



UNIVERSITÀ
DI TRENTO

Dipartimento di
Ingegneria Industriale

**Notes of the course
“Intelligent Distributed Systems”**

2023/2024

Daniele Fontanelli

Beta version - Images and text to be fixed: any correction is
more than welcome!

Contents

1 The big picture	7
1.1 Industrial Cyber-Physical Systems	9
1.2 In this notes	10
1.3 What is real-time?	11
1.3.1 Time-based versus event-based	15
1.3.2 Delay and jitter	16
2 Communication Systems	18
2.1 Basic pillars of communication systems	22
2.1.1 Asynchronous and synchronous transmission	24
2.1.2 Industrial choices	28
2.2 Open System Model	28
2.3 Industrial Networks	41
2.3.1 Interconnection Systems	44
3 IEC 61499 Standard	47
3.1 Design views	53
3.2 Models	55
3.3 Function Block Types	57
4 Fieldbuses	61
4.1 Fieldbus	61
4.1.1 Foundation Fieldbuses: H1 Protocol	63
4.1.2 Why fieldbuses?	80
4.2 Fieldbuses examples	83
4.2.1 CAN bus	84
4.2.2 DeviceNet	87
4.2.3 PROFIBUS	89
5 Industrial Networks and QoS	99
5.1 Parametrisation of Industrial Networks	100

5.2	Transmission times	101
5.3	Delays and jitters	102
5.3.1	Additional QoS measures	106
5.3.2	QoS vs QoC	107
5.4	Control networks: Final comments	107
6	Ethernet as Control Network	111
6.1	Profinet	118
6.2	Profinet CBA	121
6.3	EtherCAT	122
6.4	Ethernet Powerlink (EPL)	124
7	Wireless in the Industrial Domain	126
7.1	WiFi: IEEE 802.11	128
7.2	WiFi: IEEE 802.15.4	131
7.3	Common Industrial Protocol	132
7.3.1	Hybrid networks	132
8	PLC Standards	136
8.1	Relay Logic	138
8.2	PLC	140
8.2.1	Ladder Diagram	140
8.3	Software Issues	143
8.3.1	Preemption	145
8.3.2	Communication	147
8.3.3	Ladder Diagram Revisited	149
8.4	The Future of PLCs	151
9	SCADA	153
9.1	A bird's eye view	153
9.1.1	SCADA Components	154
9.1.2	Acquisition, Processing, Control and Failures	161
9.1.3	Evaluate SCADA	162
9.1.4	The Four SCADA Levels	163
9.2	SCADA, DCS and PLCs	163
10	Measurement Processes	165
10.1	Measurement processes	168
10.1.1	The sensor	170
10.1.2	Random effects	175
10.1.3	Modelling	184

10.2 Estimation Algorithms	185
11 Background on Statistics	188
11.1 Probability	188
11.1.1 Bayesian inference example: The Monty Hall Problem .	191
11.1.2 Random variables	192
11.1.3 Multivariate Pdfs	201
11.1.4 Propagation of errors	220
11.2 Stochastic Processes	222
11.2.1 Properties of Stochastic Processes	224
11.2.2 White processes	229
11.2.3 Auto and Cross-correlation characteristics	230
11.2.4 Markovian processes	232
12 Estimation Algorithms	235
12.1 Best Linear Unbiased Estimator	239
12.2 Bayesian approaches	240
12.3 Most Popular Estimators	249
12.3.1 Maximum Likelihood	249
12.3.2 Maximum A Posteriori	250
12.3.3 Least Squares	253
12.3.4 Minimum Mean Square Error	254
12.3.5 Comments on LS, MAP and MMSE	255
12.3.6 Unbiased estimators	256
12.3.7 Multidimensional MMSE for Gaussian uncertainties .	258
13 Estimators	263
13.1 Linear Estimators	263
13.1.1 The Least Squares solution	263
13.1.2 The Kalman filter	272
13.2 Nonlinear estimators	286
13.2.1 The Non-linear Least Squares solution	286
13.2.2 The Extended Kalman filter	290
13.2.3 Unscented Kalman Filter	291
13.3 Application of the Least Squares in Distributed Systems .	297
13.3.1 Clock synchronisation example	297
13.3.2 Smart grid state estimation example	313
14 Advanced Estimators	319
14.1 Maximum Likelihood Estimation	319
14.1.1 Finding the MLE	320

14.1.2	Properties of the MLE	322
14.1.3	Numerical Determination of the MLE	327
14.1.4	Extension to a Vector Parameter	331
14.1.5	Properties of MLE for Vector Parameters	332
14.1.6	Signal Processing Example: Sinusoidal Parameter Estimation	333
15	An example of distributed estimation	339
15.1	A simplified example with relative measurements	341
16	Digital Systems	345
16.1	Discretisation of SISO Linear Continuous Time Systems	345
16.2	Forced and Unforced Response of a Linear System	351
16.3	Discretisation of Linear Continuous Time Systems	354
16.4	An Example for Nonlinear Discretisation	354
16.4.1	The Application of the Extended Kalman Filter	356
16.5	Some issues for discretisation	356
16.5.1	Noise discretisation	356
17	Introduction to Linear Consensus	359
17.1	A Practical Example	361
17.1.1	Variable topology	366
17.2	Examples	369
17.2.1	Node counting	369
17.2.2	Minimum variance estimates	370
17.2.3	Vehicle rendezvous	371
17.3	A Theoretical Approach to Linear Consensus	371
17.3.1	Stochastic matrices	375
17.4	Linear Consensus	376
17.4.1	Continuous time systems	379
17.4.2	Design of the consensus algorithm	380
17.5	Linear Consensus with Networks	384
18	Distributed Estimation Algorithms	386
18.1	Distributed WLS	386
18.2	Distributed Kalman Filter	390
19	Introduction to Distributed Control	393
19.1	Distributed coverage control with Voronoi-based tessellation	393
19.1.1	Application to actual agents	395
19.1.2	Extensions	396

19.2	Linear consensus protocol for second-order dynamics systems	398
19.2.1	Time invariant information exchange topologies: convergence analysis	399
19.2.2	Switching information exchange topologies: convergence analysis	401
19.2.3	Vehicle Model	402
19.2.4	Communication Simulation	405
19.2.5	Communication Scheduling: Round Robin	409
19.2.6	Communication Range	411
20	Robotics Perception	416
20.1	Sensing systems	419
20.1.1	Proprioceptive sensors	420
20.1.2	Potentiometers	421
20.1.3	Inertial sensors	422
20.1.4	Tachometers	424
20.1.5	Strain Gages	425
20.2	Exteroceptive sensors	427
20.2.1	Contact Sensors and Bumpers	427
20.2.2	Light Dependant Resistors and Photodiode	427
20.2.3	Sonars	428
20.2.4	Laser	429
20.2.5	Camera	430
20.3	A taxonomy of robotics problems and their solutions	430
20.3.1	Robot manipulators	430
20.3.2	Mobile robots	431
20.3.3	Common issues	431
20.4	Interactive Multiple Models	431
20.4.1	Gaussian Pseudo-Bayesian Estimator	431
20.4.2	IMM	431
Appendices		
Appendix A	Real-Time	433
A.1	The scheduler	439
A.1.1	Popular schedulers	441
A.1.2	Schedulability test	443
A.2	Implication for Digital Control and Estimation	444

Chapter 1

The big picture

The basic components of a distributed system are basically a set of (possibly heterogeneous) *plants* to be controlled, a set of input devices (*sensors*) and output devices (*actuators*) deployed in the environment, a set of *processing units* implementing the system control and a set of (possibly heterogeneous) *communication* links.

Definition 1 (Distributed Control Systems). *Distributed Control Systems (DCSs) are control systems in which the controller elements are not geographically central but are distributed throughout the system.*

They comprise a set of components, i.e., *sub-systems*, controlled by one or more controllers. The entire system of controllers is connected by networks for communication and for monitoring, i.e. sensing, while the DCS can be connected directly to physical equipments by means of actuators. Due to technology push, DCS are becoming very close to the *Supervisory Control And Data Acquisition* (SCADA) systems, which will be also covered in this notes.

In Figure 1.1 an example of an industrial DCS with the physical and logical views.

The DCS are also similar, it is more appropriate to say that they are the precursors, of the *Systems of Systems* (SoS), which, according to [1], can be defined as follows.

Definition 2. *A SoS is an integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.*

A SoS generally comprises systems where most of the components have some (control and operational) *independence* and the purpose of the whole

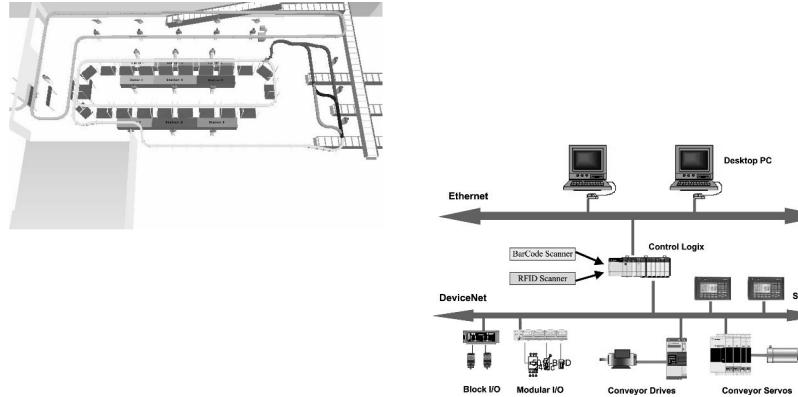


Figure 1.1: Examples of a distributed system: physical plant (top left) and logical representation (down right).

system is to provide a function or service that *cannot be provided by the individual systems independently* or cannot be provided in an efficient manner as by the overall system. Its structure, the connectivity and the “membership” of the components *can change dynamically over time*, i.e. components can be added or connected and disconnected and be dynamically reconfigured. The main features of the SoS are probably the *geographical distribution* and the *emergent behaviour*. The latter is usually a possible behaviour of the whole system and not necessarily an intended property of it; in this sense, the role of the engineers is to establish the local rules that leads the system to a *controlled global behaviour* rather than a behaviour that emerges “naturally” and by chance. The compounding systems of a SoS may be totally or partially autonomous (in such a case human interventions are needed). This terminology is usually adopted by computer scientists.

Figure 1.2 depicts an example of such a global system: the *Smart Grid* that promises a revolutionary advance over today’s power grids. Indeed, a smart grid enables a two-way flow of both electricity and information, a seamless integration of renewable resources at distribution and customer points, a widespread use of storage technologies and battery-electric and plug-in-hybrid-electric vehicles and a response for efficiency and peak reduction.

Another example is the *Traffic management* system (Figure 1.3), which represents a highly complex SoS that faces the increasing demand for additional capacity, greater safety and lower costs while meeting strict environmental regulations. This is a crucial ecosystem since the global car fleet is predicted to double from currently 800 million vehicles to over 1.6 billion vehicles by 2030, and hence there is a demand of integration of information and flow control to avoid congestions. Moreover, it needs to integrate already

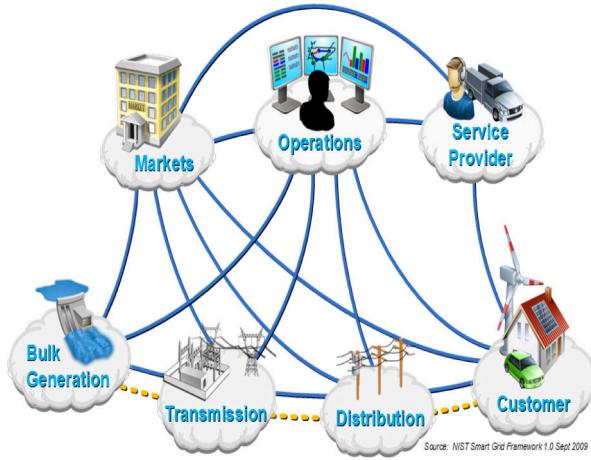


Figure 1.2: The *Smart Grid* example of a distributed system.

embedded intelligence, mobile phone, car-to-car and car-to-infrastructure communication for increased awareness, more efficient mobility and automated driver safety systems.

1.1 Industrial Cyber-Physical Systems

Cyber-Physical Systems (CPS) are systems where real-time computing and physical systems interact tightly. CPS are sometimes used as synonymous for *embedded systems*, i.e. a computer system with a specific function often with *real-time* computing constraints, even though CPSs have a very stronger emphasis on the interaction with the physical world by means of actuators of different nature. According to “*Agenda CPS, Intermediate Results*”, Acatech, 2010, CPSs typically comprise embedded systems (as parts of devices, buildings, vehicles, routes, production plants, logistics and management processes etc.) that use sensors and actuators to gather physical data directly and to directly affect physical processes, that are connected to digital networks (wireless, wired, local, global), that use globally available data and services and possess a range of multi-modal human-machine interfaces (dedicated interfaces in devices, or unspecific interfaces accessed through browsers, etc.).

A nice graphical description of CPSs has been provided by the *Rensselaer Polytechnic Institute* and reported given in Figure 1.4. In this realm, the embedded systems and the cyber-physical systems can be defined has overarching technological components, as represented in Figure 1.5. In their



Figure 1.3: A pictorial description of the traffic management system.

industrial application, the CPSs break with the traditional *automation pyramid* and remove the traditional hierarchical structure with *distribution and decentralisation*. An example of industrial cyber-physical systems for DCSs for the production area are summarised in the standard ISA 95: automation control (L1), supervisory control (L2), manufacturing operations management (L3), and business planning and logistics (L4), and represented in Figure 1.6 and Figure 1.7. Figure 1.8 instead represents the industrial cyber-physical systems for logistics applications. Finally, Figure 1.9 depicts the situation for a smart grid application. In this 4th Industrial Revolution, i.e. Industry 4.0, cyber-physical components are required to be *smaller, more intelligent* and *modular* entities that are function oriented.

