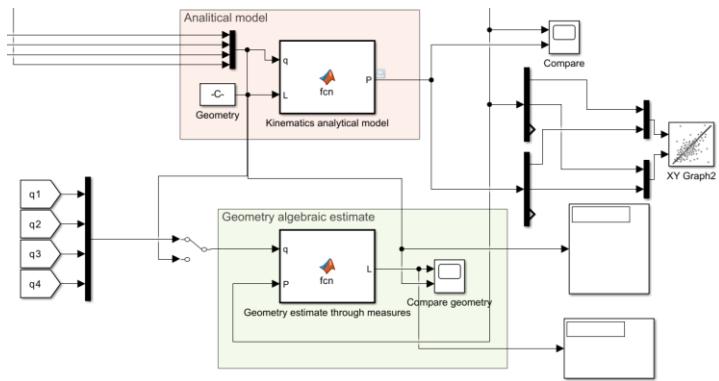
Example of basic link modeling, including an external file link geometry and a solid and inertia backup blocks (commented) to be used in case a simplified modeling is desired.



Model include a simple state state machine for managing operative condition like undesired contact with objects present in operative workspace. Obviously this include a contact model.



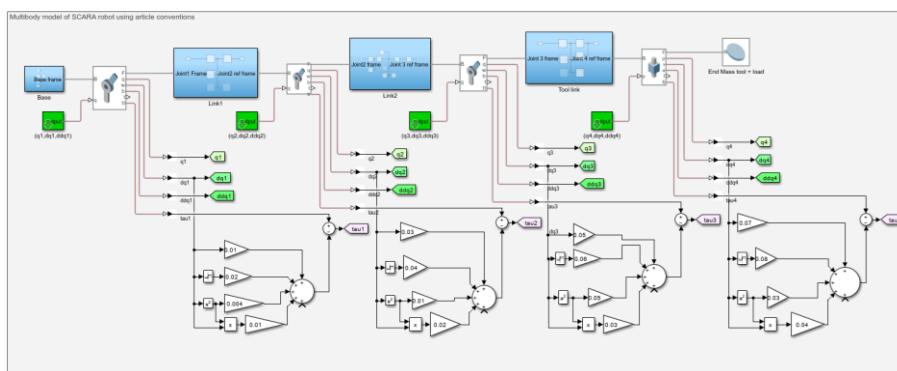
In addition some blocks for performing some kinematics estimate check were given.

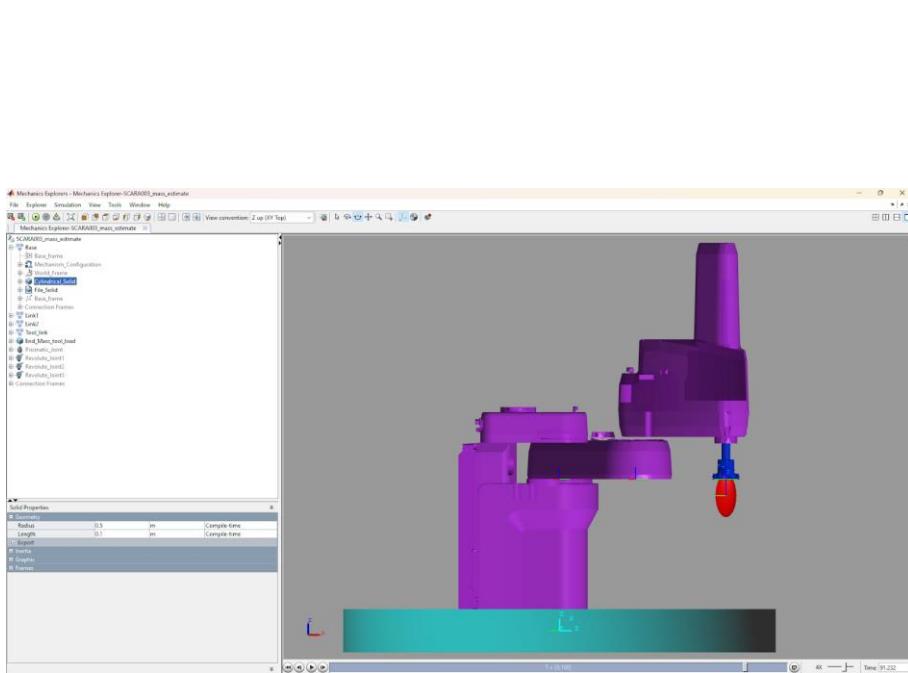


Simscape model of SCARA robot used for identification

Above presented model was used for initial modeling and exercise, providing insight on potentiality of modeling for digital twin development in a general sense. For specific identification problems a modified version of the same model was developed, cutting out some functionalities and relative blocks making it heavier to run and debugging. Model used for identification purposes was as follows. The model was inclusive of friction parameters.

Purpose was creating a digital twin able to generate a virtual experiment dynamic response of a real robot, complete with friction effects (optionally) and of measurement *noise* (added) a posteriori in data processing for limiting model running and providing control con noise scale and characteristics.. Anyway its implementation in the Simulink® model would be easy. Potentially the model could be extended and complexified at please introducing additional features.





SCARA Robot dynamic modeling

Using notions illustrated in previous chapters we're able to derive both a kinematic and dynamic model for a SCARA robot. This is a classic manual case in robotics, because of its relative simplicity allowing one to manage it in close form even by hands, without excessive burden, and resulting in a synthetic enough analytical model to be reported and inspected by a reader. Recalling then model proposed in the end of section [Dynamics], we can integrate it with required equations for third and fourth degree of freedom, getting full 4 dof dynamic frictionless inertial model.

In a second step we'll include a friction model for extending identification and modeling exercise including further physical phenomena.

Fortunately SCARA robot dynamics is considerably more simple than that of many other common manipulator robots thanks to its prominently planar motion and the orthogonality of the prismatic vertical extension tool axis, avoiding many coupling naturally occurring in manipulators arms, originating many complex terms.

Indeed inertia matrix for this robot is much more simple and can be easily found in literature as well in some of common books (see for instance [Murray] and [Modern Robotics: Mechanics, Planning, and Control]), but can also be easily derived from almost any robotics book, where it is used as standard example of robot., by an opportune collection and factoring of dynamic equations system.

In murray for instance inertia matrix is already given as *parameterized* for groups of inertial parameters linearly involved in the expression. Note the process is slightly different from traditional method that starts from dynamics equations, however derived. We start from a simpler and more compact group of expressions , using some check properties of inertia matrix like symmetry and recursivity properties that allow use some easy check. In most of

traditional literature inertia matrix calculation is considered a second step respect dynamics calculation, but we highlight that direct process of calculation using CRBA or using jacobians or GDAH is faster and more efficient, nonetheless easier to customize in case one want to introduce some grouped factor.

$$M(q) = \begin{bmatrix} \alpha + \beta + 2\gamma cq & \beta + 2\gamma cq & \delta & 0 \\ \beta + 2\gamma cq & \beta & \delta & 0 \\ \delta & \delta & \delta & 0 \\ 0 & 0 & 0 & m_4 \end{bmatrix}$$

where:

$$\begin{aligned} \alpha &= I_{1zz} + m_1 r_1^2 + m_2 l_1^2 + m_3 l_1^2 + m_4 l_1^2 \\ \beta &= I_{2zz} + I_{3zz} + I_{4zz} + m_3 l_2^2 + m_4 l_2^2 + m_2 r_2^2 \\ \gamma &= m_2 r_2 l_1 + m_3 l_1 l_2 + m_4 l_1 l_2 \\ \delta &= I_{3zz} + I_{4zz} \end{aligned}$$

This set of parameters can be seen as a linear combination of standard inertial parameters and is thus a legit set of parameters.

Moreover it is an essential set of parameters since all of them are potentially equally identifiable.

Using inertia matrix we get dynamic equations using one of possible ways, like calculating energy scalar function and applying Eulero Lagrange equations, or using directly inertia matrix differentiation respect coordinates, in order to get inertial terms, and then adding potential related terms (generalized positional forces) after. We used the first option in this case, in order to show possible automation of partial derivatives based in symbols form of dynamics equations.

$$K = \frac{1}{2} ((\alpha + \beta + 2\gamma cq_2)\dot{q}_1^2 + 2(\beta 2\gamma cq_2)\dot{q}_1\dot{q}_2 + 2(\beta 2\gamma cq_2)\dot{q}_1\dot{q}_2 + \delta(2\dot{q}_1\dot{q}_3 + 2\dot{q}_2\dot{q}_3 + \dot{q}_3^2) + m_4 \dot{q}_4^2)$$

$$\left. \begin{aligned} &\frac{(b + c \cos(q_2(t))) \frac{\partial^2}{\partial t^2} q_2(t)}{2} + \left(\frac{a}{2} + \frac{b}{2} + c \cos(q_2(t)) \right) \frac{\partial^2}{\partial t^2} q_1(t) + \left(\frac{b}{2} + \frac{c \cos(q_2(t))}{2} \right) \frac{\partial^2}{\partial t^2} q_3(t) + \frac{(a + b + 2c \cos(q_2(t))) \frac{\partial^2}{\partial t^2} q_1(t)}{2} + d \frac{\partial^2}{\partial t^2} q_3(t) - c \sin(q_2(t)) \left(\frac{\partial}{\partial t} q_2(t) \right)^2 - 2c \sin(q_2(t)) \frac{\partial}{\partial t} q_2(t) \frac{\partial}{\partial t} q_1(t) \\ &\left(c \sin(q_2(t)) \frac{\partial}{\partial t} q_1(t) + \frac{c \sin(q_2(t)) \frac{\partial}{\partial t} q_2(t)}{2} \right) \frac{\partial}{\partial t} q_1(t) + \frac{(b + c \cos(q_2(t))) \frac{\partial^2}{\partial t^2} q_1(t)}{2} + \left(\frac{b}{2} + \frac{c \cos(q_2(t))}{2} \right) \frac{\partial^2}{\partial t^2} q_1(t) + k \frac{\partial^2}{\partial t^2} q_2(t) + d \frac{\partial^2}{\partial t^2} q_3(t) - \frac{c \sin(q_2(t)) \frac{\partial}{\partial t} q_2(t) \frac{\partial}{\partial t} q_1(t)}{2} \\ &d \frac{\partial^2}{\partial t^2} q_1(t) + d \frac{\partial^2}{\partial t^2} q_2(t) + d \frac{\partial^2}{\partial t^2} q_3(t) \\ &m_4 \frac{\partial^2}{\partial t^2} q_4(t) \end{aligned} \right]$$

Comparing with a manipulator inertial dynamic equations are by far simpler and manageable.

In all cases is convenient when an explicit formulation of additional torques (especially non conservative one, like viscous terms, or related to hard to manage functions (like not

differentiable on) to add them in a second step. When they depend only on specific degree of freedom the job is simpler, but is not necessarily a condition.
In all ways we get a model complete of inverse dynamics.

SCARA robot identification

One of applicative use of digital twin considered in this work include the study of inertial and friction parameters identification process. Usually this experimental activity requires some intensive preparative task adopting analytical symbolic models and digital twins models for performing some data simulation activity.

SCARA Inertial base parameters

From formula of base parameter estimate we account for 19 base parameter.

$$b_m \leq 7n_r + 4n_t - 3 - \bar{\sigma}_1 - 2n_{g0} = \\ 7 \cdot 3 + 4 \cdot 1 - 3 - 1 - 2 = 19$$

By the way in this kind of robot we usually make simplification assumptions: neglect inertial tensor off diagonal terms, where used, and fourth and third link mass offset. Parallel axes simplify dynamic relations between links. Center of mass is usually considered to be located on the x axis, reducing by other 2 terms base parameters required. In the end simplified model accounts for link masses, inertia around joint axis and mass barycentric offset along joint to joint axis, but since not all of them are observable alone, resulting parameters are less, reducing of 15 parameters the account.

Here we use an alternative approach, example in deduction of inertial base parameters, exploiting inertia matrix parametrization, that in the end should be equivalent, since both methods lead to minimal set of parameters allowing for dynamic definition of the system. Standard inertial parameters playing similar roles are naturally grouped by their role in dynamics. Remarks in this way there's no need to apply base inertial parameters selection procedure neither formally (as suggested by Khalil) nor numerically (using QR or SVD methods as suggested by Gautier). Implicitly many standard inertial parameters and groups are left out of the identification process since they do not affect the model by evidence. In this case lack of identifiability is structural. Unless using different methods from mobile base theory, or the use of different kinds of observations, or exploiting a priori information like manufacturer declared values, CAD model estimate, or any other, some inertial parameters would be not observable. On the contrary, we shall keep in mind for purely dynamic prediction purposes they're not even strictly necessary, although single parameter identification in grouped terms would be preferable to grouped estimates. They're reported in table

	Link1	Link 2	Link 3	Link 4
<i>m</i>				

Commented [27]: Eliminate?

Commented [28]: Eliminate?

mc_x				
mc_y				
mc_z				
I_{xx}				
I_{xy}				
I_{xz}				
I_{yy}				
I_{yz}				
I_{zz}				

Commented [29]: Eliminate?

Commented [30]: Eliminate?

Commented [31]: Eliminate?

Commented [32]: Eliminate?

Commented [33]: Eliminate?

Commented [34]: Eliminate?

Commented [35]: Eliminate?

Commented [36]: Eliminate?

Commented [37]: Eliminate?

From Wensing-Slotine [Wensing et al. 17] we have a complete evaluation of SCARA robot inertial parameter identifiability:

	1(R)	2(R)	3(P)	4(R)
m	x		☆	★
mc_x	x	☆		★
mc_y	x	☆		★
mc_z	x	x	x	x
I_{xx}	x	x	x	x
I_{yy}	x	x	x	x
I_{zz}	☆	☆		★
I_{yz}	x	x	x	x
I_{xz}	x	x	x	x
I_{xy}	x	x	x	x
I_m	★	★	★	★
$\dim(\mathcal{V}_i)$	1 (2)	3 (4)	4 (4)	4 (4)
$\dim(\mathcal{K}_i)$	1 (1)	4 (4)	4 (4)	4 (4)
$\dim(\mathcal{T}_i)$	9	7	9	7

Commented [38]: Eliminate

Table 2. SCARA identifiable (★) and unidentifiable (x) parameters. A minimal parameter set is indicated (☆). Unmarked entries are only identifiable in linear combinations with ☆-type parameters. Numbers in parentheses indicate the dimension of the set within the algorithm when considering gravitational effects.

We've then five parameters among inertia can be identified. We used the least square method applied to observations derived from Simscape™ model simulation and used them for identification passing them through a simple linear regression process.

We used as trajectory a very simple sinusoidal motion for all joints, because in that way we know we're giving a solicitation suitable for exciting all parameter enough to assure a good level of confidence in estimation

All has been made using symbolic methods. We've used inertia matrix

Then we've calculated by automatic differentiation the Jacobian associated with the equations set, resulting to be a regressor for our model. Just mentioned models can include both inertial and friction parameters as well any other linearly playing parameter or group of parameters of the model.

$$\tau = Y \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ m_4 \end{pmatrix}$$

or using an extended model with friction parameters

$$\tau = Y \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ m_4 \\ f_{c_1} \\ f_{c_2} \\ f_{c_3} \\ f_{c_4} \\ f_{v_1} \\ f_{v_2} \\ f_{v_3} \\ f_{v_4} \end{pmatrix}$$

Robot regressor

Regressor matrix can be obtained simply from a dynamic model taking jacobian (partial derivative) with respect desired parameters vector or collecting respect them (in linear case that's equivalent).

The product of regressor by estimate parameters vector gives expected output.

Conversion of currents and motor model

Trajectory generation

Identification trajectory is designed using *truncated Fourier series* method, considering a finite number of harmonics with a prescribed motion amplitude limit. The use of a limited length fourier series, as well of amplitude limits, or anyway limited frequency input is fundamental for assuring that a proper consistency exists between robot actuators bandwidth and speed limitations and actuators real actuation capabilities in terms of torque and speed limits. This kind of excitation is considered most effective and simple for most identification. The use of a periodic function allows easy data processing and noise error prediction. In addition joint boundary conditions could be imposed directly embedding them on trajectory. In this regard see [Park 06]. Other authors preferred more traditional trajectories based on polynomial expressions. Both approaches allow closed form analytical treatment.

$$q_j(t) = a_{0j} + a_{1j}t + a_{2j}t^2 + a_{3j}t^3 + \cdots + a_{nj}t^5$$

Since Simscape™ model requires the input of joint position and first and second time derivatives (speed and accelerations) we first perform symbolic calculation

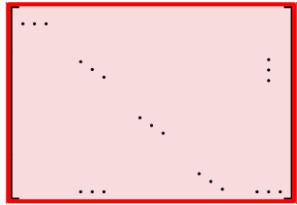
$$\begin{aligned} q_j(t) &= \sum_{i=1}^{n_a} C_{ij} \sin(\omega_i t) + S_{ij} \cos(\omega_i t) \\ \dot{q}_j(t) &= \sum_{i=1}^{n_a} C_{ij} \omega_i \cos(\omega_i t) + S_{ij} \omega_i \sin(\omega_i t) \\ \ddot{q}_j(t) &= -\sum_{i=1}^{n_a} C_{ij} \omega_i^2 \cos(\omega_i t) + S_{ij} \omega_i^2 \sin(\omega_i t) \end{aligned}$$

Commented [39]: keep this or simpler?

$$\begin{aligned} q_j(t) &= \sum_{i=1}^{n_a} C_{ij} \sin(\omega_i t) \\ \dot{q}_j(t) &= \sum_{i=1}^{n_a} C_{ij} \omega_i \cos(\omega_i t) \\ \ddot{q}_j(t) &= -\sum_{i=1}^{n_a} C_{ij} \omega_i^2 \cos(\omega_i t) \end{aligned}$$

Commented [40]: Mention of Sin-cos model?

Practically speaking the use of a couple of base frequencies was enough to perform the identification process. Substituting trajectory input into an analytical regressor determined afore, leads to an expression of arbitrary complexity (depending on input given complexity, potentially arbitrary, but in truth limited by machine computational capabilities). A short example is given here

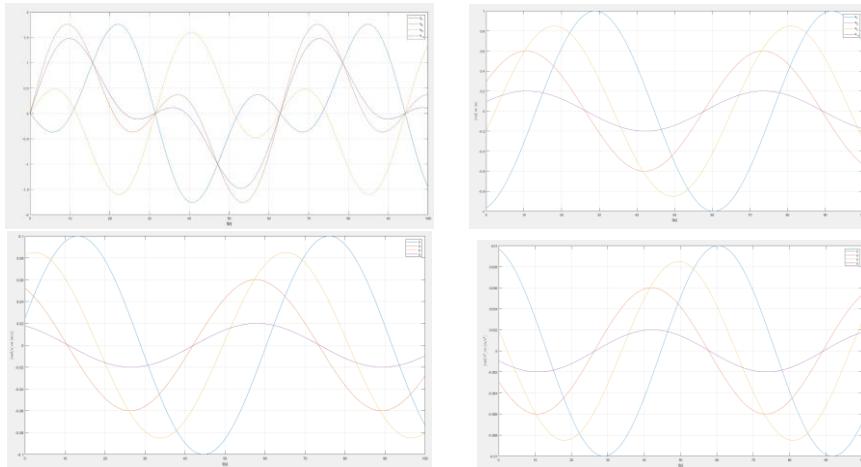


Commented [41]: Change matrix

Numerical values to be provided in input to the model are calculated in a second step as results of time span definition

$$t = t_0 : t_f = (t_0, \dots, t_k, t_{k+1}, \dots, t_f)$$

We can use a fixed sample time interval as well as an arbitrary set of time points freely as desired, in order to optimize input trajectory in purpose to improve results and efficiency/effectiveness. In our algorithm we decided to postpone numerical instantiation as late as possible to exploit at best symbolic treatment of model properties.



Also multiple harmonic inputs are possible, but even single harmonic in ideal conditions have shown to be enough for identification solutions with few points (in absence of noise and errors).

Trajectory optimization

In spite of the fact that the trajectory should be optimized, in our case in absence of measured signal errors and noise, estimation process was carried out effectively without need of a specific optimized trajectory. Of course in case of a real experiment design a trajectory refinement would be required in order to match identification objectives, once signal characteristics are clarified. Willing to implement trajectory optimization imply a set of simulations on different trajectory parameters and selection of best results.

Numerical instantiation

In order to exploit digital twin simulation and virtual experiment execution, we need to convert symbolic parametrized experiment defined previously into a numerical input choosing numerical values associated with the experiment.

In automatic mode a random assignment of parameters value is used, but specific experiment design based valorisation is as well possible. Choice of key parameters include

- Number of included harmonics
- Pulsation values
- Number of time samples
- Time span interval
- Joint motion amplitude

As mentioned in the scaling [Modeling and Control of Robot Manipulators] paragraph, all these parameters are not independent but interdependent, and their value are results of a design trade off between simulation-experiment length, dynamic solicitation, number of points managed.

Symbolic expected output (torques and currents)

The combination of input trajectory designed inside robot dynamic model in regression allows us to evaluate expected output in symbolic form as well as in numerical form. The numerical result is expected to match digital twin response and real system response less errors and noise factors.

Input to Simscape model and inverse dynamics simulation

After numerically assigned experiment key values, input vectors are generated and given in input to the digital twin model, for an inverse dynamics experiment with a given current model setup.

Experiment output includes joint position rereading and torques, with respective simulation time.

Normal equations and solution

Solution was obtained by means of normal equations, since no particular constraint or issues need to be considered for this case. Remark in addition there's no need to evaluate regressor matrix rank deficiency for this case. Since we need to estimate five parameters and we've just four observations, we need at least two separate samples for solving our problem (eight observables). In presence of noise and errors more samples, better if covering different configurations, is warmly suggested in order to mitigate effects on resulting estimate.

$$\chi = (A^T W A)^{-1} A^T y$$

Observations are collected as sampling t_s from output joint torques, and in case added with noise contributes. All observations are stacked on observation vector y . Remark is not required to use a full set of joint actuation for every sample, is just required to have enough observations for each joint, even if token randomly. Obviously the observation matrix shall be built following the same samples and observed actuations.

$$y = \tau(t_s) + \epsilon = \begin{pmatrix} \tau_1(t_{s1}) \\ \vdots \\ \tau_4(t_{s1}) \\ \tau_1(t_{s2}) \\ \vdots \\ \tau_4(t_{s2}) \\ \vdots \\ \tau_4(t_{sn}) \end{pmatrix}$$

Trials have proved that effective identification was possible with just two samples, in absence of noise.

