Industrial Robotics M

Dante Piotto

spring semester 2023

Contents

Chapter 1

Elements of Kinematics and Dynamics of Industrial Robots

1.1 Introduction

A manipulator may be mechanically described as the interconnection of rigid bodies *(links)* through kinematic pairs *(joints)*. In this course we deal with open chains of links (serial configuration). The first link is the *base*, while the last link is the *end effector*. We will consider the links to be rigid bodies for the sake of simplicity.

A joint is an element that constrains one or more relative motion directions between two links. In robotics only two types of joints are used: rotoidal joints and prismatic joints.

1.1.1 Degrees of Freedom of a Manipulator

Each joint is characterised by the number of independent motion directions allowe between two consecutive links. This number defines the *degrees of freedom* of the joint. Rotoidal and prismatic joints have 1 degree of freedom (dof of the joint. Rotoidal and prismatic joints have 1 degree of freedom (dof). In place of spehrical joints, 3 rotoidal joints are used, forming a *spherical wrist*. The spherical wrist is typically at the end of the kinematic chain. If a joint has k dof, then the relative configuration between two rigid bodies may be expressed as a function of k variables $q_1, q_2, \ldots, q_k$, called *joint variables*.

1.1.2 Joint Space and Work Space

Joint Space: Let us stack all the joint variables in a vector

$$q = \begin{bmatrix} q_1 & q_2 & \ldots & q_n \end{bmatrix}^T$$

with $q \in \mathcal{Q} \subset \mathbb{R}^{N_a of}$. The set $\mathcal{Q}$ is called *joint space*

Work Space: The work space is a subset of the Euclidean space $\mathbb{E}$ in which the robot executes its tasks. It is the set of all the points that the mechanical structure may assume, and in general is a 3D subset of $\mathbb{E}$. Each point of the work space is indicated by a vector x of proper dimension, $x \in \mathbb{R}^y$, where usually $y = \{3, 6\}$

configuration of a manipulator: It takes into account both the position and orientation of a reference frame fixed to the manipulator extremity (end effector). Then (locally):

- $x \in \mathbb{R}^3$ in a plane
- $x \in \mathbb{R}^6$ in space

Classification of manipulators: if $\mathbb{R}^n$ is the joint space and $\mathbb{R}^m$ is the work space:

- $n = m$: "normal" cases
- $n < m$: *defective* manipulators
- $n > m$: *redundant* manipualtors

1.1.3 Types of Robotic Manipulators

Common kinematic structures for robotic manipulators:

- Cartesian robots: 3 perpendicular prismatic joints. Simplest robot structure. Structure can be made to be very robust and therefore very large. Usually meant to move large payloads. Simple to control (1 motor for each direction of motion). Simple mapping between joint space and work space.
- Antropomorphic robots (majority): 3 rotational joints, 1 with a vertical axis and 2 with horizontal axes. More dexterous, harder to control, more complex coupling between joint space and work space. Errors on eaech motor stack in the final motion
- SCARA robots: Selective Compliance Assembly Robot Arm. Rigid along z, more compliant along x and y.
- Cylindrical and spherical robots: not used
- Spherical wrists: typically attached at the end of Cartesian, Antropomorphic or SCARA bots.

The first 3 joints define the basic kinematic structure

1.2 Kinematic model of a manipulator

x is made of 3 elements determining position an 3 determining orientation. These 3 represent angles, taht may be Roll-Pitch-yaw, or Euler angles (Z-Y-Z Euler angles are used in robotics)

1.2.1 Forward kinematics

mapping of joint space variables into work space variables

$$x = f(q) \quad q \in \mathbb{R}^n, x \in \mathbb{R}^m$$

It is possible to define different kinematic models for a robotic manipulator, i.e. different $f(q)$ functions. They are however mathematically equivalent

Homogenous trasformation matrix

the configuration of a rigid body b_i in a 3D space can be described by a *homogenous transformation matrix* betweeen a reference frame FF_i and the base frame FF_0 thus constructed:

$${}^0T_i = \begin{pmatrix} {}^0R_i & {}^0o_i \\ \mathbf{0} & 1 \end{pmatrix}$$

where ${}^0R_i$ is a rotation matrix belonging to $SO(3)$, and ${}^0o_i \in \mathbb{R}^3$ is a position vector. It can be noted that the determinant of the homogenous transformation matrix is also equal to 1.

For computational reasons, a new position vector $p = [p_x, p_y, p_z, 1]^T$ is defined. This way, the position ${}^0p$ of a generic point ${}^1p$ of the rigid body can be obtained as:

$${}^0p = {}^0\ T_1\ {}^1p$$

or, with some abuse of notation:

$$\begin{bmatrix} {}^0p \\ 1 \end{bmatrix} = {}^0\ T_1 \begin{bmatrix} {}^1p \\ 1 \end{bmatrix} = \begin{bmatrix} {}^0R_1\ {}^1p + {}^0\ o_1 \\ 1 \end{bmatrix}$$

in general

$${}^0T_n = {}^0\ T_1\ {}^1T_2 \ldots {}^{n-1}\ T_n$$

due to homogenous transformation being a linear operation.

Denavit-Hartenberg convention

- each link is numbered, from 0 to n, so that it is uniquely identified in the mechanical chain: $L_0, L_1, \ldots, L_n$. The base link is conventionally identified as L_0, the distal link as L_n. A manipulator with $n+1$ links has n joints.
- Joints are numbered from 1 to n. J_i connects links L_{i-1} and L_i
- Frames $\mathcal{F}_i$ are associated to each joint, according to the Denavit-Hartenberg procedure, and matrices ${}^{i-1}T_i$ are computed:
  - all the z axes are directed along the direction of motion
  - axis $x+1$ intersects and is perpendicular to axis z_i
- The Denavit-Hartenberg procedure requires 4 parameters for each link $(d_i, \theta_i, a_i, \alpha_i)$

This procedure provides a 4x4 matrix, not a vector, for x. The vector x can be computed as a function of the homogeneous trasformation matrix.

1.2.2 Inverse kinematics

mapping of work space variables into joint space variables

$$q = g(x) = f^{-1}(x) \quad q \in \mathbb{R}^n, x \in \mathbb{R}^m$$

A standard procedure to obtain the inverse kinematic model does not exist.
Moreover, we may have

- no solution (if x does not belong to the workspace
- a finite set of solutions
- infinite solutions

A *closed form* solution is preferrable to numerical procedures (better computation, easier to select a particular solution in the set of solutions)
A closed form solution to the inverse kinematic problem may be obtained, if it exists, by trying one of these approaches:

- Algebraic approach: elaboration of the kinematic equatoins in order to obtain a set of 'simple' equations that can be solved in the uknown joint variables q
- Geometric approach: based on considerations dependent on the geometric structure of the manipulator

Algebraic approach

The values of the homogenoeus trasnformation matrix are known, and the homogeneous trasnformation matrix can be written as a function of joint variables. This yields 12 scalar equations (usually more than needed). Hopefully there are a sufficient number of equations that can be solved easily enough. As an additional resource, one can perform algebraic manipulations on the involved matrices to try and find a better set of equations. This procedure is in general not so simple

Geometric approach

Also called Pieper approach.
For several kinematic structures it is possible to apply the *kinematic decoupling* principle, that allows to decompose the overall 6Dof problem into two 3 Dof sub-problems:

- inverse kinematic problem for position
- inverse kinematic problem for orientation

Sufficient condition to solve the inverse kinematic problem for a 6DoF manipulator:

- 3 consecutive rotational joints whose axes intersect in a point (spherical wrist: notice that the position of the center of rotation of the spherical wrist does not depend on the joint variables of the spherical wrist but rather on the other joints)
- 3 consecutive rotation joints with parallel axes

1.2.3 Differential kinematics - the Jacobian

Ohter relations of intrest in robotics are those between:

- end-effector velocity and joint velocities
$$\begin{pmatrix} v \\ \omega \end{pmatrix} \iff \dot{q}$$
- force applied on the environment by the manipulator and corresponding joint torques:
$$\begin{pmatrix} f \\ n \end{pmatrix} \iff \tau$$

These relations are based on a matrix known as the *Jacobian of the manipulator*. The Jacobian is also used for:

- studying the singularities of the mechanical structures
- defining numerical algorithms to solve the inverse kinematics
- studying manipulability properties

The six-dimensional velocity vector (twist) in the woek-space is defined with three linear and three rotational components

$$\dot{x} = \begin{pmatrix} v \\ \omega \end{pmatrix}$$

Two expressions of the Jacobian matrix can be defined:

- the analytic Jacobian, used when the rotational component in $\dot{x}$ is defined as $\dot{\gamma}$, the time-derivative of a triple of orientation variables
- the geometric Jacobian, used when the rotational component in $\dot{x}$ is defined as ω, the rotational velocity vector.

It is possible to relate the two expressions by means of a proper matrix $T(\gamma)$

$$J_G(q) = T(\gamma)J_A(q) = \begin{bmatrix} I_3 & 0 \\ 0 & \hat{T}(\gamma) \end{bmatrix} J_A(q)$$

Analytic Jacobian

The analytic expression of the Jacobian is obtained by differentiating the six-dimensional vector $x = f(q) = [p^T, \gamma^T]^T$ containing the position (p) and orientation (γ) of the end-effector in the work-space wrtnthe base frame FF0; usually γ are the Euler or RPY angles
By differentiating $f(q)$ we get:

$$dx = \frac{\partial f(q)}{\partial q} dq = J(q)dq \qquad (\dot{x} = J(q)\dot{q})$$

where the $m \times n$ matrix

$$J(q) = \begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \frac{\partial f_m}{\partial q_2} & \cdots & \frac{\partial f_m}{\partial q_n} \end{pmatrix}, \qquad J(q) \in \mathbb{R}^{m \times n}$$

Is called the *Jacobian matrix*
We have that the relationship between $\dot{x}$ and $\dot{q}$ is linear. The Jacobian can be conceptually split in two blocks, the first concerning the linear part of the velocity, the second concerning the angular part:

$$\dot{x}_A = J_A(q)\dot{q} = \begin{bmatrix} \dot{p} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} J_p \\ J_\gamma \end{bmatrix} \dot{q}$$

Geometric expression

In practice, the Jacobian is always computed in its geometric expression as it is much simpler. The geometric expression of the Jacobian is obtained considering as rotational components of the velocity vector $\dot{x}$ the rotational vector ω and not the derivative of a triple of angles γ

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\gamma} \end{bmatrix} \quad \Longleftrightarrow \quad \dot{x} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix}$$

$$\dot{x}_G = J_G(1)\dot{q} = \dot{x} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \dot{q} = \begin{bmatrix} J_{v_1} & J_{v_2} & \dots J_{v_n} \\ J_{\omega_1} & J_{\omega_2} & \dots J_{\omega_n} \end{bmatrix}$$

this leads to

$$v = J_{v_1}\dot{q}_1 + J_{v_2}\dot{q}_2 + \cdots + J_{v_n}\dot{q}_n$$
$$\omega = J_{\omega_1}\dot{q}_1 + J_{\omega_2}\dot{q}_2 + \cdots + J_{\omega_n}\dot{q}_n$$

it is possible to show, applying the superimposition principle by considering only the $i-th$ joint to be moving, that the generic $i-th$ column of the geometric Jacobian is defined as

$$\begin{bmatrix} J_{vi} \\ J_{\omega i} \end{bmatrix} = \begin{bmatrix} {}^0z_{i-1} \times ({}^0p_n - {}^0p_{i-1}) \\ {}^0z_{i-1} \end{bmatrix} \qquad \text{rotational joint}$$

$$\begin{bmatrix} J_{vi} \\ J_{\omega i} \end{bmatrix} = \begin{bmatrix} {}^0z_{i-1} \\ 0 \end{bmatrix} \qquad \text{rotational joint}$$

Once the forward kinematics of the manipulater is know these expressions are simple to obtain as the vectors ${}^0p_i$ and ${}^0z_i$ are obtained from the matrices ${}^0T_i$. This does not represent a computational burden with respect to the computation of the forward kinematic model.

Force domain

By exploiting the *virtual work principle*, a mapping similar to that in the velocity domain can be obtained for the force domain.
Because work is invariant wrt the frame in which it is computed:

$$w^T dx = \tau^T dq$$

where $w = (f^T n^T)^T$ is a 6-dimensional vector (*wrench*) collecting the linear forces f and the torques n applied to the manipulator. τ is the n-dimensional vector of the *joint forces/torques* since

$$dx = J(q)dq$$

we obtain by substituting

$$\tau = J^T(q)w$$

Singularities

When the Jacobian is rank deficient, there are directions in $\mathbb{R}^6$ that the robot cannot move in. Furthermore, limited velocities of the end-effector may correspond to infinite veolocities in the joint space. There is no well defined solution to the inverse kinematic problem: either no solution or infinite solutions. Singularities typically correspond to positions at the border of the workspace.

Inverse differential kinematics

The forward mapping betweeen joint and work-space is given by

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

Then, the inverse mapping is given as the solution of a linear algebraic equation

- $m = n$: if the manipulator is not in a singular configuration, it is possible to use the inverse of the jacobian:

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

- $m \neq n$: the Moore-Penrose pseudoinverse of the Jacobian is used:

$$\dot{q} = J^{\dagger}(q)\dot{x}$$

if $m < n$ (right pseudoinverse):

$$J^{\dagger} = J^T(JJ^T)^{-1}$$
$$J^{\dagger} = I_m$$

if $m > n$ (left pseudoinverse):

$$J^{\dagger} = (J^T J)^{-1}J^T$$
$$J^{\dagger} = I_n$$

When the joint space is larger than the work space ($n > m$), the solution of $\dot{x} = J\dot{q}$:

- $rank(J) = min(m, n) = m \quad \implies \mathcal{R}(J) = \mathbb{R}^m$
- $\forall \dot{x}, \exists \dot{q}$ such that $\dot{x} = J\dot{q}$ more than one exists
- $\dot{q} = J^{\dagger}\dot{x} + \dot{q}_N \implies \dot{x} = J(J^{\dagger}\dot{x} + \dot{q}_n = \dot{x} \quad \forall \dot{q}_N in \mathcal{N}(J)$
  $\implies \dot{q} = J^{\dagger}\dot{x} + (I - J^{\dagger}J)y$ is a general expression of the solution
  $(I - J^{\dagger}J)$ is a base of $\mathcal{N}(J)$
- $\dot{q} = J^{\dagger}\dot{x}$ has *minimum norm*

Inverse kinematics with the Jacobian

The Jacobian may be used to solve the inverse kinematic problem if a closed form solution is not available. A method could be to compute

$$q_{k+1} = q_k + J^{-1}(q_k)v_k dt$$

where a numerical integration of the position is performed. Unfortunately, this method is subject to numerical drifts and initialization problems, and it is therefore difficult to obtain the desired solution.