1.2 In this notes

In this notes, the term DCS can be interpreted as a *sub-class* of the SoS, with particular emphasis to *industrial applications*, i.e.

Definition 3. *The DCS is a complex physical system that is interacting with a number of CPSs for monitoring, control and management.*

Examples of large DCSs are: the electrical grid, a power plant, a vehicle, e.g., an airplane or a ship, a manufacturing process with many cooperat-

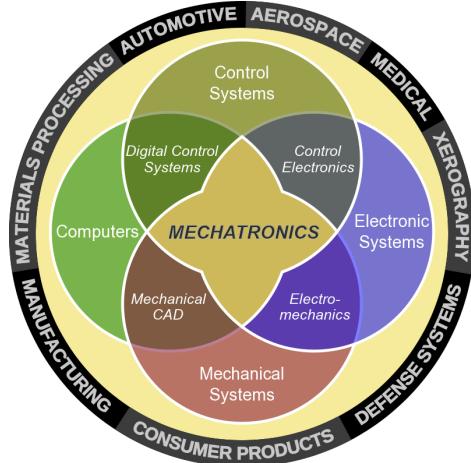


Figure 1.4: Mechatronics: Rensselaer Polytechnic Institute view.

ing elements as e.g. robots, machines, warehouses and conveyer belts, a smart building with advanced distributed Heating, Ventilation and Air Conditioning (HVAC) control, etc. Nonetheless, we will introduce concepts on distributed control by means of the so-called *consensus theory*.

1.3 What is real-time?

It will be evident that the control applications involved in the DCS will have stringent *Real-Time* (RT) constraints, hence obeying to the following definition.

Definition 4. *A real-time computer system is a computer system in which correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instants at which these results are produced [3].*

Definition 5. *A real-time (RT) application is an application where the correctness of the application depends on the timelines and predictability of the application as well as the results of computations.*

For a RT-system, the following requirements must be fulfilled:

- *Communication and synchronisation* between the different application components;
- *Predictable response* to events (e.g., clock triggers, availability of sensor data, etc) and reliable input/output (I/O);

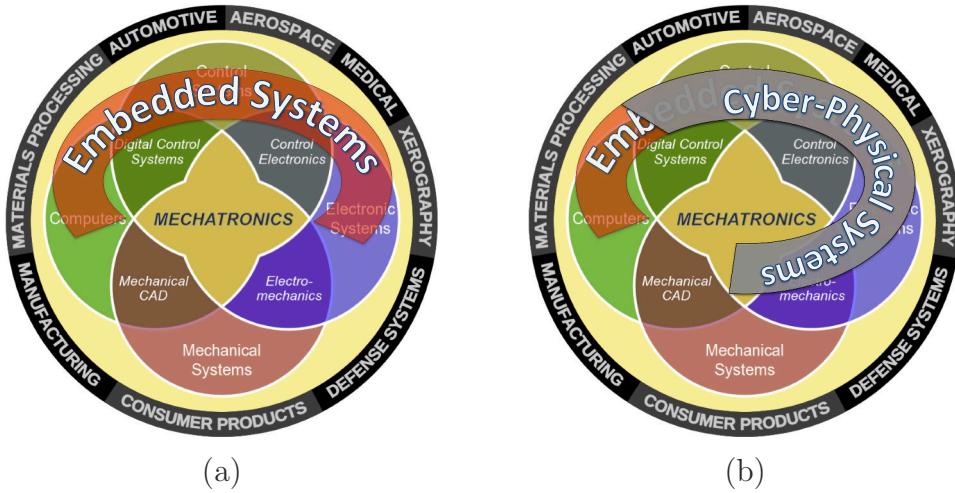


Figure 1.5: Embedded systems (a) and cyber-physical systems (b) in the mechatronics world.

Name	ISA95 level	Scope	Year
BHP Billiton	L2	Process control [23]	1995
Yokogawa	L1	Machinery control [24]	1998
MASCADA (Daimler-Benz pilot)	L2	Manufacturing control [25]	1999
Daimler Chrysler	L2	Manufacturing control [26]	2001
LIAZ	L3	Production planning [27]	2002
Skoda	L3	Production planning [27]	2002
Cambridge packing cell	L2	Manufacturing control [28]	2003
Watchdog Agent	L1	Machinery monitoring [29]	2003
FABMAS	L2	Process control [30]	2003
PABADIS	L1-L4	Manufacturing control [31]	2004
ABAS	L2	Manufacturing simulation and control [32]	2005
Saarstahl	L3	Production process planning and monitoring [33]	2005

Figure 1.6: Example of DCS production applications [2].

- *Prompt resource management* (e.g., computing power, sensor/actuator access, memory allocation, file synchronisation etc).

Real-time applications provide an action or an answer to an external event in a timely and predictable manner, which *does not* mean fast, since time scale may differ for different applications. For example, controlling a vehicle engine requires a time scale of some microseconds, while temperature control in a HVAC acts on the tens of seconds time scale. In both cases, the success of both applications depends on well-defined *time requirements* and, hence, may rely on largely different resource requests.

The concept of *predictability* is wide, however in RT applications it generally means that a certain operation or application can always be completed

SOCRADES	L1-L4	SOA-ready devices, SOA-based cross-layer integration (device-to-ERP) [34–36]	2007
NovaFlex	L2	Manufacturing control [37]	2007
AGP	L2	Manufacturing execution system [38]	2008
ADACOR-FMS	L2	Manufacturing control, and reconfiguration [39]	2008
Axion-Holding	L3	Manufacturing scheduling [40]	2010
IMC-AESOP	L1-L4	SOA-ready devices, cloud-based SCADA/DCS, SOA & cloud-based services [12, 36, 41]	2010
Kuznetsov	L3	Manufacturing scheduling [42]	2013
GRACE (Whirlpool pilot)	L2	Self-adaptation [43]	2013
IDEAS	L2	Reconfiguration and plug-and-produce [44]	2013
ARUM (Airbus and IHF pilots)	L3	Production planning and scheduling [45]	2015
ADACOR2	L2	Manufacturing control and reconfiguration [46]	2015
PRIME	L2	Manufacturing plug-and-produce [47]	2015

Figure 1.7: Example of DCS production applications [2].

Name	Scope	Year
Southwest Airlines	Ground floor operations optimization [73]	2001
ABX Logistics	Real-time transport optimization [74]	2005
Tankers International	Real-time scheduling [75]	2006
GIST	Real-time routing and scheduling [75]	2007
Air Liquide America	Logistics optimization [76]	2008
Airport ground services operations	Planning and scheduling [77]	2008
Addison Lee	Real-time taxi scheduling [78]	2009
Avis	Rent a car optimization [79]	2009
Ciudad Real Central Airport	Airport ground handling management [80]	2012
Prologics	Real-time truck scheduling and routing [81]	2012
RusGlobal	Real-time truck scheduling and routing [81]	2012
Lego	Real-time scheduling [82]	2013
MASDIMMA	Monitoring & real-time adaption in airline operations [83]	2014
Russian railways	Real-time train scheduling [84]	2015

Figure 1.8: Example of DCS logistics applications [2].

within a *predetermined amount of time*. An unpredictable RT control application generates loss of sensing and actuation, hence may lead to faults and/or system instability. In industrial applications two standards have been historically considered for real-time description: [IEC 61131-3](#) and [IEC 61499](#).

In general, responding in real-time means to fulfil precise time constraints. In all the engineering applications, there is always a trade-off between the time scale to respect and the complexity of the decisions that have to be made. For example, in typical feedback control applications, e.g. control the velocity of a motor, the time intervals are short. Instead, if strategical planning decisions have to be taken, the time intervals are long. In the industrial domain, there are usually four different abstraction levels: control level (the lowest, which is the closest to the *field*), supervisory level, execution level and planning level (the highest, the most abstract and strategical). For each of the four different levels, a “reaction time” is defined, which is directly proportional to the complexity of the decision to be taken (see Figure 1.10). This is usually referred to as the hierarchical representation.

Name	Automation functions	Year
CRISP	Active Control, Distributed Control and Monitoring [50]	2005
GridAgents	Distributed Control and Monitoring, Demand Response [51]	2008
Fenix	Demand Response, Distributed Control and Monitoring, Active Control [52]	2008
IDAPS	Distributed Control and Monitoring, Demand Side Management [53]	2009
SmartHouse/ SmartGrid	Demand Side Management, Demand Response, Distributed Control and Monitoring, Active Control [19, 50, 54, 55]	2009
OPTIMATE	Distributed Control and Monitoring [56]	2010
More Microgrids	Demand Side Management, Distributed Control and Monitoring, Active Control [19, 54, 57]	2010
Integral	Distributed Control and Monitoring, Active Control [58]	2011
MASGrid	Distributed Control and Monitoring, Active control, Self-Optimization [59]	2011
BeyWatch	Demand Side Management, Demand Response, Distributed Control and Monitoring [60]	2011
EcoGrid	Demand Side Management, Self-optimization [61]	2012
UNLV pilot	Demand Side Management, Peak Load Management [62]	2012
GRID4EU	Demand Side Management, Peak Load Management [63]	2013
NOBEL	User-bidding in Energy Marketplace [64, 65]	2013
E2SG	Distributed Control and Monitoring, Network Reconfiguration [66, 67]	2014

Figure 1.9: Example of DCS smart grid applications [2].

A major problem in engineering applications is the following: How can be ensured predictability of executions? To have control over the predictability of an application, the engineer should take care of the time bounds. For example, considering control applications running on *Embedded Computing Unit* (ECU), the average *execution time* of an application can be considered, which leads to probabilistic guarantees. Otherwise, the *Worst Case Execution Time* (WCET) can also be considered. What changes in the two cases?

The real-time computer system and the enclosing environment form a larger system called the *real-time system*, which must react to measures made on the controlled system within time intervals specified via the *deadlines*. If a result has utility even after the deadline has passed, the deadline is classified as *soft*. On the other hand, deadlines are called *hard deadlines* if catastrophic consequences can result from missing a deadline. As a consequence, a real-time system is a *soft real-time system* if *all* the deadlines are soft (i.e. no catastrophic consequences arise through *deadline violations*). It is a *hard real-time system* if *at least one* deadline is hard, which explains why a hard real-time system is sometime called a *safety-critical real-time system*. To summarise:

Definition 6. Hard RT applications require a response to events within a predetermined amount of time (hard deadline) for the application to function properly.

In this case, the choice is on the WCET. Again, predictability *does not* mean fast response. Hence, Hard RT *does not* necessarily imply a fast computing system, but instead a 100% reliable system.

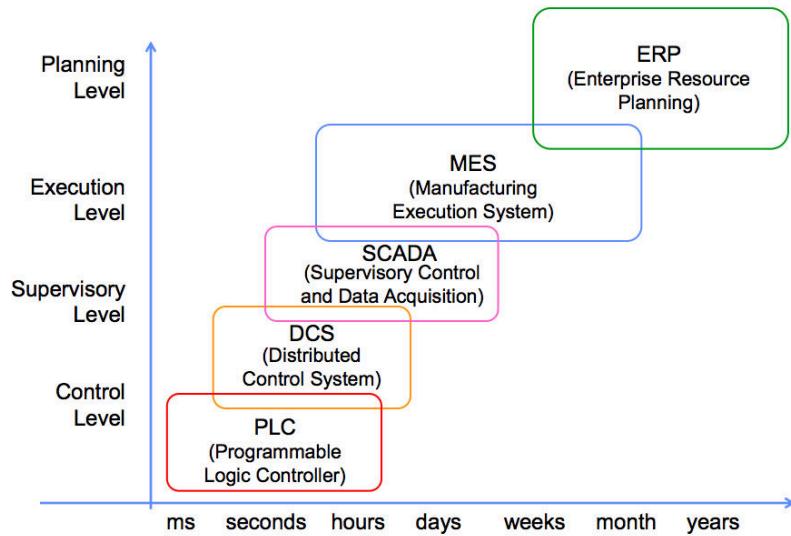


Figure 1.10: Decision levels versus reaction times.

Definition 7. Soft RT applications preferably require a response to events within a predetermined amount of time (soft deadline) for the application to function properly.

In this case, the choice can be on the average execution time (or any other statistical measure) since a Soft RT application tolerates occasional deadline misses. Again, the violations of the deadline *does not* mean that the actions should be necessarily slow.

1.3.1 Time-based versus event-based

Two different paradigms are prevalent for real-time architectures: in *event-triggered* systems the real-time actions are *triggered* by an environmental *event* (usually, a measure performed by a sensor); in *time-triggered* systems the real-time actions are triggered by the progression of a *global* time. Notice that if the system is distributed, the time must be *common*, i.e. the components should be *synchronised*.

Definition 8. An *event* is a change of state, occurring at an instant.

Definition 9. An *event trigger* is a control signal that is derived from an event, i.e. a state change in a real-time entity.

Definition 10. A *time trigger* is a control signal that is generated at a particular point in time of a synchronised global time base.

The major contrast between event-triggered and time-triggered approaches is in *how* the actions are fired. Time-triggered systems are in essence *autonomous* and interact with the environment according to an internal predefined *schedule*. On the other hand, event-triggered systems are under the control of the environment and must respond to its stimuli whenever they occur.