Anyway some numerical errors already exist even in “perfect or ideal” conditions, driven, we guess, probably from numerical approximations making results anyway not errorless.

We indeed also made an attempt to solve all analytically to see the difference and no appreciable discrepancy was found on the final numerical solution.

Nonetheless we've one time further found an issue with limits of symbolic modeling since involved M by M matrix inversion or equivalently system closed form solution involved in normal equations solution becomes easily unaffordable yet for a bunch of degree of freedom, posing serious limitations to a general solution of the problem, even considering some simplification matrix properties like symmetry.

Further research could provide improvements, but not being the core point of this work has been relegated to a side attempt to take an original line of work and research.

Adding some numerical noise to artificial observations, we can simulate a real measurement. Trials have been proved that estimate can be performed and last almost correct (although some error shall be accepted by force in this case) all inertial and friction parameters, but more observations are opportune in order to reduce the impact of noise/error on final estimate.

Another observation done is that increasing the number of estimated parameters makes parameter vector estimates less accurate, redistributing errors among parameters.

Backmapping to standard parameters

As seen we've developed the whole process using a set of independent parameters used to parameterize inertia matrix,

As seen this is not possible for all inertial parameters unless some assumption is made, because, except for those of standard inertial parameters that cannot be in any case recovered, also grouped ones cannot be determined alone, and only directly observable one are accessible straight. Indeed one can easily see that the map of standard inertial parameters on used identification parameters is underdetermined and not invertible. For solving this problem one should recur to some assumption.

Overcoming physical and experimental limitations using digital twins

Indeed it wasn't possible for us to access to a real SCARA robot to get desired data. In addition the fact of using a numerical or simulated model in place of a real model allow us to have a priori knowledge on parameters value and check them out as getting out of the process, eliminating other potential sources of uncertainty like real physical value, measurement instruments and setup and many other, usually in a laboratory experiment are sources of uncertainty and potential sources of doubt on solution reliability. This is another factor of importance in the use of digital twins: they allow to provide ideal experiment posing base for controlled conditions where one can develop and test his algorithm in absolute safety, virtually costless, and easy to control and modify with respect to a real setup. For instance one can choose to provide observation with friction or without or setting some parameter to zero for suppressing some model effect, showing considerable flexibility.

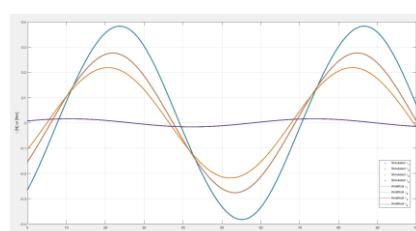
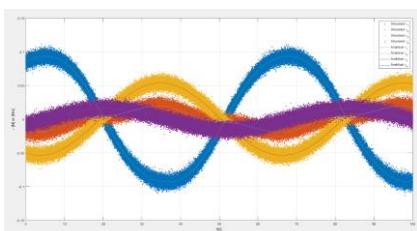
Moreover this approach allows us to extend our analysis to robot configurations other to physical real ones.

Simulation results and considerations

A set of simulations have been performed in order to check effective feasibility of a virtualized identification of inertial and dynamic parameters even in presence of noise.

Results show effectively inertial parameters are correctly identified although numerical errors are present even in absence of errors. Introduction of simulated error increases discrepancy between real value (known a priori) and estimated value, and increasing of used samples and distribution only partially can mitigate the effect, proving intrinsic need for the introduction of uncertainty on estimates carried out.

Furthermore, friction parameters (or better the whole set of dynamic parameters, because they're estimated together) are correctly found, with some numerical uncertainty. Shall be regarded that the use of an ideal model strongly facilitates the exercise, and real estimates are expected to be much more sensible to unmodeled factors affecting friction measures.



Future works and evolutions

The good performances of digital twins modeling for identification experiments opens the way to diverse improvements, studies and challenges.

Some are suggested here:

- Extensive study of numerical and simulated errors and noise effects (considering coloured noise and other kind of not gaussian white noise) on identification uncertainty and estimated parameters reliability
- Study of inertia matrix inversion using multibody techniques and applications
- Use of alternative algorithms and comparisons, like Slotine-Lie regressor formulation, alternative methods for optimization problem solution etc.
- Use of identification algorithm on real data vs digital twin response for a comparative analysis
- Use of online estimation of dynamic parameters in real robot operations using digital twin philosophy

From offline least squares to recursive least square and online estimate

Using a simple model we're confident we've got the right equations. We were able to perform a sample model using recursive least square (RLS) that in essence is not much different from a Kalman filter.

Implementation has been provided by a script simulation, a real process in postprocessing and also using a simulink implementation making a real on-line estimate.

results were compliant with offline and a priori expected solution.

Used implementations in both case recur to symbolic form expression can be used and recycled for both models easily, showing flexibility and generality of symbolic support based approach allowing for that flexibility of sue that most of mathematical formulations have respect traditional programming based approach where environment specific feature usually contaminate original form, making it less portable and reusable.

In addition the method promise very well in extending the workflow to generality, avoiding much of concernings one can arise when dealing with models.

So we expect that same procedure can be applied to almost all robots.

Online estimate do not just include inertial or friction parameters (these ones of most importance because they can effectively change during time, while inertial parameters usually are constant) and allows for real time monitoring of them during operations or in other phases of robot functioning.

SCARA robot direct dynamics and inertia matrix inversion

Inverting inertia matrix we can get a direct dynamics model usable both for a numerical simulations as well for a control law design purpose, as well for both them.

Happily we get inversion of inertia matrix is in this case rather simple and accessible.

Is worthy to see is not the case for most of more general inertia matrix because:

- are involving large expressions with many terms by themself for each position
- even in case of few terms inversion is computational burden intensive

Indeed as seen in the inertia matrix section, there are some helpful inversion algorithms for getting this naively, or with alternative algorithms as in many modern literature approaches. See [\[rodriguez1992spatial\]](#)

Problem states in general form:

$$\ddot{q} = M^{-1}(\tau - h(q, \dot{q}))$$

In SCARA robot inverse of inertia matrix is

$$\begin{pmatrix} -\frac{b-d}{c^2 \cos(q_2(t))^2 - ab + ad} & \frac{b-d+c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & -\frac{c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & 0 \\ \frac{b-d+c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & -\frac{a+b-d+2c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & \frac{a+c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & 0 \\ -\frac{c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & \frac{a+c \cos(q_2(t))}{c^2 \cos(q_2(t))^2 - ab + ad} & -\frac{ab-c^2 \cos(q_2(t))^2}{d(c^2 \cos(q_2(t))^2 - ab + ad)} & 0 \\ 0 & 0 & 0 & \frac{1}{m_4} \end{pmatrix}$$

or putting in evidence determinant

$$\frac{1}{-d m_4 (c^2 \cos(q_2(t))^2 - ab + ad)} \begin{pmatrix} b-d & b-d+c \cos(q_2(t)) & c \cos(q_2(t)) & 0 \\ b-d+c \cos(q_2(t)) & a+b-d+2c \cos(q_2(t)) & a+c \cos(q_2(t)) & 0 \\ c \cos(q_2(t)) & a+c \cos(q_2(t)) & \frac{ab-c^2 \cos(q_2(t))^2}{d} & 0 \\ 0 & 0 & 0 & \frac{1}{m_4} \end{pmatrix}$$

$$\begin{pmatrix} r_1^2 & l_1^2 & l_1^2 & l_1^2 & 1 & 0 & 0 & 0 \\ 0 & r_2^2 & l_2^2 & l_2^2 & 0 & 1 & 1 & 1 \\ 0 & l_1 r_2 & l_1 l_2 & l_1 l_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Once estimated value of inertia parameters (and in case of other parameters like friction ones) we're able to solve direct dynamics problems and control problems. That's one of reasons why identification is rather common task in robotics, especially when dealing with adaptive control where estimation of actual inertial parameters on line helps in compensating in best way acting loads (that may change because of changes in friction terms value or because of changes in load on end effector), improving control performances.

Multiphysics

Beyond all mechanics seen so far what is most interesting of Simscape™ and generally speaking of environments allowing for multiphysics modeling, is the ability to allow for including into models also other physical equations, like those involving temperature dynamics due to friction dissipation or electromagnetic behavior including voltage and currents.

Temperature effects on friction parameters

Friction terms are not constant but usually depend on temperature. Temperature dependence affects

An example of identification with temperature effects is given in:

[Dynamic and Friction Parameters of an Industrial Robot: Identification, Comparison and Repetitiveness Analysis](#)

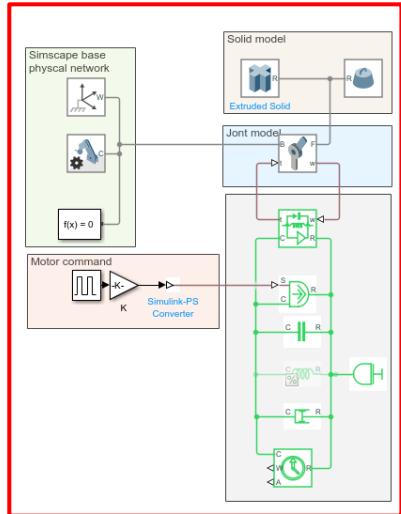
Commented [42]: Eliminate

Additional modeling

Inclusion of stateflow state machine has been tested together

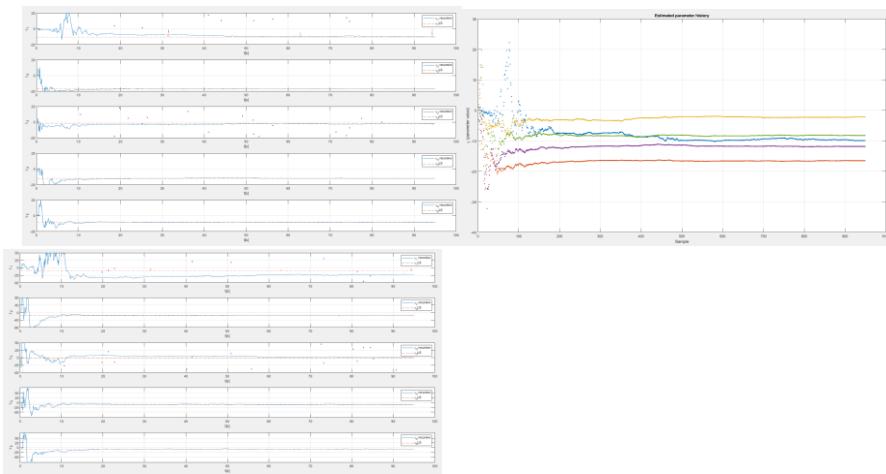
Modeling motor actuation

Motor model (rotational type) include a rotational inertia and some coupled joint frictional terms of Coulomb type, viscous and actuator inertia. In case even a torsional elastic term can be included. To couple them with a joint, a specific interface shall be included and connected.



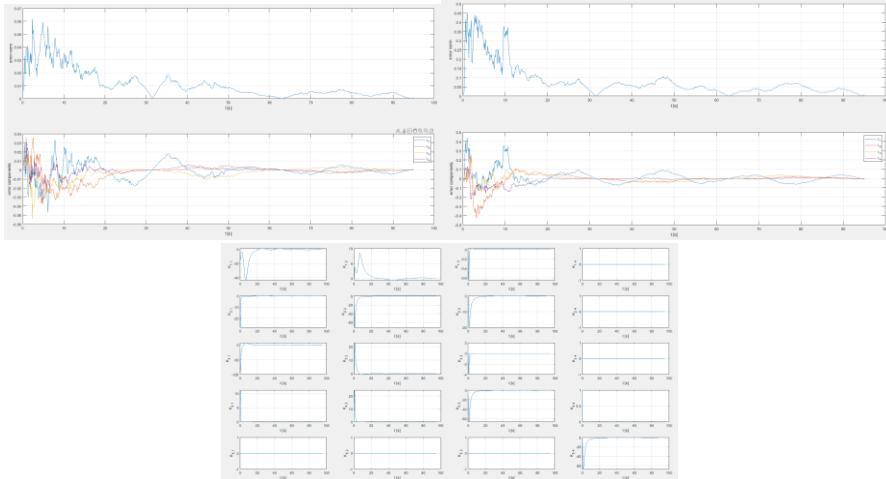
identification using recursive least square in matlab environment

Using digital twin simulation data it is possible to replicate an off-line or batch real time example of recursive least squares using recursive formulation with covariance matrix and Kalman gain. Initial estimate covariance gives initial error estimate respect to initial values (estimated) of parameters. After some time the algorithm converges to correct parameters value. Converge speed depends on Kalman gain and consequently on initial covariance matrix values.

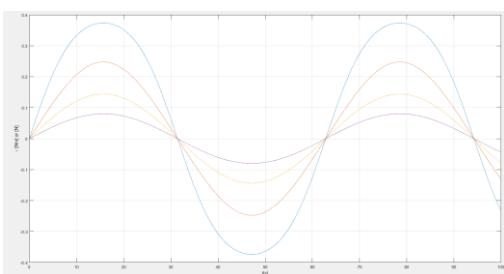


Estimate parameter time series shows trajectories configurations where position is zero lead to deviation in estimate of least square are canceled by recursive method. Kalman gain values

are not constant but change over time as covariance estimate changes, but usually in normal conditions converge to small correction values under convergence conditions. The more the error or noise, the larger the error and slower the convergence and residual error.



Trajectory used is a simple sinusoidal synchronous position joint command.



Identification using online LS estimator in Simulink/Simscape environment

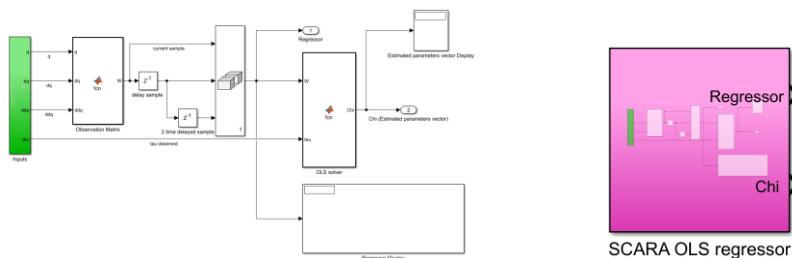
The process of identification seen previously was implemented in the Matlab script environment, where exploitation of symbolic manipulation is possible.

In a Simulink® environment the use of symbolic manipulation is not possible, and one shall proceed numerically. Nonetheless implementation of OLS or KF is anyway possible once an analytical model is available, because it's possible to create a Matlab function or a Simulink® block performing numerical operations provided by the method. Since it is possible to take directly joint position and derivatives as measures, these were used, but in a real case a direct measure wouldn't be available and numerical differentiation used for reconstruction.

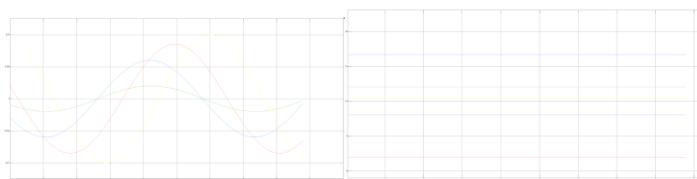
Purpose of this exercise is to give proof of the possibility of real time and online implementations of RLS or OLS or WLS implementation of identification processes. In spite of the lack of an in field evaluation of computational burden and feasibility on real target hardware, the simplicity of equations used, and the power of modern available hardware, allows to predict a reasonable feasibility expectation. Workload on the Simulink® model was negligible.

Major concerns would consist in measurement noise and un-modeled phenomena.

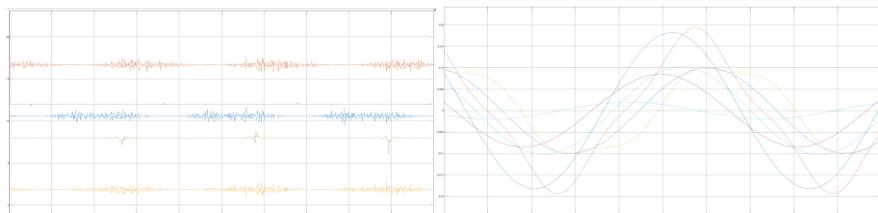
Used Simulink® setup is presented here (OLS). Remark more than one sample is required even in ideal conditions in order to carry out a solution



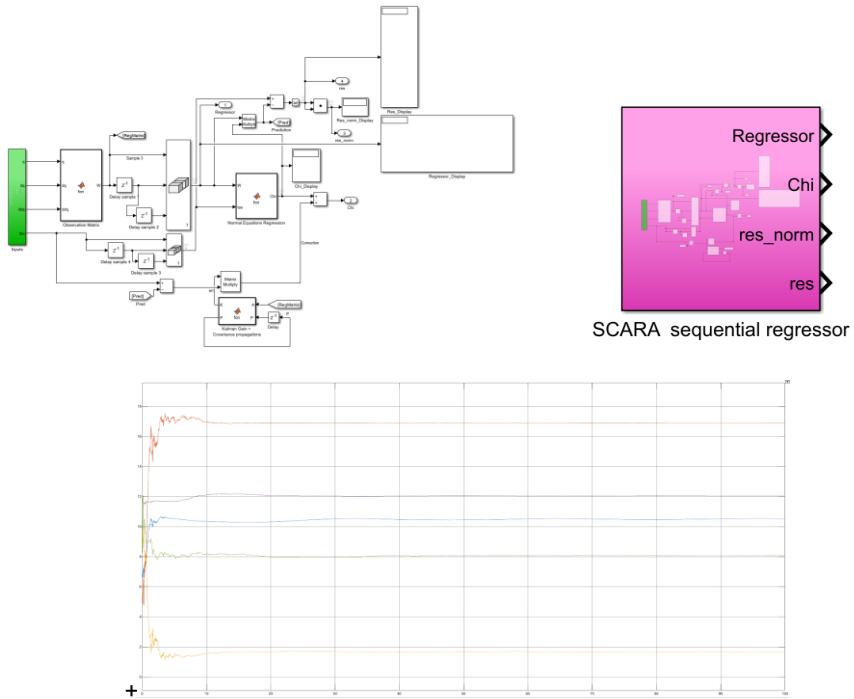
with some exemplificative results



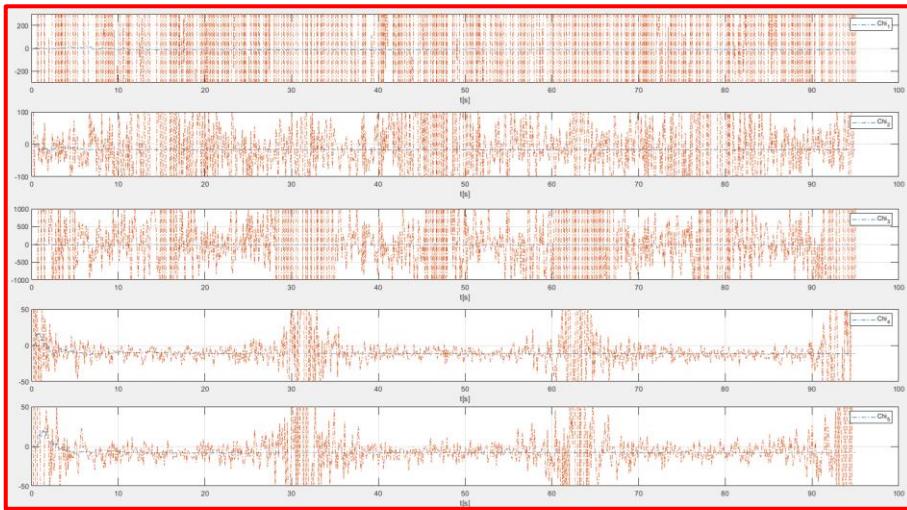
Using some simulated noise we can see how a 3 sample estimate responds to a white noise power 1e-10. Regressor on the right. Increased resilience can be obtained using multiple samples, but then we prefer to go for a recursive filter straight.



A recursive version allows us to overcome sample collection limits. The use of opportune weighting leading to inclusion of covariance matrix and propagation, in order to get optimal covariance of signal under gaussian noise assumption provide an improved version of basic RLS algorithm, could anyway be reconducted to by imposing unitary covariance.



Also some simulations using noisy simulated measurements has been performed to reproduce a more realistic performance.



Remarkable, most of the model when implemented with Matlab functions is largely reusable since I/O block is vectorised and just regressor structure and visualization shall be modified.

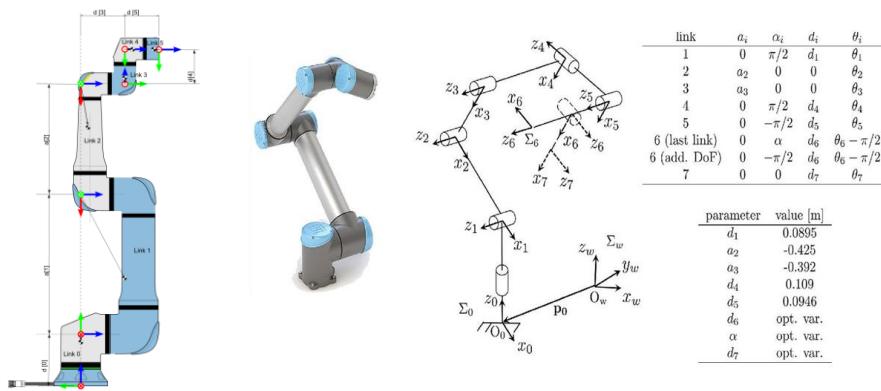
[Kalman filter using simulink model](#)

Direct dynamics and control

Case study II: UR5 articulated manipulator 6dof robot

UR5 robot present a traditional manipulator architecture with a root vertical joint, a second horizontal joint (shoulder) and a second parallel joint (elbow) plus a non spherical wrist composed of three *non concurrent joints*. The use of offsets on all joints is fundamental for reducing singular configurations effects and potential locks. Totally we've 6 degree of freedom, as for most of manipulators, due to trade off between manipulability and redundancy, enough for assuring an operative space reachability suitable for most of operations in 3D space and avoiding joint redundancy. This architecture is very common in articulated arm manipulators and in some measure emulate human arms kinematic structure although usually robotic joints have larger extension respect human articulations allowing for more complex and extensive motions. Anyway given arm geometry cause some limitations on close basis reachability.