An alternative method is based on a feedback loop, by defining an error in the work-space. Given

$$e = x_d - x$$

by differentiating, we obtain

$$dote = \dot{x}_d - J(q)\dot{q}$$

it is necessary to define $\dot{q}$ in such a way that the error e converges to 0

Algorithm 1

if

$$\dot{q} = J^{-1}(q)(\dot{x}_d + Ke)$$

with $K = K^T > 0$, then

$$\dot{e} = \dot{x}_d - JJ^{-1}(\dot{x}_d + Ke)$$
$$\dot{e} = -Ke$$

Typically K is chosen as a diagonal matrix, leading to

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \vdots \\ \dot{e}_n \end{bmatrix} = \begin{bmatrix} K_1 & \cdots & & \\ \vdots & K_2 & & \\ & & \ddots & \\ & & & K_n \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

Which is completely decoupled and converging to 0.

Algorithm 2

We use a Lyapunov technique:
Given $\dot{x}_d = 0$, if

$$\dot{q} = J^T(q)Ke$$

with $K = K^T > 0$, then

$$V(e) = \frac{1}{2}e^T Ke \implies \dot{V}(e) = e^T K\dot{e} = e^T K(\dot{x}_d - J\dot{q})$$
$$= -e^T KJ(q)\dot{q}$$

we choose $\dot{q} = J^T Ke$, leading to:

$$\dot{V}(e) = -e^T KJJ^T Ke \leq 0$$

1.2.4 Manipulability Measures

The Jacobian may be used to evaluate the achievable performaces of a manipulator, in terms of velocities and applicable forces, through the *manipulability ellipsoids*

Velocity manipulability ellipsoids

Given a sphere with unit radius radius in the joint velocity space, $\dot{q}^T\dot{q} = 1$, that may be considered as a"cost" (energy input to the manipulator), we want to obtain its equivalent expression in the work-space. From $\dot{x} = J(q)\dot{q}$ we obtain:

$$\dot{q} = J^\dagger \dot{x}$$
$$\implies \dot{x}J^{\dagger T}J^\dagger \dot{x} = 1$$
$$\implies \dot{x}^T(JJ^T)^\dagger \dot{x} = 1$$

which is an ellipsoid in the work-space $\mathbb{R}^m$

- the principal axes are directed along the eigenvectors of JJ^T
- the length of the principal axes are given by the singular values of J, $\sigma_i = \sqrt{\lambda_i(JJ^T)}$

Force manipulability ellipsoids

Let us now consider the equation $\tau^T \tau = 1$ from which we obtain:

$$w^T J J^T w = 1$$

This equation describes an ellipsoid in the $\mathbb{R}^m$ force space

- the principal axes are directed along the eigenvectors of JJ^T
- the length of the principal axes are given by the inverse of the singular values of J, $\sigma_i = \sqrt{\lambda_i(JJ^T)}$

This result confirms the duality of the velocity/force spaces: along the directions in which high velocity performances are possible, low force performances are achievable and viceversa

Chapter 2

Dynamic Model of Robotic Manipulators

Dymanics of a robot manipulator: analysis of the relationship between the applied forces/torques and the resulting motion of a robot. For the dynamics, it is possible to define two "models":

- Direct model: goven the forces/torques applied at joints, the hoint position and velocity, compute the acceleration
$$\ddot{q} = f(q, \dot{q}, \tau)$$
and then
$$\dot{q} = int\ddot{q}dy \qquad q = \int qdotqdt$$
- Inverse model: given the desired joint acceleration, velocity and position, compute the corresponding force/torque to be applied
$$\tau = f^{-1}(\ddot{q}, \dot{q}, q) = g(\ddot{q}, \dot{q}, q)$$

In deriving the dynamic model of a robot, we refer to a chain of rigid bodies mutually, and ideally, connected among them The dynamic model of a robot is used for:

- simulation: testing motion behaviour without resorting to physical experiments
- analysis and synthesis of control laws
- alaysis of the structural properties of a manipulator in the design phase

2.1 Euler-Lagrange model

The dynamic model obtained through this method is simpler and more intuitive, and also better suited to understanding the effects of changes in the mechanical parameters. The links are considered altogether, and the model is obtained analytically. Drawbacks: the model is obtained starting from the kinetic and potential energies (non intuitive) and the model is not computationally efficient.

Lagrange coordinates: independet variables used to describe the position of rigid bodies in space. For the same physical system, several choices of Lagrangian coordinates are usually possible. In robotics, we use *joint variables* $q_1, q_2, \ldots, q_n$. Input torques should be defined according to the chosen Lagrangian coordinates $\Longrightarrow$ joint torques $\tau_1, \tau_2, \ldots, \tau_n$

From physics, we know it is possible to define:

- the Kinetic Energy function $K(q\dot{q})$
- the Potential Energy function $P(q)$

And therefore the Lagrangian function

$$\mathcal{L}(q\dot{q}) = K(q,\dot{q}) - P(q)$$

The EEuler-Lagrange equations for a system described by n Lagrange coordinates q_i are

$$\psi_i = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} \qquad i = 1, \ldots, n$$

Being ψ_i the non-conservative (external or dissipative) generalized forces performing work on q_i. In robotics, we consider:

- τ_i joint actuator torque
- $[J^T F_c]_i$ term due to external (contact) forces
- $d_{ii}\ \dot{q}_i$ joint friction torque

Therefore:

$$\psi_i = \tau_i + [J^T F_c]_i - d_{ii}\dot{q}_i$$

It can be noted that

$$K = \sum_{i=1}^{n} K_i \qquad P = \sum_{i=1} nP_i$$

2.1.1 obtained model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau$$

Non-linear wrt the state. It can be made linear wrt a set of parameters α

2.1.2 Identification of the dynamic parameters

From:

$$Y(q,\dot{q},\ddot{q})\alpha = \tau$$

by measuring along a proper trajectory the quantities $q, \dot{q}, \ddot{q}, \tau$ ($N > p$ samples) one obtains:

$$\tau_0 = Y_0\alpha$$
$$Y_0^T\tau_0 = Y_0^T Y_0\alpha$$
$$\alpha = (Y_0^T Y_0)^{-1} Y_0^T \tau_0 = Y_0^\dagger \tau_0$$

2.1.3 Dynamic model in simulink

When only one of the two of the joints has friction energy still leaks out of the system because the system is coupled so movement of each link affects the other link causing the energy to be lost.

2.1.4 Linearization of the Euler-Lagrange model

By neglecting friction effects ($D = 0$), and considering the vector $C(q,\dot{q})\dot{q} = c(q,\dot{q})$, the non linear dynamic model of a robot manipulator is:

$$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = u$$

It is possible to obtain a linear dynamic model which is valid only locally around a ginven operative condition (not used, very bad, uselss, stupid, why even bother):

- linearization around a (constant) equilibrium state
- linearization along a (nominal, time-varying) equilibrium trajectory

skipped because excessively useless and pointless

2.1.5 Symbolic computation of the dynamic model

The Euler-Lagrange formulation of the dynamic model of a robot arm can be expressed in a form suitable to be implemented in a symbolic package (Mathematica or Matlab Symbolic toolbox) in order to obtain, given the kinematic and mechanical parameters of each link/joint, the model expressed in a symbolic form. Parameters needed: DH parameters, masses, inertia matrices, position of centers of mass.

2.2 Newton-Euler model

Based on a computationally efficient recursive technique that expoits the serial structure of an industrial manipulator. On the other hand, the model is not expressed in closed form. The model is based on the balance of forces and torques acting on each link.

The dynamic equations are written separately for each link and a *recursive formulation* is obtained: forward recursion (from Link 0 to Link n) for the computation of the "propagation" of accelerations and velocities, backward recursion (from Link n to Link 0) for the computation of the "propagation" of forces and torques. The basic equations are:

- Newton's second law of motion: (conservation of momentum mv)

$$f = \frac{d(mv)}{dt}$$

  where m is mass, v is the linear velocity, f the sum of all the applied external forces and mv the momentum. Since in robotics the masses of the links are constant (and considered as concentrated in the center of mass) we have the well known Newton equation $f = ma = m\ddot{p}_C$

- Euler's law of motion(conservation of the angular momentum $L = I\omega$)

$$\ni = \frac{d({}^0I^0\omega}{dt}$$

  where ${}^0I$ is the moment of inertia (rotational inertia) expressed in an inertial frame FF0 placed in the center of mass, ${}^0\omega$ the rotational veolcity, τ the sum of the torques applied to the rigid body.

Let us consider the generic $i-th$ link of the manipulator. Define, in an inertial frame, the following parameters:

m_i	mass of the link	$\ddot{p}_{C_i}$	acceleration of C_i
I_i	rotational inertia	$\ddot{\omega}_i$	acceleration of C_i
R_{i-1,C_i}	vector from FFi-1 to the CoM C_i	g_i	gravity acceleratio
r_{i,C_i}	vector from FFi tot he Com C_i	f_i	force applied on I
$r_{i-1,i}$	vector from FFi-1 to FFi	$-f_{i+1}$	force applied on I
$\dot{p}_{C_i}$	linear velocity of C_i	ν_i	torque applied on
$\dot{p}_i$	linear velocity of FFi	$-\nu_{i+1}$	torque applied on
ω_i	rotational velocity of the link		

NOTE: write them at some point please (slide 102)

Therefore, for each link we have:
Newton law:

$$f_i - f_{i+1} + m_i g = m_i \ddot{p}c_i$$

Euler law:

$$\nu_i + f_i \times r_{i-1,C_i} - \nu_{i+1} - fi + 1 \times r_{i,C_i} = \frac{d}{dt}(I_i\omega_i)$$

The Euler law results:

$$\begin{aligned}\nu_i + f_i \times r_{i-1,C_i} - \nu_{i+1} - fi + 1 \times r_{i,C_i} &= \frac{d}{dt}(I_i\omega_i) = I_i\dot{\omega}_i + \frac{d}{dt}(R_i^i I_i R_i^T)\omega_i \\ &= I_i\dot{\omega}_i + (\dot{R}_i^i I_i R_i^T + R_i^i I_i \dot{R}_i^T)\omega_i \\ &= I_i\dot{\omega}_i + S(\omega_i)(R_i^i I_i R_i^T(\omega_i) + R_i^i I_i R_i^T S(\omega_i)\omega_i \\ &= I_i\dot{\omega}_i + \omega_i \times (I_i\omega_i) \qquad [S(\omega_i)\omega_i = \omega_i \times \omega_i = 0]\end{aligned}$$

slide 104-105-106-107 couldn't keep up
After forward and backward recursion, we obtain forces/torques applied on the links. We require torques to be

applied by the motors. The generalized force τ_i applied on th $i-th$ joint by the link L_{i-1} can be computed as the projection of f_i or ν_i along the z_{i-1} axis. We have:

$$\tau_i = \nu_i^T z_{i-1}$$
$$\tau_i = f_i^T z_{i-1}$$

Friction may be considered at this point, for example

$$\tau_i = \nu_i^T z_{i-1} + \mu_i \dot{q}_i$$
$$\tau_i = f_i^T z_{i-1} + \mu_i \dot{q}_i$$

Overall algorithm

1. given joint poistions, velocities and accelerations link velocities and accelerations are computed through forward recursion
2. forces and torques are computed through backward recursion
3. joint torques are computed through force/torque projection

This algorithm allows for a variety of useful outputs: slide 113

2.3 Dynamic model in the work-space

It may be of interest to obtain the dynamic model in the work-space and not in the joint-space. For the sake of simplicity, let us consider a 6 dof manipulator, not in a singualr configuration. Since $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau + J(q)^T F_a$, then

$$\ddot{q} = M^{-1}(q)\left[\tau + J^T(q)F_a - C(q,\dot{q})\dot{q} - D\dot{q} - g(q)\right]$$

Moreover, by deriving $\dot{x} = J(q)\dot{q}$ with respect to time, we have:

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

Therefore

$$\hat{M}(x)\ddot{x} + \hat{C}(c,\dot{x})\dot{x} + \hat{D}\dot{x} + \hat{g}(x) = F + F_a \quad \tau = J^T F$$

with

$$\hat{M} = (JM^{-1}J^T)^{-1}$$
$$\hat{C} = \hat{M}$$

Chapter 3

Control of robot manipulators

3.1 Nominal case

All the parameters are known exactly and they are assumed to be constant in time.
Control problem: definition of the input signals for the joints (e.g. torques or actuator input voltages) in order to achieve a predefined behaviour for the manipulator. The achievable performances can vary due to:

- the many control techniques available to solve such a problem
- the hardware used to implement the control algorithms
- the mechanical configuration of the robot (anthropomorphic, cartesia, ...)

The robot performances are mainly influeneced by the mechanical design and by the actuation system. For example, gearboxes "decouple" the load from the motor (disturbances on the load are diminished towards the motor through the gearbox), however they usually introduce non linear effects such as dead-zones, friction, elasticity,...; Direct Drive motors (can provide high torque with low speed without reduction gears) ensure better performances and do not introduce non-linearities in the transmission chain; however a more relevant dynamic coupling between joints is present.