The implication of a time-based approach is that the *communication controller* decides autonomously when a message is transmitted. Due to this behaviour, the system behaviour is far more predictable and, hence, the time-triggered is far more adopted in industry. On the other hand, the event-triggered control paradigm may be preferred due to *higher flexibility and better resource exploitation*, since messages and computations take place only when strictly needed. Event-triggered architectures support *dynamic resource allocation strategies and resource sharing*.

1.3.2 Delay and jitter

The speed of computation or communication for a digital system can be quantified using the following definition.

Definition 11. *The throughput, synonymous of digital bandwidth consumption, is a measurement of bit-rate (available or consumed) of a data resource expressed in bits per second.*

For DCS we have to take into account the presence of (possibly multiple) communication channels, which have their own characteristics and throughput. However, it is evident from this discussion that a communication system that is able to ensure *predictability* of communications, hence to satisfy RT requirements, and a high *throughput*, hence to transfer the desired amount of data timely, is what we are looking for a DCS system.

However, there are some other characteristics that are still missing. Let us recall a standard output $y(t)$ feedback system, in which the control inputs are computed as:

$$u(t) = f(y(t)).$$

In a DCS there can be a communication channel between the output given by the available sensors and the system computing the control input $u(t)$. Moreover, the control input is usually computed by an embedded computing system, which will take some time to accomplish the desired computation. Therefore, we may model this unavoidable effect by:

$$u(t + \delta t) = f(y(t)),$$

the more δt is prominent with respect to the system time-scale, the higher will be the discrepancy between the hypothesised behaviour and the real behaviour. This effect is broadly known with the name *input/output delay*.

But it can be even worse. Indeed, if the delay δt is known upfront, it can be reasonably considered and compensated in the controller design, for example by adding a *unitary delay* on purpose to the plant to be controlled or by designing a properly robust control. However, it may happen that the delay *changes in time* and that is even *unpredictable*. This is a widely known problem in the literature, since it is the common situation for computing systems as well as for communication systems, and hence a proper definition is needed.

Definition 12. *The Jitter is defined as the deviation of a periodic signal from pure periodicity.*

The meaning of jitter is *context dependent*. For digital communication networks, it is the *variation in latency* as measured in the variability over time of the packet delay across a network. For computing systems, it is the variability of the time incurring *between the start of the computation and its end*. Under the presence of the jitter, a stable system may easily become *unstable*, hence, we have to add this problem in the picture derived so far for DCS.

To summarise, an efficient DCS should rely on:

- *Predictability* of communications and computations, i.e., to satisfy RT requirements;
- A high *throughput*, to transfer the desired amount of data and to process it;
- A predictable and possibly short *jitter*, to have a limited and known time-delay between the data transmission, reception and processing.

Chapter 2

Communication Systems

Basically, the *communication systems* are used to transfer *information* from one source, i.e., the *transmitter*, to a destination, i.e., the *receiver*. The transmission/reception of messages takes place using *communication modules* that are interfaces between the transmitter/receiver and the physical *medium* upon which the signals are transmitted.

Definition 13. *In communications, physical media (singular is medium) refers to the physical components used to transmit information.*

Examples: copper wire, optical fibre, coaxial cable, air, etc.

Definition 14. *A communication channel, or channel, refers to a logical connection over a multiplexed medium.*

Examples: the medium per se, the radio channel, etc.

The *transmission quality* is determined by the selected *medium* and the *bandwidth*. There are *guided media* in which the transmission is directed by the medium, i.e. the wire. In this case, the medium determine the transmission quality. Instead, for *unguided media* the transmission is in the air (*wireless communication*). In this case the bandwidth of the signal produced by the *transmitting antenna* is more important than the medium for the transmission quality.

Transmission speed, covered distance and electromagnetic immunity are all functions of the specific media adopted. Examples of commonly adopted guided media are:

- *Pair of twisted wires:* the cheapest, the simplest, the slowest. There are *unshielded twisted pair* (UTP, as in Figure 2.1) and the *shielded twisted pair* (STP, Figure 2.2-a). *Twisting* reduces the cross-talk noise, as depicted in Figure 2.2-b;

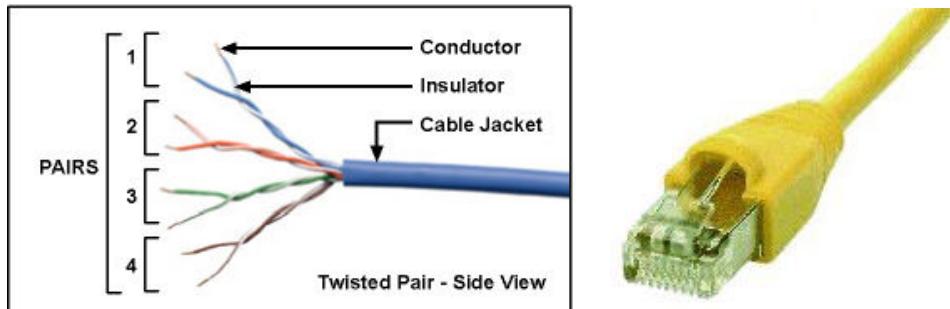


Figure 2.1: Pair of twisted wires UTP with the popular RJ-45 Jack (courtesy of www.infocellar.com).

- *Coaxial cable*: it consists of a copper conductor surrounded by grounding shielding and an external conductor (Figure 2.3). It has excellent *electrical properties* and it is conceived for high speed transmission;
- *Optical fibre*: messages are transmitted with light signals. It is conceived for *harsh industrial environments* where signals are transmitted over *long distances* (Figure 2.4). It is also the most expensive.

One common question when dealing with a communication media is: how fast can be sent data over a transmission line? To answer this question we have to first define the *bandwidth* that is a channel capacity for transmitting information and is measured in [Hz]. In particular, the *bandwidth* refers to the range of frequencies that a medium can pass without a loss of one-half of the power (-3 dB) contained in the signal. The speed depends also on the *number of levels* that can be transmitted, i.e. the quantity of information per bit.

In general, the more is the *noise* in the transmission line, the less is the speed. Depending on the choice made for transmitting the information, the noise is more or less disrupting. An *analog signal* has a possible infinite set of values (i.e. levels), while a *digital signal* represents a limited set of values that can be *coded* in *bits*. The number of bits in the code determines the maximum number of unique characters or symbols that can be represented. The most common character set in the Western World is the *American Standard Code for Information Interchange* (or ASCII).

An *analog signal* requires a *band pass channel*, since the analog signal per se has some specific frequency that are transmitted, of course the one in the pass-band of the channel. Indeed, *analog transmission* takes place whenever a medium carries a *carrier*, i.e. a sinusoidal wave. Transmitting an *analog signal* (the *baseband*) is then possible by *modulating* the carrier

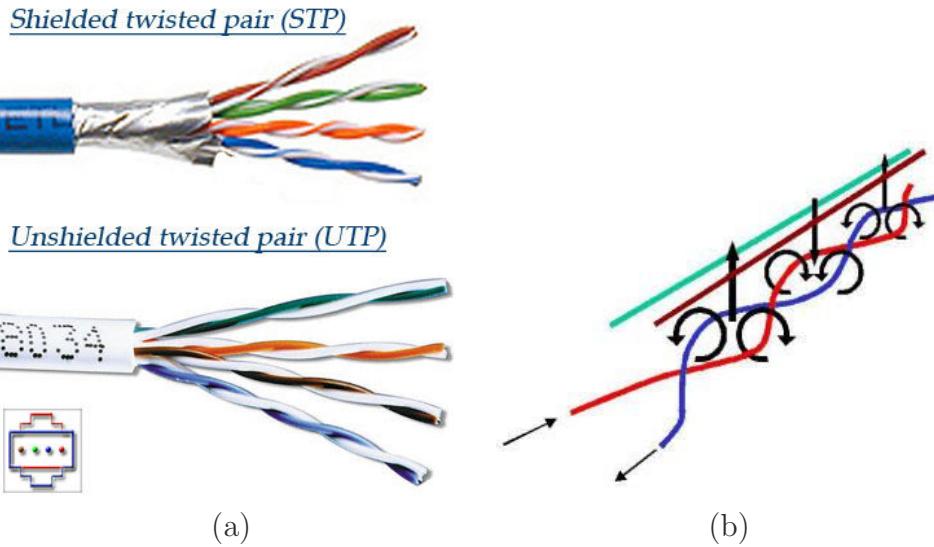


Figure 2.2: UTP and STP and the minimisation of the cross-talk noise by twisting.

(Figure 2.5). Example: AM, FM, PM, etc. It is also possible to transmit a *digital signal* over an *analog medium*, for example using the ASK, FSK (using two carriers), PSK, OOK (or BASK), QAM, etc. (Figure 2.6). Each of the selected transmission inherits some specific robustness to noise (Figure 2.7). Therefore, the buzz word here is *modulation*.

Instead, a *digital signal* requires a bandwidth from 0 to (ideally) *infinite* (recall the Fourier Transform). Since the channel is usually low-pass, the higher is the maximum bandwidth, the better is the approximation of the digital signal. Since a *digital transmission* takes place efficiently whenever bits are correctly transmitted, which must consider the channel *number of signal levels and synchronisation*. Here it is important the encoding, i.e. unipolar, polar, bipolar (see examples in Figure 2.8). Furthermore, it is also possible to transmit an *analog signal* over a *digital medium* after *sampling and quantisation*, for example using the PAM, Delta modulation, etc. The buzz word here is *encoding*. Even though the number of bits is finite, any possible value can be sent with an appropriate encoding, as reported in Figure 2.9. This idea is radically similar to the classic digital control paradigm (Figure 2.10), where a continuous time plant can be controlled by a digital system. As a final remark, we point out the definition of the bit leads also to another definition of *bandwidth*, that is the *data rate* ([bit/s] or [bps], bits per second).

To summarise, in *analog communications* the shapes of the transmitted

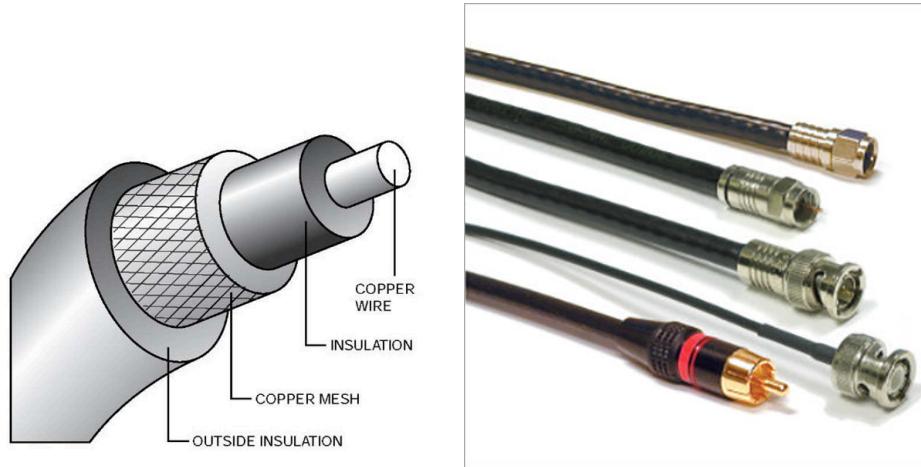


Figure 2.3: Coaxial cables.

waveforms can be very widespread, virtually infinite. Instead, in *digital communications* show robustness towards noise, since only a finite set of known waveforms are transmitted (Figure 2.11). Indeed, they are easier to be regenerated and more robust to distortion and interference, they make use of digital signal processing algorithms to regenerate the signal and they are tractable with digital hardware, which also have a lower cost with respect to the analog circuitry. The drawbacks are a larger bandwidth, the need of heavy signal processing and synchronisation between the transmitter and the receiver and it is a typical “all or nothing” approach, i.e. either the signal is entirely received or otherwise nothing can be retrieved. Usually, this problem is overcome using *redundancy*.

For a telecommunication engineer, it is of major relevance the *characterisation of the medium* and the *noise* it introduces and, hence, the objective of the study in that field are the *communication modules* as a whole, with special emphasis on the modulation/demodulation algorithms and signal processing algorithms involved. For a control engineer, instead of major relevance the nuisances (e.g. delay, noise, jitter) the communication medium generates on the transmitted signal using some level of abstraction. Our view is somehow in the middle of the previously reported stand-points: the knowledge of the *technological issues* related in a communication network are needed for properly design an industrial network. On the same time, the *latencies*, *delays* and *level of synchronism* is also of interest to properly model all the actors involved in the distributed control system.

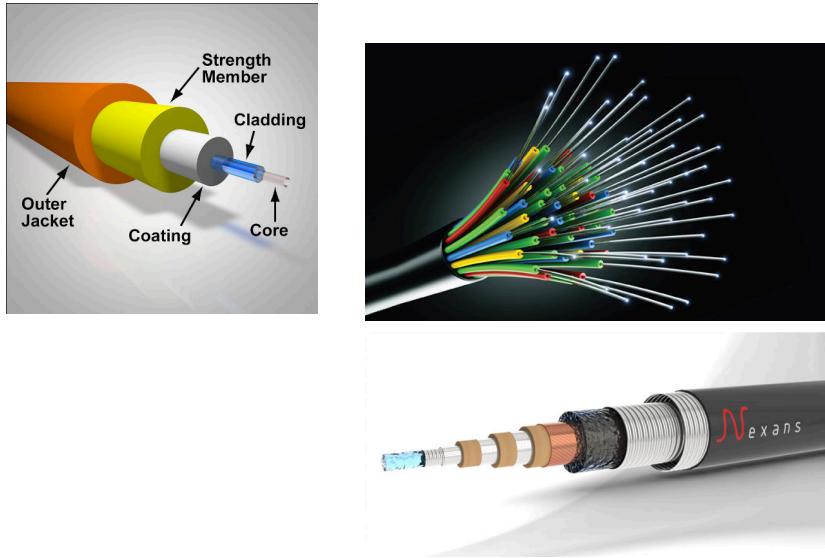


Figure 2.4: Optical fibre.