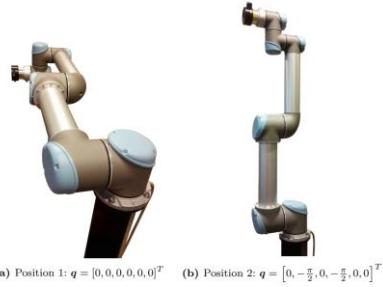
Robot layout and Denavit Hartenberg parameters table is given in the following:



A first obvious limitation of robotic arm workspace is due to maximum arm extension delimiting workspace.



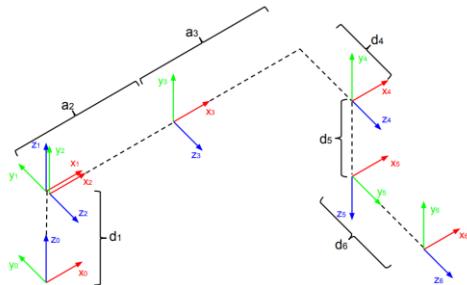
Layout position of robot in default configuration is given in pictures



(a) Position 1: $\mathbf{q} = [0, 0, 0, 0, 0]^T$ (b) Position 2: $\mathbf{q} = [0, -\frac{\pi}{2}, 0, -\frac{\pi}{2}, 0, 0]^T$

UR5 kinematics equations

UR5 robot kinematics can be defined in terms of DH parameters. Schematic illustration of sequence is given in following picture



Coordinate frames for UR arm. Joints rotate around the z-axes and are pictured at $\theta_i = 0$ for $1 \leq i \leq 6$.

Using common definition of trigonometric variables and usual indexing of revolute joint coordinates.

$${}^0T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^6_5 T {}^5_4 T {}^4_3 T {}^3_2 T {}^2_1 T {}^1_0 T$$

$$\begin{aligned}
r_{11} &= c_1 c_2 c_{234} c_5 c_6 + c_6 s_1 s_5 - c_1 \\
r_{21} &= c_{234} c_5 c_6 s_1 - c_1 c_6 s_5 - s_1 s_{234} s_6 \\
r_{31} &= c_5 c_6 s_{234} + c_{234} s_6 \\
r_{12} &= -c_1 c_{234} c_5 c_6 - s_1 s_5 s_6 - c_1 c_6 s_{234} \\
r_{22} &= -c_{234} c_5 s_1 s_6 + c_1 s_5 s_6 - c_6 s_1 s_{234} \\
r_{32} &= -c_5 s_{234} s_6 + c_{234} c_6 \\
r_{13} &= -c_1 c_{234} s_5 + c_5 s_1 \\
r_{23} &= -c_{234} s_1 s_5 - c_1 c_5 \\
r_{33} &= -s_{234} s_5
\end{aligned}$$

$$\begin{aligned}
p_x &= r_{13} d_6 + c_1 (s_{234} d_5 + c_{23} a_3 + c_2 a_2) + s_1 d_4 \\
p_y &= r_{23} d_6 + s_1 (s_{234} d_5 + c_{23} a_3 + c_2 a_2) - c_1 d_4 \\
p_z &= r_{33} d_6 - c_{234} d_5 + s_{23} a_3 + s_2 a_2 + d_1
\end{aligned}$$

Singular configurations

Jacobian matrix determinant is given by

$$\det(J) = s_3 s_5 a_2 a_3 (c_2 a_2 + c_{23} a_3 + s_{234}) d_5$$

determining potential singular configurations causing not invertibility of task-space to coordinate space backmapping by matrix inversion (note is a 6x6 square matrix).

Three types of singularities (shoulder, wrist, and elbow) exist for this robot. Information about them can be found in [], and they are briefly described next.

- A *shoulder singularity* happens when the last factor in equation, which involves angles q_2, q_3, q_4 is equal to zero. One example can be seen in **Figure 2**, which shows that the end effector cannot be moved along z_6
- *Wrist singularities* exist when $s_5 = 0$ which mathematically happens when $q_5 = 0$ or $q_5 = \pm\pi$ meaning $z_4 // z_6$ are parallel
- An *elbow singularity* is present when $s_3 = 0$ which happens when $q_3 = 0$ or $q_3 = \pm\pi$. This means that the arm is fully stretched or bent; however, only the former case is physically possible.

For practical purposes one should consider that any conditions making $\det(J)$ close to zero indicates nearly singular configurations where mobility and manipulability ellipsoid shrinks.



Center of mass position

Here we report links center of mass position as function of joint coordinates:

$$p_{c_2} = \begin{pmatrix} a_1 + s_1 (d_2 + p_{c2z}) + c_1 (a_2 + c_2 p_{c2x}) \\ s_1 (a_2 + c_2 p_{c2x}) - c_1 (d_2 + p_{c2z}) \\ d_1 + p_{c2x} s_2 \end{pmatrix}$$

$$p_{c_3} = \begin{pmatrix} a_1 + s_1 (d_2 + d_3 + p_{c3z}) + c_1 (a_2 + a_3 c_2 + c_{23} p_{c3x}) \\ s_1 (a_2 + a_3 c_2 + c_{23} p_{c3x}) - c_1 (d_2 + d_3 + p_{c3z}) \\ d_1 + a_3 s_2 + p_{c3x} s_{23} \end{pmatrix}$$

$$p_{c_4} = \begin{pmatrix} a_1 + c_1 (a_2 + a_3 c_2 + a_4 c_{23} - p_{c4y} s_{234}) + s_1 (d_2 + d_3 + d_4 + p_{c4z}) \\ s_1 (a_2 + a_3 c_2 + a_4 c_{23} - p_{c4y} s_{234}) - c_1 (d_2 + d_3 + d_4 + p_{c4z}) \\ d_1 + c_{234} p_{c4y} + a_3 s_2 + a_4 s_{23} \end{pmatrix}$$

$$p_0 = \begin{pmatrix} a_1 + s_1 (d_2 + d_3 + d_4 - c_1 p_{c0x}) + c_1 (a_2 - s_2 (s_3 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) + c_2 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_5 + p_{c0z}))) + c_2 (a_2 + c_2 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) - s_3 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) \\ s_1 (a_2 - s_2 (s_3 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) + c_2 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_5 + p_{c0z}))) + c_2 (a_2 + c_2 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) - s_1 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) - c_1 (d_2 + d_3 + d_4 - c_2 p_{c0x}) \\ d_1 + c_2 (s_2 (s_3 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) + c_2 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_5 + p_{c0z}))) + s_2 (a_2 + c_2 (a_4 - s_4 (d_5 + p_{c0x}) + c_1 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z}))) - s_3 (s_4 (a_5 - p_{c0y} s_5) + c_1 (d_6 + p_{c0z})) \end{pmatrix}$$

UR5 dynamics modeling

UR5 dynamic model can be derived using Euler-Lagrange method, RNEA method, or calculating first mass matrix and then deriving EOM.

We illustrate the use of RNEA method for exemplificative purposes.

First step consist in kinematic propagation of angular velocities and accelerations. Using vectorial notation without components (avoiding frame projection for simplicity).

$$\begin{aligned} \omega_1 &= \omega_0 + \hat{z}_1 \dot{q}_1 & \dot{\omega}_1 &= \dot{\omega}_0 + \hat{z}_1 \ddot{q}_1 + \dot{q}_1 \omega_0 \times \hat{z}_1 \\ \omega_2 &= \omega_1 + \hat{z}_2 \dot{q}_2 & \dot{\omega}_2 &= \dot{\omega}_1 + \hat{z}_2 \ddot{q}_2 + \dot{q}_2 \omega_1 \times \hat{z}_2 \\ \omega_3 &= \omega_2 + \hat{z}_3 \dot{q}_3 & \dot{\omega}_3 &= \dot{\omega}_2 + \hat{z}_3 \ddot{q}_3 + \dot{q}_3 \omega_2 \times \hat{z}_3 \\ \omega_4 &= \omega_3 + \hat{z}_4 \dot{q}_4 & \dot{\omega}_4 &= \dot{\omega}_3 + \hat{z}_4 \ddot{q}_4 + \dot{q}_4 \omega_3 \times \hat{z}_3 \\ \omega_5 &= \omega_4 + \hat{z}_5 \dot{q}_5 & \dot{\omega}_5 &= \dot{\omega}_4 + \hat{z}_5 \ddot{q}_5 + \dot{q}_5 \omega_4 \times \hat{z}_4 \\ \omega_6 &= \omega_5 + \hat{z}_6 \dot{q}_6 & \dot{\omega}_6 &= \dot{\omega}_5 + \hat{z}_6 \ddot{q}_6 + \dot{q}_6 \omega_5 \times \hat{z}_5 \end{aligned}$$

Inertial wrench are calculated using Eulero's equations

$$\begin{aligned} \mu_{I1} &= I_1 \dot{\omega}_1 + \omega_1 \wedge I_1 \omega_1 & f_{I1} &= m_1 \dot{v}_1 + \omega_1 \wedge m_1 v_1 \\ \mu_{I2} &= I_2 \dot{\omega}_2 + \omega_2 \wedge I_2 \omega_2 & f_{I2} &= m_2 \dot{v}_2 + \omega_2 \wedge m_2 v_2 \\ \mu_{I3} &= I_3 \dot{\omega}_3 + \omega_3 \wedge I_3 \omega_3 & f_{I3} &= m_3 \dot{v}_3 + \omega_3 \wedge m_3 v_3 \\ \mu_{I4} &= I_4 \dot{\omega}_4 + \omega_4 \wedge I_4 \omega_4 & f_{I4} &= m_4 \dot{v}_4 + \omega_4 \wedge m_4 v_4 \\ \mu_{I5} &= I_5 \dot{\omega}_5 + \omega_5 \wedge I_5 \omega_5 & f_{I5} &= m_5 \dot{v}_5 + \omega_5 \wedge m_5 v_5 \\ \mu_{I6} &= I_6 \dot{\omega}_6 + \omega_6 \wedge I_6 \omega_6 & f_{I6} &= m_6 \dot{v}_6 + \omega_6 \wedge m_6 v_6 \end{aligned}$$

Using backward loop on bodies from end effector to root one get (avoiding projection on frames, that for inertial forces is body frame)

$$\begin{aligned} f_6 &= f_{I6} + f_{ee} & \mu_6 &= \mu_{I6} + \mu_{ee} \\ f_5 &= f_{I5} + f_6 + f_{e5} & \mu_5 &= \mu_{I5} + \mu_6 + \mu_{e5} \\ f_4 &= f_{I4} + f_5 + f_{e4} & \mu_4 &= \mu_{I4} + \mu_5 + \mu_{e4} \\ f_3 &= f_{I3} + f_4 + f_{e3} & \mu_3 &= \mu_{I3} + \mu_4 + \mu_{e3} \\ f_2 &= f_{I2} + f_3 + f_{e2} & \mu_2 &= \mu_{I2} + \mu_3 + \mu_{e2} \\ f_1 &= f_{I1} + f_2 + f_{e1} & \mu_1 &= \mu_{I1} + \mu_2 + \mu_{e1} \end{aligned}$$

Resulting actuator torques are given by projecting wrenches on actuation axis considering joint nature that for our case is $\bar{\sigma} = 1$ and $\sigma = 0$ for all joints. Then forces do not count since are transmitted directly to base. This last set gives EOM of system

$$\begin{aligned}\tau_6^j &= \bar{\sigma} \hat{z}_6 \cdot (\mu_6) + \sigma \hat{z}_6 \cdot (f_6) \\ \tau_6 &= \bar{\sigma} \hat{z}_6 \cdot (\mu_6) + \sigma \hat{z}_6 \cdot (f_6) \\ \tau_5^j &= \bar{\sigma} \hat{z}_5 \cdot (\mu_5) + \sigma \hat{z}_5 \cdot (f_5) \\ \tau_5 &= \bar{\sigma} \hat{z}_5 \cdot (\mu_5) + \sigma \hat{z}_5 \cdot (f_5) \\ \tau_4^j &= \bar{\sigma} \hat{z}_4 \cdot (\mu_4) + \sigma \hat{z}_4 \cdot (f_4) \\ \tau_4 &= \bar{\sigma} \hat{z}_4 \cdot (\mu_4) + \sigma \hat{z}_4 \cdot (f_4) \\ \tau_3^j &= \bar{\sigma} \hat{z}_3 \cdot (\mu_3) + \sigma \hat{z}_3 \cdot (f_3) \\ \tau_3 &= \bar{\sigma} \hat{z}_3 \cdot (\mu_3) + \sigma \hat{z}_3 \cdot (f_3) \\ \tau_2^j &= \bar{\sigma} \hat{z}_2 \cdot (\mu_2) + \sigma \hat{z}_2 \cdot (f_2) \\ \tau_2 &= \bar{\sigma} \hat{z}_2 \cdot (\mu_2) + \sigma \hat{z}_2 \cdot (f_2) \\ \tau_1^j &= \bar{\sigma} \hat{z}_1 \cdot (\mu_1) + \sigma \hat{z}_1 \cdot (f_1) \\ \tau_1 &= \bar{\sigma} \hat{z}_1 \cdot (\mu_1) + \sigma \hat{z}_1 \cdot (f_1)\end{aligned}$$

Remark actuation torque here accounts for link dynamics and applied *external wrenches*, requires for accounting also for friction forces/torques (to be accounted as *internal wrenches*, consequently not affecting dynamic balance afore, because subject to action-reaction principle) can be added directly into previous equations or lately as we do specify distinction between actuator forces (at load side) and joint required forces. In this model we didn't considered the contribute of actuator and transmission inertia.

$$\begin{aligned}\tau_6 &= \tau_6^j + f_{c_6} \text{sign}(\dot{q}_6) + f_{v_6} \dot{q}_6 + f_{v_6^2} \dot{q}_6^2 + \dots \\ \tau_5 &= \tau_5^j + f_{c_5} \text{sign}(\dot{q}_5) + f_{v_5} \dot{q}_5 + f_{v_5^2} \dot{q}_5^2 + \dots \\ \tau_4 &= \tau_4^j + f_{c_4} \text{sign}(\dot{q}_4) + f_{v_4} \dot{q}_4 + f_{v_4^2} \dot{q}_4^2 + \dots \\ \tau_3 &= \tau_3^j + f_{c_3} \text{sign}(\dot{q}_3) + f_{v_3} \dot{q}_3 + f_{v_3^2} \dot{q}_3^2 + \dots \\ \tau_2 &= \tau_2^j + f_{c_2} \text{sign}(\dot{q}_2) + f_{v_2} \dot{q}_2 + f_{v_2^2} \dot{q}_2^2 + \dots \\ \tau_1 &= \tau_1^j + f_{c_1} \text{sign}(\dot{q}_1) + f_{v_1} \dot{q}_1 + f_{v_1^2} \dot{q}_1^2 + \dots\end{aligned}$$

Actuators motor and transmission inertia contribution can be added directly before projection or later as here, accounting for transmission ratio k. In UR5 model robot transmission ration is 100, giving a not negligible contribute. Equations give torque required at load side.

$$\begin{aligned}\tau_6^a &= \tau_6 + k_6^2 (I_{m_6} + I_{r_6}) \ddot{q}_6 \\ \tau_5^a &= \tau_5 + k_5^2 (I_{m_5} + I_{r_5}) \ddot{q}_5 \\ \tau_4^a &= \tau_4 + k_4^2 (I_{m_4} + I_{r_4}) \ddot{q}_4 \\ \tau_3^a &= \tau_3 + k_3^2 (I_{m_3} + I_{r_3}) \ddot{q}_3 \\ \tau_2^a &= \tau_2 + k_2^2 (I_{m_2} + I_{r_2}) \ddot{q}_2 \\ \tau_1^a &= \tau_1 + k_1^2 (I_{m_1} + I_{r_1}) \ddot{q}_1\end{aligned}$$

Vectoral equations should then be projected on proper frame to have resulting equations

Kinetic energy is given by

$$K = \underbrace{\frac{1}{2}\omega_1^T I_1 \omega_1}_{\text{body frame 1}} + \underbrace{\frac{1}{2}\omega_2^T I_2 \omega_2}_{\text{body frame 2}} + \underbrace{\frac{1}{2}\omega_3^T I_3 \omega_3}_{\text{body frame 3}} + \underbrace{\frac{1}{2}\omega_4^T I_4 \omega_4}_{\text{body frame 4}} + \underbrace{\frac{1}{2}\omega_5^T I_5 \omega_5}_{\text{body frame 5}} + \underbrace{\frac{1}{2}\omega_6^T I_6 \omega_6}_{\text{body frame 6}} + \underbrace{\frac{1}{2}v_1^T m_1 v_1 + \frac{1}{2}v_2^T m_2 v_2 + \frac{1}{2}v_3^T m_3 v_3 + \frac{1}{2}v_4^T m_4 v_4 + \frac{1}{2}v_5^T m_5 v_5 + \frac{1}{2}v_6^T m_6 v_6}_{\text{frame at choice for each term}}$$

Using symbolic manipulation algorithm based on Jacobian matrix

Inertia matrix

Inertia matrix can be directly computed using Jacobians derived from kinematics. Resulting expression is a 6x6 matrix very complex, that even after simplifications and trigonometric parameters conventional renaming result by far hard to read and to interpret. Unlike SCARA case trigonometric terms included are by fare more heterogeneous and complex, so that its treatment is unpractical for direct inspection. The fact of using negligible values of inertia helps little. On the other side adopting symbols simplifications allow for some considerable reduction in complexity and reveal underlying structure, still very complex but quasi-affordable at least for last degree of freedom.

$$M = M_m + M_I = J_{v_1} m_1 J_{v_1}^T + J_{v_2} m_2 J_{v_2}^T + J_{v_3} m_3 J_{v_3}^T + J_{v_4} m_4 J_{v_4}^T + J_{v_5} m_5 J_{v_5}^T + J_{v_6} m_6 J_{v_6}^T + J_{\omega_1} I_1 J_{\omega_1}^T + J_{\omega_2} I_2 J_{\omega_2}^T + J_{\omega_3} I_3 J_3^T + J_{\omega_4} I_4 J_{\omega_4}^T + J_{\omega_5} I_5 J_{\omega_5}^T + J_{\omega_6} I_6 J_{\omega_6}^T$$

$$\begin{aligned} M(0,0) &= m_1 p_{v_1}^2 \\ M(0,1) &= -m_1 p_{v_1} s_1 (d_1 + p_{\omega_1}) \\ M(0,3) &= -m_1 p_{v_1}^2 + m_1 (a_1^2 + d_1^2 + p_{\omega_1}^2 + p_{v_1}^2 + 2d_1 p_{v_1} + 2m_1 p_{v_1} s_1) \\ M(4,4) &= m_1 (a_1^2 + d_1^2 + p_{v_1}^2 + p_{\omega_1}^2 + 2d_1 p_{v_1} - p_{v_1}^2 - 2m_1 p_{v_1} s_1 + m_1 p_{v_1}^2) \\ m_1 (a_1^2 + d_1^2 + d_1^2 (1 - s_1^2) + p_{v_1}^2 + p_{\omega_1}^2 + 2d_1 p_{v_1} + 2m_1 p_{v_1} s_1 + 2m_1 p_{v_1} a_1 - 2d_1 p_{v_1} s_1 + 2d_1 p_{v_1} a_1 + m_1 p_{v_1} p_{\omega_1} c_1 s_1) \\ M(3,3) &= m_1 p_{v_1}^2 + m_1 (a_1^2 - 2d_1 p_{v_1} s_1 + p_{\omega_1}^2) + m_1 (a_1^2 + d_1^2 + 2d_1 p_{v_1} - p_{v_1}^2 + p_{\omega_1}^2 - 2m_1 p_{v_1} s_1 - 2d_1 p_{v_1} a_1 - 2m_1 p_{v_1} s_1 + 2d_1 p_{v_1} a_1 - 2d_1 p_{v_1} c_1 s_1) \\ m_1 (2d_1 p_{v_1} - a_1^2 p_{v_1}^2 - p_{v_1}^2 d_1^2 - p_{\omega_1}^2 + a_1^2 + d_1^2 + p_{v_1}^2 + p_{\omega_1}^2 - 2d_1 p_{v_1} s_1 + 2m_1 p_{v_1} a_1 + m_1 p_{v_1} p_{\omega_1} c_1 s_1 + 2m_1 p_{v_1} c_1 s_1 + 2m_1 p_{v_1} a_1 s_1 - 2d_1 p_{v_1} a_1 s_1 - 2d_1 p_{v_1} c_1 s_1 + 2m_1 p_{v_1} c_1 s_1 + 2m_1 p_{v_1} a_1 s_1 + 2d_1 p_{v_1} c_1 s_1 + 2p_{v_1} c_1 s_1)$$

The presence of joints with offsets make the matrix tendentially full, and richer of terms than in absence of.

So far we've considered only symbolic expressions of inertia matrix, parametrizing robot mass, and center of mass position. Since manufacturer declares their nominal values we can use them for defining an as designed inertia matrix, whose dependence on joint positions is kept as explicit parameters.

Inertia matrix expression is still very cumbersome, and in author opinion even less readable than in symbols form, but on the other side can be directly used for numerical evaluations and practical purposes. Remark matrix would still depend on symbolic variable and inversion n closed form would be unpractical, unless using inversion methods from multibody and spatial operator algebra seen in [Rodriguez], [Jan et al].