Control problems:

- Control of the robot's motion (position control schemes)
  - joint-space control
  - workspace control
- Control of the interaction with the workspace (force control schemes)

Control schemes:

- Decentralized (or independent) control schemes (SISO)
- Centralized control schemes (MIMO)

Dynamic model of a manipualator:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau + J^T(q)F_a$$

Control problem: define the generalized forces τ to be applied to the joints in order to obtain a desired trajectory $q_d(t)$
In non-linear control schemes it is best to choose a sampling frequency as large as possible

3.2 Decentralized position control

Actuators: velocities $\omega_m(t)$, positions q_m, torques $\tau_m(t)$
Gearboxes: reduction ratio K_r
Joints: velocities $\omega(t)$, positions q, generalized forces $\tau(t)$

$$K_r\omega = \omega_m \quad K_r q = q_m \quad \tau_m = K_r^{-1}\tau$$

K_r is a diagonal matrix with elements $\gg 1$ we can observe that:

$$\begin{aligned} q &= K_r^{-1} q_m \\ \dot{q} &= K_r^{-1}\dot{q}_m \\ \ddot{q} &= K_r^{-1}\ddot{q}_m \\ \tau &= K_r\tau_m \end{aligned}$$

we get

$$M(q)K_r^{-1}\ddot{q}_m + C(q,\dot{q})K_r^{-1}\dot{q}_m + DK_r^{-1}\dot{q}_m + g(q_m) = K_r\tau_m$$

and therefore

$$K_r^{-1}M(q)K_r^{-1}\ddot{q}_m + K_r^{-1}C(q,\dot{q})K_r^{-1}\dot{q}_m + K_r^{-1}DK_r^{-1}\dot{q}_m + K_r^{-1}g(q_m) = \tau_m$$

The diagonal of the matrix $M(q)$ is composed by two kinds of elements

- inertia terms that do not depend on the robot's configuration
- terms that depend on the robot's configuration

Therefore:

$$M(q) = \bar{M} + \Delta M(q)$$

where $\bar{M}$ is a diagonal matrix with constant elements (i.e. the mean values of the joints inertia). From the robot dynammic model it follows that

$$\tau_m = (K_r^{-1}\bar{M}K_r^{-1})\ddot{q}_m + D_m\dot{q}_m + d$$

having renamed

$$\begin{aligned} D_m &= K_r^{-1}DK_r^{-1} \\ d &= (K_r^{-1}\Delta M(q)K_r^{-1})\ddot{q}_m + (K_r^{-1}C(q,\dot{q})K_r^{-1})\dot{q}_m + K_r^{-1}g(q_m) \end{aligned}$$

D_m is a matrix collecting the motors friction coefficients, d is a term that can be considered as a disturbance that collects nonlinearities and coupling effects. The disturbance terms are divided by K_r^2 and therefore greatly reduced (the gravity is only divided by K_r) Based on this we can design a linear controller, with the hope that d may be small enough. d grows with the speed and acceleration of the robot.

How to reduce the effects of the disturbance d:

- high gain
- integral action to annhiliate the steady state error (e.g. the effect of the gravitational term)

Possible solution: PI controller

$$C(s) = K_c\frac{1+sT_c}{s}$$

3.2.1 possible control scheme

A cascade configuration can be employed If inertia is constant, current control implies acceleration control as torque is directly proportional to acceleration.
In robotics (and in general in motion control systems), feedwarward control actions are usually adopted with very positive results:

- velocity ff action

- acceleration (torque) ff action

With ff actions, it is possible to compensate (partially) also some parts of the "distrubance" term $d(t)$. Since $d(t)$ is known from the dynamic model, given the desired position, velocity and acceleration signals it is possible to compute a ff control action to compensate for $d(t)$:

$$d_d = (K_r^{-1}\Delta M(q_d)K_r^{-1})\ddot{q}_{md+}(K_r^{-1}C(q_d,\dot{q}_d)K_r^{-1})\dot{q}_{md} + K_r^{-1}g(q_{md})$$

It is to be noted that in any case $d \neq d_d$

3.3 Centralized control

We introduce two centralized control schemes:

3.3.1 PD + gravity compensation

Given a desired reference configuration q_d, the goal is to define a controller ensuring the global asympototic stability of the nonlienar dynamical system described by:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = \tau + J^T(q)F_a$$

For this purpose, let us define the error as

$$\tilde{q} = q_d - q$$

and consider a dynamic system with state x given by

$$x = \begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix}$$

The *direct Lyapunov method* is exploited for the control law definition. Let us consider the following candidate Lyapunov function:

$$V(x) = V(\tilde{q},\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}\tilde{q}^T K_P \tilde{q} > 0 \quad \forall \dot{q}, \tilde{q} \neq 0$$

Where K_P is a square $(n \times n)$ positive definite matrix. Usually, it is chosen as a diagonal matrix. Function $V(\tilde{q},\dot{q})$ is composed by two terms, the first expressing the kinetic energy of the system, the second can be interpreted as eleastic energy stored by springs with stiffness K_P. These springs are a physical interpretation of the proportional control.

We compute the time derivative of the candidate Lyapunov function:

$$\begin{aligned} \dot{V}(x) &= \dot{q}^T M\ddot{q} + \frac{1}{2}\dot{q}^T\dot{M}\dot{q} - \dot{q}^T K_P\tilde{q} \\ &= \dot{q}^T(u - C\dot{q} - D\dot{q} - g(q)) + \frac{1}{2}\dot{q}^T\dot{M}\dot{q} - \dot{q}^T K_P\tilde{q} \\ &= \frac{1}{2}\dot{q}^T(\dot{M} - 2C)\dot{q} - \dot{q}^T D\dot{q} + \dot{q}^T(u - g(q) - K_P\tilde{q}) \end{aligned}$$

note that $\frac{1}{2}\dot{q}^T(\dot{M} - 2C)\dot{q} = 0$ due to the choice of C(Christoffel symbols. Also, by choosing

$$u = g(q) + K_P\tilde{q} + K_D\dot{q}$$

where K_D is a positive definite matrix, we obtain a proportional and derivative control with gravity compensation, and we have:

$$\dot{V}(x) = -\dot{q}^T D\dot{q} - \dot{q}K_D\dot{q}$$

note that the term introduced by the derivative control acts like a friction, bringing the energy of the system towards 0 faster.

The PD control scheme can be interpreted as a spring and a damper.

$$\dot{V}(x) = -\dot{q}^T(D + K_D)\dot{q}$$

So far, we have guaranteed that the system stops, but we do not know where.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = K_P\tilde{q} - K_D\dot{q} + g(q)$$

we observe that all the terms other than the one with $\tilde{q}$, in steady state cancel out:

$$K_P\tilde{q} = 0$$

That is, as K_p is positive definite

$$q = q_d$$

A perfect compensation of the gravity term is necessary, otherwise it is not possible to guarantee the stability of the system.

The "PD+$g(q)$" approach can be extended to the position control in work-space.

3.3.2 Inverse dynamics control

The manipulator is considered as a nonlinear MIMO system described by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = u$$

or in short:

$$M(q)\ddot{q} + n(q,\dot{q}) = u$$

Two assumptions are necessary:

- the model is linear in the control input: $\dot{x} = f(x) + \alpha u(t)$
- The matrix $M(q)$ is invertible for any configuration fo the manipulator: $\exists f^{-1}(x)$

Let us choose the control input u as:

$$u = M(q)y + n(q,\dot{q})$$

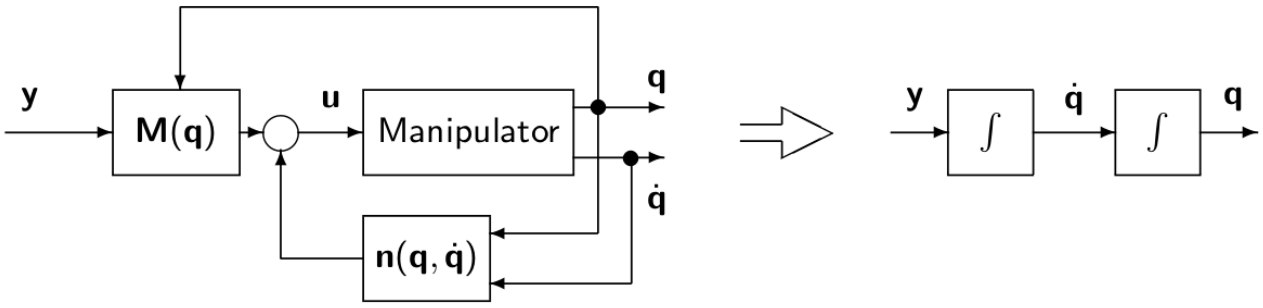
it follows that

$$M\ddot{q} + n = My + n$$

and thus

$$\ddot{q} = y$$

where y is the new input of the system



The system results to be completely decoupled and linear. However, the resulting system has two poles in the origin and is therefore unstable. In order to stabilyze the system y can be chosen as:

$$y = -K_D\dot{q} - K_P q + r$$

This means adding proportional and derivative control. From $\ddot{q} = y$ it follows:

$$\ddot{q} + K_D\dot{q} + K_P q = r$$

Which is asymptotically stable as long as K_P and K_D are positive-definite. If matrices K_P, K_D are diagonal matrices defined by

$$K_P = diag\{\omega_n^2\} \qquad K_D = diag\{2\delta\omega_n\}$$

the dynamics of the i-th component is characterized by the specified natural frequency and damping coefficient. A predefined trajectory $q_d(t)$ can be tracked by defining

$$r = \ddot{q}_d + K_D \dot{q}_d + K_P q_d$$

Then the dynamics of the tracking error is

$$\ddot{\tilde{q}} + K_D \dot{\tilde{q}} + K_P \tilde{q} = 0$$

which converges to zero.

Chapter 4

Advanced Position Control Schemes

4.1 Robust Control

Aims at compensating directly possible errors deriving from model errors and/or external disturbances. Some of the most known robust control techniques:

- H-infinity loop shaping
- Loop transfer recovery (LQG,LTR)
- Sliding mode control (SMC), a variation of variable structure control (VSS)

4.1.1 Variable Structure Control

Example

Consider a second order dynamic system described by

$$\ddot{x} = -\psi x \quad \text{with} \quad \psi \in \{a_1^2, a_2^2\} \quad (a_1^2 > 1 > a_2^2)$$

Assuming that ψ is a control input, by using the commutation law:

$$\psi = \begin{cases} a_1^2 \text{ if } x\dot{x} > 0 \\ a_2^2 \text{ if } x\dot{x} < 0 \end{cases}$$

An asymptotically stable system is obtained

Example

Consider the system

$$\ddot{x} - \xi\dot{x} + \psi x = 0 \quad \xi > 0$$

where we choose

$$\psi = \begin{cases} -a \quad (A) \\ a \quad (B) \end{cases}$$

The system is unstable (different signed elements of the equation).

In case (A), there are two real eigenvales, given by:

$$\lambda_{1,2} = \frac{\xi}{2} \pm \sqrt{\frac{\xi^2}{4} + a}$$

one is stable (negative) and one unstable (positive)

In case (B) there are complex conjugate eigenvalues with positive real part given that $a > \frac{\xi^2}{4}$. By choosing

$$\psi = \begin{cases} -a \quad \text{if} \quad xs < 0 \quad (A) \\ a \quad \text{if} \quad xs > 0 \quad (B) \end{cases} \qquad s = -cx + \dot{x}$$

where

$$c = \frac{\xi}{2} - \sqrt{\frac{\xi^2}{4} + a} < 0$$

is the inclination of the line representing the stable mode in the (A) case. The resulting system is asympotitically stable.

This type of control scheme is called Variable structure control or Sliding Mode Control (as the system goes towards a stable mode and "slides" towards the origin) and can be generalized:

We define a *Sliding Surface* $S(x) = 0$ based on which we decide which one of two control laws is used:

$$u_c = \begin{cases} K_1(x) & \text{if} \quad S(x) > 0 \\ K_2(x) & \text{if} \quad S(x) < 0 \end{cases}$$

Two steps are involved in the design of variable structure control:

- choice of the sliding surface $S(x)$ so that the control system has the desired dynamic behaviour
- choice of the control law u_i $(i = 1, \ldots, m)$ in order to force the state on the sliding surface $S_i(x)$ even in case of parameter variations or external disturbances.

Example

Let us consider the second order system

$$\ddot{y} + a\dot{y} + by = \psi + u \qquad G(s) = \frac{1}{s^2 + as + b}$$

where u is the control and ψ is an external disturbance, supposed to be bounded with bounded first order derivative

$$|\psi| < \Delta_0 \qquad |\dot{\psi}| < \Delta_1$$

Choice of the sliding surface:

$$S(x) = \dot{y} + cy = 0 \quad \text{with} \quad x = (y, \dot{y})$$

When the state $x = (y, \dot{y}) \in S(x) = 0$:

- the dynamics of the controlled system is exponentially asymptotically stable

$$\dot{y} = -cy \quad \Longrightarrow \quad y(t) = y(0)e^{-ct}, \quad c > 0$$

- the output of the system $y(t)$ goes to zero with a velocity that depends only the parameter "c", that is on the chosen sliding surface
- the behaviour of the controlled system does not depend neither on the external disturbance nor on changes in the parameters a and b. (ROBUST)

Choice of the control: It is possible to prove that the discontinuous control law

$$u(t) = -K \operatorname{sgn} S(x)$$

is able to force the state towards the sliding surface $S(x) = 0$ as long as $K > \Delta_0$, with total disturbance rejection. This control can only take the state of the system to 0. Substituting y and $\dot{y}$ with the tracking error: $x = (e, \dot{e})$, we can track a given trajectory with the chosen dynamics, with $e = y_d - y$. If $y_d \neq 0$ the set point is to be considered another disturbance to the system as it is an exogenous input to the system, requiring $y_d \times G(0)$ control effort to be compensated. The lower bound on K therefore becomes:

$$K > \Delta_0 + y_d G(0)$$

Continuous time

When the state x is on the sliding surface $S(x) = 0$, we have the *ideal sliding mode*:

- the control action $u(t)$ commutes with infinite frequency
- the oscillation produced on the output variable $y(t)$ has a null amplitude

Discrete time

If the VS control law is implemented as a discrete-time system we have:

- a finite commutation frequency
- a residual oscillation on the output variable $y(t)$ (*chattering*)

The amplitude of the residual oscillation is proportional to the sampling period T and the value of K of the control action

$$|y(t)| \leq KT$$

Therefore, it is not possible to force the state "exactly" on the surface $S(x) = 0$, but it is rather possible to keep it within a sufficiently small neighbourhood of it (*discrete sliding mode*)

4.1.2 The chattering problem

From a practical point of view, it is not possible to commute the control between the two values u^+, u^- at an infinite frequency. Therefore, "oscillations" are generated in the controlled system (or in the actuator), typically in a frequency range within the system's bandwidth[1]. Then, non modelled dynamics may be excited and undesired behaviours may be obtained. Some methods have been proposed to avoid this problem, among them:

- Boundary layers
- DIC (Discrete Integral Control)

Boundary Layers

A region with proper amplitude is defined around the sliding surface; in this region, the control action is not discontinuous, but rather it changes in a continuous way, linealry proportional to the error.