2.1 Basic pillars of communication systems

Digital transmission relies on the exchange of a flow of bits. The transmission can take place using different *modalities*, whose main purpose is to define the *direction of the information flow* (radio stations broadcasting their messages are different from a telephone call), the *data per message* (the information is sent in a sequence or all together?), the *synchronism between sender and receiver* (you need to know the schedule to see your favourite TV show but not to receive a phone call) and the *data encoding* (is it nice to see the content of a video podcast by looking at an oscilloscope attached to the cable?).

For the information flow, three main approaches are used: *simplex*, *half duplex* and *full duplex* (Figure 2.12). A *simplex* system is one that is designed for sending messages in one direction only. Since it is not possible to send feedback (unless for particular network topologies, i.e., *ring*), it is of limited interest in an industrial communications system. Examples: Radio and television broadcast, remote controllers (e.g. garage door openers, TV, HiFi, etc.), baby monitors, surveillance cameras, the PC mouse, GPS, printers, etc. *Half duplex* communications occurs when data flows *alternatively* in both directions (e.g. RS-485 physical standard). Example: walkie talkie, telegraph lines, etc. *Full duplex* systems allows the data flow in both directions simultaneously. Examples of hardware standards supporting full duplex are the physical standard EIA-232E (sometimes referred to as RS-232C). Examples: telephone lines. To divide forward and reverse communication channels on

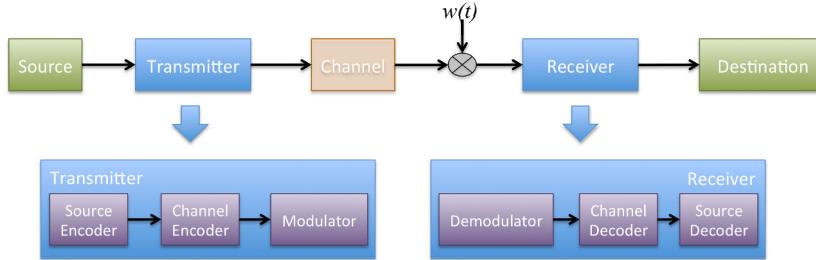


Figure 2.5: Analog transmission system.

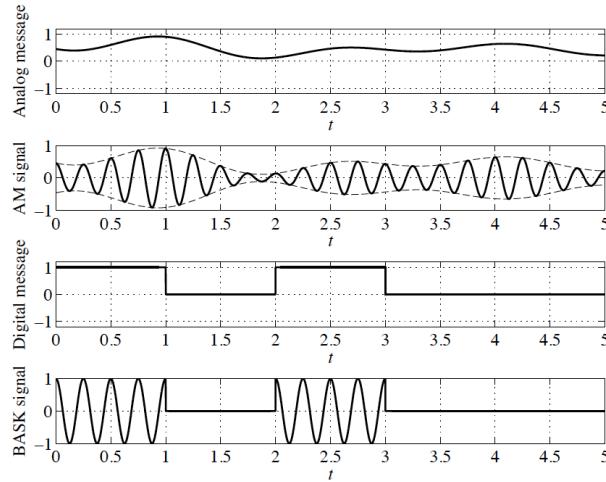


Figure 2.6: Examples of analog transmissions: AM and BASK [4].

the *same* physical communications medium, *duplexing methods* are used, i.e. *time-division duplexing* and *frequency-division duplexing*.

Another modality is the parallel versus serial communication.

Definition 15. In a *parallel* transmission, which is usually used for short distances, the bits in a byte are transmitted *simultaneously* over separated paths.

The reason why they are used for short distances is mainly due to the close vicinity of neighbouring paths, which tends to create *interference*. Multiple electric wires are usually used. Due to differences on the electric characteristics of the wires may *skew* the signals, i.e. some of the signals may arrive earlier.

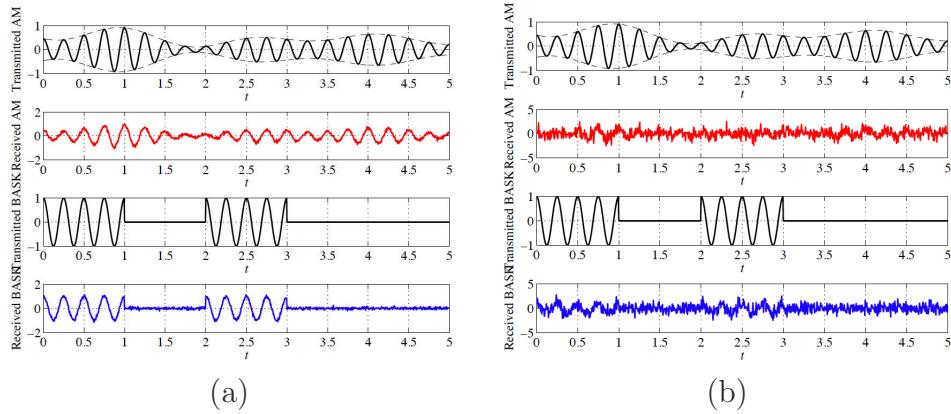


Figure 2.7: Examples of analog transmissions: AM and BASK affected with low (a) and high (b) noise levels [4].

Definition 16. In a [serial](#) transmission the bits in a byte are transmitted one after the other.

The transmission happens on the same wire, frequency or optical path sequentially. The serial transmission is simpler than parallel transmission, hence less *signal processing* and less *transmission errors* are usually obtained.

Encoded data are transmitted by means of *frames*.

Definition 17. A frame is a digital data transmission unit.

Hence, the frame contains a sequence of bits with various meaning, as shown in Figure 2.13. Transmitting and receiving nodes need to agree when the signal begins and ends so the signals can be correctly measured and interpreted. This is called *framing*, *clocking* or *bit synchronisation*. This concept that can be found in any “natural” communication.

Indeed what happens when a text is written without a proper start and end for each word.

That is the purpose of the efficient encoding, i.e. to generate a *self-synchronising code*. Since the encoded data is in a frame, hence the name *framing*.

2.1.1 Asynchronous and synchronous transmission

Asynchronous transmission is very basic and allows low data rate. Data are transmitted irregularly in time (variable bit rate) rather than in a steady stream, but the time interval between two successive bits in the message is fixed and known. Any timing required to recover data from the communication symbols is not given by an external clock (hence, *no clock synchronisation* between the sender and the receiver) but it is *encoded within the symbols*,

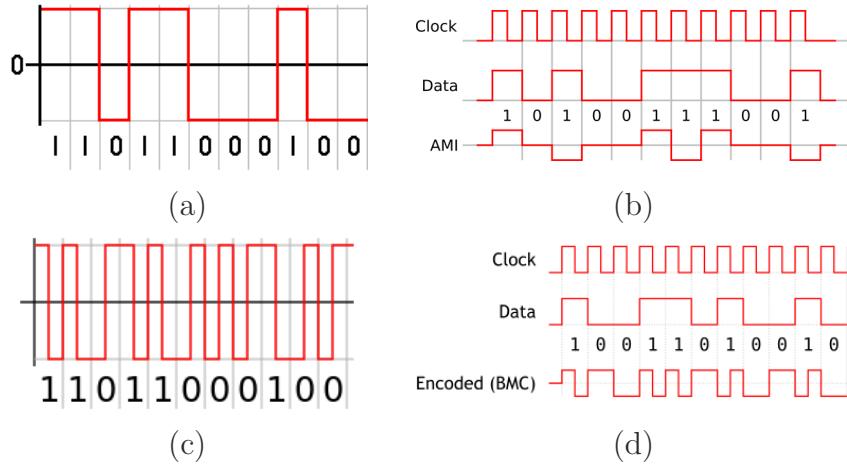


Figure 2.8: (a) Polar Non-Return-to-Zero (NRZ). (b) Bipolar (AMI). (c) Manchester. (d) Biphase Mark Code (BMC).

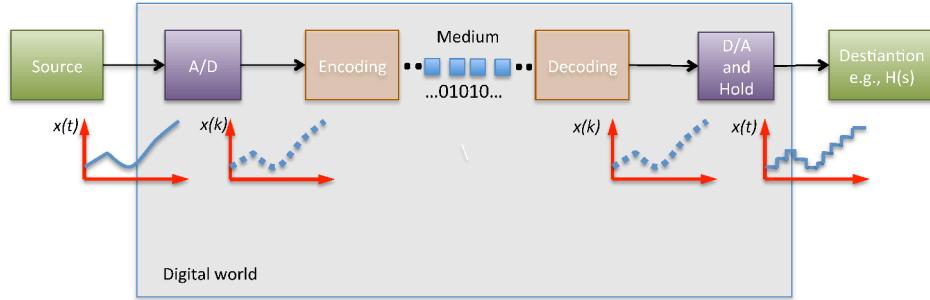


Figure 2.9: Connections between the digital and the analog world.

i.e. framing. The receiver does not start detection until it receives the *first bit*, known as the start bit. The start bit is in the opposite voltage state to the idle voltage and allows the receiver to synchronise to the bits following (Figure 2.13). The elements of the frame are usually:

- Start Bit: Signals the start of the frame;
- Data: Usually 8 bits of data;
- Parity Bit: Optional Error detection bit;
- Stop bits: Usually 1 or 2 bits.

A parity bit is '0' if the number of one's in the message is even, '1' otherwise. Parity bits are used as the simplest form of error detecting code.

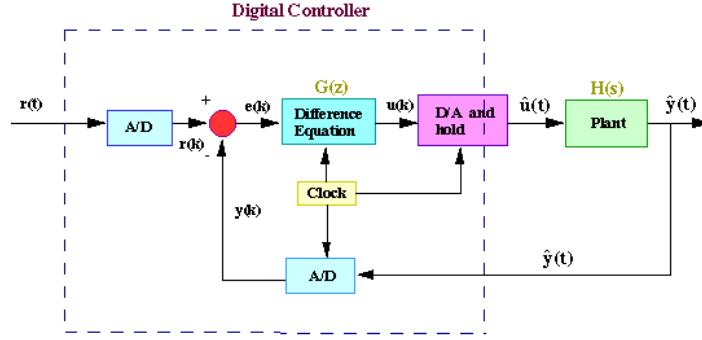


Figure 2.10: Connections between the digital and the analog world in control systems.

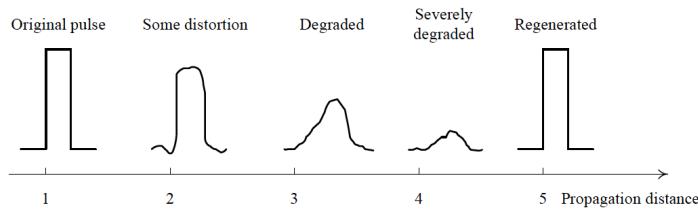
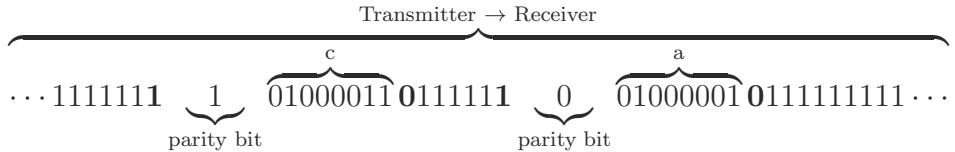


Figure 2.11: Robustness of a transmitted digital bit [4].

For example, the following sequence for the transmitter to the receiver



where the bold numbers are respectively the start (**0**) and end (**1**) bits, while the idle (or *mark*) state is 1. Notice that each data transmitted (in this case a character) needs a start and an end bit. It is evident that an overhead of bits are sent for each transmission, hence, this approach is not suitable for a large amount of data.

On the other hand, *synchronous* transmission allows high data rate since data is transmitted continuously. Synchronous transmission requires the communicating devices to maintain *synchronous clocks* during the entire connection. The sending device transmits on a specific schedule and the receiving device accepts the data on that same fixed schedule. A synchronous system uses a string of bits transmitted with data signals, usually the *preamble of the frame*, to keep the two clocks, transmitter and receiver, synchronised (see Figure 2.14). Synchronous systems detect bits by a change in voltage rather

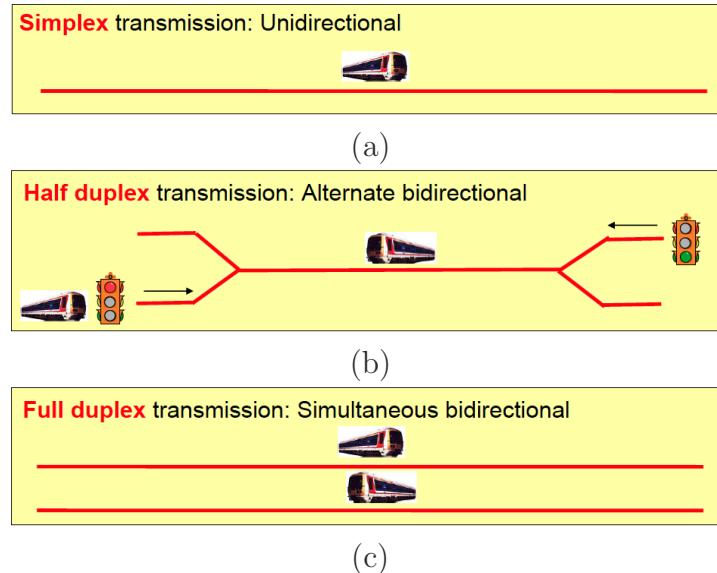


Figure 2.12: (a) Simplex, (b) half-duplex and (c) full duplex communication methods (Courtesy of Schneider Electric).