Jacobians matrix

Here we report at illustration purpose Jacobian matrices for links center of mass used in mass matrix calculation and in other relevant quantities implied in modeling robot dynamics and acting wrenches.

$$J_{C_1} = \begin{pmatrix} -c_1 p_{c1y} & 0 & 0 & 0 & 0 & 0 \\ -p_{c1y} s_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}'$$

$$J_{C_2} = \begin{pmatrix} c_1 (d_2 + p_{c2z}) - s_1 (a_2 + c_2 p_{c2x}) & -c_1 p_{c2x} s_2 & 0 & 0 & 0 & 0 \\ s_1 (d_2 + p_{c2z}) + c_1 (a_2 + c_2 p_{c2x}) & -p_{c2x} s_1 s_2 & 0 & 0 & 0 & 0 \\ 0 & c_2 p_{c2x} & 0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & 0 \\ 0 & -c_1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}'$$

$$J_{C_3} = \begin{pmatrix} c_1 (d_2 + d_3 + p_{c3z}) - s_1 (a_2 + c_2 (a_3 + c_3 p_{c3x}) - p_{c3x} s_2 s_3) & -c_1 (s_2 (a_3 + c_3 p_{c3x}) + c_2 p_{c3x} s_3) & -c_1 (c_2 p_{c3x} s_3 + c_3 p_{c3x} s_2) & 0 & 0 & 0 \\ s_1 (d_2 + d_3 + p_{c3z}) + c_1 (a_2 + c_2 (a_3 + c_3 p_{c3x}) - p_{c3x} s_2 s_3) & -s_1 (s_2 (a_3 + c_3 p_{c3x}) + c_2 p_{c3x} s_3) & -s_1 (c_2 p_{c3x} s_3 + c_3 p_{c3x} s_2) & 0 & 0 & 0 \\ 0 & c_2 (a_3 + c_3 p_{c3x}) - p_{c3x} s_2 s_3 & c_2 c_3 p_{c3x} - p_{c3x} s_2 s_3 & 0 & 0 & 0 \\ 0 & s_1 & s_1 & 0 & 0 & 0 \\ 0 & -c_1 & -c_1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}'$$

We can observe once again as expressions arising from kinematic chain involved are quickly increasing complexity. Moreover observe as every column is associated in a open chain, to a joint position index in the sequence of link numbering, and consequently kinematics vectors, and jacobian columns cannot depend on higher index, and usually depend on all (or most) of predecessor joint-link indexes.

Jacobian matrices have multiple utility use like:

- Determination of static condition joint torques for applied forces (including gravity)
- Determination of link's center of mass linear speeds as function of joints angular speeds
- Determination of links angular speed as function of joint angular speed
-

Gravitational terms

In this model we've assumed as reasonable assumption that gravitational acceleration acts along global reference \hat{z}_0 axis. Dominant terms in slow dynamic is due to self weight of arm links:

$$\left(\begin{array}{c} c_2 g m_2 p_{c2x} + c_{23} g m_3 p_{c3x} - g m_4 p_{c4y} s_{234} + a_3 c_2 g m_3 + a_3 c_2 g m_4 + a_4 c_{23} g m_4 \\ c_{23} g m_3 p_{c3x} - g m_4 p_{c4y} s_{234} + a_4 c_{23} g m_4 \\ -g m_4 p_{c4y} s_{234} \\ 0 \\ 0 \end{array} \right)'$$

neglecting contributes of mass m_6 and m_5 expression reduces to rather easy expressions

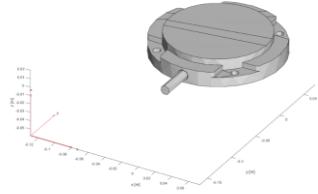
$$\left(\begin{array}{c} 0 \\ c_2 g m_2 p_{c2x} + c_{23} g m_3 p_{c3x} - g m_4 p_{c4y} s_{234} + a_3 c_2 g m_3 + a_3 c_2 g m_4 + a_4 c_{23} g m_4 \\ c_{23} g m_3 p_{c3x} - g m_4 p_{c4y} s_{234} + a_4 c_{23} g m_4 \\ -g m_4 p_{c4y} s_{234} \\ 0 \\ 0 \end{array} \right)'$$

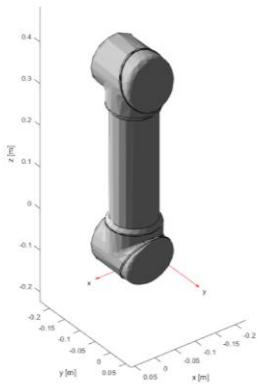
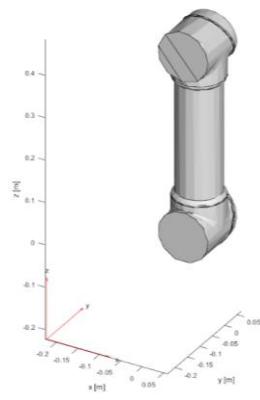
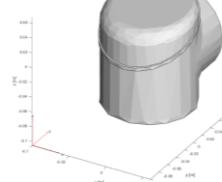
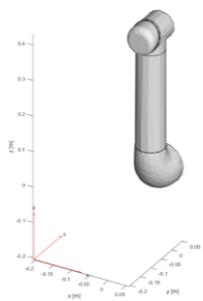
Consequently even in static condition and without payload joints are solicited with link's self weight loads, depending on robot configuration. Consequently every joint is solicited with a base torque for keeping robot configuration. These loads shall be included in actuators design, in addition to the consideration that loads on end effectors and generally of higher index links causes higher torque loads on lower index joints. Indeed shoulder and elbow actuators are designed for higher torques level.

Gravitational torques are enough for identification of part of inertial parameters relevant for dynamics, and since robots inertia are negligible, for a rough identification of robot, without requiring specific trajectories design and experiment, since can be performed even during operative phase in unloaded conditions. Consequently also payload mass can be evaluated in static conditions during operations, subtracting self weight contribute.

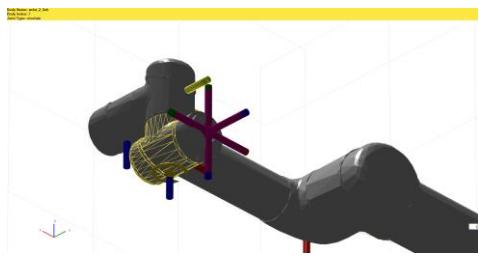
UR5 robot geometrical model parts

We present here below a decomposition of robot geometry as described into Robotic Matlab toolbox.





UR5 frames description

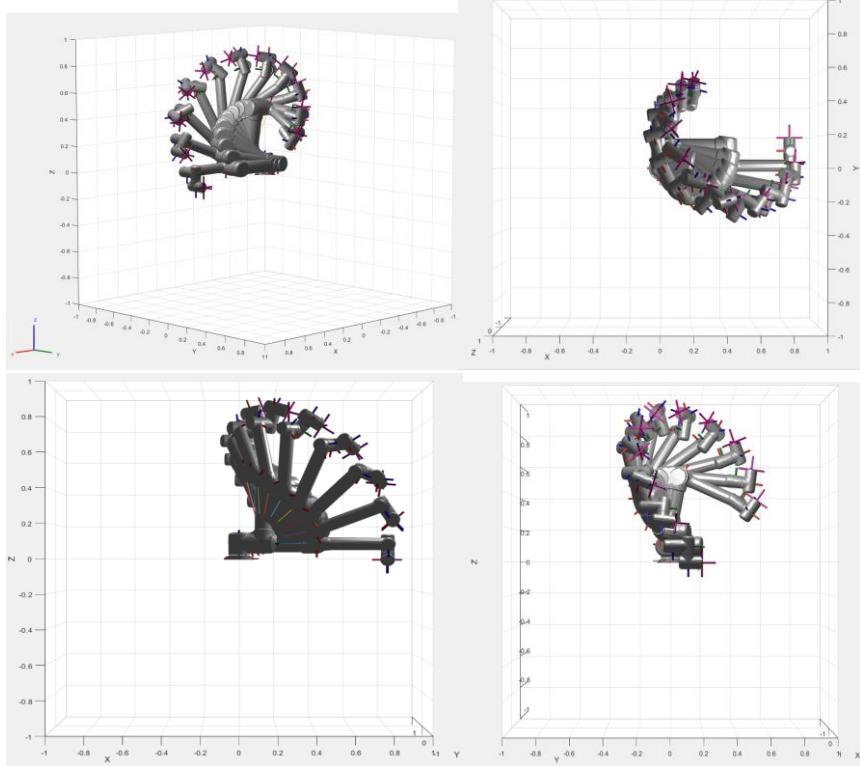


UR5 robot inertial properties

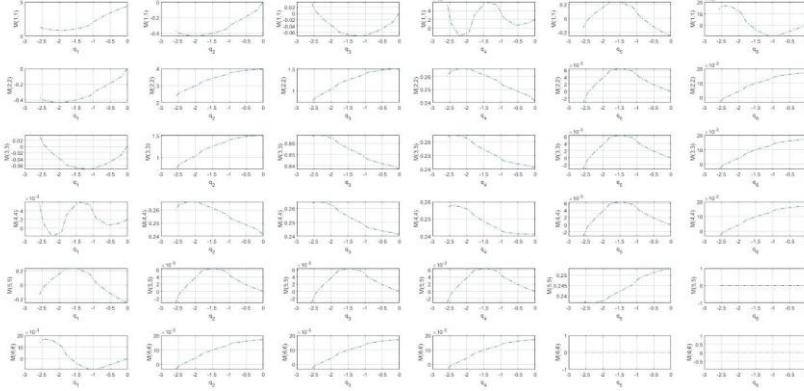
Mass matrix of robot with numerical simulation

We've seen challenging task of analytical interpretation of inertia matrix. Alternative to symbolic expression we can use numerical inspection. Using Matlab Robotic Toolbox we are able to numerically derive most of interest quantities used in robot kinematics and dynamics.

- Gravitational torque
- Mass matrix
- Coriolis and centrifugal terms matrix
- External forces
- geometric jacobians



Resulting mass matrix over trajectory coordinates (and time) is depicted below. We immediately remark what's obvious: matrix elements are not constant but depend on configuration assumed from the robot along trajectory. Some elements relative to latter joints are essentially unloaded due to lack of significative link inertia. Obviously is symmetric. But some terms are essentially negligible. The closer to the root the larger values as intuitive to expect mostly. We do not observe a full decoupling of first joint respect others because of the presence of offsets respect more abstract model of articulated manipulator.



Reduced order fitting of UR5 planned trajectories

A simplified least square method

Usually when writing least squares problems, we need to stack systems of equations blocks for each observation sample, in order to build up observation matrix. After that is also possible to apply a column and/or row reordering by means of a permutation matrix, without changing the overall problem nature (rank, determinant, conditioning number do not change for this kind of transformations), neither results. Nevertheless this operation allow to put in evidence the structure of regressor when present like mentioned for robot inertial parameters identification, where a block structure is present.

When equations is just a single one the process of building up observation matrix is by far easier and straightforward, since we just need put in a row all column vector of partials relative to all observation samples. This last also for non linear case. Implementation of this kind of least squares is way easier as well as interpretation. When dealing with a model contemplating multiple outputs that means splitting outputs and fit them independently, that's a different type of problem respect original one. Indeed:

- Parameters set used may be different (but this is not limitative since we can prove using a parameter collecting all, at price of a larger regressor with rank deficiency, would be equivalent)
- Equivalent to take a subset of observations and parameters
- Optimal results may differ, since observation set are different also, then constraint of using for all observations same parameters value is left out, allowing for easier accommodation of model

On the other side this fitting is easier and allow to get some preliminary estimate of parameters that are anyway likely to approximate full model. In that sense this approach can be considered a model order reduction.

Order of magnitude discrimination

Usually in practice and in literature no separation is made between different identification affecting factors in terms of magnitude, although this is of fundamental importance in purpose of understanding and behavior of the system.

When dealing with collaborative robotics, for safety reasons, usually only limited dynamics trajectories are of effective interest, since high dynamics of many industrial robots are not applicable to a collaborative case, and some protection, like cages, or separate environment are required. Even in case of collaborative robotics are required some measure to prevent hurting caused by undesired human-robot contact and forcing.

Resulting trajectories are characterized by limited values of dynamics, making inertial loads of secondary order of magnitude respect gravitational load, even when payloads are present.

That means in spite of traditional inertial analysis (whose intention is anyway differently motivated, like verification of robot characteristics), its value for practical purpose is limited, since a static model or quasi static model would be enough for having a first order idea of loads applied or to be applied for

Dynamic lead loads are of lower magnitude and limited to levels comparable in some cases to friction loads. Their effects are mostly relevant where accelerations are higher, namely at beginning and end of a trajectory.

Finally, although in collaborative robotics joint speed is not high enough to create significative loads due to viscous friction, coulomb friction may be still significative especially for precision application, leading to the need to keep it as low as possible. Consequently we repute friction loads should be relegated to a third order of magnitude.

Trajectory independence of observations

Is worthy to note that trajectory imposition is not demanded from least squares, since observations are not required to be complete, sequential, but just sufficient and independent and distributed well enough in regressor space to allow a proper estimate of parameters.

That means we could use for estimate even samples coming from different trajectories provided parameters have all same or similar values, as well they do not require to rely on same model or observation type. The use of a trajectory is just a matter of practical application. This is easiest approach, and some care should be used when adopting different sets of observations for having similar conditions to justify their use in same set of sed data. Indeed even in a single trajectory or acquisition some secondary conditions could potentially change affecting results.

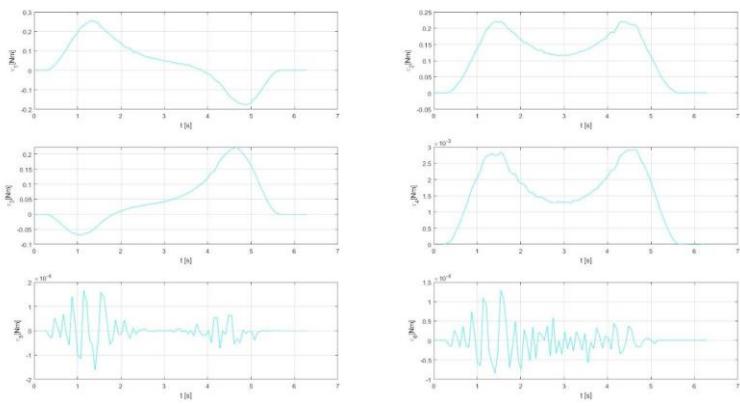
Friction model and contributes

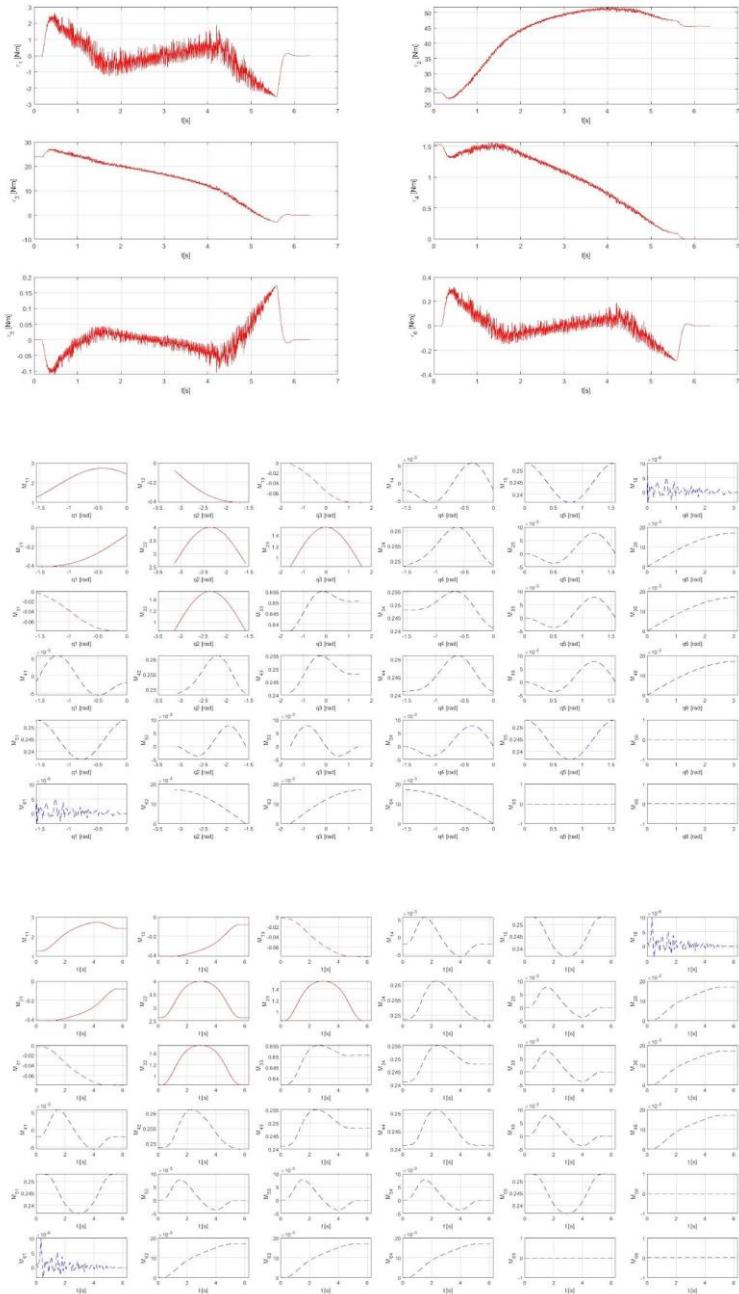
From data analysis we get some evidence of presence of friction factors:

- Coulomb friction (due to relative motion)
- Viscous terms friction

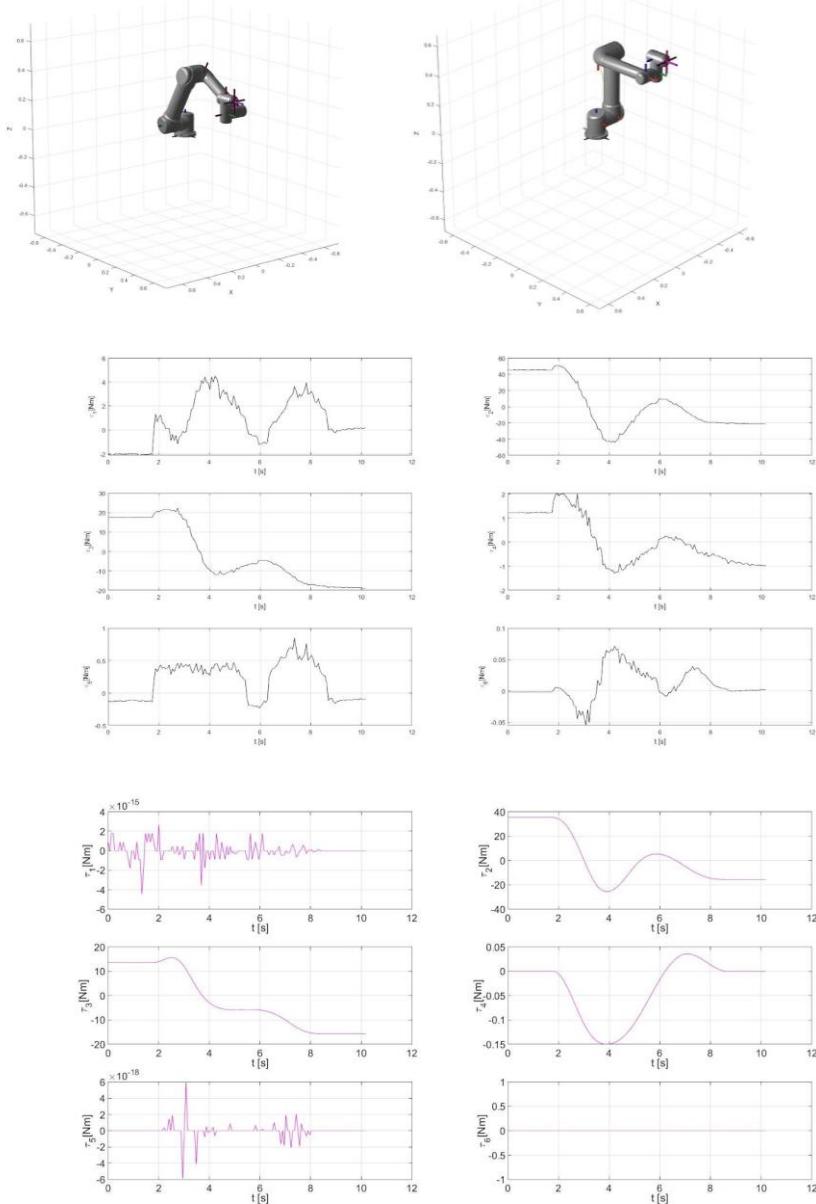
Study trajectories

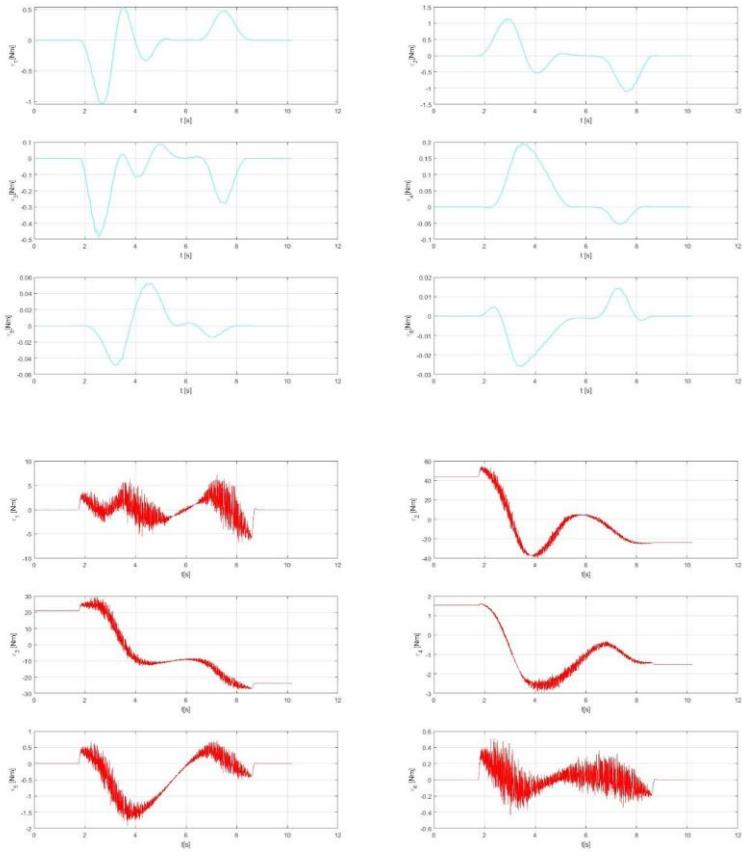
Three study trajectories have been studied, named A, B and C. Hereafter are reported analysis performed on data to derive preliminary results and fittings.