DIC (Discrete Integral Control)

Let us assme that the disturbance is bounded with bounded first derivative:

$$|\psi| < \Delta_0 \qquad |\dot{\psi}| < \Delta_1$$

In addition to the standard VS control term $(-k \operatorname{sgn} S(x))$, two additional contributions are used: a term proportional to the error $-\lambda S(x)$ and one proportional to the integral of its sign

$$\begin{cases} u(t) = -\lambda S(x) - k \operatorname{sgn} S(x) - \tilde{\psi}(t) \\ \dot{\tilde{\psi}}(t) = h \operatorname{sgn} S(x) \end{cases}$$

It is possible to prove the following:
If the parameters k, h and λ of the control law are chosen such that:

$$k > 0, \quad h > \Delta_1, \quad h\lambda k > \Delta_1^2(1 - ln2)$$

then the controlled system is globally asymptotically stable, and the state reaches the sliding surface $S(x) = 0$ in a finite time. Note that there is no lower bound on k, and it can therefore be chosen to be quite small to reduce the chattering effect. The lower bound on h, means that the implemented controller must be faster than the disturbance (have a higher derivative).

The resulting controller is composed of a proportional, integral and a switching component. If the proportional part is not present, i.e. $\lambda = 0$, the control action is still able to sabilize the system, but only for "small" initial conditions.

[1] if the control input is discontinuous, all the frequencies are excited and the natural frequencies of the system are necessarily excited

4.1.3 Robust control of industrial manipulators

In general, it is difficult to compensate exactly the dynamics of a robot manipulator. In practice, what we get is only a partial compensation. Let us consider the inverse dynamics control:

$$u = \hat{M}(q)y + \hat{n}(q, \dot{q})$$

where $\hat{M}$ and $\hat{n}$ represent the known part of the dynamic model. In general, uncertainties may be expressed as

$$\tilde{M} = \hat{M} - M \qquad \tilde{n} = \hat{n} - n$$

By using the above equation, we have

$$M\ddot{q} + n = \hat{M} + \hat{n}$$

Since matrix M is invertible, we have (adding and subtracting y)

$$\begin{aligned} \ddot{q} &= y + (M^{-1}\hat{M} - I)y + M^{-1}\tilde{n} \\ &= y - \eta \end{aligned}$$

with

$$\eta = (M^{-1}\hat{M} - I)y + M^{-1}\tilde{n}$$

note that, if $\hat{M} = M$ and $\hat{n} = n$ then $\eta = 0$ and $\ddot{q} = y$ By using the same control as in the ideal case:

$$y = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q)$$

The error dynamics $\tilde{q}$ is given by

$$\ddot{\tilde{q}} + K_D\dot{\tilde{q}} + K_P\tilde{q} = \eta$$

and it is therefore not possible to guarantee that $\tilde{q}$ goes to zero (η depends on the uncertainties of the system) From

$$\ddot{q} = y - \eta$$

we have

$$\dddot{\tilde{q}} = \ddot{q}_d - y + \eta$$

Therefore, by defining $\xi = \begin{bmatrix} \tilde{q}^T & \dot{\tilde{q}}^T \end{bmatrix}^T$ as the state vector, we obtain the first order differential equation

$$\dot{\xi} = H\xi + D(\ddot{q}_d - y + \eta)$$

with

$$H = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{2n \times 2n} \qquad D = \begin{bmatrix} 0 \\ I \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

The problem with tracking a given trajectory $(q_d, \dot{q}_d, \ddot{q}_d)$ may be solved by designing a control law y that stabilizes the error dynamics ξ, that is nonlinear and time variant (because of η). Because of the properties of any assigned trajectory (boundedness of velocity/acceleration) and considering the properties of the dynamic model fo a robot manipulator(boundedness) it is possible to estimate the range of variablity of the uncertainties η affecting the error dynamics.

In order to compensate for the uncertainty term η, we define a control law as:

$$y = \ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q} + w = \ddot{q}_d + w + \begin{bmatrix} K_P & K_D \end{bmatrix} \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}$$

the error dynamics in this case results

$$\dot{\xi} = H\xi + D(\eta - w)$$

where the eigenvalues of the matrix

$$\tilde{H} = (H - DK) = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}$$

have negative real part (as K_P, K_D are positive definite) By properly choosing the two matrices K_P, K_D for example as

$$K_P = diag\{\omega_{n1}^2, \ldots, \omega_{nn}^2\} \qquad K_D = diag\{2\delta_1\omega_{n1}, \ldots, 2\delta\omega_{nn}\}$$

it is possible to obtain a desired (and decoupled) behaviour for the linear part of the error dynamics. Notice that if $\eta = 0$, then by choosing $w = 0$ we obtain the previous control scheme. Viceversa, if $\eta \neq 0$, a proper value for w must be defined. The Lyapunov method is adopted, choosing as a candidate Lyapunov function

$$V(\xi) = \xi^T Q \xi > 0 \quad \forall \xi \neq 0$$

where Q is a symmetric, positive definite matrix By derivation, since $\dot{\xi} = \tilde{H}\xi + D(\eta - w)$, we have:

$$\begin{aligned} \dot{V} &= \dot{x}^T Q\xi + \xi^T Q\dot{x} \\ &= (\tilde{H}\xi + D(\eta - w)^T Q\xi + \xi^T Q/\tilde{H}\xi + D(\eta - w)) \\ &= \xi^T(\tilde{H}^T Q + Q\tilde{H})\xi + 2\xi^T QD(\eta - w) \\ &= -\xi^T P\xi + 2\xi^T QD(\eta - w) \\ &= -\xi^T P\xi + 2z^T(\eta - w) \end{aligned}$$

where

$$z = d^T Q\xi$$

and, since $\tilde{H} < 0$, the equation

$$\tilde{H}^T Q + Q\tilde{H} = -P, \quad \forall P > 0$$

has a unique solution Q, symmetric and positive definite. Notice that the first term is negative definite. We need to prove negativity of the second term. The vector z is known, therefore we can choose the control input:

$$w = \frac{\rho}{\|z\|} z \quad \rho > \|\eta\|$$

we obtain

$$z^T(\eta - w) = z^T\eta - \frac{\rho}{\|z\|} z^T z \leq \|z\|\|\eta\| - \rho\|z\| \leq 0$$

and thus

$$\dot{V} < 0$$

Overall control scheme: This is akin to sliding mode control as the direction of z cannot be assumed to be continuous. The sliding subspace $S(\xi) = D^T Q\xi = 0$ depends on the choice of the matrix Q. The chattering problem may be solved e.g. by the 'boundary layers' method:

$$w = \begin{cases} \frac{\rho}{\|z\|} z & \text{when } \|z\| \geq \epsilon \\ \frac{\rho}{\epsilon} z & \text{when } \|z\| < \epsilon \end{cases}$$

Discrete Integral Control may exhibit some unexpected behaviour and implementative challenges due to the complexity of the system.

4.2 Adaptive Control

Aims at modifying the control parameters in order to adapt to different working conditions, or to different values of the model parameters. There are two main families of adaptive controllers:

- Self Tuning Regulator (indirect)
- Implicit regulator (direct)

We assume the dynamics of the process to be linear wrt the parameters to be estimated, and we require the change of parameters to be slower than the dynamics of the system.

4.2.1 Robot parameters

For design purposes:

- kinematic parameters are assumed to be known (DH parameters, 3 per link)
- uncertain (constant) parameters that can be identified off line: masses(1), positions of CoMs(3), inertia matrices(6) (10 per link)
- parameters that are (slowly) varying during operation: viscous, dry and stiction friction at each joint (3 per joint)
- uknown and abruptly changing parameters: mass, CoM, inertia matrix of the payload wrt the tool center point (10)

so for an n-dof robot: $(3 + 10 + 3) \times n + 10 = 16 \times n + 10$ parameters

The main goals of an adaptive control scheme may be defined as:

- given a twice differentiable desired joint trajectory $q_d(t), \dot{q}_d(t), \ddot{q}_d(t)$
- execute this trajectory under large dynamic uncertainties $\tilde{\alpha} = \alpha - \hat{\alpha}$ with a tracking error that vanishes asymptotically:
$$\tilde{q} = q_d - q \to 0, \quad \dot{\tilde{q}} = \dot{q}_d - \dot{q} \to 0$$
guaranteeing global stability
- paramenter identification (i.e. $\tilde{\alpha} \to 0$) is not of particular concern. If the desired trajectory is persistently exciting, one obtaines parameter identification as a by-product ($\tilde{\alpha} \to 0$)
- indirect adaptive control schemes are more complex, but in principle allow also systematic convergence of dynamic coefficients to their true values

It is always possible to write the Euler-Lagrange formulation of the dynamics of an n dof robot as

$$Y(q, \dot{q}, \ddot{q})\alpha = u$$

where:

- vector α contains only uknown or uncertain coefficients
- each component of α is in general a combination of the robot physical parameters
- the model regression matrix Y depends: linearly on $\ddot{q}$, quadratically on $\dot{q}$, nonlinearly on q

In general, only a partial knowledge of the dynamic parameters is available, and therefore we may describe the dynamics of a robot as:

$$\hat{M}(q)\ddot{q} + \hat{C}(q, \dot{q})\dot{q} + \hat{D}\dot{q} + \hat{g}(q) = Y(q, \dot{q}, \ddot{q})\alpha = u$$

where $\hat{M}, \hat{C}, \hat{D}, \hat{g}$ estimate α. Some controllers are better suited to adaptive control with robot manipulators.

4.2.2 Inverse dynamics feedforward + PD

Given a desired trajectory it is possible to implement an inverse dynamics ff + PD control action defined as:

$$u = M(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + D\dot{q}_d + g(q_d) + K_P\tilde{q} + K_d\dot{\tilde{q}}$$

being $\tilde{q}$ and $\dot{\tilde{q}}$ the position and velocity errors. Since the dynamic parameters are in general not exactly known, only an approximation of the ff term can be computed. It can be shown that the control law leads to an error dynamics described by

$$\ddot{\tilde{q}} + K_d\dot{\tilde{q}} + K_P\tilde{q} = \eta$$

being η a nonlinear function of the parameter uncertainties and of the nl dynamics of the robot, so the error does not vanish asymptotically

4.2.3 NL control based on fb linearization

The error dynamics behaves analogously to the previous case, and does not vanish asymptotically.

4.2.4 Nonlinear tracking control law without cancelations

Let us consider a candidate control law:

$$u = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + D\dot{q}_d + g(q) + K_p\tilde{q} + K_d\dot{\tilde{q}}$$

that is a nonlinear trajectory tracking control law having global asymptotic stabilization properties. In principle, the control law could be made adaptive by defining

$$\dot{\hat{\alpha}} = \gamma(\hat{\alpha}q, \dot{q}, q_d, \dot{q}_d)$$

where $\gamma(\cdot)$ is a proper update law for the robot's parameters

It can be shown that with this control law the velocities could track the desired ones (eventually with zero error), but a permanent residual position error may occur if parameters are not exactly known. A possible solution is to modify the velocity reference as

$$\dot{q}_d \to \dot{q}_r = \dot{q}_d + \Lambda(q_d - q), \quad \Lambda > 0$$

where ofthe the value $\Lambda = K_d^{-1}K_p$ is chosen

In robotics, in order to design an adaptive control law we exploit the property of the dynamic model to be linear wrt the parameters

$$Y(q,\dot{q},\ddot{q})\alpha = u$$

Let us consider the control law:

$$u = M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + D\dot{q}_r + g(q) + K_D\sigma$$

where

$$\dot{q}_r = \dot{q}_d + \Lambda\tilde{q}$$
$$\ddot{q}_r = \ddot{q}_d + \Lambda\dot{\tilde{q}}$$

and K_D a positive definite matrix. The term $K_D\sigma$ is equivalent to a PD control action on the tracking error assuming

$$\sigma = \dot{q}_r - \dot{q} = \dot{\tilde{q}} + \Lambda\tilde{q}$$

The overall dynamics is the described by

$$M(q)\dot{\sigma} + C(q,\dot{q})\sigma + D\sigma + K_D\sigma = 0$$

To prove stability of the system we use the Lyapunov method. We choose as a candidate Lyapunov function

$$V(\sigma,\tilde{q}) = \frac{1}{2}\sigma^T M(q)\sigma + \frac{1}{2}\tilde{q}^T A\tilde{q} > 0 \quad \forall\sigma,\tilde{q} \neq 0$$

where A is positive definite. The time derivative of V is

$$\begin{aligned}
\dot{V} &= \sigma^T M(q)\dot{\sigma} + \frac{1}{2}\sigma^T\dot{M}(q)\sigma + \tilde{q}^T A\dot{\tilde{q}} \\
&= \sigma^T[-C\sigma - D\sigma - K_D\sigma] + \frac{1}{2}\sigma^T\dot{M}\sigma + \tilde{q}^T A\dot{\tilde{q}} \\
&= \frac{1}{2}\sigma^T(\dot{M} - 2C)\sigma - \sigma^T D\sigma - \sigma^T K_D\sigma + \tilde{q}^T A\dot{\tilde{q}} \qquad \sigma = \dot{\tilde{q}} + \Lambda\tilde{q} \\
&= -\sigma^T D\sigma - (\dot{\tilde{q}} + \Lambda\tilde{q})^T K_D(\dot{\tilde{q}} + \Lambda\tilde{q}) + \tilde{q}^T A\dot{\tilde{q}} \qquad A = 2\Lambda K_D \\
&= -\sigma^T D\sigma - \dot{\tilde{q}}^T K_D\dot{\tilde{q}} - \tilde{q}^T\Lambda K_D\Lambda\tilde{q} < 0
\end{aligned}$$

Therefore $\dot{V} < 0$, and $\dot{V} = 0$ only when $\tilde{q} = \dot{\tilde{q}} \equiv 0 \implies [\tilde{q}^T, \sigma^T]^T = 0$ is GAS. The system evolves on $\sigma = 0$ without a high.frequency control action.