Figure 2.13: Typical asynchronous frame.

than by reading an absolute value as with asynchronous systems. In each frame we may find:

- Preamble: This comprises one or more bytes that allow the receiving unit to synchronise with the frame;
- Start of Frame Delimiter (SFD): It signals the beginning of the frame;
- Destination: The address to which the frame is sent;
- Source: The address from which the frame is sent;
- Length: Indicates the number of bytes in the data field;
- Data: The actual message;
- Frame Check Sequence (FCS): It is used for error detection.



Figure 2.14: Typical synchronous frame.

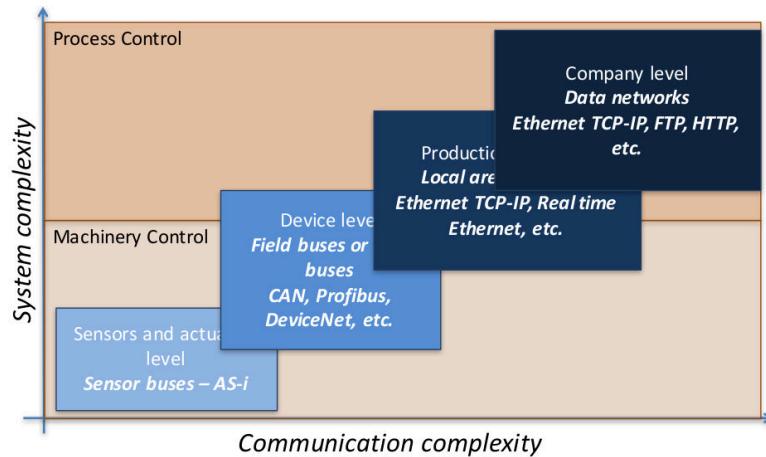


Figure 2.15: Main industrial networks and buses: company level.

2.1.2 Industrial choices

Historically, due to cost, durability, reliability and simplicity, most of the communication network adopted in the industrial domain is the *half duplex asynchronous serial digital communication*. However, more efficient and powerful buses are gaining market shares in these days, as succinctly reported in Figure 2.15. Again, there is a tradeoff between communication and system complexity.

2.2 Open System Model

In digital data communications, wiring together two or more devices is *one of the first steps* for establishing a network, as well as a proper *software* definition must also be addressed. The *Open System Interconnection* (OSI) model proposed by the *International Organisation for Standardisation* (ISO) is a standard way to structure communication software that is applicable to any network. The model has been standardised by ISO and *International Telecommunication Union* (ITU) *Telecommunication Standardisation Sector* (ITU-T) which is the organisation coordinating standards for telecommunications. The ISO OSI, or simply OSI, model was developed by Pouzin and

Zimmermann in the '70 to structure telecommunication *protocols*.

The communication is based on low-level message passing between the communicating systems.

- A wants to communicate with B ;
- A builds a message addressing B ;
- A executes a call to the communication module to send the message to B .

Of course, A and B need to speak the same language, i.e., they need to agree on the meaning of the bits being sent, and this is called a *protocol*.

Definition 18. *The protocols define the rules for communication. When data is exchanged through a computer network, the system rules are called a network protocol.*

A protocol must define the *syntax*, *semantics*, and *timing* of communication (i.e. *how*, *what* and *when*); the specified behaviour is typically independent of how it is to be implemented. *Syntax* refers to the structure or the format of the data; *Semantics* defines the way in which the bit patterns are interpreted; *Timing* specifies when the data can be sent and how fast it will be. Another term is *synchronisation*.

A protocol can be implemented as *hardware*, *software*, or *both*. To address heterogeneity in networks, instead of using a single universal protocol to handle all transmission tasks, a set of cooperating protocols fitting the *layering scheme* (see Figure 2.16) has been adopted:

- Protocols are structured into *layers*;
- Each layer deals with a *specific aspect* of communication;
- Each layer *uses the services of the layer below it*. An interface specifies the services provided by the lower layers to the upper layers;
- The upper layer sees the lower layer as a *black box*, implementing a specific *protocol*.
- The n -th layer in the sender node *logically speaks* with the n -th layer in the receiver node;
- The protocols for each layer are organised in a *protocol stack*;
- Each protocol adds a *header* to the message to be sent.

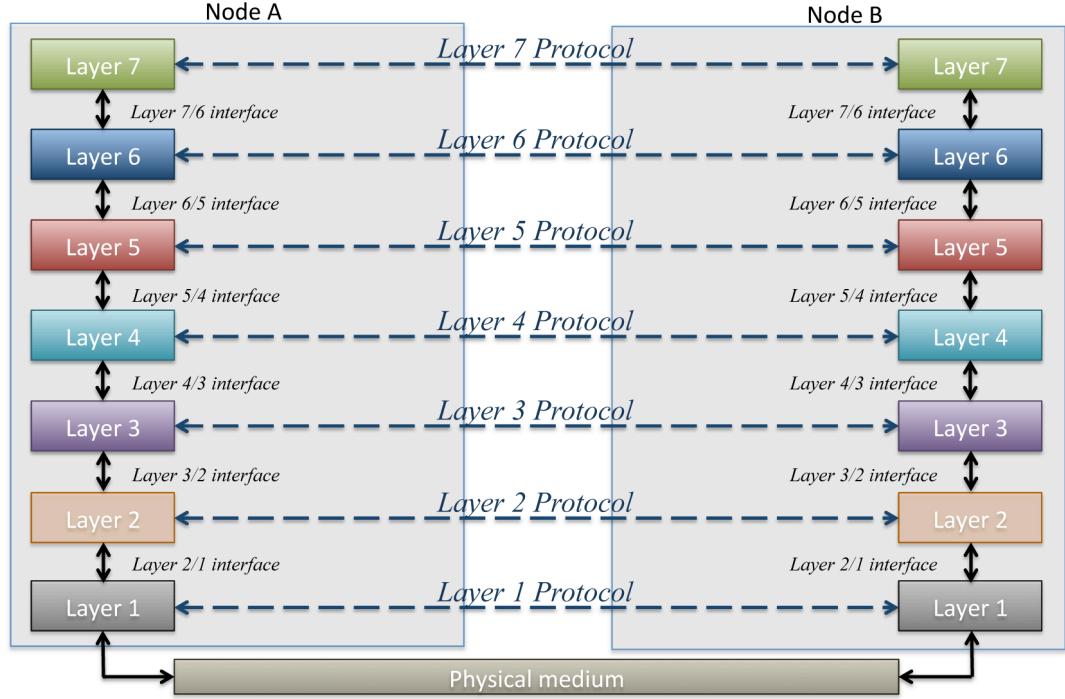


Figure 2.16: Layering scheme.

The OSI (that is a *model* *not* a *protocol*) has been standardised by ITU-T and ISO as ISO/IEC 7498 and almost all communication protocols can be mapped to it. The OSI can be mapped to *industrial communication systems*, since it is specifically designed to allow *open systems* to communicate. A *protocol* is a *particular instantiation* of the OSI *model*, i.e., it specifies all the details of each layer (see Figure 2.17). The message to be sent in Node A moves downwards in the stack, while the received message in Node B moves upwards: each layer adds a header to the message (see Figure 2.18). The information in the layer n -th header is used for the layer n -th protocol, thus generating independence among headers.

The low layers implement the basic functions of a computer network. The *Physical layer* defines the electrical and mechanical definition of the system (e.g., RS485). Hence, it is related to the actual transmission of bits (0s and 1s) and standardises the electrical, mechanical and signalling interfaces so that when A sends a 0 bit, it is actually received as a 0 bit. The *Physical medium parameters* are:

- Coding for 0 and 1;
- Adopted modulation;

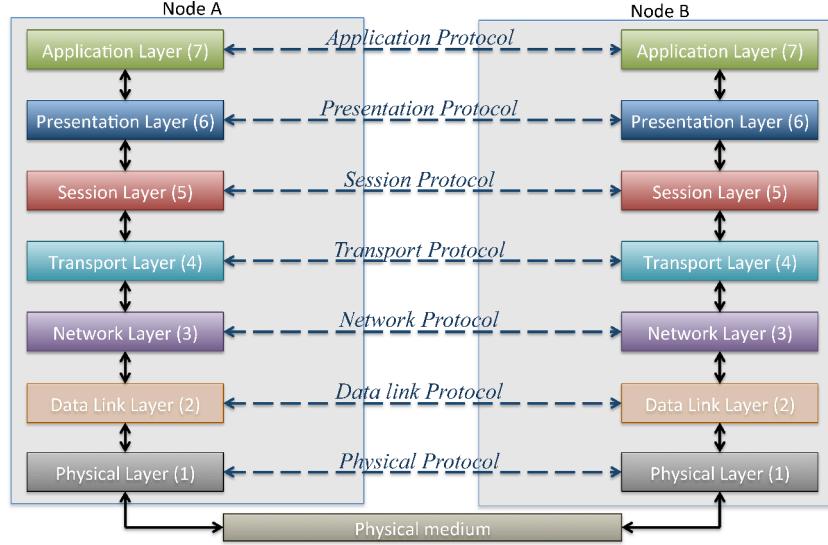


Figure 2.17: Layers of the OSI Model.

- Bits per second, i.e., data rate;
- Electrical and mechanical coupling.

Example 1. For example, the [Ethernet](#) family of network computers is based on the [Ethernet Physical Layer](#). It comprises different media interfaces and different media, i.e. coaxial cable, twisted pair and optical fibre. Network protocol stack software will work similarly on all physical layers. The speed ranges from 1 Mbit/s to 100 Gbit/s. The physical layer is divided into [three sublayers](#):

- [Physical Coding Sublayer](#): performs auto-negotiation and encoding.
- [Physical Medium Attachment sublayer](#): performs framing and synchronisation.
- [Physical Medium Dependent sublayer](#): consists of a [transceiver](#) for the physical medium.

At the [Data link layer](#) it is defined the framing and error correction format of the data (e.g., network protocol [High-Level Data Link Control](#) (HDLC)). This layer groups bits into frames and sees that each frame is correctly received; puts a special bit pattern at the start and end of each frame (to mark them) as well as a checksum; if checksums differ, requests a retransmission; assigns sequential numbers to frames; controls the flow of data. In

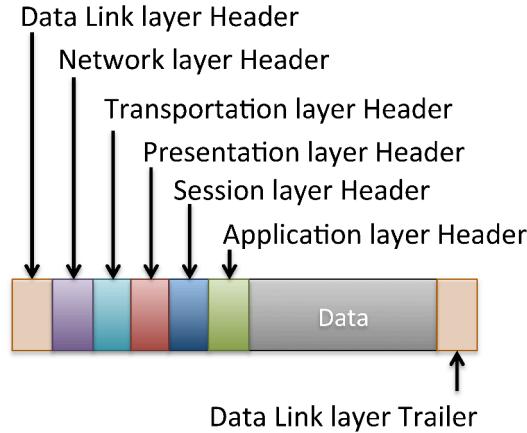


Figure 2.18: Headers added by the OSI Model layers.

Figure 2.19, an example of data exchange at the Data Link Layer between the sender node A and the receiving node B is reported. The use of the checksum is to understand if the communication has been established correctly.

One method to increase the number of systems that can communicate on the same medium and implemented in the *Data-Link layer* is to *virtually parallelise* the shared resource.

Definition 19. *The multiplexing is the technique combining multiple analog or digital signals over a shared medium.*

The data link layer comprises two additional sub-layers. The *Logical Link Control* (LLC), which provides multiplexing mechanisms for successful coexistence of several network protocols within a multipoint network. It can also provide flow control and automatic repeat request error management mechanisms. The *Media Access Control* (MAC) that provides *addressing and channel access control* mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. Ethernet. The hardware that implements the MAC is referred to as a *Medium Access Controller*. There are different methods for determining which node can send a message and this critically determines the efficiency of the LAN. The *Media Access Method* refers to the manner a computer gains and controls access to the network's physical medium (e.g., defines how the network places data on the cable and how it takes it off). One of the primary concern with media access is to prevent packets from *colliding*.

Definition 20. *A collision occurs when two or more computers transmit signals at the same time.*

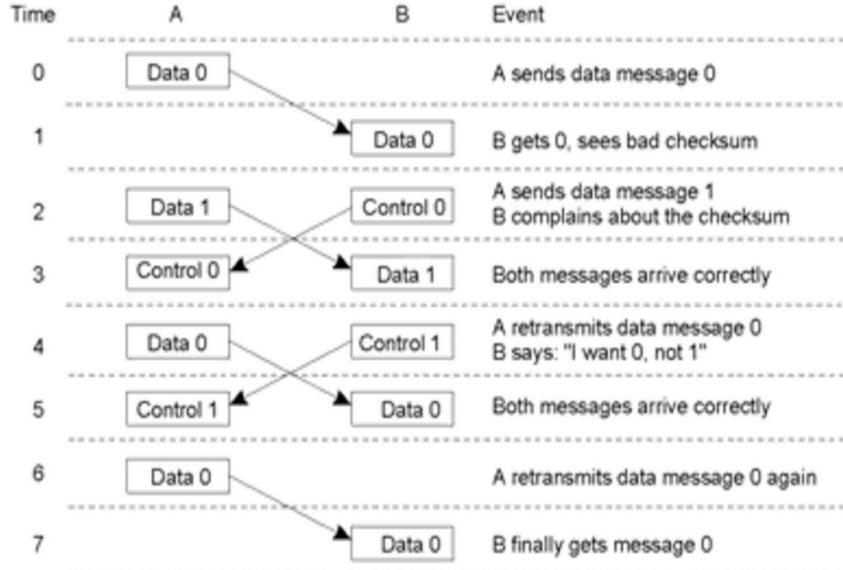


Figure 2.19: Message exchange at the DLL Layer when the communication is not carried out properly.