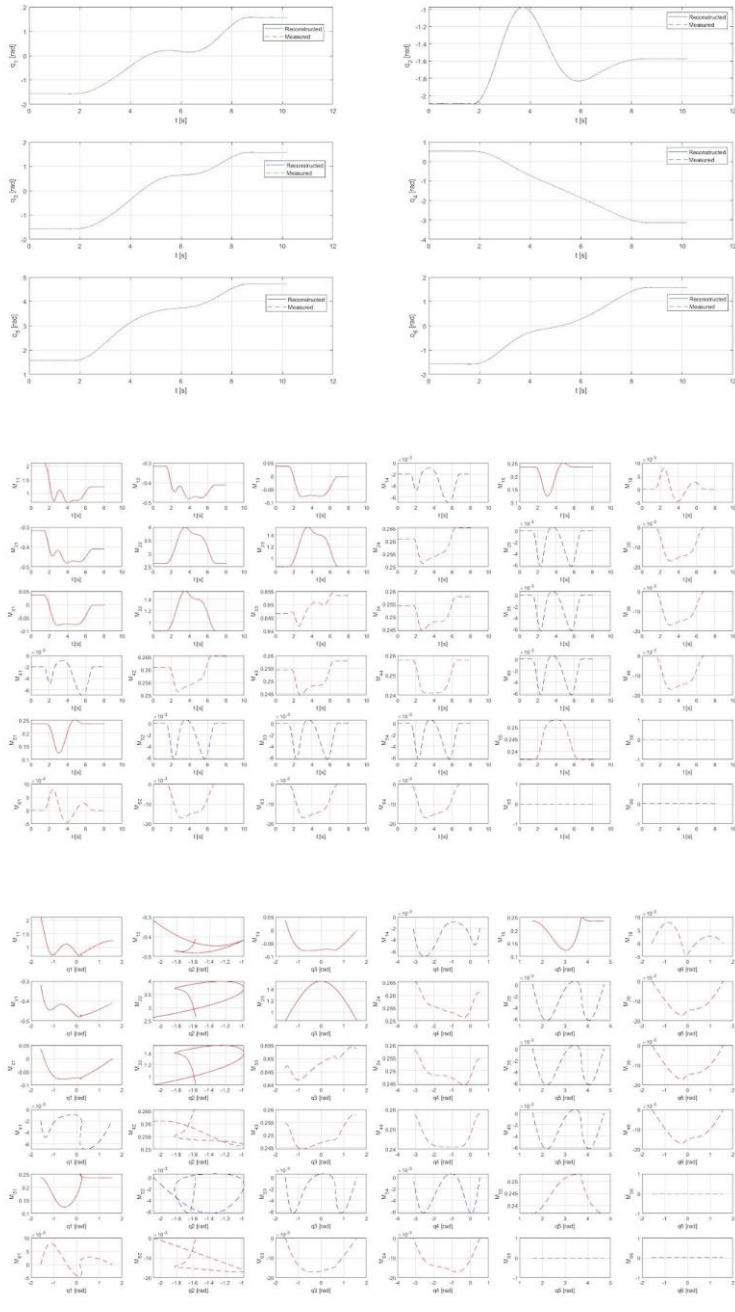


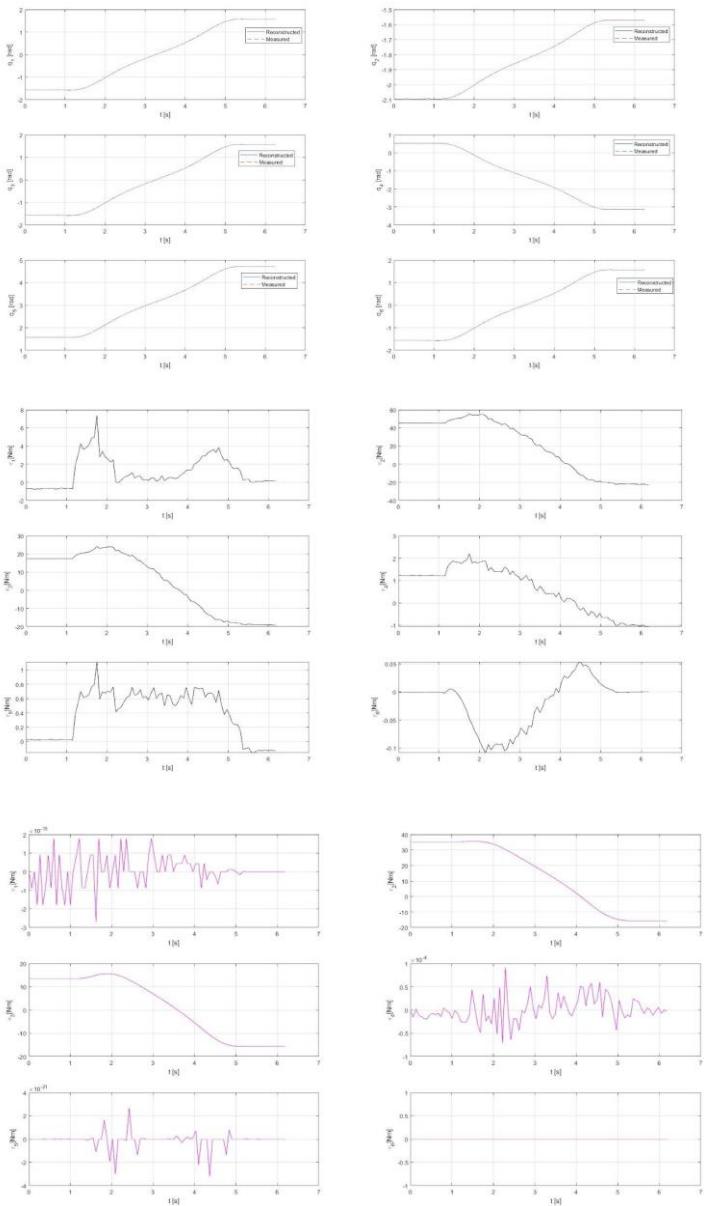


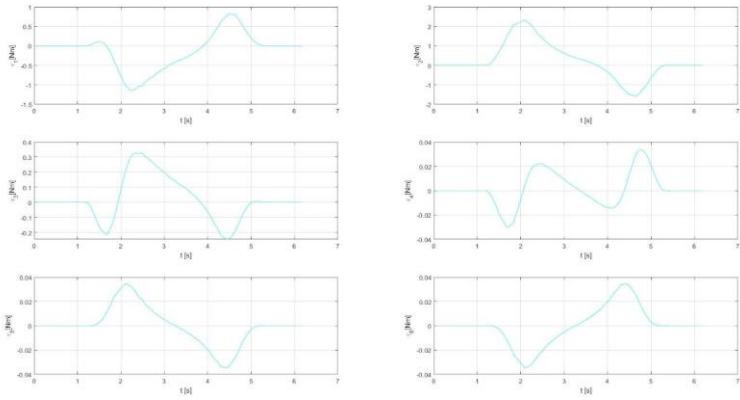
Initial and final robot configurations are depicted in the following by means of the support of Matlab robotic toolbox

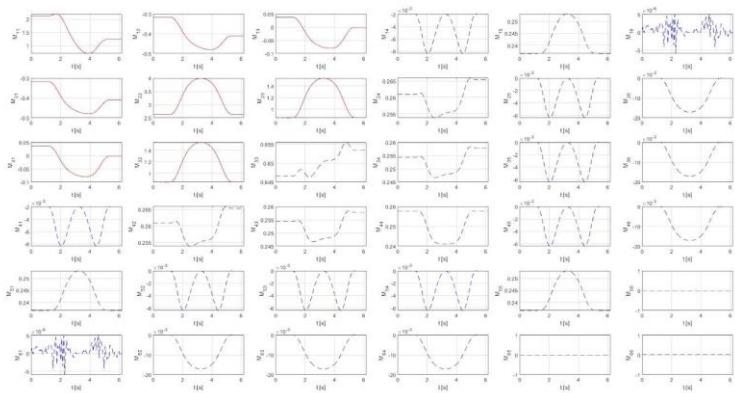












Base inertial parameters for UR5

In literature different approaches have been used for the identification of the robot UR5, leading to rather similar (or linearly related) set of base parameters, as should be. Base parameters start from a full set of 60 standard inertial parameters that are reduced and regrouped. Using formula for base parameters we've

But usually significative simplifications comes from assumptions on links inertia and center of mass positions. Unfortunately axis orientation in this case do not lead to simplifications.

A first simplified model more explainable is related to a reduced and simplified model taking into account most relevant dynamic contributes those of first 3 degree of freedom. Second group of non-spherical wrist joints named BSE is neglected at first, since expected lower impact into dynamics loads). Leading to:

$$p_{BW} = \left\{ \begin{array}{l} 0.2m_6p_{6G_6z} - 0.2m_5p_{5G_5y} \\ m_5p_{5G_5z} \\ m_5p_{5G_5z} + m_6p_{6G_6z} \\ J_{5xy} \\ J_{5xz} \\ 0.2m_6p_{5G_5z} \\ J_{5yz} \\ J_{5zz} \\ m_6p_{6G_6x} \\ m_6p_{6G_6y} \\ J_{6xx} - J_{6yy} \\ J_{6xy} \\ J_{6xz} \\ J_{6yz} \\ J_{6zz} \end{array} \right\}$$

$$p_{B_W} = \begin{pmatrix} 0.2m_6p_{6G_{6z}} - 0.2m_5p_{5G_{5y}} \\ m_5p_{5G_{5z}} \\ m_5p_{5G_{5z}} + m_6p_{6G_{6z}} \\ J_{5_{xy}} \\ J_{5_{xx}} \\ 0.2m_6p_{6G_{6z}} \\ J_{5_{yz}} \\ J_{5_{zz}} \\ m_6p_{6G_{6x}} \\ m_6p_{6G_{6y}} \\ J_{6_{xx}} - J_{6_{yy}} \\ J_{6_{xy}} \\ J_{6_{xz}} \\ J_{6_{yz}} \\ J_{6_{zz}} \end{pmatrix}$$

A more complete work suggest, as result of QR decomposition of a random regressor:

$$\mathbf{p}_B = \left(\begin{array}{l} i_{G,1}^2 C_{M,1} + C_1 + C_2 + C_3 + C_4 + 0.01285 m_2 + \\ + 0.01191 m_5 + 0.01191 m_6 \\ m_2 \rho_{S_{z2}} \\ A_2 - C_2 \\ i_{G,2}^2 C_{M,2} + B_2 \\ m_3 + m_4 + m_5 + m_6 \\ 0.3922 m_4 + 0.3922 m_5 + 0.3922 m_6 + m_3 \rho_{S_{z3}} \\ A_3 - C_3 + 0.1539 m_4 + 0.1539 m_5 + 0.1539 m_6 \\ B_3 + 0.1539 m_4 + 0.1539 m_5 + 0.1539 m_6 \\ 0.1092 m_5 + 0.1092 m_6 + m_4 \rho_{S_{y4}} \\ A_4 + B_5 - C_4 + 0.008959 m_6 \\ B_4 + B_5 + 0.008959 m_6 \\ 0.09465 m_6 + m_5 \rho_{S_{z5}} \\ A_5 - B_5 + C_6 \\ C_5 + C_6 \\ m_6 \rho_{S_{y6}} \\ A_6 - C_6 \\ B_6 \\ i_{G,3}^2 C_{M,3} \\ i_{G,4}^2 C_{M,4} \\ i_{G,5}^2 C_{M,5} \\ i_{G,6}^2 C_{M,6} \\ r_{v_1} \\ \vdots \\ r_{e_6} \end{array} \right)$$

A further approximation comes from neglecting links inertia that are less significant respect mass effect terms. First we can consider each link inertia tensor as diagonal, eliminating centrifugal products. Second we can consider link center of mass offset only in x and y directions as relevant. This way we can eliminate many superfluous parameters.

Regressor structure

Full regressor structure as mentioned is triangular for a serial open chain unbranched manipulator. It's full expression is by far excessively long and complex for a paper reporting an almost no work attempt to report its full form.

a better insight into robot physics allow a simplified (although not correct) regressor matrix structure, leading to a decoupled structure, where shoulder triad (BSE) and wrist (W) coordinates results, as expected decoupled (for physical reasons)

UR5 robot and drive train parameters

Appendix A. UR5 Robot and Drive Train Parameters

Table A1. Mass properties and Denavit-Hartenberg Parameters of the UR5 [36].

Link/Joint	Mass in kg	Center of Mass in m	θ in rad
1	3.7	[0, -0.02561, 0.00193]	0
2	8.393	[0.2125, 0, 0.011336]	0
3	2.33	[0.15, 0, 0.0265]	0
4	1.219	[0, -0.0018, 0.01634]	0
5	1.219	[0, 0.0018, 0.01634]	0
6	0.1879	[0, 0, 0.001159]	0
Link/Joint	a in m	d in m	α in rad
1	0	0.089159	$\pi/2$
2	-0.425	0	0
3	-0.39225	0	0
4	0	0.10915	$\pi/2$
5	0	0.09465	$-\pi/2$
6	0	0.823	0

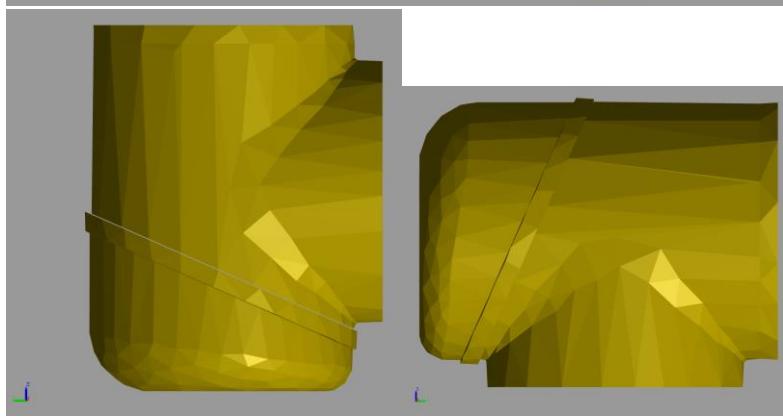
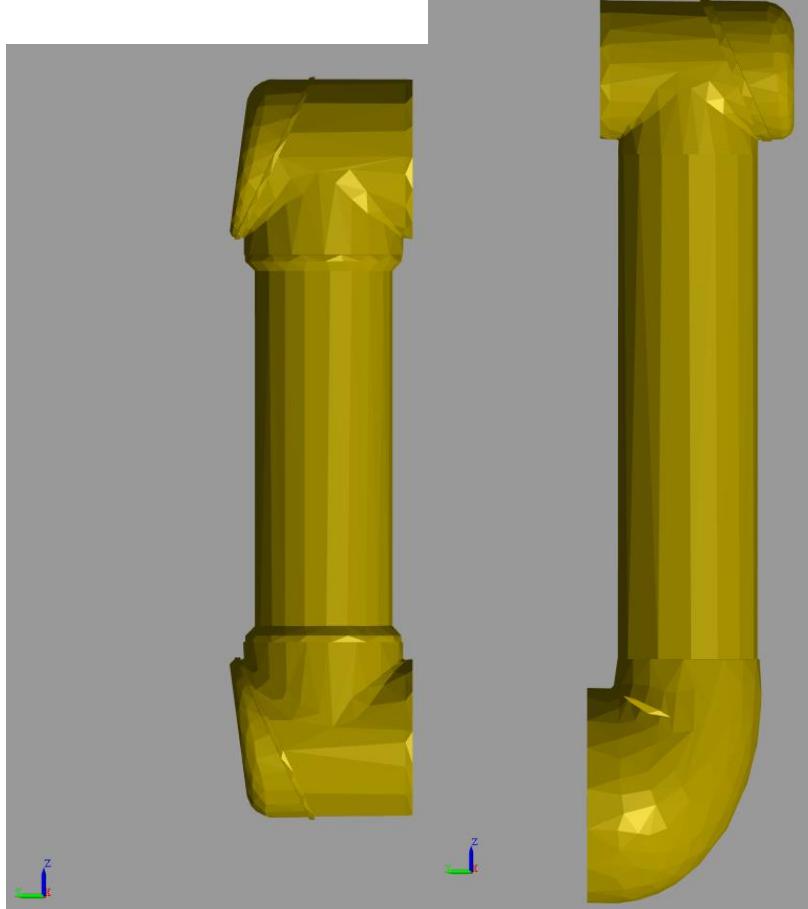
Table A2. Properties of the reducer, based on the family HFUS-2SH [18].

Joints	Size	i_g	Mass in kg	Inertia in kg m^2
1, 2, 3	25	100	1.44	1.07×10^{-4}
4, 5, 6	14	100	0.45	9.1×10^{-6}
Joints	L_r in h	\dot{q}_r in rad/s	u_r in N m	c
1, 2, 3	3.5×10^4	1.152	67	3
4, 5, 6	3.5×10^4	1.152	7.8	3

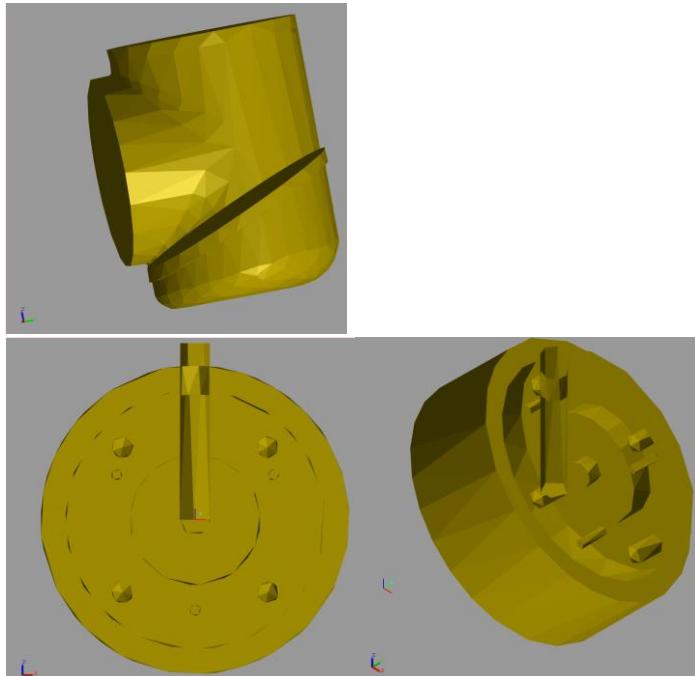
Table A3. Properties of the motor with friction brake, based on the RDU family based on [37].

Joints	Size	Mass in kg (Estimated)	Inertia in kg m^2	k_m in N m/A
1, 2, 3	70 × 18-HW	0.71	9×10^{-5}	0.106
4, 5, 6	50 × 08-HW	0.24	2.5×10^{-5}	0.057
Joints	Size	k_b in Vs/rad	L in H	R in Ω
1, 2, 3	70 × 18-HW	0.106	7.2×10^{-4}	0.552
4, 5, 6	50 × 08-HW	0.057	8×10^{-4}	0.47

Simscape model of UR5



Wrist joints detailed view



Commented [43]: Eliminate

Study Case III: Workflow automation and future perspectives

Automation of design and use of robot models provide significative advantages for users, saving much of work required for modeling and instrumenting a model for purposed use.

Using tools provided with matlab and mathworks is possible easily to setup an automated workflow for analyzing even complex systems like humanoid robots with dof up to some tens, and simulating their dynamics imposing a desired motion and recording resulting actuators required forces, or vice versa, imposing a set of solicitations getting resulting motions (Direct dynamics problem).

This can be done for an arbitrary robot either already present in the Simulink® toolbox, but also for any robot build at user like.

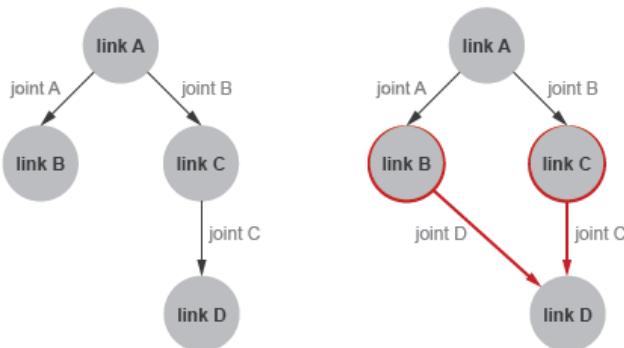
Aforementioned workflow is described in the following

Building kinematic structures and associated attributes

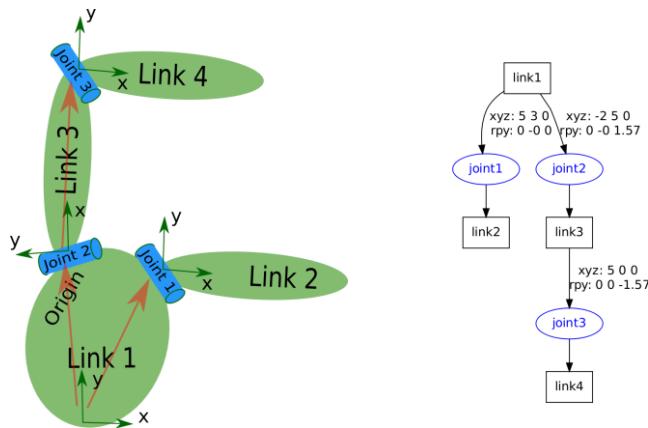
URDF file description

URDF files is common exchange format very commonly found in many applications and in user community interested in robotics. *Unified Robotics Description Format, URDF*, is an XML specification used in academia and industry to model *multibody systems* such as robotic manipulator arms for manufacturing assembly lines and animatronic robots for amusement parks. URDF is especially popular with users of Robotics Operating System, ROS. For example of URDF field see section [UR5 URDF File](#). The use across different platform of a common exchange format is fundamental for interoperability between different environments. URDF files contains informations about elements with respective attributes for basic robot objects: <robot>, <links>, <joints>. Like other types of XML files, URDF files comprise various XML elements, such as <robot>, <link>, <joint>, nested in *hierarchical* structures known as XML trees. The <link> and <joint> elements are said to be *children* of the <robot> element and, reciprocally, the <robot> element the parent of the <link> and <joint> elements. Tree structures and hierarchical structure naturally provide topological connection for a kinematic tree, without need of additional informations, but is not suitable for natural description of a generic mechanism containing kinematic loops. Parallel robots and closed loop mechanisms are then left out. A similar issue is presented by joints. URDF links connect through *joints* in hierarchical structures not unlike those formed by nesting XML elements in a URDF file. <joint> elements *enforce* these hierarchies through <parent> and <child> elements that identify one link as the parent and the other as the child. Parent links can themselves be children and child links parents of other links in the model.