On the basis of the previous result, let us consider the control law

$$u = \hat{M}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{D}\dot{q}_r + \hat{g}(q) + K_D\sigma = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\alpha} + K_D\sigma$$

computed using the estimation $\hat{\alpha}$ of the parameters of the dynamic model. Notice that Y does not depend on $\ddot{q}$. With this control, the overall dynamics are described by:

$$\begin{aligned} M(q)\dot{\sigma} + C(q,\dot{q})\sigma + D\sigma + K_D\sigma &= -\tilde{M}(q)\ddot{q}_r - \tilde{C}(q,\dot{q})\dot{q}_r - \tilde{D}\dot{q}_r - \tilde{g}(q) \\ &= -Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{\alpha} \end{aligned}$$

where

$$\tilde{M} = \hat{M} - M, \qquad \tilde{C} = \hat{C} - C, \qquad \tilde{g} = \hat{g} - g, \qquad \tilde{g} = \hat{g} - g, \qquad \tilde{\alpha} = \hat{\alpha} - \alpha$$

In order to prove the stability of the system under this new control law, we once again use the Lyapunov method:

$$V(\sigma,\tilde{q},\tilde{\alpha}) = \frac{1}{2}\sigma^T M(q)\sigma + \tilde{q}^T\Lambda K_D\tilde{q} + \frac{1}{2}\tilde{\alpha}^T K_\alpha\tilde{\alpha} > 0 \qquad \forall\sigma,\tilde{q},\tilde{\alpha} \neq 0$$

Where K_σ is a symmetric positive definite matrix. It can be shown that the time derivative of V is now :

$$\dot{V} = -\sigma^T D\sigma - \dot{\tilde{q}}^T K_D\dot{\tilde{q}} - \tilde{q}^T\Lambda K_D\Lambda\tilde{q} + \tilde{\alpha}^T(K_\alpha\dot{\tilde{\alpha}} - Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma)$$

The function $\dot{V}$ is negative if the parameters are "adapted" according to

$$\dot{\hat{\alpha}} = K_\alpha^{-1}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma$$

As a matter of fact, we have

$$\dot{V} = -\sigma^T D\sigma - \dot{\tilde{q}}^T K_D\dot{\tilde{q}} - \tilde{q}^T\Lambda K_D\Lambda\tilde{q} < 0$$

Therefore, with the control law

$$u = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\alpha} + K_d(\tilde{q} + \hat{\Lambda}\dot{\tilde{q}})$$

where the parameters are computed according to

$$\dot{\hat{\alpha}} = K_\alpha^{-1}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma$$

The overall dynamics converges to

$$\sigma = 0, \quad \tilde{q} = 0 \quad (\dot{\tilde{q}} = 0)$$

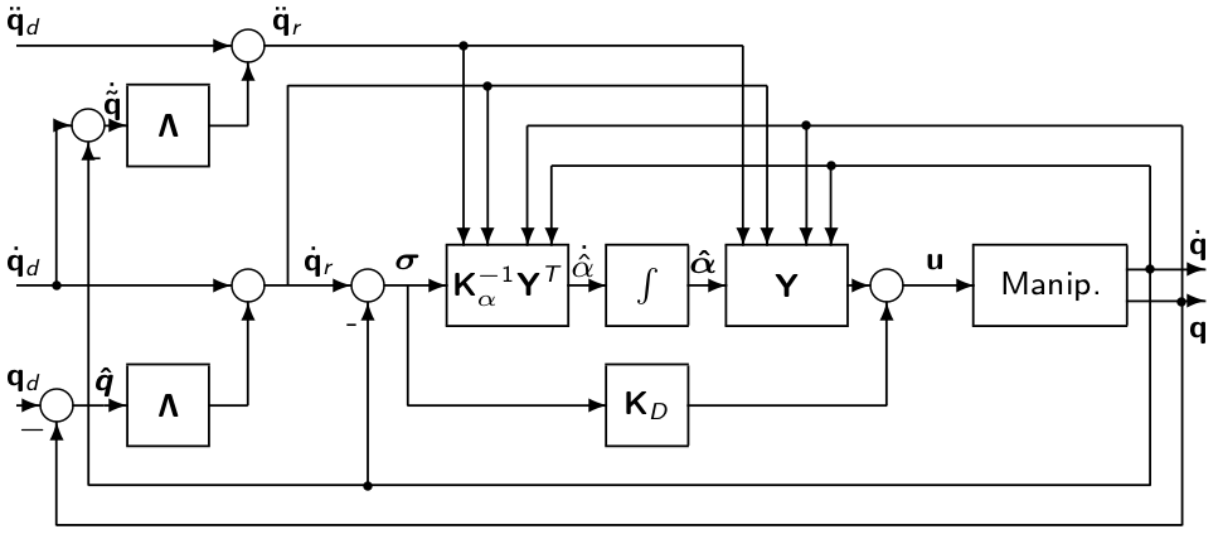
and then $\tilde{q}, \dot{\tilde{q}}$ converge to zero and moreover $\hat{\alpha}$ is bounded. Notice that from

$$M(q)\dot{\sigma} + C(q,\dot{q})\sigma + D\sigma + K_D\sigma = -Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\tilde{\alpha}$$

in steady state ($\sigma = \dot{\sigma} = 0$)) we have

$$Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)(\hat{\alpha} - \alpha) = 0 \qquad \not\Longrightarrow \qquad \hat{\alpha} \to \alpha$$

It is not possible to conclude that $\hat{\alpha} = \alpha$, but only that $\tilde{\alpha} \in Null\{Y\}$ The overall control scheme is



This is an implicit adaptive control. There are three contributions:

1. term $Y\hat{@}a$, a control action similar to the inverse dynamics approach
2. term $K_D\sigma$, PD stabilizing action
3. estimated parameter vector $\hat{\alpha}$, updated according to a *gradient* method, matrix K_α defines the speed of convergence of the estimated parameters.

To sum up

Adaptive control:

- does not compensate directly external disturbance
- depends on the considered model
- tries to compensate the effects of disturbances by modifying the parameters
- "smooth" control action

Robust Control:

- direct compensation of disturbances
- chattering phenomenon

Chapter 5

Learning/Repetitive Control

If a robot has to repeat a given task cyclically, following every time the same trajectory, possible uncertainties in the system may be compensated by adopting learning or repetitive control schemes, by means of which the contrl system, by repeating the task, learns the optimal control input to be applied in order to have zero tracking error. There are learning or repetitive control schemes to solve

- trajectory tracking problems
- control to steady state configurations.

Two control actions:

- feedback, based on the error $e = y_d - y$
- feedforward, updated at each iteration

5.1 PD control without gragity compensation

Let us consider the following case:

- Control to arbitrary constant (equilibrium) configuration of a robot subject to gravity
  - without explicit knowledge of the dynamic coefficients and of the gravity term
  - without using "high gain" position control terms
- Use of an iterative control scheme based on
  1. PD control on the joint position error + constant compensation (ff) term
  2. iterative update of the ff term at each steady-state intermediate condition

Assumptions:

- dynamic model of the robot
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$
- boundedness of the gradient of the gravity term
$$\|\frac{\partial g}{\partial q}\| \leq \alpha$$
- joint-based PD controller (without g compensation)
$$u = K_p(q_d - q) - K_d\dot{q} \quad K_p > 0, K_d > 0$$
- it can be shown that the systeem is stable and that in steady-state we have:
$$q = \bar{q}, \quad \dot{q} = 0, \quad g(\bar{q}) = K_p(q_d - \bar{q}), \quad \tilde{q} = q_d - \bar{q}$$

Control scheme:

- at the $i-th$ iteration, the control action is

$$u(t) = \frac{1}{\beta} K_p(q_d - q(t)) - K_d\dot{q}(t) + u_{i-1} \quad \beta > 0, \quad q(0) = \bar{q}_{i-1}$$

  where

  - the term u_{i-1} is a proper, constant ff action
  - $u_0 = 0$ is the usual initial value
  - q_0 is the initial robot configuration

- at the steady-state of the i-th iteration we have

$$g(\bar{q}_i) = \frac{1}{\beta} K_p(q_d - \bar{q}_i) + u_{i-1}$$

- the ff action (for the next iteration) is updated as

$$u_i = \frac{1}{\beta} K_p(q_d - \bar{q}_i) + u_{i-1} \quad (= g(\bar{q}_i))$$

Theorem (sufficient condition for convergence)

If the following conditions:

1. $\lambda_{min}(K_p) \geq \alpha$
2. $0 < \beta \leq 0.5$

hold, then the sequence $\{q_0, \bar{q}_1, \bar{q}_2, \dots\}$ converges to q_d from any initial condition $(q_0, \dot{q}_0)$ (*global convergence*) Condition (1) is sufficient for guaranteeing the global asymptotic stability (a unique equilibrium point exists for the closed loop system) with the control law. The additional sufficient condition (2) guarantees the convergence of the iterative scheme and in particular that

$$\lim_{i \to \infty} u_i = g(q_d)$$

Proof

Define $\tilde{q} = q_d - q$. At the end of each iteration, because of the update law, we have $u_i = g(\bar{q}_i)$ and therefore

$$\|u_i - u_{i-1}\| = \|g(\bar{q}_i) - g(\bar{q}_{i-1})\| \leq \alpha \|\bar{q}_i - \bar{q}_{i-1} \leq \alpha(\|\tilde{q}_i\| + \|\tilde{q}_{i-1}\|)$$

5.1.1 Final comments

- convergence is achieved in few iterations
- the condition on the proportional gain is sufficient:

5.2 Repetitive control

5.2.1 Internal Model Principle

Accurate control can be achieved only if the control system encapsulates (either implicitly or explicitly) some representation of the process to be controlled.

Once the controlled system contains an internal model of the signal to be tracked, and *the overall system is stabilized*, perfect tracking is achieved.

5.2.2 Fourier Decomposition and IMP

Since our initial hypothesis states that we have to deal with periodic reference signals of known period τ, let's see how these signals can be modelled and how we can connect them with the IMP idea:

The Fourier Decomposition Theorem states that any periodic function (or periodic signal) can be decomposed into the sum of a (possibly infinite) set of simple oscillating functions represented by sines and cosines.

In particular, we can express a periodic reference signal $y_{ref}(t)$ with period τ as

$$y_{ref}(t) = \sum_{k=0}^{\infty} A_k sin(k\omega_\tau + \phi_k) \quad \text{with} \quad \omega_\tau = \frac{2\pi}{\tau}$$

In general, in most control applications we deal with (well known) simple signals. Therefore, we assume that the model of any of these (periodic) functions is knwon and represented by a pair of poles on the imaginary axis

$$\frac{1}{s^2 + (k\omega_\tau)^2} \rightarrow s = \pm jk\omega_\tau$$

According to the IMP, if any of these modes (coming from the Fourir Decomposition of the periodic reference) is embedded in the control systemm, perfect tracking will be achieved by the control system. In practice we need a system that acts as internal model for any periodic signal of period τ, thus a system having infinitely many poles on the imaginary axis at multiple integers of ω_τ. The transfer function of such a system can be represented as

$$\frac{1}{s\prod_{k=1}^{\infty}(s^2 + k^2\omega_\tau^2)}$$

A system with this dynamic characterization is called *Repetitive Compensator* and represents the core of any RC system. Let us consider the following scheme: It generates any periodic signal of period τ (with a suitable initial function stored in the delay block). It works as the internal model for periodic signals of period τ. The pure delay present in the positive feedback loop implements the dynamic system with the desired poles on the imaginary axis.

The repetitive compensator presents a closed loop trasnfer function

$$\frac{e^{-\tau s}}{1 - e^{-\tau s}}$$

The poles of this system are those complex numbers such that

$$e^{-\tau s} = 1$$

By setting $s = \sigma + j\omega$ it results

$$e^{-\tau\sigma}e^{-j\omega\tau} = 1 = e^{j2k\pi}$$

from which

$$\sigma = 0$$
$$\omega = \frac{2k\pi}{\tau} = k\omega_\tau, \quad k \text{ integer}$$

Thus exactly the infinite number of poles palced where needed. This proves that the repetitive compensator acts as internal model for any periodic signal of period τ. Stability analysis is necessary to evaluate the functionality of the scheme.

5.2.3 Nyquist-Driven Stability Analysis

In order to analyse the stabilty properties, we can consider the following system which has the same characteristic equations (and consequently the same poles) of the previous one. As a matter of fact, the overall transfer fucntion is given by

$$G_0(s) = \frac{e^{-\tau s}(P(s))}{1 + e^{-\tau s(P(s)-1)}}$$

and therefore, the characteristic equation is

$$1 + e^{-\tau s}(P(s) - 1) = 0$$

In order to analyse the stability properties, we can consider the following system which has the same characteristic equations (and consequently the same poles) of the previous one with the same characteristic equation as $G_0(s)$

$$H_0(s) = \frac{e^{-\tau s}(P(s) - 1)}{1 + e^{-\tau s(P(s)-1)}}$$

Nyquist Stability Criterion

The Nyquist Stavility Criterion is a very effective tool to determine whether a closed loop system is stable or not without computing explicitly the closed loop poles of the system, i.e. the roots of $1 + L(s) = 0$

The criterion states that the closed loop system with open loop tf $L(s)$ (stable), is stable iff the polar plot of $L(jw)$ does not encircle or touch the critical point -1 on the complex plane.