To classify the industrial networks, the data link layer is the primary differentiator among network protocol types. The MAC sublayer protocol within the data link layer describes the protocol for *obtaining access to the network*. Hence it is responsible for:

- Satisfying the *time-critical / real-time* response requirement over the network;
- The *quality* and *reliability* of the communication between network nodes.

There are three main types of medium access control used in control networks (all implemented in the *Data Link Layer*). The first is the *Time-Division Multiplexing* (TDM), such as *master-slave* or *token-passing*; the second is the *Random access with retransmission* when collisions occur, such as Ethernet and most wireless mechanisms; the third is the *Random access with prioritisation* for collision arbitration such as CAN buses. Hybrid solutions of the previous may exist.

In the *master-slave* (known also as *source-sink* model) there is a fixed assignments of roles. The *master* grants the access to the medium. The *slave* is the entity which accesses the medium after requesting it from the master, i.e., *arbitration*. One example of this bus is the *Profibus-DP*.

In the *token ring*, the members of a *logical* ring gain access to the network upon receipt of a token. The *token* is a group of bits that is passed in a

rotating address sequence from one node to another, e.g., a group of players playing cards. When an available token reaches a node, that node can access the network for a maximum predetermined time, before passing the token on. An example is the *Modbus*. This deterministic access method guarantees that *every node will get access to the network* within a given length of time, usually in the order of a few milliseconds. Because each node gets its turn within a fixed period, deterministic access methods are more efficient on networks that have *heavy traffic*. To transmit, the node first marks the token as “in use”, and then transmits a data packet, with the token attached.

An example of *Random access* approach is the *Carrier Sense Multiple Access*, which is based on a first-come-first-served approach. This operates in a similar manner to polite human communication. We *listen before speaking*, deferring to anyone who already is speaking. Non deterministic method, hence not suitable for networks with heavy traffic. Usually there are two different modalities: the *Carrier Sense Multiple Access - Collision Detect* (CSMA/CA) is an approach where the collision is destructive (example: Ethernet). The steps are the following:

- Collision detection;
- Stop of the emitted frame;
- Scrambling frame emission;
- Wait a random time;
- Frame re-emission.

The *Carrier Sense Multiple Access - Collision Avoidance* is instead a non destructive method (example: CAN bus). In this case:

- Non destructive collision detection;
- The device with the lower priority stops its transmission;
- End of the high priority frame transmission;
- The device with lower priority can send its frame.

At the *Network layer*, optimum *routing* (i.e., choosing the best path minimising the delay) of messages and possibly message segmenting (e.g., *Internet Protocol* (IP)) is mainly enforced. In details this layer is responsible for *source-to-destination* message delivery (it *establishes, maintains* and *terminates* the connections), for *logical addressing* and *message priorities* and

performs *message routing* by dealing with the fact that communication might require multi-hops, i.e., traverse multiple nodes, and at each hop determines through which link to forward the packet.

Example 2. *The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite. It can be considered as the Network layer, even though in the Internet Protocol Suite there it is called Internet layer. The IP delivers packets from the source to the destination using only IP addresses in the packet headers. Therefore, it defines packet structures for data encapsulation and the addressing methods adopted.*

The *Network layer* defines the type of connections. For *connection-oriented* protocols, before exchanging data, the sender and the receiver must establish a connection, possibly negotiate the protocols to be used, and release the connection when done. Modelled after the *telephone system* has been developed. For *connectionless* protocol: no setup in advance. Modelled after the *Postal System* has been adopted.

Remark 1. *The message changes its name when it goes downwards the protocol stack. Frame: physical layer representation of the message. Packet: Network or transport layer representation of the message.*

The middle layers transform the underlying network and its characteristics into a an object that an application developer can use. The *Transport layer* ensures end-to-end flow control and error recovery (e.g. *Transport Protocol Class 0-4* (TP0-4)). In details, it provides the actual communication facilities for most distributed systems, by managing the *end-to-end communication*, it breaks a message received by the application layer into packets and assigns each one of them a sequence number and send them all, it guarantees reliable connection-oriented transport connections built on top of connection-oriented (all packets arrive in the correct sequence, if they arrive at all) or connectionless network services. For instance, the header contains which packets have been sent, received, there is room for, need to be retransmitted.

Example 3. *In the Internet protocol suite there is also the Transmission Control Protocol (TCP). It is the complement of the Internet Protocol, yielding the well known TCP/IP. TCP provides reliable, ordered, and error-checked delivery of the messages between applications on an IP network. The most used internet applications, i.e. the World Wide Web, e-mail, file transfer, rely on TCP.*

Example 4. *The User Datagram Protocol (UDP) is another Transport layer part of the Internet protocol suite. It is used, as the TCP, to let computer*

applications send messages, referred to as datagrams, to other host applications on the IP networks without establishing transmission channels or data paths upfront. Hence, applications that do not require reliable data stream service may use the UDP, which provides a connectionless datagram service that emphasises reduced latency over reliability. To this end, UDP provides minimal protocol functionalities, such as checksums for data integrity, but it does not offer any handshaking mechanism, hence data delivery of messages cannot be guaranteed.

In the highest layers, only the application layer is used in industrial networks. The *Session layer* is responsible for the organisation and synchronisation of the data exchange (e.g., ISO 8326). It establishes and terminates *connections* between the local and remote application, maintains “logical” sessions using as many transport connections as necessary, performs *dialog management*: determines who speaks, when and how long and, hence, deals with simplex, half-duplex or full-duplex characteristics, and establishes checkpointing, adjournment, termination, and restart procedures.

The *Presentation layer* arranges data format or representation (e.g., *Abstract Syntax Notation One* (ASN 1) used for notation, software description and encoding procedures). It deals with *non-uniform data representation* (describing the messages in a platform-independent format and sending the descriptions along with data) and with *data compression*, with syntax and semantics of the data, and defines and *encrypts/decrypts* data types from given by the *Application layer*. For instance, MIDI, MPEG, and GIF are presentation layer formats.

The *Application layer* is where the interface towards the application and it is responsible for the network services, such as file Transfer, message exchange (e.g., *Simple Mail Transfer Protocol* (SMTP)). Moreover, it deals with the *user application*, it manages the *Common Application Service Elements* (CASE), which are login, password checks, etc., it controls the *specific application services*: file transfer, e-mails, message handling, telefax, videotex, etc. Many application protocols are directly implemented on top of transport protocols, doing a lot of application-independent work. At the *Application Layer*, there are concepts defined for:

- *Standardisation of components*, to allow interchangeability. This is necessary for *open systems*;
- *Characteristics of the traffic*, in order to carry out optimal counteractions in the presence of (burst) heavy traffic;
- *Mechanism to access the offered applicative service*, in order to ensure correct information sharing.

Another important concept in this layer is the *open system*. It comprises *interoperable* (i.e., the ability to communicate intelligibly with other devices) and *interchangeable* (i.e., the ability to replace one device with another, possibly supplied by a different manufacturer) components. Interoperability is achieved by means of strict adherence to *protocol specifications*; interchangeability is achieved by means of adherence to *profile specifications*. All manufacturers reserve the right to define whether or not they wish to offer manufacturer-specific functions in addition to those which are part of the minimum profile or core. A *profile*, described with a *strict syntax*, is a standardised way of describing functions which ensure components can be interchanged. Data is grouped by function:

- *Identification*: product name, reference, version, family, manufacturer;
- *Characteristics related to communication*: Speeds supported, type and size of messages exchanged, etc.;
- *Characteristics related to the application*: Variables which can be accessed in write mode, in read mode, when stopped, when running, etc.

Most profiles are provided in electronic file format: *EDS* file, *GSD* file, etc. The profile settings and types are expressed in the available industrial standards, such as the *IEC61499* (mainly related to *abstraction* of network devices building up the Distributed Control System by means of *function blocks*, a concept that is quite close to the *object oriented software*) and the *IEC61131* (mainly related to the design of *control software* in the industrial domain and has a tight relation with *Programmable Logic Devices*).

At the *Application layer* level, the services offered by the applications running on the network are ruled by mechanisms that establish *who is the recipient of a certain service*. One the most used approaches is the *client-server*. The *client* is an entity requesting a service on the network. The *server* is the entity which responds to a request from a client (example: Modbus). Another approach is the *producer-consumer*: the *producer* is a single entity which produces information, while the *consumer* is an entity which uses it (several entities can use the same information). An example is the CAN bus.

Another important aspect at this level is the *traffic type*. The traffic can be categorised according to the particular data exchange rate involved. *Cyclic data* refers to data that is refreshed periodically according to a pre-determined time (example: sampled data for a control loop). In essence it is a *small amount* of information *refreshed frequently*. Instead, the *acyclic data* is generated according to a request or to an event. This is used at start-up for configuration and setup, or for diagnostics in the event of a fault

(example: event based control). In essence it is a *lot of information* without time constraints.

To summarise:

- *Physical layer*: specification and implementation of bits, transmission between sender and receiver.
- *Data link layer*: packages raw bits from the physical layer into frames, it performs error and flow control.
- *Network layer*: Routes *packets*.
- *Transport layer*: End-to-end flow control and error recovery.
- *Session layer*: Management of connections.
- *Presentation layer*: Definition and conversion of the data formats.
- *Application layer*: All services directly called by the end user, such as mail, file transfer, etc.

In practice, the lower three layers, *Physical*, *Data-link* and *Network* layers are the responsibility of the *network*. The upper four layers, *Transport*, *Session*, *Presentation* and *Application* layers are the responsibility of the *host*, i.e. the computer.

As a final remark, the OSI Model can be visualised as a collection of entities, such as software programs situated at each of the seven layers. It provides an overall framework for the vendor in which to package their communications solutions comprising the hardware communications links and the protocols. The OSI model is the reference for all industrial communication. Even when some layers are skipped, the concepts are generally implemented. Examples of commercial solutions can be found in the Figure 2.20.

- Telephone network modems
 - IRDA physical layer
 - USB physical layer
 - 10BASE-T, 100BASE-T, 1000BASE-T, 10GBASE-T, 100GBASE-T, 1000GBASE-T, 10000GBASE-T, 100000GBASE-T and other varieties
 - Ethernet physical layer 10BASE5, 10BASIC, 10BASE2, 10BASE5, 100BASE5, 100BASE7, 1000BASE5 and other varieties
 - Varieties of 802.11 Wi-Fi physical layers.
 - DSL
 - ISDN
 - ITU-T and other E-carrier links, and E1 and other E-carrier links
 - SONET/SDH
 - Optical Transport Network (OTN)
 - GSM Um air interface physical layer
 - Bluetooth physical layer
 - ITU Recommendations: see ITU-T
 - IEEE 1394 interface
 - FireWire
 - Etherloop
 - ARINC 818 Avionics Digital Video Bus
 - G.hn/G.9960 physical layer
 - CAN bus (controller area network) physical layer
 - Mobile Industry Processor Interface physical layer
 - Infrared
 - wi-fi
- ARP Address Resolution Protocol
 - ARChet - Attached Resource Computer NETwork
 - CDP Cisco Discovery Protocol
 - DCAP Data Link Switching Client Access Protocol
 - Distributed Multi-Link Trunking
 - Distributed Split Multi-Link Trunking
 - Dynamic Trunking Protocol
 - Ethernet
 - FDDI Fiber Distributed Data Interface
 - Frame Relay
 - ITU-T G.hn Data Link Layer
 - HLOC High-Level Data Link Control
 - IEEE 802.11 WiFi
 - IEEE 802.16 WiMAX
 - LAG Link Aggregation Control Protocol
 - Latency
 - LocalTalk
 - L2TP Layer 2 Forwarding Protocol
 - L2TP-Layer 2 Tunneling Protocol
 - LAPD Link Access Procedures on the D channel
 - LLDP Link Layer Discovery Protocol
 - LLDP-MED Link Layer Discovery Protocol - Media Endpoint Discovery
 - PAgP Port Aggregation Protocol
 - PPP Point-to-Point Protocol
 - PPTP Point-to-Point Tunneling Protocol
 - Q.931 Simplified Message Transfer Part
 - Multi-link trunking Protocol
 - RPR IEEE 802.17 Resilient Packet Ring
 - SLIP Serial Line Internet Protocol (obsolete)
 - StarLAN
 - STP Spanning Tree Protocol
 - Split-brain link-trunking Protocol
 - Token ring a protocol developed by IBM; the name can also be used to describe the token passing ring logical topology that it popularized.
 - VTP VLAN Trunking Protocol
 - VLAN Virtual Local Area Network