Model topology is important in URDF. The connectivity graph of a model can take the shape only of a kinematic tree a kinematic chain, branched or unbranched, that is always open. Kinematic loops, each a closed chain formed by joining the ends of an otherwise open chain, are disallowed. This restriction impacts how <link> elements can connect in a URDF model. The restriction translates to the following rule: no <link> element can serve as a child node in more than one <joint> element. Put another way, no <link> element can have more than one parent element in the model's connectivity graph. Only the root link, that at the origin of the connectivity graph, can have a number of parent nodes different from one (zero). Only one root link is allowed in a model.



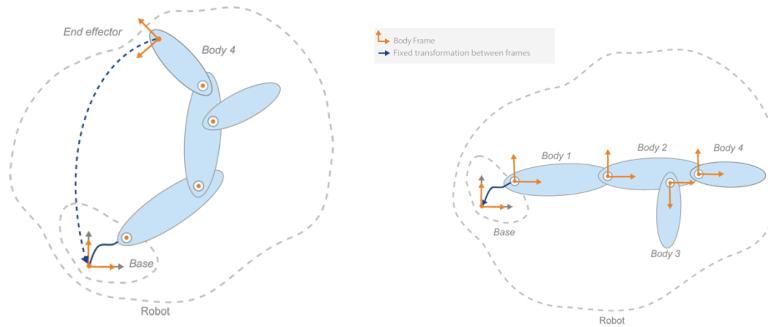
Child elements, such as `<link>` and `<joint>` under `<robot>`, can in turn have child elements of their own. For example, the `<link>` element has the child elements `<inertial>` and `<visual>`, `<collision>`, `<sensors>`. The `<visual>` element has the child elements `<geometry>` and `<material>`. And the `<material>` element has the child element `<color>`. Such chains of child elements are essential to define the properties and behavior of the parent elements. In addition to child elements, the XML elements in a URDF model can have attributes. For example, the `<robot>`, `<link>`, and `<joint>` elements all have the attribute `<name>`—a string that serves to identify the element. The `<color>` element has the attribute `rgba`—a numeric array with the red, green, blue, and alpha (or opacity) values of the link color. Attributes such as these help to completely define the elements in the model.



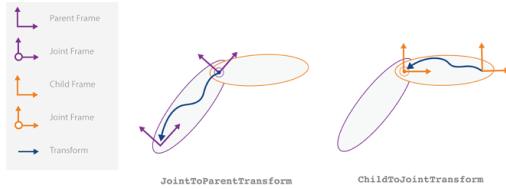
This description closely follows robotics conventions and is compatible with Robotic toolbox description as well as with Simscape multibody description.

Kinematic tree description used in Matlab Robotic-toolbox

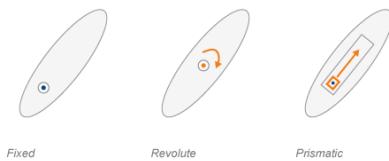
A similar description of robot kinematic tree structure is given in Matlab robotic toolbox



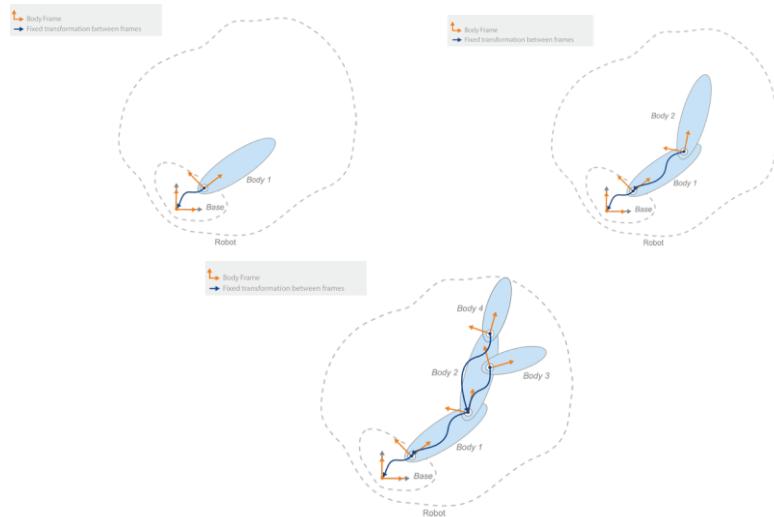
Each body has an associated body frame, and transformation between frames is managed by transformation between joint-to-parent and child-to-joint.



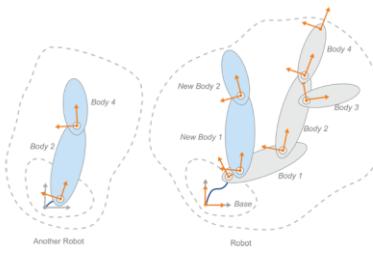
Allowed joints are floating (6dof), revolute (1dof), prismatic (1 dof) and fixed (0dof).



A kinematic tree robot can be built up step by step using a recursive process associating body to body by means of joints, starting from a base (frame) treated as a fixed body. The process allow to add more than one body to a same parent body, leading to branched tree robots (kinematic tree).



Basic elements allow for assembling up a kinematic tree from existing tree, branched or unbranched (robots), establishing proper body-body connections, attaching a base to a any parent body. The specified body name acts as a base for attachment, and all transformations on the subtree are relative to that body frame. Before you add the subtree, you must ensure all the names of bodies and joints are unique.



More details are given in [rigid body tree robot model](#).

Once kinematic structure is defined, respective bodies and joints properties shall be specified. For bodies shall be specified:

- joints associated
- mass
- inertia vector
- center of mass position
- parent and children
- visual properties
- collision geometry

For joints shall be specified

- joint type
- transformations

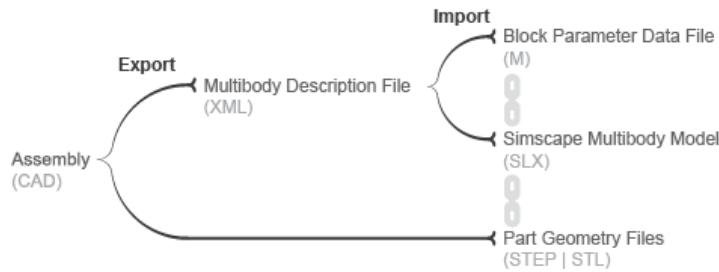
Some exercises have been provided in order of illustrating potential of automatize building process of a robot using mentioned toolset.

Parsing URDF files

Many free parser for URDF files are available online, under different environment.

Generation of a URDF file

Generally there's no need to manually create your own URDF files. For more complex models, it can be preferable to obtain URDF files from other sources. Robotics manufacturers and consultants often provide URDF models for their robotic systems. CAD applications such as SolidWorks® and PTC® Creo™ support URDF exporters that convert your CAD assemblies into URDF models. Consider these options when working with complex robotics models that may not be simple to create manually.



In matlab environment is possible either to generate an URDF file from a desired robot configuration, or generating respective URDF file describing created robot. It is worthy to note an URDF file could contain much more informations respect simple robot description, as outlined in the previous paragraph.

Many sources provide URDF files for free. See resources section in [Bibliography](#) for details.

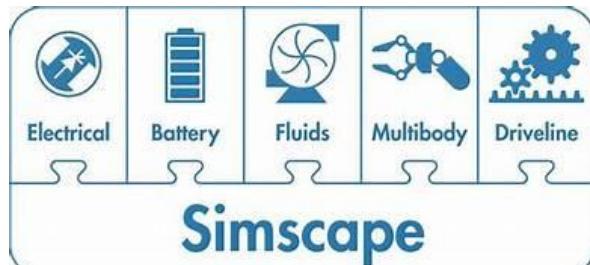
Conversion of URDF files to simscape and automatic instrumenting

You can import URDF models into the Simscape™ Multibody™ environment using [smimport](#) function to create a multibody dynamics model for simulation, analysis, or control design tasks.

In matlab environment is possible to convert directly a robot structure into a simscape model, or taking an URDF file convert it to a simscape model, and programmatically or manually instrument and simulate it or even change it at your desire.

This approach is very practical in sight of a workflow process automation reducing workload and risk of erroneous operations.

SIMSCAPE multiphysics



SIMSCAPE is a toolset allowing for modeling and simulation of multiphysics systems, including multibody, electrical, thermal and fluid behavior. Consequently it is then a perfect environment for the development of a digital twin of a robotic system, considering possible extensions for multi physics simulation and analysis.

Automatic generation of tree like models

A script function has been written for generating arbitrary tree models, given as input

- Number of bodies
- Branch factor (1 for serial robots): average number of childs for each link
- Skew angle (joint axis)

This function make it easier to generate a complete rigid body tree model from few specified parameters, since we use symbolic and parametric factors for its definition.

This function can be used for *explorative* studies on different sample models.

Output model can be used as source for calculation of dynamic model using a Newton-Euler script (customized for dealing trees), as well for direct generation of inertia matrix.

Dynamic model for complex robots may be excessive as computational and memory burden, so for managing general case a Matlab script is generated in place of explicit closed form expression. This script can be used as function or script for any purpose in substantially equivalent way. In addition intermediate quantities are calculated and can be used.

Dynamic model manipulation

Once dynamic model is available it can be manipulated in many ways, for different purposes

- Naive calculator of inertia matrix, as symbolic matrix
- Naive calculation of regressor model respect a desired set
- Specialized model for specific values of model parameters
- Integration with additional forces (for example friction)
- other?

Identification of model

A set of functions were developed for automatize identification process once dynamic equations model is given calculated, automatically.

Dynamic model can be built basing on kinematic model description, automatically using RNEA or CRBA method, using symbolic standard inertial parameter and including other dynamic effects like friction models (even of higher order but still linear).

Since for identification purpose we need to assign values to identified parameters (symbolic identification is a speculative perspective not having capability for symbol-parametrised simulations in Simulink® environment). Used parameters are set randomly, in order to make automatic process of estimation, and do not provide physically meaningful values, because they're for demonstrative purpose. Parameters values are instantiated on symbolic model by means of a substitution process.

Trajectory is automatically designed using a simple standard input harmonic solicitations, chosen a priori (but can be tined by user attempt).

A set of input trajectories are generated automatically using random values for amplitude, using an harmonic motion imposed, with a number of pulsations specified at pleasure, first symbolically and later numerically evaluated on arbitrary vector of sample time.

Dynamic model is then evaluated for each sample time giving respective set of joint coordinates, speeds and accelerations by substitution and evaluation. This open also opportunity of keeping some free parameter keeping model parametrized.

For identification we need to have numerical values of output joint actuation, to be downsampled as observations vectors.

Noise can be added artificially to simulated observations.

Regressor model in symbolical form is built up as jacobian (or differentiating) respect selected parameters vector, or alternatively using naive evaluation.

Resulting regressor model can be checked for rank deficiency using numerical QR decomposition method. Resulting parameters vector to be used in estimate is in symbolic or numerical form.

Given trajectories can be used directly on analytical model or in simscape model for providing simulated output to use for identification. At current state for lack of time a check of dynamic model coherence in different environments was not proved, limiting the check to closure of dynamic model in symbolic-analytical way, for checking identification process in a basilar case less error prone, for generic kinematic structure.

Regressor instantiated values then calculated by substitution in a similar way and staked up.

Once observations and regression matrix are available in numerical form parameters are estimated and checked respect expected values.

This workflow is designed in a way allowing for managing an arbitrary robot structure, but its effective applicability on real case is still to be explored.

In a future it is expected to become a full automated or semiautomated process for identification but also for other purposes.

Reuse of automated process for other purposes

The automated process can be naturally be used not only for identification purposes but also for other uses.

Case study: importing and simulation of Valkyrie humanoid robot

Using automated approach is possible to load and simulate in near real time even a complex rigid body architecture as NASA Valkyrie humanoid robot. Corresponding system is mounted of 88



Resulting model considered other robots, is substantially more complex, but not exaggeratedly. It can be instrumented, compiled and simulated in some minutes, substantially more than traditional robots, but still a reasonable time we can deal with.

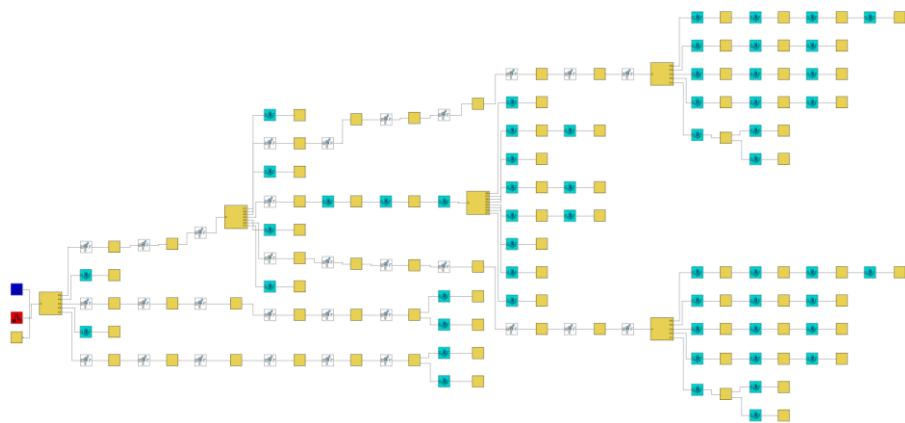
Peculiarities of humanoid robot

Humanoid robots identification treatment case is mostly similar to traditional identification problems in spite of the fact that usually it needs to be treated as a floating base systems.

Model conversion from URDF into Simscape environment

A complex humanoid robot model from URDF file has been used as stress case for automation process of translation and instrumentations. Demonstrating that the method would be in principle viable and usable for purposes from control, direct and inverse kinematics as well for identification.

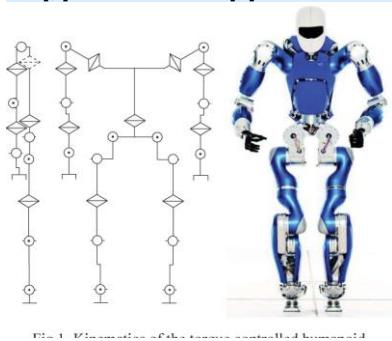
Resulting simscape imported model is depicted in following, and has been run effectively in a reasonable (although not too short) amount of time, with a default input planned motion in inverse kinematic mode.



The use of simscape models (digital twins) allows for create a simulation environment for experiment virtualization cannot be afforded using a physical hardware setup because of lack of a target for experiment as well of surrounding experimental setup hardware.

Inertial parameters identification in humanoid robots

Methods for humanoid robots identification and base parameters definition could differ from classic robot methods, because also identification method can be different. Usually camera based motion tracking is used in combination with gyro measurement and contact forces. (with force plates). Indeed in place of dynamic equations set a mobile base equations set shall be used, and usually base motion equation, lead to simpler identification, and resulting inertial parameter used for identification results to be substantially different. See for details [Jovic et al. 15], [Jovic et al. 16], [Ayusawa et al. 08a], [Ayusawa et al. 08b], [Ayusawa et al. 14], [Iwasaki et al. 12], [Nakamura et al. 00], [Ogawa et al. 14], [Venture et al. 08a], [Venture et al. 08b], [Venture et al. 09], [Baelemans et al. 13].



Commented [44]: Da eliminare

Future works and developments

Conclusions

Across this work I'd opportunity to get in depth of some of modern subjects of robotics collecting extensive bibliographic resources and references worthy in themselves as state of the art, after comparison of numerous among textbooks and papers among most referred, constituting a little personal treasure of knowledge I hope to have opportunity to cultivate further in future. Through this work we've proved the effectiveness of application of current commercial tools for multibody and robotic modeling and simulation and control design in order to develop a robot digital twin. Simscape modeling supported by symbolic modeling and Matlab/Simulink environment for the development of robots digital twins offering opportunity of extension to a variety of different physical aspects modeling. Robotic systems present advantages with respect to other environments, since requirements of precise modeling and twinning are from one side easier to realize, deeply studied in literature and effectively applicable for control purposes as well for identification. Customer support have been proved efficiency and care in answering question posed and accepting suggestion proposed. By the way tools have also shown some margin for improvement, especially for some robotic specific need, not offering yet explicitly designed toolsets.

Adopted environment allows for geometrical and physical modeling natural integration by means of present primitives and library blocks, as well as customization are possible with rather good ease by means of Simulink® development environment as well as of Matlab

environment and scripting support in combination with symbolic modeling. Would be nice to have a more straightforward symbolic modeling connection in place of code generation, for study purpose and for academic research purpose. Application of symbolic modeling is potentially interesting for traditional applications like dynamic model derivation, computed torque law, model based identification, but also other applications can benefit. Different approaches have been proved, and direct mass matrix calculator using geometrical methods or CRBA method have shown some advantages respect traditional RNEA method. Experimentation considered also high dof cases, showing the need for the use of intermediate variable adoption, leading to semi-implicit models, as unique viable way to get a dynamic model.

Identification processes have been proven offline and simulated online as prototypes on digital twin target as development process. Lack of resources for a real system application leaves experimental integration for future work.

Acknowledgement

I extend my deepest gratitude to my supervisor, Professor Paolo Boscaroli, for her unwavering support and insightful critiques throughout my research journey. Her deep commitment to academic excellence and meticulous attention to detail have significantly shaped this dissertation. Once more I want to express gratitude for the freedom conceded in following my curiosity and deepening direction I considered worthwhile for my personal enrichment.

I am equally thankful to the members of my thesis committee for their constructive feedback and essential suggestions that enhanced the quality of my work.

My appreciation also goes to the faculty and staff in the Department of Management Engineering at University of Padova, whose resources and assistance have been invaluable. I would also like to acknowledge my peers for their camaraderie and the stimulating discussions that inspired me throughout my academic journey. Their collective wisdom and encouragement have been a cornerstone of my research experience.

Finally, my sincere thanks to the technical staff whose expertise in managing laboratory equipment was crucial for my experiments. Their patience and readiness to assist at all times have left a profound impact on the completion of my project.

Bibliography

Books

[Principles of Dynamics] D.T. Greenwood: Principles of Dynamics (PrenticeHall, Englewood Cliffs 1988)

[Dynamics of Multibody Systems] R.E. Roberson, R. Schwertassek: Dynamics of Multibody Systems (Springer, Berlin, Heidelberg 1988)

[Handbook of robotics] B. Siciliano, O. Khatib, Handbook of Robotics, (Springer, Berlin, Heidelberg 2008)

[A Mathematical Introduction to Robotic Manipulation] R.M. Murray, Z. Li, S.S. Sastry: A Mathematical Introduction to Robotic Manipulation (CRC, Boca Raton 1994)

[Introduction to Robotics: Mechanics and Control] J.J. Craig: Introduction to Robotics: Mechanics and Control (Addison-Wesley, Reading 1986)

[Rigid body dynamics algorithms] R. Featherstone: Rigid Body Dynamics Algorithms (Kluwer, Boston 2007)

[Robot Dynamics Algorithms] R. Featherstone: Robot Dynamics Algorithms (Kluwer, Boston 1987)

[Efficient Dynamic Simulation of Robotic Mechanisms] K.W. Lilly: Efficient Dynamic Simulation of Robotic Mechanisms (Kluwer, Boston 1993)

[Modeling and Control of Robot Manipulators] L. Sciavicco, B. Siciliano: Modeling and Control of Robot Manipulators (Springer, London 2000)

[Robot Force Control] B. Siciliano, L. Villani: Robot Force Control (Kluwer, Boston 1999)

[Applied Nonlinear Control] J.J..E. Slotine, W. Li: Applied Nonlinear Control (Prentice Hall, Englewood Cliffs 1991)

[Robot and Multibody Dynamics: Analysis and Algorithms] A. Jain, Robot and Multibody Dynamics: Analysis and Algorithms (Springer, New York 2011)

[Modeling, Identification and Control of Robots] W. Khalil, E. Dombre: Modeling, Identification and Control of Robots (Taylor Francis, New York 2002)

[Fundamentals of Robotic Mechanical Systems] J. Angeles: Fundamentals of Robotic Mechanical Systems, 2nd edn. (Springer, New York 2003)

[Applied Dynamics] F.C. Moon: Applied Dynamics (Wiley, New York 1998)

[Dynamics, Theory and Applications] T.R. Kane, D.A. Levinson: Dynamics, Theory and Applications (McGraw-Hill, New York 1985)

[Modern Robotics: Mechanics, Planning, and Control] K.M. Lynch., F.C. Park, Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press (2017).