- The polar plot can be constructed by evaluating on the complex plane the image of $L(jw)$ parametrized for w ranging from 0 to $+\infty$ and then closing the plot in a specular way about the real axis
- the overall plot will always form a closed curve because it derives from a mapping that has a closd curve as domain (called *Nyquist Path*)
- The Nyquist path is a curve that has as interior the whole right half plane

If the loop transfer function is not rational but contains some delay term it is still worth to apply the Nyquist criterion for stability analysis of the closed loop systems. Let's consider a system with loop transfer function

$$L(s) = e^{-\delta s}L'(s)$$

The greater the value of the delay δ, the greater the negative phase shift. This behaviour can easily cause instability in the closed loop system. The polar plot of such a system presents rotational behaviour with increasing speed with growing values of ω. A very important corollary of the Nyquist criterion that will be used in the following states that a sufficient condition for stability of the closed loop system is

$$|L(j\omega)| < 1 \quad \forall\omega$$

By applying the Nyquist Criterion to the RC scheme, it follows that a sufficient condition for stability of the RC scheme is

$$\|P(j\omega) - 1\| < 1 \quad \forall\omega$$

This condition is never satisfied unless P is biproper (rel degree 0). However, it requires too much: tracking arbitrary periodic signals (even discontinuous ones).

Possible Remedies

The nonexistence of a repetitive controller for a strictly proper pant is not surprsing: if $P(s)$ has non null relative degee it "integrates" the input at least once, and hence the output will be smoothed out to some extent making it impossible to track a signal with an infintely sharp edge. To overcome this problem two solutions have been found:

1. Introduction fo a filter in the delay (often called Modified Repetitive Control Scheme)
2. Make it a discrete time system, that provides a natural upper bound to the maximum meaningful frequency value

5.2.4 Modified Repetitive Control System

The modified RC system presents a low-pass filter Q(s) that multiplies the pure delay in the repetitive compensator.

- Exact internal model is lost: the repetitive compensator acts as internal model for any periodic signal of period τ within a certain frequency range
- problems in high freq tracking
- the poles escape away from the imaginary axis for high values of ω

5.2.5 Discret Time Repetitive Control

The most commonly adopted solution is to make the system a digital system. Any periodic signal with period N (N samples) can be generated by an *N-step delay with positive feedback loop*. This means that such a system must represent the repetitive compensator in discrete time version. We can interpret fthe complex function z^{-1} as the one-step delay operator.

The discrete time repetitivecompensator has a closed loop transfer function

$$\frac{z^{-N}}{z^{-N}}$$

the poles of this system are those complex numbers such that $z^{-N} = 1$. By setting $z = \rho e^{-j\theta N}$ it results

$$\rho e^{j\theta N} = 1 = e^{j2k\pi}$$

from which

$$\rho = 1$$
$$\theta = \frac{2k\pi}{N}, \quad k \text{ integer}$$

Thus exactly N poles on the unit circle in the z-plane. In discrete time signal analysis these poles represent the simple oscillation modes whose linear combination can produce any periodic signal of period N (Discrete Fourier Decomposition). This proves that the discrete time repetitiev compensator acts as internal model for any periodic signal of period N. Note that tracking is only achievable at the sample points.

Repetitive Control applied in a nonlinear system framework has been studied and in some cases satisfactory results have been achieved. Since in most industrial robotics applications cyclic operations are needed, RC can be a robust solution for the accomplishment of these tasks considering the lack of knowledge of the parameters can be seen as a periodic disturbance as the reference is periodic.

Chapter 6

Force Control

Force control is rather difficult whith very stiff obects, therefore it is appropriate to introduce some compliance between the robot and the environment.

6.1 Passive Compliance

mechanical devices properly installed on the wrist of the robot, able to react to external forces (RCC: Remote Center of Compliance)

- advantages: low cost, simple devices
- low flexibility, use limited to specific applications

The center of compliance is a point in space where linear forces correspond to pure linear displacements and torques correspond to pure rotation.

6.2 Active Compliance

Achieved my means of suitable control laws: the robot reacts in a programmed manner to external forces applied to the end-effector (a force sensor is usually requirede).

- advantages: high operational flexibility
- drawbacks: computational complexity, an envirnment model could be required

Different techniques are available:

- Pure force control: (theoretically) used in quasi-static applications
- Active stiffness control: control of interaction forces is obtained on the basis of position control with an elastic model of the environment
- Impedance control: a desired dynamic behaviour of the end-effector is imposed (mechanical impedance: ratio between force and velocity)
- Hybride position/force control: the directions in the operational space are separated in position-controlled and force-controlled directions (on the basis of kinematic models of the task and of the dynamic model of the robot)

6.2.1 Stiffness control

Basic idea:
Force on the environment: $f_e = k_e(x - x_e)$

system dynamics: $m\ddot{x}+k_e(x-x_e)=f$ where x_e is the original position of the contact point of the environment, and x is the actual position of the contact. We choose to use a PD control:

$$m\ddot{x}+k_e(x-x_e)=f=k_p(x_d-x)-k_v\dot{x}$$
$$m\ddot{x}+k_v\dot{x}+(k_p+k_e)x=k_px_d+k_ex_e$$

In steady state

$$x=\frac{k_px_d+k_ex_e}{k_p+k_e}$$

and then

$$f_e=k_e\left[\frac{k_px_d+k_ex_e}{k_p+k_e}-x_e\right]=\frac{k_e}{k_p+k_e}k_p(x_d-x_e)$$

which is equivalent to two springs in series, a robot "spring" + and environment "spring"

- if $k_e \gg k_p$ then $f_e \approx k_p(x_d-x_e)$ (compliant robot)
- if $k_p \gg k_e$ then $f_e \approx k_e(x_d-x_e)$ (compliant environment)

We obtain force control by means of position control. The stiffness of the system is defined by the gain k_p

Let us consider now a n dof robot: Elastic environment:

$$w=\begin{bmatrix}f\\n\end{bmatrix}=\begin{bmatrix}K_f & 0\\0 & K_v\end{bmatrix}\begin{bmatrix}dp\\\omega dt\end{bmatrix}=K\begin{bmatrix}dp\\\omega dt\end{bmatrix}$$

By using work-space coordinates (RPY,...)

$$w=KT_A(x)dx=KT_A(x)(x_e-x)$$

With the analytic expression of the jacobian we have (from $J=T_AJ_A$ and $\tau=J^Tw=J_A^Tw_A$)

$$w_A=T_A^T(x)KT_A(x)dx=K_A(x)(x_e-x)$$

K_A is the (positive semi-definite) *stiffness matrix*: the environment may not be able to generate forces along all the directions. Note that this matrix is not constant as it depends on the position x In case it may be defined, K_A^{-1} is the *compliance matrix*. The range space of matrix K_A, i.e. $\mathcal{R}(K_A)$, defines the directions in the work space along which forces may be applied.
Dynamic model of the robot: $M(q)\ddot{q}+C(q,\dot{q})\dot{q}+D\dot{q}+g(q)=u+J^T(q)w$
Control PD+g(q): $u=g(q)+J_A^T(q)K_P\tilde{x}-J_A^TK_DJ_A(q)\dot{q}$
This control is obtained by transforming the standard PD+g(q) control into the joint space by the analytic Jacobian. As veolcities in the workspace are not measured, we make use of the equivalence $\dot{x}_A=J_A\dot{q}$.
in steady-state:

$$^T(q)K_P\tilde{x}=-J^T(q)w \quad (=-J_A^T(q)w_A=-J_A^T(q)T_A^Tw)$$

and, if the inverse of the Jacobian exists

$$\tilde{x}=-K_P^{-1}T_A^T(x)w=-K_P^{-1}w_A$$

The compliance is defined by K_P^{-1}. If it is diagonal, the compliance referred to forces is linear and configuration-independent; this is not true for the torques due to the presence of T_A. We have

$$\tilde{x}=-K_P^{-1}K_A(x)(x_e-x)$$

Stiffness control in joint space

6.2.2 Impedance control

The goal is not to track specific desired position/veolcity or force trajectories, but rather to define a dynamic relationship between velocities and forces. We define a desired work-space dynamics. We have:

$$\frac{w(s)}{\dot{x}(s)}=Z(s) \quad \text{or} \quad \frac{w(s)}{x(s)}=sZ(s)$$

By defining

$$sZ(s) = -(M_d s^2 + D_d s + K_d)$$

where M_d is the desired inertia matrix, D_d is the desired damping and K_d is the desired stiffness, we have

$$M_d\ddot{x} + D_d\dot{x} + K_d x = -w$$

that defines the desired behaviour of the manipulator

We have a two-step design procedure:

1) Linearization in the work-space

$$\hat{M}(x)\ddot{x} + \hat{C}(x,\dot{x})\dot{x} + \hat{g}(x) = F + F_a$$

where

$$\hat{M} = (JM^{-1}J^T)^{-1} = J^{-T}MJ^{-1}$$
$$\hat{C} = \hat{M}(JM^{-1}CJ^{-1} - dotJJ^{-1}) = J^{-T}CJ^{-1} - \hat{M}\dot{J}J^{-1}$$
$$\hat{g} = \hat{M}JM^{-1}g = J^{-T}g$$

by defining

$$F = [\hat{M}(x)y + \hat{C}(x,\dot{x})\dot{x} + \hat{g}(x) - F_a] = [\hat{M}(x)y + \hat{n}(x,\dot{x}) - F_a]$$

we obtain the linear model

$$\ddot{x} = y$$

2) Definition of the desired impedance bhaviour

$$y = \ddot{x}_d + M_d^{-1}[D_d(\dot{x}_d - \dot{x}) + K_d(x_d - x) + F_a]$$

from Which

$$M_d(\dot{x}_d - \ddot{x}) + D_d)\dot{x}_d - \dot{x}) + K_d(x_d - x) = -F_a$$

or

$$M_d\ddot{\tilde{x}} + D_d\dot{\tilde{x}} + K_d\tilde{x} = -F_a$$

F_a must be added because if it is not, the robot will go to $\tilde{x} = 0$ ignoring external forces, leaind to potential damage In conclusion, the overall control input F is:

$$F = \hat{M}\ddot{x}_d + \hat{M}M_d^{-1}[D_d(\dot{x}_d - \dot{x}) + K_d(x_d - x) + F_a] + \hat{n}(x,\dot{x}) - F_a$$

note that if we choose $M_d = M(= J^{-\hat{T}}MJ^{-1})$ then the control law is greatly simplified since

$$F = \hat{M}\ddot{x}_d + [D_d(\dot{x}_d - \dot{x}) + K_d(x_d - x) + F_a] + \hat{n}(x,\dot{x})$$

and it does not even need a F/T sensor to be implemented. Since typically the control actino is applied at the joint level, we have

$$u = J^T F$$

Choice of the desired impedance

The robot should:

- avoid high impact forces due to uncertainties on position/shape of the environment
- adapt to the stiffness properties of the environment
- mimic a "human" behaviour: fast and rigid in free-space motion, slow and compliant iduring the approaching and contact phases

Then:

- high values of $M_{d,i}$ and small values of $K_{d,i}$ along directions in the work space where contacts are expected
- high values of $K_{d,j}$ and small values of $M_{d,j}$ along free directions in the woek space (no contacts)
- if the environment is stiff, small values of $K_{d,i}$
- the values of $D_{d,k}$ are used to modify the transient phases

6.3 Hybrid position/force control

When a task involves at the same time both motion and force application, it often happens that directions along which forces are applied are different with respect to directions along which motion takes place.

It is always possible to deifine, in a proper reference frame

- Natural constraints: directions along which the task gives constraints in position (velocity) and force
- Artificial constraints: directions along which it is possible to assign desired constraints with the control system.

These constraints are complementary and altogether define the 12 dofs of a given task: 6 dof in terms of position/velocity and 6 dof in terms of forces/torques.

The artificial constraints define directions in the workspace along which it is possible to assign desired values of a given variable (velocity or force). It is then simple to design a "hybrid" control structure able to impose proper values alonog these directions only.

Consider now the velocity domain v. Given the natural and artificial constraints, it is possble to decompose v in two components:

$$v = v_a + v_d$$

where the "free" directions (velocities) v_a are complementary to the directions in v_n, that is

$$v_a^T v_n = 0$$

Note that the vectors cannot be said to be orthogonal as their norm cannot be defined, due to them being comprised of both linear and angular parts, leading to the vector space of the velocities v not having a norm. Given a desired task, it is possible to define a base matrix A_v that describes all the free velocities v_a:

$$A_v \in \mathbb{R}^{6\times n_a} \quad \text{such that} \quad v_a = A_v y$$

and a base matrix N_v that describes all the remaining components

$$N_v \in \mathbb{R}^{6\times 6-n_a} \quad \text{such that} \quad v_n = N_v y$$

with

$$A_v^T N_v = 0$$

Given the matrix A_v, the goal is to control motions in $\mathcal{R}(A_v)$

Therefore, it is desired to control the components of a generic velocity vector v satisfying

$$A_v^T v = A_v^T A_v y + A_v N_v z$$

from which by setting $v_n = 0$

$$v_a = A_v y = A_v (A_v^T A_v)^{-1} A_v^T v = \Sigma_v v$$

Matrix Σ_v is known as the *selection matrix* and, if defined with respect to the contact reference frame, results a diagonal matrix with elements "1" and "0" on the diagonal
NB: Σ_v is a projector matrix from $\mathbb{R}^6$ in $\mathcal{R}(A_v)$

Similarly, in the force domain it is possible to define a base matrix A_f of the artificial contstrainsts, and the force components that it is possible to control are described by

$$\Sigma_f = A_f (A_f^T A_f =^{-1}) A_f^T$$

In order to establish a Euclidean scalar product in the force/velocity vector spaces it is necessary to define specific metrics in these spaces. Once the selection matrices Σ_v, Σ_f have been defined, it is possible to use these general control schemes:

- Control in joint space:
- Control in work space

Chapter 7

Trajectory planning for Robot Manipulation

The planning modalities for trahectories may be quite different:

- point-to-point
- with pre-defined path

Or:

- in the joint space
- in the work space, either defining some points of interest (initial and final points, via points) or the whole geometric pat $x = x(t)$

For planning a desired trajectory, it is necessary to specify two aspects:

- geometric path
- motion law

wiht constraints on the continuity of the trajectory and on its time derivatives up to a given degree.