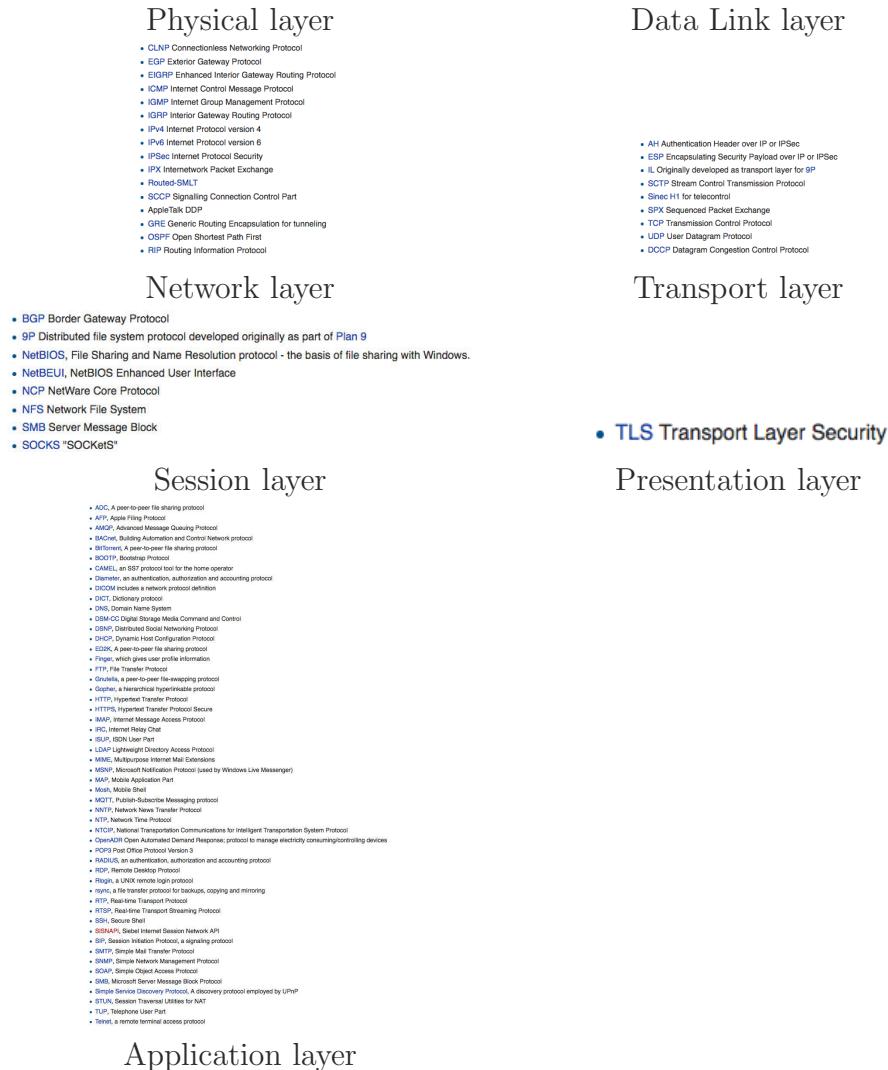


Figure 2.20: Layer examples (Courtesy of Wikipedia).

Example 5. The Controller Area Network (*CAN bus*) is a bus standard defined to let micro-controllers and devices to communicate with each other without the presence of a host computer. It is a message-based protocol originally used in the Automotive industry and it comprises: the Physical layer; the Transfer layer; the Object layer; the Application layer.

At the physical layer, the *CAN bus* defines the use of a medium with multiple-access at the bit level through the use of dominant and recessive states. The electrical aspects are specified in the ISO 11898-2:2003. In its definition, the mechanical aspects (connector type and number, colours, labels, pin-outs) are not formally specified, even though the 9-pin D-sub type male connector with the *CAN-Low (CAN-)* pin 2, *GND (Ground)* pin 3, *CAN-High (CAN+)* pin 7 and *CAN V+ (Power)* pin 9.

The Transfer layer receives messages from the Physical layer and transmits those messages to the Object layer and it is mainly responsible for bit timing and synchronisation, message framing, arbitration and error detection.

The Object layer is responsible for message filtering and message and status handling. In practice the Transfer and the Object layers implements the Data link layer functionalities.

The CANopen is a communication protocol and device profile specification for embedded systems used in automation. It consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. In practice it usually builds upon *CAN bus*, hence it implements the layers above the Data Link layer. Nonetheless, the CANOpen can be implemented over other communication means, such as Ethernet Powerlink or EtherCAT.

Example 6. The IEEE 802.11 is a set of Media Access Control (*MAC*) and Physical layer specifications, released in 1997, for implementing wireless local area network. The bandwidth is usually in the 900 MHz and 2.4, 3.6, 5, and 60 GHz frequency bands. The standard defines the basis for the Wi-Fi network products.

Example 7. The Manufacturing Automation Protocol (*MAP*), a standard released in 1982, focused on real-time control of manufacturing robots of various types. It implements the Physical layer, the Data Link layer and the Application layer, the latter implementing the so called Manufacturing Message System. This stack is intended just for the robots themselves. Although promoted and used by manufacturers such as General Motors, Boeing, and others, it lost market share to the contemporary Ethernet standard and was not widely adopted.

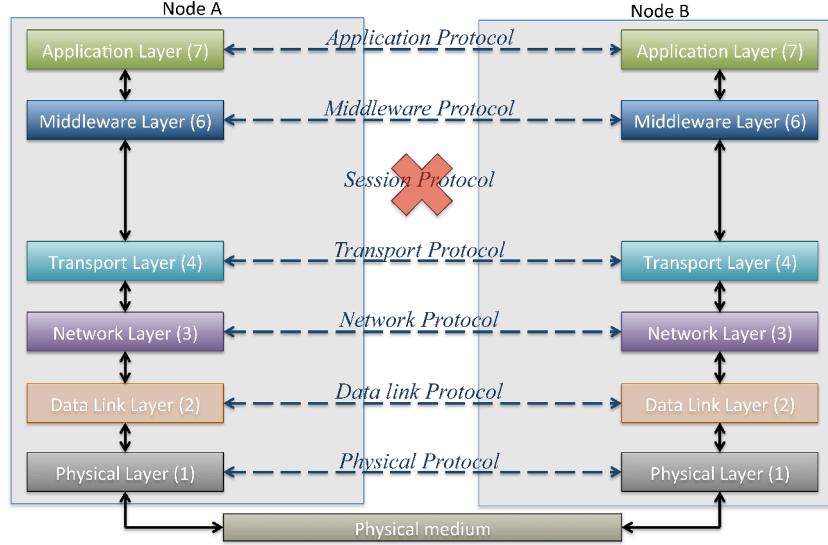


Figure 2.21: Middleware example.

There is a final approach that has become widespread and applies on the OSI model thus defined. The *middleware* has been invented to provide common services and protocols that can be used by a rich set of communication protocols, but which allow different applications to communicate. The relation with the OSI model is reported in Figure 2.21. The middleware is: “The software layer that lies between the operating system and applications on each side of a distributed computing system in a network.” (OW2 Consortium - developers of open source middleware). Services that can be regarded as middleware include *enterprise application integration*, *data integration*, *Message Oriented Middleware* (MOM), *Object Request Brokers* (ORBs), and the *Enterprise Service Bus* (ESB).

Essentially, the *Middleware* abstracts away the particular implementation of a communication system and allows the connected components to communicate using higher level function calls. In practice, devices that are connected to the network are not requested to know in details the specificity of the hardware/software components adopted at the lower levels.

2.3 Industrial Networks

The objective of the industrial network, aka *Local Area Networks* (LAN), is to share information and resources. To share information among the network nodes, they must be connected by some transmission medium organised in a *network topology*. Since the transmission medium is shared, the *Media Access*

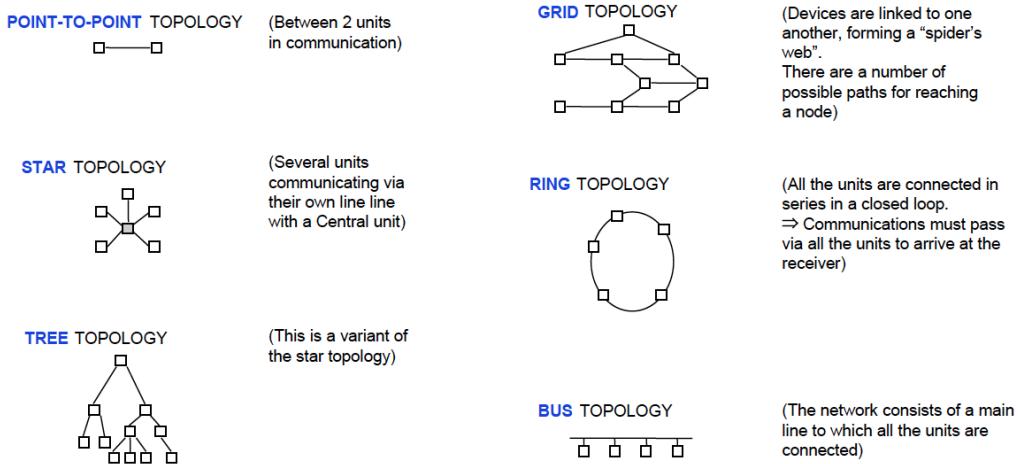


Figure 2.22: Most common topologies (Courtesy of Schneider Electric).

Control (MAC), implemented in the *Data Link Layer*, ensures minimisation of disruption of an established sender. One important concept for networks, and especially for industrial networks, is related to the type of connections among the different elements, which is called *topology*.

Definition 21. *The topology specifies the way the nodes are connected to form a network.*

Basically, a physical topology defines the wiring layout for a network. This specifies how the elements in the network are connected to each other electrically, as reported in Figure 2.22. One of the most common industrial connection is the *bus*, in which each node is connected to a common single communication channel. In a bus, sometimes called a *backbone* as it provides the “spine” for the network, every node can hear each transmitted message packet. Each node checks the *destination address* that is included in the message packet to determine whether that packet is intended for the specific node. When the signal reaches the end of the bus, an electrical terminator absorbs the packet energy to keep it from reflecting back again along the bus cable, hence avoiding interference. Therefore, each end of a bus cable must be terminated. In a long bus topology, the signal strength is boosted up by some form of amplification, or *repeater*. The main advantages of the bus topology are:

- *Simplest wiring*: A bus uses relatively little cable compared to other topologies and offer a simple arrangement of wires;

- *Scalability and flexibility*: Since nodes are connected by high *impedance tappings* across a backbone cable, it's easy to add or remove nodes from a bus. This makes it easy to extend a bus topology;
- *Data broadcasting*: The broadcasting of messages is advantageous for one-to-many data transmissions.

The main disadvantages of the bus topology are:

- *Security*: each node sees each message;
- *Fault isolation*: A faulty node can be everywhere along the bus line;
- *No handshake*: There is no automatic acknowledgement of messages, since messages get absorbed at the end of the bus and do not return to the sender;
- *Low throughput*: When the network traffic gets increasingly heavy, the MAC grants the access with an increasing difficulty, hence the can spend much of their time trying to access the network.

A *star topology* is a physical topology in which multiple nodes are connected to a central component, generally known as a *hub*. The hub may be a wiring centre or may actually be a file server. All signals, instructions, and data going to and from each node *must pass through the hub* to which the node is connected, e.g., telephone lines. There are *not so many* industrial LAN implementations that use a logical star topology. The main advantages of the star topology are:

- *Fault isolation*: Troubleshooting is very easy and a failure of a single node does not isolate any other node;
- *Scalability and flexibility*: It is easy to add or remove nodes, and to modify the cable layout;
- *Monitoring*: The inclusion of a central hub allows easier monitoring of traffic for management purposes.

The main disadvantages of the star topology are:

- *Centralisation*: If the hub fails, the entire network fails;
- *Cabling*: A star topology requires a lot of cabling compared to the number of nodes.

A *ring topology* is distinguished by the fact that message packets are transmitted sequentially from node to node, e.g., point-to-point system, in a predefined order. Hence, each node is connected to exactly two other nodes. Nodes are arranged in a *closed loop*, so that the initiating node is the last one to receive a packet. Hence, *simplex communication*. Equivalence of physical and logical description. In a ring topology, *each node can act as a repeater*, boosting the signal before sending it on. The node packets have an address, so *each node checks* whether the message packet's destination node matches its address. When the packet reaches its destination, the destination node accepts the message, then sends it back to the sender, to *acknowledge receipt*. Ring topologies use *token passing* to control access to the network. The main advantages of the ring topology are:

- *Cabling*: A physical ring topology has minimal cable requirements;
- *No centralisation*: No wiring centre or central server is needed;
- *Automatic acknowledgment*: The message can be automatically acknowledged;
- *Regeneration*: Each node can regenerate the signal.

The main disadvantages of the ring topology are:

- *Fault tolerance*: If any node goes down, the entire ring goes down;
- *Fault isolation*: Troubleshooting is difficult because communication is only one-way (simplex);
- *Scalability and flexibility*: Adding or removing nodes disrupts the network;
- *Physical distance*: There will be a limit on the distance between nodes.

2.3.1 Interconnection Systems

In each and every network presented, the nodes are interconnected to form different topologies. We will now consider the *interconnecting components* among the different branches of the network. There are different kind of components that can be used to create a topology, which will be presented next. Before going into details, an interesting naming convention is useful: by the IEEE definition, a *network segment* would correspond to the individual connection between *end station to network equipment* (e.g., repeater) or the connections between *different pieces of network equipment*.

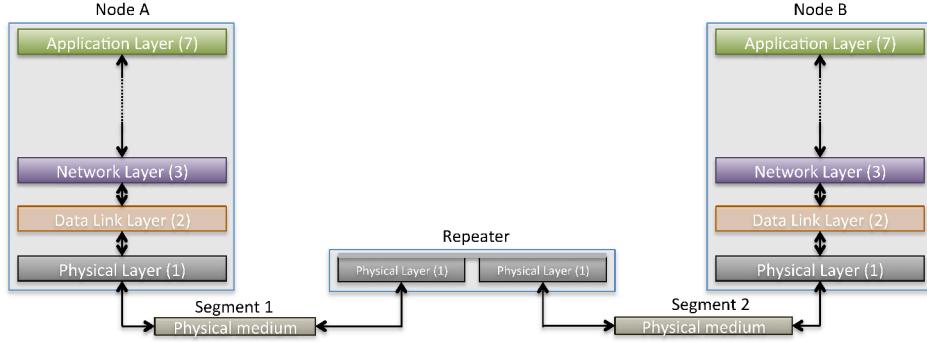


Figure 2.23: Repeater.