[Spacecraft Dynamics] T.R. Kane, P. Likins, D. Levinson,: Spacecraft Dynamics. McGraw-Hill, New York (1983)

[Methods of Analytical Dynamics] L. Meirovitch., Methods of Analytical Dynamics. McGraw-Hill, New York (1970)

[] L. Meirovitch: Fundamentals of vibrations (McGraw-Hill, Boston 2001)

[] A. Isidori: Nonlinear Control Systems: An Introduction, Lecture Notes in Control and Information Sciences, Vol. 72 (Springer, New York 1985)

[Dynamic Programming] R. Bellman: Dynamic Programming (Princeton Univ. Press, Princeton 1957)

[Geometrical Fundamentals of Robotics] J. Selig,: Geometrical Fundamentals of Robotics. Springer, New York (2004)

[Course of Theoretical Physics] L.D. Landau, E.M.Lifshitz, Course of Theoretical Physics. Pergamon, Oxford University (1979)

[Robot Modeling and Control] M.W. Spong, S. Hutchinson, M. Vidyasagar: Robot Modeling and Control (Wiley, Hoboken 2006)

[Robot Control] M.W. Spong, F.L. Lewis, C.T. Abdallah (Eds.): Robot Control (IEEE, New York 1989)

[Applied Optimal Control] A.E. Bryson Jr., Y.C. Ho: Applied Optimal Control (Hemisphere, Washington 1975)

[Modelling and Identification in Robotics] K Kozlowski Modelling and Identification in Robotics Springer (1998)

[Dynamics of Tree-Type Robotic Systems] S.V. Shah, S.K. Saha, J. K. Dutt Dynamics of Tree-Type Robotic Systems. Springer (2018)

[Fondamenti di controlli automatici] P. Bolzern, R. Scattolini, N. Schiavoni Fondamenti di controlli automatici McGraw-Hill (2008)

[Optimization by Vector Space Methods] D.G. Luenberger: Optimization by Vector Space Methods (Wiley, New York 1969)

[Robot Analysis and Control] H. Asada, J.J.E. Slotine: Robot Analysis and Control (Wiley, New York 1986)

[Fundamentals of Robotics: Analysis and Control] R.J. Schilling: Fundamentals of Robotics: Analysis and Control (Prentice Hall, Englewood Cliffs 1990)

[Manipulators: Mathematics, Programming and Control] R. Paul: Robot Manipulators: Mathematics, Programming and Control (MIT Press, Cambridge 1982)

[Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach] C.A. Balafoutis, R.V. Patel: Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach (Kluwer, Boston 1991)

[Simulating and Generating Motions of Human Figures] K. Yamane: Simulating and Generating Motions of Human Figures (Springer, Berlin, Heidelberg 2004)

[Fundamentals of Multibody Dynamics: Theory and Applications] F.M.L. Amrouche: Fundamentals of Multibody Dynamics: Theory and Applications (Birkhäuser, Boston 2006)

[] M.G. Coutinho: Dynamic Simulations of Multibody Systems (Springer, New York 2001)

[] E.J. Haug: Computer Aided Kinematics and Dynamics of Mechanical Systems (Allyn and Bacon, Boston 1989)

[] R.L. Huston: Multibody Dynamics (Butterworths, Boston 1990)

[] C.H. An, C.H. Atkeson, J.M. Hollerbach: Model-Based Control of a Robot Manipulator (MIT Press, Cambridge 1988)

[] J.M. McCarthy: Introduction to Theoretical Kinematics (MIT Press, Cambridge 1990)

[] S. Sastry: Nonlinear Systems: Analysis, Stability, and Control (Springer, Berlin, Heidelberg 1999)

[] M. Vukobratovic, N. Kircanski: Real-Time Dynamics of Manipulation Robots (Springer, New York 1985)

[] S. Hirose: Biologically Inspired Robots: SnakeLike Locomotors and Manipulators (Oxford Univ. Press, New York 1993)

[] J. Duffy. Analysis of Mechanisms and Robot Manipulators. Edward Arnold Ltd., London, 1980.

[] M. Hall. The Theory of Groups. Macmillan, (1959).

[] L. C. Young. Lectures on the Calculus of Variations and Optimal Control Theory. Chelsea, New York, second edition, (1980).

[] J-P. Serre., W. A. Benjamin, Lie Algebras and Lie groups., New York, (1965).

[] R. M. Rosenberg. Analytical Dynamics of Discrete Systems. Plenum Press, New York, (1977)

[Numerical Methods for Least Squares Problems] A. Bjorck, Numerical Methods for Least Squares Problems; Siam: Philadelphia, PA, USA, (1996); Volume 51.

[Robot calibration] R Bernhardt, SL. Albright, Robot calibration, Chapman & Hall, London, UK (1993)

[Dynamic model identification for industrial robots] J. Swevers, W. Verdonck, and J. De Schutter, Dynamic model identification for industrial robots, IEEE Control Systems Mag., vol. 27, no. 5, pp. 58–71, 2007.

[Linear Systems Theory] W.J. Rugh Linear Systems Theory. 2nd edition. NJ: Prentice-Hall, Inc., Hoboken, (1996)

[] Da Forno R., Dal Corpo rigido al Robot con Matlab, McGraw-Hill, (1998).

[] L.W. Tsai: Robot Analysis and Design: The Mechanics of Serial and Parallel Manipulators (Wiley, New York 1999)

[] E. Otten, Inverse and forward dynamics: models of multi–body systems Philos. Trans. R. Soc. Lond. Ser. B (2003)

[Space Robotics: Dynamics and Control] Y. Xu, T. Kanade (Eds.): Space Robotics: Dynamics and Control (Kluwer, Boston 1993)

Thesis, Lectures and notes

[R. Tedrake Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines Course Notes for MIT 6.832 Massachusetts Institute of Technology](#)

[] K. Kufieta, Force Estimation in Robotic Manipulators: Modeling, Simulation and experiments (Diploma Thesis) (Department of Engineering Cybernetics NTNU Norwegian University of Science and Technology, 2014).

[] [Universal Robots \(2022\). Real-time data exchange \(RTDE\) guide](#) <https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/>

[] B. Paden. Kinematics and Control Robot Manipulators. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1986.

[Aubin 91] Aubin A., Modélisation, identification et commande du bras manipulateur TAM, Thèse de Doctorat, INPG, Grenoble, France, (1991).

[] A. Karger, J. Novak: Space Kinematics and Lie Groups (Routledge, New York 1985)

[Lagrangian dynamics] A. De Luca, Lectures on robotics, Dynamic model of robots: Analysis, properties, extensions, uses.

[Newton-Euler approach] A. De Luca, Lectures on robotics, Dynamic model of robots: Newton-Euler approach

[Robots with Flexible Joints] A. De Luca, Robots with Flexible Joints: Modeling and Control, 2023 International Graduate School on Control Course M16

[Robots with Flexible Links] A. De Luca, Robots with Flexible Links: Modeling and Control, 2023 International Graduate School on Control Course M16

[Linear parametrization and identification] A. De Luca, Lectures on robotics, Linear parametrization and identification of robot dynamics

[Modeling and control of robots with flexible joints]
<https://www.diag.uniroma1.it/deluca/flexiblejoints.html> (Site page)

[Jain lectures] <https://dartslab.jpl.nasa.gov/References/> (Jain lectures 2024)

Articles by thematic and author

Identification and dynamic parameters

Khalil W.

[Profile](#)

[Khalil et al. 96] W. Khalil, P. P. Restrepo , An efficient algorithm for the calculation of the filtered dynamic model of robots. Proceedings of IEEE International Conference on Robotics and Automation [10.1109/ROBOT.1996.503797](https://doi.org/10.1109/ROBOT.1996.503797)

Gautier-Khalil

[Profile Gautier](#)

[Gautier 97] M. Gautier: Dynamic identification of robots with power model, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1997) pp. 1922–1927

[Gautier 91] M. Gautier: Numerical calculation of the base inertial parameters, J. Robotics Syst. 8, 485–506 (1991)

[Khalil et al. 07] W. Khalil, M. Gautier, P. Lemoine: Identification of the payload inertial parameters of industrial manipulators, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2007) pp. 4943–4948

[] W. Khalil, F. Bennis: Symbolic calculation of the base inertial parameters of closed-loop robots, Int. J. Robotics Res. 14, 112–128 (1995)

[] M. Gautier, W. Khalil: Exciting trajectories for inertial parameter identification, Int. J. Robotics Res. 11, 362–375 (1992)

[] P.O. Vandajon, M. Gautier, P. Desbats: Identification of robots inertial parameters by means of spectrum analysis, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1995) pp. 3033–3038

[Gautier et al. 95] M. Gautier, W. Khalil, P. P. Restrepo, Identification of the dynamic parameters of a closed loop robot, Proc. IEEE Int. Conf. on Robotics and Automation, Nagoya, Japan, May 1995, p. 3045-3050.

[Gautier 96] M. Gautier, A comparison of filtered models for dynamic identification of robots, Proc. IEEE 35° Conf. on Decision and Control, Kobe, Japan, December 1996,p. 875-880.

[Gautier 01] M. Gautier, P. Poignet: Extended Kalman filtering and weighted least squares dynamic identification of robot, Control Eng. Pract. 9, 1361–1372 (2001)

[Gautier et al. 08] Gautier M, Janot A, Vandanjon. DIDIM: a new method for the dynamic identification of robots from only torque data. In: Proceedings of the IEEE international conference on robotics and automation, Pasadena, CA, USA, 2008. p. 2122-7.

[] Gautier, M.; Vandanjon, P.O.; Janot, A. Dynamic identification of a 6 dof robot without joint position data. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 234–239.

[] Gautier, M.; Briot, S. Global identification of drive gains parameters of robots using a known payload. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 2812–2817.

[] M. Gautier, A. Janot, P.O. Vandanjon: A New Closed Loop Output Error Method for Parameter Identification of Robot Dynamics, IEEE Trans. Control Syst. Techn. 21, 428–444 (2013)

[] M. Gautier, S. Briot, Global Identification of Joint Drive Gains and Dynamic Parameters of Robots. J. Dyn. Syst. Meas. Control 2014, 136, 1–9.

[] Gautier, M.; Venture, G. Identification of standard dynamic parameters of robots with positive definite inertia matrix. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 5815–5820.

[] M. Gautier, G. Venture Identification of Standard Dynamic Parameters of Robots with positive definite inertia matrix 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems [10.1109/IROS.2013.6697198](https://doi.org/10.1109/IROS.2013.6697198)

[] Gautier M. Optimal motion planning for robot's inertial parameters. Proceedings of the 31 conference on decision and control, Tucson, Arizons, 1992. pp. 70–3.

[Khalil et al. 97] W. Khalil, D. Creusot: SYMORO+: A system for the symbolic modeling of robots, Robotica 15, 153–161 (1997)

[Khalil 14] W. Khalil, A. Vijayalingam, B. Khomutenko, I. Mukhanov, P. Lemoine, G. Ecorchard: OpenSYMORO: An open-source software package for symbolic modelling of robots, Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron. (2014) pp. 126–1211

[Gautier-Khalil 94] M. Gautier, W. Khalil Direct calculation of minimum set of inertial parameters of serial robots IEEE Transactions on Robotics and Automation, 1994, 10 (1), pp.78-79.

Janot

[] A. Janot, P.O. Vandanjon, M. Gautier: A Generic Instrumental Variable Approach for Industrial Robots Identification, IEEE Trans. Control Syst. Techn. 22, 132–145 (2014)

[] A. Janot, P.O. Vandajon, M. Gautier A revised Durbin-Wu-Hausman test for industrial robot identification. *Control Eng. Pract.* 2016, 48, 52–62.

[] A. Janot, On the Identification of Continuous-Time Inverse Dynamic Model of Electromechanical Systems Operating in Closed Loop with an Instrumental Variable Approach: Application to Industrial Robots. Ph.D. Thesis, Centre Midi Pyrenees Toulouse, Toulouse, France, 2017.

[] A. Janot, P.O. Vandajon, M. Gautier A generic instrumental variable approach for industrial robot identification. *IEEE Trans. Control Syst. Technol.* 2014, 22, 132–145.

[] Janot, A.; Wensing, P.M. Sequential semidefinite optimization for physically and statistically consistent robot identification. *Control Eng. Pract.* 2021, 107, 104699.

Luh

[Luh et al. 80] J.Y.S. Luh, M.W. Walker, R.P.C. Paul: On-line computational scheme for mechanical manipulators, *Trans. ASME J. Dyn. Syst. Meas. Control* 102(2), 69–76 (1980)

[] J.Y.S. Luh, M.W. Walker, R.P.C. Paul: Resolved– acceleration control of mechanical manipulator, *IEEE Trans. Autom. Control* 25(3), 468–474 (1980)

Walker-Orin

[] J.Y.S. Luh, M.W. Walker, R.P.C. Paul: On-line computational scheme for mechanical manipulators, *Trans. ASME J. Dyn. Syst. Meas. Control* 102(2), 69–76 (1980)

[Walker 82] M.W. Walker, D.E. Orin: Efficient dynamic computer simulation of robotic mechanisms, *Trans. ASME J. Dyn. Syst. Meas. Control* 104, 205–211 (1982)

[] D.E. Orin, W.W. Schrader: Efficient computation of the jacobian for robot manipulators, *Int. J. Robotics Res.* 3(4), 66–75 (1984)

[] S. McMillan, D.E. Orin: Efficient computation of articulated-body inertias using successive axial screws, *IEEE Trans. Robotics Autom.* 11, 606–611 (1995)

[] K.W. Lilly, D.E. Orin: Efficient O(N) recursive computation of the operational space inertia matrix, *IEEE Trans. Syst. Man Cybern.* 23(5), 1384–1391 (1993)

Featherstone

[] R. Featherstone: The calculation of robot dynamics using articulated-body inertias, Int. J. Robotics Res. 2(1), 13–30 (1983)

[] R. Featherstone, S. Sonck, O. Khatib: A general contact model for dynamically-decoupled force/motion control, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Detroit (1999) pp. 3281–3286

[] R. Featherstone, A. Fijany: A technique for analyzing constrained rigid-body systems and its application to the constraint force algorithm, IEEE Trans. Robotics Autom. 15(6), 1140–1144 (1999)

[] R. Featherstone: A divide-and-conquer articulated-body algorithm for parallel $O(\log(n))$ calculation of rigid-body dynamics. Part Trees, loops and accuracy, Int. J. Robotics Res. 18(9), 876–892 (1999)

[] R. Featherstone, D.E. Orin: Robot dynamics: Equations and algorithms, Proc. IEEE Int. Conf. Robotics Autom., San Francisco (2000) pp. 826–834

[Featherstone 04] R. Featherstone: An empirical study of the joint space inertia matrix, Int. J. Robotics Res. 23(9), 859–871 (2004)

[Featherstone 05] R. Featherstone: Efficient factorization of the joint space inertia matrix for branched kinematic trees, Int. J. Robotics Res. 24(6), 487–500 (2005)

[] [R. Featherstone](#) Plucker basis vectors Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. [10.1109/ROBOT.2006.1641982](https://doi.org/10.1109/ROBOT.2006.1641982)

[Featherstone 10] R. Featherstone: Exploiting sparsity in operational space dynamics, Int. J. Robotics Res. 29(10), 1353–1368 (2010)

Arimoto

[] M. Takegaki, S. Arimoto: A new feedback method for dynamic control of manipulators, Trans. ASME J. Dyn. Syst. Meas. Control 103, 119–125 (1981)

[] S. Arimoto, F. Miyazaki: Stability and robustness of PID feedback control for robot manipulators of sensory capability. In: Robotics Research, ed. by M. Brady, R. Paul (MIT Press, Cambridge 1984) pp. 783–799

[] S. Arimoto: Mathematical theory or learning with application to robot control. In: Adaptive and Learning Control, ed. by K.S. Narendra (Plenum, New York 1986) pp. 379–388

Denavit-Hartenberg

[Denavit-Hartenberg 55] J. Denavit, R.S. Hartenberg: A kinematic notation for lower-pair mechanisms based on matrices, J. Appl. Mech. 22, 215–221 (1955)

[Denavit-Hartenberg 64] R. Hartenberg, J. Denavit: Kinematic Synthesis of Linkages (McGraw-Hill, New York 1964)

Uicker

[] Uicker 67] J.J. Uicker: Dynamic force analysis of spatial linkages, Trans. ASME J. Appl. Mech. 34, 418–424 (1967)

[] Uicker, J. J., On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices. PhD thesis, Northwestern University. (1965)

[] J.J. Uicker, Dynamic behavior of spatial linkages. Journal of Engineering for Industry (1969) <https://doi.org/10.1115/1.3591539>

[] J.J. Uicker Jr., J. Denavit, R.S. Hartenberg: An interactive method for the displacement analysis of spatial mechanisms, J. Appl. Mech. 31, 309–314 (1964)

Lie groups

[] F.C. Park, J.E. Bobrow, S.R. Ploen: A lie group formulation of robot dynamics, Int. J. Robotics Res. 14(6), 609–618 (1995)

[] M.E. Kahn, B. Roth: The near minimum-time control of open-loop articulated kinematic chains, J. Dyn. Syst. Meas. Control 93, 164–172 (1971)

[] B. Paden, S. Sastry: Optimal kinematic design of 6R manipulators, Int. J. Robotics Res. 7(2), 43–61 (1988)

Hollerbach

Profile

[] J.M. Hollerbach: A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Trans. Syst. Man Cybern. SMC-10(11), 730–736 (1980)

[] J. Hollerbach: Dynamic scaling of manipulator trajectories, Tech. Rep. 700 (MIT, Cambridge 1983)

[] J.M. Hollerbach, K.C. Suh: Redundancy resolution of manipulators through torque optimization, IEEE J. Robotics Autom. 3, 308–316 (1987)

[] J.M. Hollerbach, K.C. Suh: Local versus global torque optimization of redundant manipulators, IEEE Int. Conf. Robotics Autom. (ICRA), Raleigh (1987) pp. 619–624

[] J.M. Hollerbach: Optimum kinematic design for a seven degree of freedom manipulator. In: Robotics Research – The Second International Symposium, ed. by H. Hanafusa, H. Hinoue (MIT Press, Cambridge 1985) pp. 216–222

Atkeson

[] C.G. Atkeson, C.H. An, J.M. Hollerbach: Estimation of inertial parameters of manipulator loads and links, Int. J. Robotics Res. 5(3), 101–119 (1986)

[An et al. 85] C. H. An, C. G. Atkeson, J. M. Hollerbach [Estimation of inertial parameters of rigid body links of manipulators \(1985\) 24th IEEE Conference on Decision and Control, 990-995](#)

Wensing

[] P. Wensing, R. Featherstone, D.E. Orin: A reduced order recursive algorithm for the computation of the operational-space inertia matrix, Proc. IEEE Int. Conf. Robotics Autom., St. Paul (2012) pp. 4911–4917

[Wensing et al. 24] P. M. Wensing, G. Niemeyer, J.J. E. Slotine A geometric characterization of observability in inertial parameter identification [The International Journal of Robotics Research \(2024\)](#)
doi.org/10.1177/02783649241258215

[] P.M. Wensing, S. Kim, J.J.E. Slotine, Linear Matrix Inequalities for Physically Consistent Inertial Parameter Identification: A Statistical Perspective on the Mass Distribution. *IEEE Robot. Autom. Lett.* 2018, 3, 60–67.

Rodriguez-Jain-Kreuz

[Rodriguez 87] G. Rodriguez: Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics, IEEE J. Robotics Autom. RA-3(6), 624–639 (1987)

[] G. Rodriguez, A. Jain, K. Kreutz-Delgado: A spatial operator algebra for manipulator modelling and control, Int. J. Robotics Res. 10(4), 371–381 (1991)

[] K. Kreutz-Delgado, A. Jain, G. Rodriguez: Recursive formulation of operational space control, Proc. IEEE Int. Conf. Robotics Autom., Sacramento (1991) pp. 1750–1753

[] A. Jain Unified formulation of dynamics for serial rigid multibody systems Publication: Journal of Guidance, Control, and Dynamics Volume: 14 (1991) doi.org/10.2514/3.20672

[] G. Rodriguez, A. Jain and K. Kreutz-Delgado Spatial operator algebra for flexible multibody dynamics Proceedings of the Fifth NASA(NSF)DOD Workshop on Aerospace Computational Control [19940010158.pdf](https://ntrs.nasa.gov/api/citations/19940010158.pdf)

[Rodriguez et al. 92] G. Rodriguez and K. Kreutz-Delgado Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix IEEE Transactions on Robotics and Automation (Volume: 8, Issue: 1, Feb 1992) [10.1109/70.127240](https://doi.org/10.1109/70.127240)

[] A. Jain and G. Rodriguez. Recursive flexible multibody system dynamics using spatial operators. *Journal of Guidance, Control, and Dynamics*, 15(6):1453–1466, 1992.

Park

[] F. C. Park and A. P. Murray, Computational aspects of the product-of exponentials formula for robot kinematics. *IEEE Transactions on Automatic Control*, 1994.

[] F.C. Park: Optimal robot design and differential geometry, *ASME J. Mech. Des.* 117, 87–92 (1995)

[] F. C. Park and R. W. Brockett. Kinematic dexterity of robotic mechanisms. *International Journal of Robotics Research*, 13(1):1–15, 1994.

[] F. C. Park and I. G. Kang. Cubic spline algorithms for orientation interpolation. *International Journal of Numerical Methods in Engineering*, 46:46–54, 1999.

[] F. C. Park and J. Kim. Singularity analysis of closed kinematic chains. *ASME Journal of Mechanical Design*, 121(1):32–38, 1999.

UR5 related articles

[] Arenas-Rosales, F.; Martell-Chavez, F.; Sanchez-Chavez, I.Y.; Paredes-Orta, C.A. Virtual UR5 Robot for Online Learning of Inverse Kinematics and Independent Joint Control Validated with FSM Position Control. *Robotics* **2023**, 12, 23 doi.org/10.3390/robotics12010023

[] The Stm, S T Mem, S T Mem, and U S B Otg. UR5 User Manual. (January):1–42, 2014.

[] E. Clochiatti, L. Scalera, P. Boscaroli and A. Gasparetto Electro-mechanical modeling and identification of the UR5 e-series robot

[] P. M. Kebria , S. Al-wais, H. Abdi, S. Nahavandi, Kinematic and Dynamic Modelling of UR5 Manipulator IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2016) doi.org/10.1109/SMC.2016.7844896

[] N. Kovincic, A. Muller, H. Gattringer, M. Weyrer, A. Schlotzhauer and M. Brandstotter Dynamic parameter identification of the Universal Robots UR5 Proceedings of the ARW & OAGM Workshop (2019) [DOI: 10.3217/978-3-85125-663-5-07](https://doi.org/10.3217/978-3-85125-663-5-07)

[] F. Piekiewski and IEEE C I S Member. Dynamic parameter identifica tion of the Universal Robots UR5.

[] Villalobos, J.; Sanchez, I.Y.; Martell, F. Alternative Inverse Kinematic Solution of the UR5 Robotic Arm. In *Advances in Automation and Robotics Research*; Moreno, H.A., Carrera, I.G., Ramírez-Mendoza, R.A., Baca, J., Banfield, I.A., Eds.; Springer: Cham, Switzerland, 2022; pp. 200–207.