7.0.1 Geometric path and motion law

The geometric path can be defined in the work-space or in the joint-space. Usually, it is expressed in a parametric form as

$$\begin{aligned} p &= p(s) \qquad \text{work-space} \\ q &= q(\sigma) \qquad \text{joint-space} \end{aligned}$$

Together, the geometric path and the motion law adefine the trajectory $p(s(t))$, or in short $p = p(t)$. Note that, with the chain rule,

$$\dot{p} = \frac{dp}{ds}\dot{s}$$
$$\ddot{p} = \frac{dp}{ds}\ddot{s} + \frac{d^2p}{ds^2}\dot{s}^2$$

Input data to an algorithm for trajectory planning are:

- data defining the path
- geometrical constraints on the path
- constraints on the mechanical dynamics
- constraints due to the actuation system

Output: the trajectory in the joint or work-space, given as a sequence (in time) of the position, velocity and acceleration values.

1. Kinematic constraints

Constraints on the minimum/maximum values for the derivatives up to a given order:

$$q_{min}^{(i)} \leq q^{(i)} \leq q_{max}^{(i)} \quad i = 1, \ldots, n$$

in some cases $|q_{min}^{(i)}| \neq |q_{max}^{(i)}$ Constraints on torque may be "converted" into kinematic constraints. Cosntraining jerk physically means limiting vibrations.

2. Constraints on the order of continuity

We have to choose a proper order n of continuity for the motion law, i.e.

$$q(t) \in C^n$$

that is the n-th derivative $q^{(n)}(t)$ of function $q(t)$ is continuous, and $q^{(n+1)}(t)$ is composed of constant segments

- the higher n, the "smoother" the profile: the frequency bandwidth will be reduced (better for vibration effects)
- For control purposes (ff) the continuity of the trajectory should guarantee a limited control action
- in high dynamics mechanical system, continuity of (at least) acceleration is required

3. Dynamic consraints

These constraints aim to reduce possible vibration effects on the mechanical load.

7.1 Joint-space trajectories

Trajectories are specified by defining some characteristic points:

- directly assigned by some specifications
- assigned by defining desired configurations x in the woek-space, which are the converted in the joint-space using the inverse kinematic model.

The algorithm that computes a function $q(t)$ interpolating the given points is characterized by the following features:

- trajectories must be computationally efficient
- the positino and velocity profiles must be continuous functions of time
- undesired effects (such as non regular curvatures) must be minimized or completely avoided.

Single joint trajectories may be classified as

- Polynomial trajectories
- Trigonometric trajectories (cycloidal, harmonic)
- Trajectories based on Fourier series expansion
- Composed trajectories (Trapezoidal velocity, double s, ...)
- Spline (B-Splines, NURBS,...)
- Trajectories based on FIR (Finite Impulse Response) filters

7.1.1 Polynomial trajectories

In the most simple cases, a trajectoru is specified by assigning initial and final conditions on: time (duration), position, velocity, acceleration, ... Then, the problem is to determine a function

$$q = q(t) \quad \text{or} \quad q = q(\sigma), \quad \sigma = \sigma(t)$$

so that those conditions are satisfied

This is a boundary condition problem, that can be easily solved by considering polynomial functions such as:

$$q(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

The degree n (3,5,...) of the polynomial depends on the number of boundary conditions that must be verified and on the desired "smoothness of the trajectory"

In general, besides the initial and final values, other constraints could be specified on the values of some time-derivatives in generic instans $t_j \in [t_i, t_f]$. In other terms, one could be interested in defining a polynomial function $q(t)$ whose k-th derivative has a specified value $q^k(t_j)$ at a given time instant t_j. Mathematically, these conditions are expressed as:

$$k!a_k + (k+1)!a_{k+1}t_j + \cdots + \frac{n!}{(n-k)!}a_n t_j^{n-k} = q^k(t_j)$$

or, expressing all condiitions in matrix form:

$$Ma = b$$

Third-order polynomial trajectories

Given an initial and final instant, t_i, t_f, a trajectory may be specified by assigning initial and final conditions on:

- initial position and veloccity
- final position and velocity

There are four boundary conditions, and therefore a polynomial of degree (at least) 3 must be considered

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

in order to solve these equations, let us assume for the time being that $t_i = 0$. Therefore:

$$a_0 = q_i$$
$$a_1 = \dot{q}_i$$
$$a_2 = \frac{-3(q_i - q_f) - (2\dot{q}_i + \dot{q}_f)t_f}{t_f^2}$$
$$a_2 = \frac{2(q_i - q_f) - (\dot{q}_i + \dot{q}_f)t_f}{t_f^3}$$

The results obtained can be generalized tot he case in whic $t_i \neq 0$:

$$q(t) = a_0 + a_1(t - t_i) + a_2(t - t_i)^2 + a_3(t - t_i)^3$$

Often, a trajectory is assigned by specifying a sequence of desired points (via-points) without indication on the velocity in these points. In these cases, the "most suitable" values for the velocities must be automatically computed. This assignment is quite simple with heuristic rules such as:

$$\dot{q}_1 = 0$$
$$\dot{q}_k = \begin{cases} 0 & sign(v_k) \neq sign(v_{k+1}) \\ \frac{1}{2}(v_k + v_k + 1) & sign(v_k) = sign(v_k + 1) \end{cases}$$
$$\dot{q}_n = 0$$

being

$$v_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}} = \frac{K_k}{T_k}$$

the slope of the k-tj tract $\begin{bmatrix} t_k - 1 & t_k \end{bmatrix}$

Fifth-order polynomial trajectories

From the above examples, it may be noticed that both the position and velocity profiles are continuous functions of time. This is not true for acceleration, that presents discontinuities among different segments. Moreover, it is not possible to specify for this signal suitable initial/final values in each segment. In many applications, these aspects do not constitute a problem, being the trajectories "smooth" enough. On the other hand, if it is requested to specify initial and final values for the acceleration (e.g. for obtaining continuous acceleration profiles), then (at least) fifth-order polynomial functions should be considered

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

In general, the smoother the function, the higher the values of velocity and acceleration need to be applied. This may be an issue with respect to motor constraints.

7.1.2 Trapezoidal trajectories

Among many other combinations, a possbile approach for planning a trajectory is to use linear segments joined with parabolic blends.

In the linear tract, the velocity is constant while, in the parabolic blends, it is a linear function of time: *trapezoidal velocity profiles*, typical of this type of trajectory, are then obtained. In trapezoidal trajectories, the duration is divided into three parts:

1. in the first part, a constant acceleration is appiled, then the velocity is linear and the position results a parabolic function of time
2. in the second, the acceleration is null, the velocity is constant and the position is linear in time
3. in the last part a (negative) acceleration is applied, then the velocity is a negative ramp and the position is a parabolic function

Usually, the acceleration and the deceleration phases have the same duration ($t_a = t_d$). Therefore, symmetric profiles, wrt a central instant $(t_f - t_i)/2$ are obtained.

The trajectory is computed according to the following equations:

Acceleration phase, $t \in [0 \div t_a]$

The position, velocity and acceleration are described by:

$$\begin{aligned} q(t) &= a_0 + a_1 t + a_2 t \\ \dot{q}(t) &= a_1 + 2a_2 t \\ \ddot{q}(t) &= 2a_2 \end{aligned}$$

The parameters are defined by constraints on the initial position q_i and velocity $\dot{q}_i$, and on the desired constant velocity $\dot{q}_v$ that must be obtained at the end of the acceleration period. Assuming a null initial velocity ($\dot{q}_i = 0$) one obtains

$$\begin{aligned} a_0 &= q_i \\ a_1 &= 0 \\ a_2 &= \frac{\dot{q}_v}{2t_a} \end{aligned}$$

In this phase, the acceleration is constant and equal to $\dot{q}_v/t_a$

Constant velocity phase, $t \in [t_a \div t_f - t_a]$

Deceleration phase, $t \in [t_f - t_a \div t_f]$

Summarizing, the trajectory is computed as

$$q(t) = \begin{cases} q_i + \dfrac{\dot{q}_v}{2t_a}t^2 & 0 \le t < t_a \\ q_i + \dot{q}_v\left(t - \dfrac{t_a}{2}\right) & t_a \le t < t_f - t_a \\ q_f - \dfrac{\dot{q}_v}{t_a}\dfrac{(t_f - t)^2}{2} & t_f - t_a \le t \le t_f \end{cases}$$

Note that in the previous computations two parameters have been assumed to be known:

- the constant velocity value $\dot{q}_v$
- the acceleration time t_a

We have seen that the parameters $t_a, \dot{q}_v$ have been assumed as known, and indeed they should be specified in order to compute the trajectory. Since $\ddot{q}_a t_a = \dot{q}_v$, it is possible to specify any two parameters among $t_a, \dot{q}_v . \ddot{q}_a$. On the other hand, it is not possible to choose any value for these variables, since there are some constraints that should be considered. Define as $L = (q_f - q_i)$ the displacement and $T = (t_f - t_i)$ the duration of the trajectory

1. From geometrical considerations, three basic constraints are:

$$t_a \le \frac{T}{2}, \quad \frac{|L|}{T} < \dot{q}_v \le \frac{2|L|}{T}, \quad \ddot{q}_a \ge \frac{4|L|}{T^2}$$

2. Moreover, from geometrical considerations, the following condition must be verified:

$$\ddot{q}_a t_a = \frac{q_m - q_a}{t_m - t_a} \begin{cases} q_a = q(t_a) \\ q_m = (q_i + q_f)/2 = q_i + L/2 \\ t_m = T_2 \end{cases}$$

$$q_a = q_i + \frac{1}{2}\ddot{q}t_a^2$$

   Therefore, from the previous equations we have

$$\ddot{q}_a t_a^2 - \ddot{q}_a(t_f - t_i)t_a + (q_f - q_i) = 0 \tag{7.1}$$

3. Finally, by computing the area below the trapezoidal velocity profile:

$$\dot{q}_v = \frac{q_f - q_i}{T - t_a} = \frac{L}{T - t_a}$$

Any pair of values $(\ddot{q}_a, t_a)$ verifying 7.1 can be considered. In general, it is possible to:

1. assign a value to t_a $(\le T/2)$ and compute $\ddot{q}_a$
2. assign a value to $\ddot{q}_a$ $(\ge 4|L|/T^2)$ and compute t_a
3. if T is not given, assigne a value to $\ddot{q}, \dot{q}_v$ (e.g. the maximum values) and compute t_a, T

Note that with this last modality for computing the trajectory, the time duration of the motion from qi to qf is not specified in advance. In fact, the period T is computed on the basis of the imposed maximum acceleration and velocity values.

Synchronization of more joints

Assume that more joints $q_i, i = 1, \ldots, n$ must be coordinated, and that the same constraints on the maximum acceleration and velocity apply to all the joints. In this situation:

1. the joint q_k with the largest displacement $|L_k|$ must be individuated. For this joint, the maximum values $\dot{q}_{max}, \ddot{q}_{max}$ for the velocity and acceleration are assigned, and then the corresponding values t_a and T are computed according to

2. for the remaining joints, the acceleration and velocity values must be computed (from (8)) on the basis of these values of t_a and T and on the basis of the given displacement L_i

$$\ddot{q}_i = \frac{L_i}{t_a(T - t_a)}, \quad \frac{L_i}{T - t_a}, \quad i = 1, \ldots, n \quad i \neq k$$

7.2 Data interpolation via Splines

7.2.1 Introduction

The problem of defining multipoint trajectories. i.e. of functions suitable for the interpolation or approximation of a set of given points (t_k, q_k), $k = 0, \ldots, n$, is now addressed. The problem is discussed in the case of a single axis of motion: the general problem related to 3D space canbe addressed with a similar approach. For this purpose, many function can be adopted, such as polynomial functions of proper degree, orthogonal and trigonometric polynomials, ...

Very effective methods for motion control applications are based on

- Spline functions
- B-Spline
- NURBS
- Bezier functions
- Nonlinear filters able to generate in real time optimal trajectories satisfying given constraints on maximum velocity, acceleration, and jerk

The problm of computing a trajectory through $n + 1$ points can be solved by means of a polynomial function of degree n:

$$q(t) = a_0 + a_1 t + \cdots + a_n t^n$$

From a mathematical point of view, the solution of the interpolation problem of $n+1$ points can be obtained by solving a linear system of $n + 1$ equations in $n + 1$ uknowns. This method is based on the following algorithm:

In addition to numerical problems, the use of a polynomial of degree n for interpolating $n + 1$ points has a number of other drawbacks, since:

1. The degree of the polynomial depends on the number of points and, for large values of n, the amount of calculations may be remarkable

2. The variation of a single point (t_k, q_k) implies that all the coefficients of the polynomial must be recomputed

3. The insertion of an additional point (t_{n+1}, q_{n+1}) implies the adoption of a polynomial of higher degree and the calculation of all the coefficients

4. The resulting trajectories are usually characterized by pronounced "oscillations" that are unacceptable in motion profiles for automatic machines

Oscillations are present at the boundaries of the trajectory, and their amplitude increases with the order of the polynomial. In numerical analysis, *Runge's phenomenon* is a well known problem of oscillation at the boundaries of an interval occurring with polynomial interpolation with polynomials of high degree over a set of *equispaced interpolation points*

Moreover, standard techniques for polynomial interpolation do not take into account further conditions on initial, final or intermediate velocities and accelerations. In this case, it is necessary to assume an higher order polynomial function and consider additional constraints on the polynomial coefficients. For instance, in order to assign initial/final velocities and accelerations (i.e. 4 additional constraints) on a trajectory interpolating $n+1$ given points, the polynomial must be of degree $n+4$.

In order to deal with all these problems, as before mentioned, other techniques are available

7.2.2 Cubic Splines

When $n+1$ points $q_0, q_1, \ldots, q_{n-1}, q_n$ are given, in lieu of a unique interpolating polynomial of degree n it is possible to use n polynomials of degree p (usually lower) each one defining a segment of the trajectory. The overall function $s(t)$ defined in this manner is called *spline* of degree p.

The value of p is chosen according to the desired degree of continuity of the spline. For instance, in order to obtain continuity of velocities and accelerations at the time instants t_k, where the transition between two consecuitve segments occurs, it is possible to assume a polynomial of degree $p = 3$ (cubic polynomial)

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

The overall function given by

$$s(t) = \{q_k(t), t \in [t_k, t_{k+1}], k = 0, \ldots, n-1\}$$

$$q_k(t) = a_{k0} + a_{k1}(t - t_k) + a_{k2}(t - t_k)^2 + a_{k3}(t - t_k)^3$$

In this way, it is necessary to compute 4 coefficients for each polynomial. Since n polynomial are necessary for the definition of a trajectory thtough $n+1$ points, the total number of coefficients to be determined is $4n$. In order to solve this problem, the following conditions must be considered:

- $2n$ conditions for the interpolation of the given points, since each cubic function must cross the points at its extremities
- $n-1$ conditions for the continuity of the velocities at the transition points
- $n-1$ conditions for the continuity of the accelerations at the transition points.

In this way, there are $2n + 2(n-1)$ conditions and therefore the remaining degrees of freedom are $4n - 2n - 2(n-1) = 2$. Then, two additional constraints must be imposed in order to compute the spline. Among the possible choices, one can assign:

1. The initial and final velocity $\dot{s}(t_0) = v_0, \dot{s}(t_n) = v_n$
2. The initial and final acceleration $\ddot{s}(t_0), \ddot{s}(t_n)$ (when set to zero, these conditions are generally referred to as *natural*)
3. The conditions $\dot{s}(t_0) = \dot{s}(t_n), \ddot{s}(t_0) = \ddot{s}(t_n)$; these conditions are usually called *cyclic* and are used when it is necessary to define a periodic spline, with period $T = t_n - t_0$
4. The continuity of the jerk at time instants t_1, t_{n-1}:

$$\left.\frac{d^3 s(t)}{dt^3}\right|_{t=t_1^-} = \left.\frac{d^3 s(t)}{dt^3}\right|_{t=t_1^+} \qquad \left.\frac{d^3 s(t)}{dt^3}\right|_{t=t_{n-1}^-} = \left.\frac{d^3 s(t)}{dt^3}\right|_{t=t_{n-1}^+}$$

In general, a spline is characterized by the following properties:

- $[n(p+1)]$ parameters are sufficient for the definition of a trajectory $s(t)$ of degree p, interpolating the given poitns
- Given $n+1$ points, and given the boundary conditions, the interpolating spline $s(t)$ of degree p is univocally determined.
- The degree p of the polynomilas used to constuct the spline does not depend on the number of data points

- The function $s(t)$ has continuous derivatives up to the order $(p-1)$
- By assuming the conditions $\ddot{s}(t_0) = \ddot{s}(t_n) = 0$, the cubic spline is, among all the functions $f(t)$ interpolating the given points and with continuous first and second derivatives, the function which minimizes the functional

$$J = \int_{t_0}^{t_n} \left(\frac{d^2 f(t)}{dt^2} \right)^2 dt$$

that can be interpreted as a sort of deformation energy, proportional to the curvature of $f(t)$

For the definition of a trajectory for an automatic machine, the condition on the continuity of the velocity profile is of fundamental importance. For this reason, a typical choice for the computation of the spline is to assign the initial and final velocities v_0 and v_n. Therefore, given the points $(t_k, q_k), k = 0, \ldots, n$ and the boundary conditions on the velocity v_0, v_n, the goal is to determine the function

$$s(t) = \{q_k(t), t \in [t_k, t_{k+1}], k = 0, \ldots, n-1\}$$

$$q_k(t) = a_{k0} + a_{k1}(t - t_k) + a_{k2}(t - t_k)^2 + a_{k3}(t - t_k)^3$$

with the conditions

$$\begin{aligned} q_k(t_k) = q_k, \quad q_k(t_{k+1}) = q_{k+1}, \qquad & k = 0, \ldots, n-1 \\ \dot{q}_k(t_{k+1}) = \dot{q}_{k+1}(t_{k+1}) = v_{k-1} \qquad & k = 0, \ldots, n-2 \\ \ddot{q}_k(t_{k+1}) = \ddot{q}_{k+1}(t_{k+1}) \qquad & k = 0, \ldots, n-2 \end{aligned}$$

That is

$$A(T)v = c(T, q, v_0, v_n)$$

where $T = [T_0, T_1, \ldots, T_{n-1}]^T, q = [q_0, q_1, \ldots, q_n]^T$

The $(n-1) \times (n-1)$ matrix A has a diagonal dominant structure, and therefore it is always invertible if $T_k > 0$ ($|a_{kk} > \sum_{j \neq k} |a_{kj}|$). Moreover, being A tridiagonal, computationally efficient techniques are available for its inversion (*Thomas algorithm*). Once the inverse of A has been computed, the velocities $v_1, \ldots, v_n - 1$ can be calculated from $v = A^{-1}c$ and therefore the problem is solved: the spline coefficients are obtained with

Splines are very flexible functions, and different goals can be assigned in their computation. For example, it is possible to design:

- Periodic splines
- Splines computed on the basis of the intermediate acceleration values
- Smoothing cubic splines: a non-precise interpolation is allowed
- Choice of the time instants t_k for optimizing cubic splines

Although a criteria for optimizing the time instants sequence t_k are possible, in the following a method to obtain the minimum total duration T is illustrated. The total duration of a spline is

$$T = \sum_{k=1}^{n} T_k = t_n - t_0$$

It is possible to define an optimality problem aiming at minimizing T. The values of T_k must be computed so that T is minimized and the constraints on the velocity and acceleration are satisfied. Formally the problem is formulated as

$$\begin{cases} \min_{T_k} T = \sum_{k=1}^{n} T_k \\ \text{such that} \end{cases}$$

7.2.3 synchronization of more motion axes

The above procedure for computing the spline is adopted also for more motion axes (joints). Notice that the matrix $A_i(T) = A(T)$ is the same for all the $i = 1, \ldots, m$ joints (it depends only on the parameters T_k), while the vector $c(T, q_i, v_{i1}, v_{in})$ depends on the specific i-th joint

7.2.4 Scaling

If

- the duration T_k of each interval is multiplied by a constant λ
- the initial and final velocities are null

one obtains that the new duration T', the velocities and accelerations of the new trajectory are:

$$T' = \lambda T$$
$$v'_k = \frac{1}{\lambda} v_k$$
$$a'_k = \frac{1}{\lambda^2} a_k$$

The parameter λ can then be defined in order to satisfy the constraints on maximum velocities/accelerations and obtain a minumum-time trajectory.

7.3 Scaling Trajectories

Due to several reasons, like limits on the actuation system (torques, accelerations, velocities,...) or computational efficiency, it is often requested to *scale* trajectories and motion laws. It is possible to adopt

- Kinematic scaling procedures
- Dynamic scaling procedures

7.4 Kinematic Scaling of trajectories

If a trajectory is expressed in parametric form as a function of a parameter $\sigma = \sigma(t)$, by changing the parametrization it is possible to obtain in a simple manner a trajectory satisfying constraints on velocity or accelerations. For this purpose, it is conveniente to express the trajectory in *normal form*, i.e.:

$$p(t) = p_0 + (p_1 - p_0)s(\tau) = p_0 + Ls(\tau)$$

being $s(\tau)$ a proper parameterization, with

$$0 \leq \sigma \leq 1, \qquad \tau = \frac{t - t_0}{t_1 - t_0} = \frac{t - t_0}{T}, \quad 0 \leq tau \leq 1$$

In this manner, it results (the notation $s^{(k)}(\tau) = \dfrac{d^k s}{d\tau^k}$ is used)

$$\frac{dp}{dt} = \frac{L}{T} s'(\tau) \quad \frac{d^2 p}{dt^2} = \frac{L}{T^2} s''(\tau)$$
$$\frac{d^3 p}{dt^3} = \frac{L}{T^3} s'''(\tau)$$
$$\frac{d^n p}{dt^n} = \frac{L}{T^n} s^{(n)}(\tau)$$

it follows that the maximum values for the velocity, acceleration, etc. are obtained in correspondence of the mazimum values of the functions $s', s'', \ldots$ These values and the correpsonding time instants $\tau(t)$ are known frmo the chosen parameterization $s(\tau)$.

Notice that if the duration T of the trajectory is changed, it is possible to satisfy in an exact manner the given constraints or to optimize the trajectory itself (minimum time). Moreover, it is easily possible to co-ordinate more motion axes.

Polynomial trajectories of degree 3

Consider a parameterization expressed by a cubic polynomial

$$s(\tau) = a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3, \qquad 0 \le s \le 1, \qquad 0 \le \tau \le 1, \quad \tau = \frac{t}{T}$$

If the boundary conditions $s_0 = 0, s_1 = 1, v_0 = 0, v_1 = 0$ are specified, one obtains

$$a_0 = 0, \quad a_1 = 0, \quad a_2 = 3, \quad a_3 = -2$$

Therefore:

$$\begin{gathered} s(\tau) = 3\tau^2 - 2\tau^3 \\ s'(\tau) = 6\tau - 6\tau^2 \\ s''(\tau) = 6 - 12\tau \\ s'''(\tau) = -12 \end{gathered}$$

Then

$$s'_{max} = s'(0.5) = \frac{3}{2} \quad \Longrightarrow \quad \dot{q}_{max} = \frac{3L}{2T} s''_{max} = s''(0) = 6 \quad \Longrightarrow \quad \ddot{q}_{max} = \frac{6L}{T^2}$$

Polynomial trajectories of degree 5 and 7

analogous considerations

It is clear that if the displacement L and the duration T of a motion are specified, the profiles of velocity, acceleration and jerk are defined by the parameterization $s(\tau)$ chosen to generate the motion profile. In particular, the maximum values for these variables are determined (considering $L > 0$ for the sake of simplicity)

7.5 Dynamic scaling of trajectories

When a trajectory is specified for a complex mechanical system, because of the dynamics of the actuation system, of the robot manipulator or of the load (dynamic couplings), torques non physically achievable by the actuator could be requested. In these cases, it is possible to scale the trajectory taking into account the dynamics of the system in order to obtain a physically achievable motion.

The dynamic model of a manipulator is (neglecting friction)

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

Then, for each joint

$$m_i^T(q)\ddot{q} + \frac{1}{2}\dot{q}^T C_i(q)\dot{q} + g_i(q) = \tau_i \quad i = 1, \dots, n$$

If

$$q = q(\sigma) \qquad \sigma = \sigma(t)$$

is a proper parameterization of the trajectory with a motion law such that

$$\dot{q} = \frac{dq}{d\sigma}\dot{\sigma}, \qquad \ddot{q} = \frac{d^2q}{d\sigma^2}\dot{\sigma}^2 + \frac{dq}{d\sigma}\ddot{\sigma}$$

By substitution in the dynamic model:

$$\left[m_i^T(q(\sigma))\frac{dq}{d\sigma}\right]\ddot{\sigma} + \left[m_i^T(q(\sigma))\frac{d^2q}{d\sigma^2} + \frac{1}{2}\frac{dq^T}{d\sigma}C_i(q(\sigma))\frac{dq}{d\sigma}\right]\dot{\sigma}^2 + g_i(q(\sigma)) = \tau_i$$

From which

$$\alpha_i(\sigma)\ddot{\sigma} + \beta_i(\sigma)\dot{\sigma}^2 + \gamma_i(\sigma) = \tau_i$$

Notice that $\gamma_i(\sigma)$ (gravitational terms) depend only on the position (σ)

Let us suppose to compute the torques τ_i necessary to achieve the motion defined by $q = q(\sigma), \sigma = \sigma(t)$:

$$\tau_i(t) = \alpha_i(\sigma(t))\ddot{\sigma}(t) + \beta_i(\sigma(t))\dot{\sigma}^2(t) + \gamma_i(\sigma(t)), \quad i = 1, \ldots, n \quad t \in [0, T]$$

If the time-axis is changed (e.g. in a linear fashion $x = kt$), a different parameterization of the trajectory is obtained

$$t \to x = kt \quad x \in [0, kT] \quad \sigma(t) \to \hat{\sigma}(x)$$

Notice that in general even a non linear parameterization $x = x(t)$ could be considered. With the new parameterization, one obtains:

$$\hat{\sigma}(x) = \sigma(t)$$
$$\dot{\hat{\sigma}}(x) = \frac{\dot{\sigma}(t)}{k}$$
$$\ddot{\hat{\sigma}}(x) = \frac{\ddot{\sigma}(t)}{k^2}$$

Therefore:

- if $k < 1$ a slower motion is obtained
- if $k > 1$ a faster motion is obtained

With the new parameterization, the torques compute as:

$$\tau_i(t) = \alpha_i(\hat{\sigma}(t))\ddot{\hat{\sigma}}(t) + \beta_i(\hat{\sigma}(t))\dot{\hat{\sigma}}^2(t) + \gamma_i(\hat{\sigma}(t)), = \alpha_i(\sigma(t))\frac{\ddot{\sigma(t)}}{k^2} + \beta_i(\sigma(t))\frac{\dot{\sigma}^2(t)}{k^2} + \gamma_i(\sigma(t))$$

7.6 Trajectories in the Workspace

If trajectories are defined in the workspace, it is necessary to use the inverse kinematic function to translate the motion specification to the joint space (where actuators operate). Since this increases the computational burden for trajectory planning, the operations of computing the trajectory and translating it to the joint space are made at a lower frequency with respect to the control frequency. Therefore, it is necessary to interpolate the data before assigning them to the low-level controllers: usually, a simple linear interpolation is adopted. There are two drawbacks to this approach:

- You can only start interpolation once you know the next point, so the system presents a delay equal to the frequence of the trajectory+inverse kinematics (the slower one)
- The cartesian positions actually achieved during the motion obtained by interpolating the points $q(t_n)$ are not those originally planned, due to points linearly interpolated in the joint space are not linearly interpolated in the work space