[] Analytic Inverse Kinematics for the Universal Robots UR-5/UR-10 Arms. Available online: <https://smartech.gatech.edu/handle/1853/50782> (last access 9 November 2024).

[] Kinematics of a UR5. Available online: rasmusan.dk/.../uploadsur5_kinematics.pdf (accessed on 9 November 2024).

[] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano and A. De Luca, Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization, *IEEE Robot Autom Lett* 4(4), 4147–4154 (2019).

Identification

[Gabiccini et al. 09] M. Gabiccini, A. Bracci, D. D. Carli, M. Fredianelli, A. Bicchi Explicit Lagrangian formulation of the dynamic regressors for serial manipulators. *Proceedings of the ...*, (2), 2009.

Other articles consulted on this work

[] Y. Xu, R. Liu [Dynamic modeling of SCARA robot based on Udwadia–Kalaba theory](#) *Advances in Mechanical Engineering*, 2017

[] G. Boschetti, T. Sinico, Designing Digital Twins of Robots Using Simscape Multibody Published in *Robotics* 14 April 2024 Engineering, Computer Science <doi.org/10.3390/robotics13040062>

[] G. Fabris, L. Scalera, P. Boscaroli, A. Gasparetto Experimental Analysis and Comparison of Friction Models Applied to the UR5e Robot Mechanism Design for Robotics (September 2024) DOI: 10.1007/978-3-031-67383-2_13

[] M. Neubauer, H. Gattringer, H. Bremer A persistent method for parameter identification of a seven-axes manipulator *Robotica* , Volume 33 , Special Issue 5: Robotics in the Alpe-Adria-Danube Region (RAAD 2013) , June 2015 , pp. 1099 - 1112 <doi.org/10.1017/S0263574714001465>

[Gautier-Pham 91] M. Gautier, C.M. Pham Essential parameters of robots. *Proceedings of the 30th IEEE Conference on Decision and Control* (1991)<10.1109/CDC.1991.261862>

[] C. Urrea, D. Saa, J. Kern Automated Symbolic Processes for Dynamic Modeling of Redundant Manipulator Robots *Processes* Volume 12 Issue 3 <doi.org/10.3390/pr12030593>

[] Urrea, C.; Pascal, J. Design, simulation, comparison and evaluation of parameter identification methods for an industrial robot. *Comput. Electr. Eng.* **2016**, *67*, 791–806.

[] Urrea, C.; Pascal, J. Parameter identification methods for real redundant manipulators. *J. Appl. Res. Technol.* **2017**, *15*, 320–331.

[Poignet et al. 00] Poignet P, Gautier M. Comparison of weighted least squares and extended Kalman filtering methods for dynamic identification of robots. In: Proceedings of the IEEE international conference on robotics and automation, San Francisco, 2000. p. 3622–7.

[] G. Garofalo, C. Ott, A. Albu-Schaffer On closed form computation of dynamic matrices and their differentiation IEEE/RSJ International Conference on Intelligent Robots and Systems (2013) [10.1109/IROS.2013.6696688](https://doi.org/10.1109/IROS.2013.6696688)

[Slotine-Li 87] J. Slotine, W. Li: On the adaptive control of robot manipulators, Int. J. Robotics Res. 6(3), 49–59 (1987)

[] W. S. Lu and Q. H. Meng, Regressor formulation of robot dynamics: Computation and applications, IEEE Transactions on Robotics and Automation, Vol. 9, no. 3, pp. 323–333, (1993).

[Yuan 95] J. Yuan and B. Yuan, Recursive computation of the Slotine-Li regressor, in Proceedings of the American Control Conference, Seattle, Washington, (1995), pp. 2327–2331.

[Safeeaa et al. 18] M. Safeeaa, R. Bearee, P. Neto Reducing the computational complexity of mass-matrix calculation for high DOF robots (2018) IEEE/RSJ International Conference on Intelligent Robots and Systems

[] M. Safeeaa, P. Neto, R. Bearee Robot dynamics: A recursive algorithm for efficient calculation of Christoffel symbols Mechanism and Machine Theory 142, 103589

[Wensing et al. 17] P. M. Wensing, G Niemeyer, J.J. E. Slotine Observability in Inertial Parameter Identification in [arXiv.org](https://arxiv.org/) 10 November Engineering, Computer Science (2017)

[Leboutet et al. 21] Q. Leboutet, J. Roux, A. Janot , J. R. Guadarrama-Olvera and G. Cheng Inertial Parameter Identification in Robotics: A Survey. Published in [Applied Sciences](https://www.appliedsciences.com/) 10 May (2021) Engineering, Computer Science [10.3390/app11094303](https://doi.org/10.3390/app11094303)

[Hao et al 21] L. Hao, R. Pagani, M. Beschi, G. Legnai Dynamic and Friction Parameters of an Industrial Robot: Identification, Comparison and Repetitiveness Analysis Robotics Volume 10 Issue 1 (2021) [10.3390/robotics10010049](https://doi.org/10.3390/robotics10010049)

[Wu et al. 10] J. Wu, J. Wang, Z. You, An overview of dynamic parameter identification of robots Robotics and Computer-Integrated Manufacturing DOI:[10.1016/j.rcim.2010.03.013](https://doi.org/10.1016/j.rcim.2010.03.013)

[] W. Huang, H. Min, M. Liu, A review of dynamic parameters identification for manipulator control Published in Cobot 21 January (2022) Engineering doi.org/10.12688/cobot.17444.1

[] M. Gautier IDENTIFICATION OF ROBOTS DYNAMICS IFAC Proceedings Volumes Volume 19, Issue 14, December (1986), [https://doi.org/10.1016/S1474-6670\(17\)59465-3](https://doi.org/10.1016/S1474-6670(17)59465-3)

[] D. Rade, P. Kurka Lagrange's , Maggi's and Kane's equations to the dynamic modeling of serial manipulator Published (2016) Engineering

[] A.Horn, F. Malvezzi, and R. M. M. Orsino. On the Use of Maggi's Equation in the Dynamic Modeling of Multibody Systems. Vibroengineering (2020) doi.org/10.21595/vp.2020.21323.

[] A. De Luca, G. Oriolo: Issues in acceleration resolution of robot redundancy, 3rd IFAC Symp. Robot Control, Vienna (1991) pp. 665–670

[] A. De Luca, G. Oriolo: The reduced gradient method for solving redundancy in robot arms, Robotersysteme 7(2), 117–122 (1991)

[] A. De Luca, G. Oriolo, B. Siciliano: Robot redundancy resolution at the acceleration level, Lab. Robotics Autom. 4(2), 97–106 (1992)

[] A. De Luca, P. Lucibello: A general algorithm for dynamic feedback linearization of robots with elastic joints, Proc. IEEE Int. Conf. Robotics Autom. (1998)

[] A. De Luca: Feedforward/feedback laws for the control of flexible robots, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Vol. 1 (2000) pp. 233–240

[De Luca 09] A. De Luca, L. Ferrajoli, A modified newton-euler method for dynamic computations in robot fault detection and control 2009 IEEE International Conference on Robotics and Automation

[] Y.C. Tsai, A.H. Soni: An algorithm for the workspace of a general n-R robot, ASME J. Mech. Trans. Autom. Des. 105, 52–57 (1985)

[] D. E. Rosenthal. An Order n Formulation for Robotic Systems. J. Astronautical Sciences, vol. 38, no. 4, pp. 511–529. (1990)

[Muscolo et al. 17] G.G. Muscolo, D.G. Caldwell, F. Cannella, Calculation of the Center of Mass Position of Each Link of Multibody Biped Robots. Applied science (2017) [10.3390/app7070724](https://doi.org/10.3390/app7070724)

[Tait 1898] Tait, P. G., On the rotation of a rigid body about a fixed point, *Proceedings of the Royal Society of Edinburgh* 25(2) 261-303 (1869). Reprinted in pp. 86–127 of Tait, P. G., *Scientific Papers, Vol. 1*, Cambridge University Press, Cambridge (1898).

[Bryan 11] Bryan, G. H., *Stability in Aviation: An Introduction to Dynamical Stability as Applied to the Motion of Aeroplanes*, Macmillan, London (1911).

[] Markley, F. Landis Spacecraft Attitude Representations

[Schuster 93] M.D. Shuster, A Survey of Attitude Representations Journal of The Astronautical Sciences (1993)

[Singla-Mortari-Junkins 05] P. Singla, D. Mortari, J. Junkins, How to avoid singularity when using Euler angles, Engineering, Physics (2005)

[Baraff 96] D. Baraff: Linear-time dynamics using lagrange multipliers, Proc. 23rd Annu. Conf. Comp. Graph. Interact. Tech., New Orleans (1996) pp. 137–146

[Baumgarte 72] J. Baumgarte: Stabilization of constraints and integrals of motion in dynamical systems, Comput. Methods Appl. Mech. Eng. 1, 1–16 (1972)

[Bertrand-Bruneau 12] S. Bertrand · O. Bruneau A clear description of system dynamics through the physical parameters and generalized coordinates in Multibody system dynamics 16 August 2012 Engineering, Physics

[Borm et al. 89] C.H. Menq, J.H. Borm, J.Z. Lai: Identification and observability measure of a basis set of error parameters in robot calibration, ASME J. Mech. Autom. Des. 111(4), 513–518 (1989)

[Sun et al. 08] Y. Sun, J.M. Hollerbach: Active robot calibration algorithm, Proc. IEEE Int. Conf. Robotics Autom. (ICRA), Piscataway (2008) pp. 1276–1281

[Pukelsheim 06] F. Pukelsheim, Optimal design of experiments, The Society for Industrial and Applied Mathematics, New York (2006)

[Gautier-Khalil 92] M Gautier, W. Khalil, Exciting trajectories for the identification of base inertial parameters of robots. International Journal of Robotics Research, 11 (4) (1992), pp. 362-375

[Lu et al. 93] Z Lu, KB Shimoga, A. Goldberg, Experimental determination of dynamic parameters of robotic arms. Journal of Robotic Systems, 10 (8) (1993), pp. 1009-1029

[Presse - Gautier 93] Presse C, Gautier M. New criteria of exciting trajectories for robot identification. In: Proceedings of the IEEE international conference on robotics and automation, Atlanta, GA, USA, (1993). p. 907–12.

[Armstrong 89] B. Armstrong, On finding exciting trajectories for identification experiment involving systems with nonlinear dynamics. International Journal of Robotics Research, 8 (6) (1989), pp. 28-48

[Swevers et al. 97] J Swevers, C Ganseman, J DeSchutter, H. VanBrussel, Generation of periodic trajectories for optimal robot excitation. Journal of Manufacturing Science and Engineering, 119 (4) (1997), pp. 611-615

[Abdellatif et al. 04] Abdellatif H, Benimeli F, Heimann B, Grotjahn M. Direct identification of dynamic parameters for parallel manipulators. In: Proceedings of the international conference on mechatronics and robotics, Aachen, Germany, 2004. p. 999–1005.

[Park 06] KJ. Park, Fourier-based optimal excitation trajectories for the dynamic identification of robots. Robotica, 24 (5) (2006), pp. 625-633

[Olsen 02] Olsen, M.M.; Swevers, J.; Verdonck, W. Maximum Likelihood Identification of a Dynamic Robot Model: Implementation Issues. *Int. J. Robot. Res.* **2002**, 21, 89–96.

[Haug 21] E. J. Haug, Multibody Dynamics on Differentiable Manifolds. ASME. J. Comput. Nonlinear Dynam. April 2021; <https://doi.org/10.1115/1.4049995>

[Gaz et al. 14] C. Gaz, F. Flacco, A. De Luca, “Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach,” IEEE Int. Conf. on Robotics and Automation, pp. 1386-1392, 2014

[Gaz et al. 16] C. Gaz, F. Flacco, A. De Luca, Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 2075–2081.

[Mayeda 84] H. Mayeda, K. Osuka , A. Kangawa , A new identification method for serial manipulator arms, Proc. IF AC 9th World Congress, Budapest, Hungary, July 1984, p. 74-79.

[Mayeda 90] Mayeda H., Yoshida K., Osuka K., "Base parameters of manipulator dynamic models", IEEE Trans, on Robotics and Automation, Vol. RA-6(3), 1990. p. 312-3

[Mayeda et al. 88] Mayeda H, Yoshida K, Osuka K Base parameters of manipulator dynamic models. Proceed. 1988 IEEE International Conference on Robotics and Automation [10.1109/ROBOT.1988.12258](https://doi.org/10.1109/ROBOT.1988.12258)

[Beschi et al. 15] M. Beschi, E. Villagrossi, N. Pedrocchi, L.M. Tosatti, A general analytical procedure for robot dynamic model reduction. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 4127–4132.

[Kawasaki et al. 91] H. Kawasaki; Y. Beniya; K. Kanzaki Minimum dynamics parameters of tree structure robot models Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation [10.1109/IECON.1991.239286](https://doi.org/10.1109/IECON.1991.239286)

[Khalil 87a] Khalil W., Chevallereau C, "An efficient algorithm for die dynamic control of robots in the cartesian space", Proc. 26th IEEE Conf on Decision and Control, Los Angeles, USA, December 1987, p. 582-588.

[Khalil 87b] Khalil W., Kleinfmger J.-F., Minimum operations and minimum parameters of the dynamic model of tree structure robots, IEEE J. of Robotics and Automation, Vol. RA3(6), December 1987, p. 517-526. 460 Modeling, identification and control of robots

[Khalil 86c] Khalil W., Kleinfmger J.-F., Oautier M., "Reducing the computational burden of the dynamic model of robots", Proc. IEEE Int. Conf on Robotics and Automation, San Francisco, USA, April 1986, p. 525-531.

[Ros et al. 15] J. Ros,A. Plaza, X. Iriarte et al. Inertia transfer concept based general method for the determination of the base inertial parameters. Multibody System Dynamics, 2015 doi.org/10.1007/s11044-014-9446-3

[Ros et al. 12] J. Ros, X. Iriarte, V. Mata 3D inertia transfer concept and symbolic determination of the base inertial parameters. Mechanism and Machine Theory Volume 49, March 2012, Pages 284-297 doi.org/10.1016/j.mechmachtheory.2011.09.006

[Kims et al. 24] J. W. Kim; P. G. Mehta Duality for Nonlinear Filtering II: Optimal Control IEEE Transactions on Automatic Control (Volume: 69, Issue: 2, February 2024) doi.org/10.1109/TAC.2023.3279208

[Heess et al. 17] N. Heess, Dhruva TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, D. Silver, Emergence of Locomotion Behaviours in Rich Environments, DeepMind

[Jovic et al. 16] Jovic J, Escande A, Ayusawa K, Yoshida E, Kheddar A and Venture G (2016) Humanoid and human inertia parameter identification using hierarchical optimization. IEEE Transactions on Robotics 32(3): 726–735. [10.1109/TRO.2016.2558190](https://doi.org/10.1109/TRO.2016.2558190).

[Jovic et al. 15] Jovic J, Philipp F, Escande A, Ayusawa K, Yoshida E, Kheddar A and Venture G (2015) Identification of dynamics of humanoids: Systematic exciting motion generation. In: IEEE/RSJ Int. Conf. on Intelligent Rob. and Sys. [10.1109/IROS.2015.7353668](https://doi.org/10.1109/IROS.2015.7353668).

[Ayusawa 08a] K. Ayusawa, G. Venture, Y. Nakamura Identification of Humanoid Robots Dynamics Using Floating-base Motion Dynamics 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems [10.1109/IROS.2008.4650614](https://doi.org/10.1109/IROS.2008.4650614)

[Ayusawa et al. 08b] K. Ayusawa; G. Venture; Y. Nakamura Identification of the inertial parameters of a humanoid robot using unactuated dynamics of the base link Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots [10.1109/ICHR.2008.4755923](https://doi.org/10.1109/ICHR.2008.4755923)

[Iwasaki et al. 12] T. Iwasaki, G. Venture, E. Yoshida Identification of the Inertial Parameters of a Humanoid Robot Using Grounded Sole Link 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012) [10.1109/HUMANOIDS.2012.6651558](https://doi.org/10.1109/HUMANOIDS.2012.6651558)

[Ayusawa et al. 14] K. Ayusawa, G. Venture and Y. Nakamura (2014) Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems. The International Journal of Robotics Research. doi.org/10.1177/0278364913495932

[Nakamura et al. 00] Y. Nakamura, K. Yamane: Dynamics computation of structure-varying kinematic chains and its application to human figures, IEEE Trans. Robotics Autom. 16(2), 124–134 (2000)

[Ogawa et al. 14] Y. Ogawa, G. Venture, Christian Ott Dynamic parameters identification of a humanoid robot using joint torque sensors and/or contact forces 2014 IEEE-RAS International Conference on Humanoid Robots [10.1109/HUMANOIDS.2014.7041401](https://doi.org/10.1109/HUMANOIDS.2014.7041401)

[Venture et al. 08] G. Venture, K. Ayusawa, and Y. Nakamura. Dynamics identification of humanoid systems. In Proc. CISIM-IFTToMM Symp. on Robot Design, Dynamics, and Control (ROMANSY), pages 301–308, 2008a.

[Venture et al. 08] G. Venture, K. Ayusawa, and Y. Nakamura. Motion capture based identification of human inertial parameters. In Proc. IEEE/EMBS Int. Conf. on Eng. in Medicine and Biology (to be published), pages 4575–4578, 2008b.

[Ott et al. 08] D. Ott, C. Lee and Y. Nakamura, Motion capture based human motion recognition and imitation by direct marker control, in Proc. IEEE/Int. Conf. on Humanoids, December 2008 (to be published).

[Baelemans et al. 13] J. Baelemans, P. van Zutven, H. Nijmeijer Model Parameter Estimation of Humanoid Robots using Static Contact Force Measurements Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven University of Technology P.O. Box 513, 5600 MB, Eindhoven, the Netherlands (2013) doi.org/10.1109/SSRR.2013.6719328

[Venture et al. 09] G. Venture, K. Ayusawa, Y. Nakamura, Identification of Human Mass Properties From Motion, IFAC Proceedings Volumes Volume 42, Issue 10, 2009, Pages 988-993 doi.org/10.3182/20090706-3-FR-2004.00164

[Grotjahn et al. 01] M Grotjahn, M Daemi, B. Heimann, Friction and rigid body identification of robot dynamics. International Journal of Solids and Structures, 38 (10–13) (2001), pp. 1889-1902

[Gaz et al. 17] C. Gaz, A. De Luca, "Payload estimation based on identified coefficients of robot dynamics – with an application to collision detection," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 3033-3040, 2017

[Ha et al. 89] I.J. Ha, M.S. Ko, S.K. Kwon , An efficient estimation algorithm for the model parameters of robotic manipulators. IEEE Trans, on Robotics and Automation, Vol. RA5(6). 1989. p. 386-394.

[Kawasaki et al. 88] H Kawasaki, K. Nishimura, Terminal-link parameter estimation and trajectory control of robotic manipulators. IEEE Journal of Robotics and Automation, 4 (5) (1988), pp. 485-490

[Serban et al. 01] R Serban, JS. Freeman, Identification and identifiability of unknown parameters in multibody dynamic systems. Multibody System Dynamics, 5 (4) (2001), pp. 335-350

Webgraphy

underactuated.csail.mit.edu/

royfeatherstone.org/

[Tree \(graph theory\). \(2009\)](http://Tree (graph theory). (2009))

[Gauss's principle of least constraint](#)

[Moment of inertia](#)

[List of moments of inertia](#)

Simscape Multibody. Available online: <https://www.mathworks.com/products/simscape-multibody.html> (accessed on 9 November 2024).

Robotic System Toolbox. Available online:

<https://www.mathworks.com/products/robotics.html> (accessed on 31 October 2024).

Further chapters and section considered in this thesis

[Lie groups and algebra for kinematics and dynamics](#)

[Motors algebra](#)

[Line representation](#)

[Describing kinematics and dynamics using homogeneous transformations](#)

[Jacobians and their use](#)

[Closed chain robot](#)

[Underactuated robots](#)

[Other topics Kinematics](#)

[Other topics inherent robot dynamics](#)

[Other methods for derivation of systems dynamics](#)

[Other methods and topics in dynamics](#)

[Spatial Motion notation for dynamics with SOA](#)

[Continuous robots](#)

[Inertia matrix example](#)

[Least squares, Kalman filtering and other techniques of optimization](#)

[Alternative methods used in identification](#)

[Reinforcement learning](#)

[PINN Physic Informed Neural Networks](#)

[URDF Files](#)

[Grasping and effectors](#)

[Full biography](#)

.Resources

royfeatherstone.org/

<https://github.com/TUM-ICS/BIRDy>

[https://doi.org/10.5281/zenodo.4728085 \(Data sources\)](https://doi.org/10.5281/zenodo.4728085)

[https://doi.org/10.5281/zenodo.4679467 \(Data sources\)](https://doi.org/10.5281/zenodo.4679467)

<https://github.com/symoro/symoro>

https://github.com/ROAM-Lab-ND/spatial_v2_extended/blob/7967e43/v3/identifiability/RPNA_Examples.m.Commit:7967e43
<https://github.com/rbdl/rbdl>
https://github.com/kkufieta/ur5_modeling_force_estimate
<https://github.com/topics/urdf-models>

[] Smits, R.; Bruyninckx, H.; Aertbeliën, E. KDL: Kinematics and Dynamics Library. 2011.
Available online: <https://github.com/rbdl/rbdl> (accessed on 9 November 2024).

All thesis material will be available also (by January 2025) at :

https://github.com/enricosold/Thesis_2024-Simscape-based-robot-digital-twin-development-supported-by-symbolic-modeling-

https://github.com/enricosold/Thesis_2024/blob/main/Readings/Moving%20base%20robots.pdf

https://github.com/enricosold/Thesis_2024-Simscape-based-robot-digital-twin-development-supported-by-symbolic-modeling-/tree/main/Readings

https://github.com/enricosold/Thesis_2024-Simscape-based-robot-digital-twin-development-supported-by-symbolic-modeling-/tree/main/Codes