Definition 22. A *segment* is an electrical connection between networked devices (IEEE standards for Ethernet).

Let us now review the different components connecting network segments. The *repeater* is used to add *segments* to a network. The repeater amplifies and restores the signal, as reported in Figure 2.23. All traffic that appears on one side of the repeater appears on both sides. Repeaters handle *only* the electrical and physical characteristics of the signal. In Ethernet networks, for example, they operate according to the so called *3-R principle*: reshaping, retiming, and retransmitting.

A basic *hub*, also known as *multi-port repeater* or *concentrators*, is merely a collection of repeaters, so no intelligence and microprocessors on board. Some hubs are instead intelligent, hence can perform basic diagnostics and test the nodes to see if they are operating correctly. The hub amplifies and restores the same type of signal on *all* ports, therefore *does not* reduces the number of collisions. They are usually used to extend star topologies.

A *switch*, also known as *switched hub*, amplifies and restores the same type of signal on a *single* port. They are usually used to reduce the number of collisions in the network. They are usually used to extend star topologies.

The *bridge* is used to connect two networks using the same protocol but different lower layers, as reported in Figure 2.24. An example is the Modbus RS485/Ethernet TCP-IP bridge. Instead, the *router* is used to connect two networks of the same type (e.g. Ethernet TCP-IP router), as reported in Figure 2.25. The *gateway* is used to connect two networks of different types, as reported in Figure 2.26.

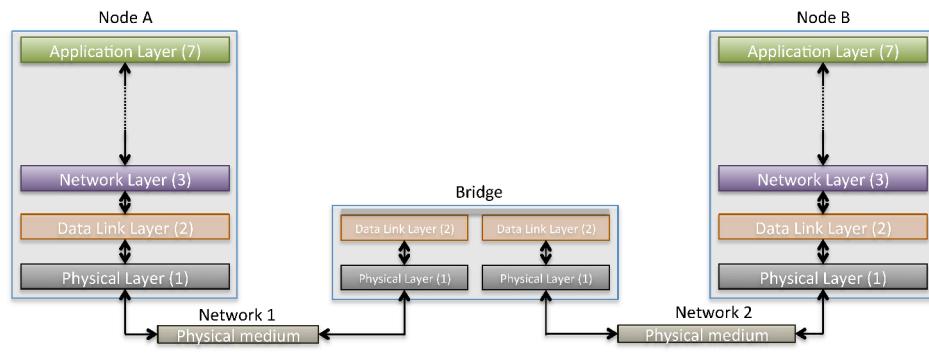


Figure 2.24: Bridge.

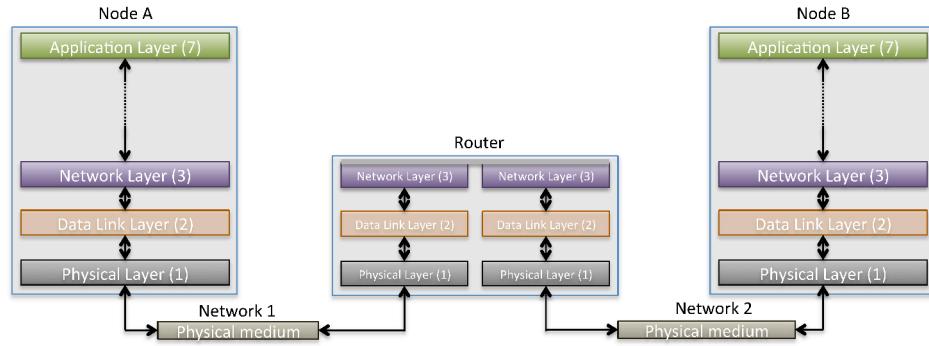


Figure 2.25: Router.

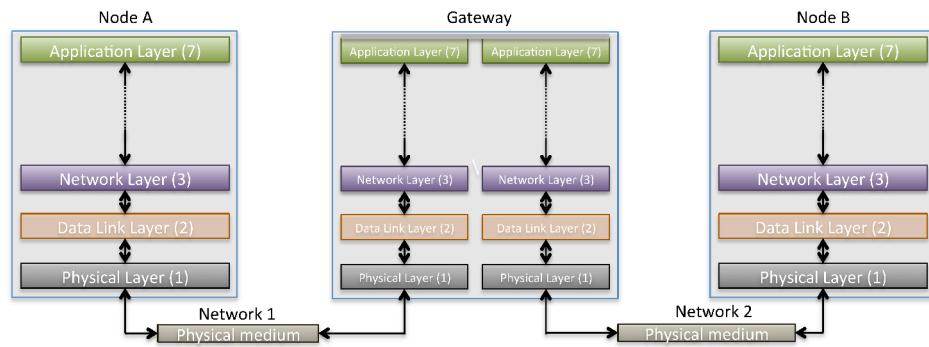


Figure 2.26: Gateway.

Chapter 3

IEC 61499 Standard

Each industrial sector will sooner or later come to a level of maturity when it is necessary to *unify different approaches*. *Standards* are specifically conceived for this purpose. Without creating the standards, development cannot progress. More precisely, the use of new standard enable:

- Installation and start-up cost reduction.
- Large stocks of standard devices.
- Components compatibility.
- Safety increase.

The industrial community has long been aware that the ready *interconnection of software components* will have major advantages, especially for end-users. The major advantages are: *improved software productivity* through the re-use of standard solutions and *improved design flexibility* by being able to plug-and-play software and devices from different vendors. In the past, standards have focused mainly on enabling *technical integration* of distributed components. Recently, the major hurdle of *semantic integration* has been addressed, i.e., *making the data exchange between software in a remote industrial controller and a control algorithm running in a PC be meaningful*.

In early 1990, Technical Committee 65 of the *International Electrotechnical Commission* (IEC TC65) received a proposal to standardise certain aspects of the application of software modules in *Distributed Industrial-Process Measurement and Control Systems*. The software modules subject of this new proposal are called *function blocks*. The *function blocks* (FB) are also part of the programming language standard IEC 61131-3 for *Programmable Logic Controllers* (PLCs).

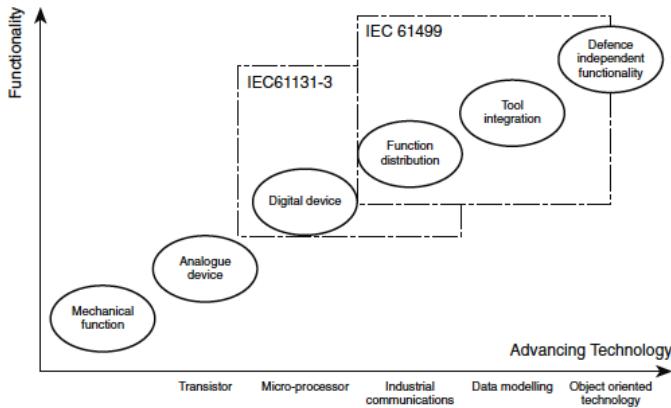


Figure 3.1: Technology development for industrial control [6].

Since the new proposal made explicit reference to the *fieldbus* standard, the request of defining a common model leads to the IEC TC65 and, finally, to the standard IEC 61499 (see [5, 6, 7]). In practice the standard defines *the basic concepts and detailed technical approach to achieve a common model for the application of FBs in DCSs*.

To make a long story short, when creating a distributed control systems, two standards have been developed from the *International Electrotechnical Commission* (IEC):

- *IEC 61131*: Used for a long time in the PLC sphere, but it does *not* directly support the creation of extensive distributed systems;
- *IEC 61499*: Offers a direct solution for *distributed control system* creation by means of FBs.

The relation between the two standards can be graphically seen in Figure 3.1.

The standard IEC 61499 has been defined in order to make the automation systems *reconfigurable, interoperable and portable*. The standard is based on the FBs conception. Strictly speaking, the concept underlying the FBs is the same that have pushed towards *object oriented programming* in computer science. To deal with the complexity of the object oriented programming, new methodologies, such as the *Universal Modelling Language* (UML), have been proposed.

Similarly, control and system engineers are also facing with increased software complexity as advances in *industrial networking* allow intelligence

to be *distributed* throughout the system from controllers, instruments, actuators and even out to the sensors themselves. The IEC 61499 deals directly with such a complexity and provides a methodology for modelling DCSs by defining concepts and models so that software in the form of FBs can be interconnected to define the behaviour of a DCS.

First, the *field area networks* (i.e. networks build up with *fieldbuses*) were used just to collect data from sensors distributed in geographic area to be elaborated in *Programmable Logic Controllers* (see IEC 61131). Then integration components architecture, such as *Modbus-IDA* or *Profinet-CBA*, where proposed to integrate the various PLCs of the network. Finally, *the intelligence has been distributed* over the network. Distribute and embed functionalities into *networked hardware devices* was identified as a *grand challenge*. The IEC 61499 architecture was conceived just for these purposes, but until the emergence of *powerful software tools* it could not find its way to industrial applications. By increasing the *design performance of automation software components*, IEC 61499 becomes the standard for distributed automation.

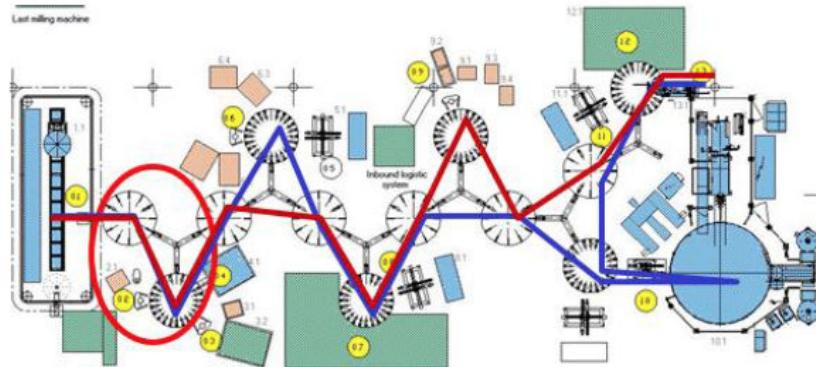
As an example, the shoe factory presented in [8] (Figure 3.2-a) is logically represented as in Figure 3.2-b. The IEC 61499 control program of the *molecular line* inherits reusability and reconfigurability of the equipment. The line controller uses *six instances* of the *Tern Control* FB. Each *Tern Control* is a combination of the *Table*, *Manipulator* and *Island* FBs (see Figure 3.3). On its turn, each of these is a composite FB. At the lowest level, each FB is programmed using *state machines language*, e.g. *Execution Control Charts*, and traditional PLC languages, e.g. *Ladder Logic Diagrams*.

FBs have been used for a long time by the process and system engineers in various forms for an effective way to *represent and model* software functionality in instrumentation and controllers. For example, the PID algorithm is an early form of a FB. However, new forms of smart devices and sensors now allow intelligence to be *distributed widely throughout a control system*. It is now very difficult to define where the main intelligence of any control system really resides! FBs solve part of the problem related to the *semantic* integration of the different software components in a DCS. A FB can provide a software solution to a small problem, such as the control of a valve, or control a major unit of a plant, such as a complete production line. FBs allow industrial algorithms to be encapsulated in a form that can be readily understood and applied by people who are not software specialists.

FB execution is controlled by *events*, with very fast responses, i.e. *event-driven execution*. Hence the FB *remains idle* unless an event is sent to one of its *event inputs*. The main motivation for event-driven execution is the desire to make *the code independent of the sequence of FB invocation*, i.e. *portability*. Nonetheless, the execution can also be *cyclical*. This is important



(a)



(b)

Figure 3.2: Shoe factory example of [8]: the plant (a) and the logical representation (b).

for many control programs in modern devices. Each block has a defined set of *input parameters*, which are read by the internal algorithm when it executes. The results of the algorithm are written to the *block's outputs*. Complete applications can be built from *networks of function blocks* formed by interconnecting block inputs and outputs. A graphical representation of a FB is reported in Figure 3.4.

FB models open to the coexistence of *several alternative* algorithms in the FB body. The various alternative are selected according to *external events* (or *conditions*) (e.g., initialisation algorithms, standard algorithms, post-failure algorithms, etc). Data inputs are updated at the same time. Hence, the basic principle of IEC 61499 consists of *event controlled function blocks*. A FB stays

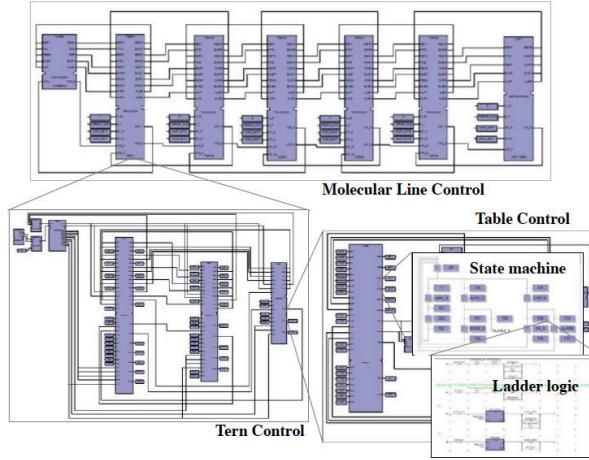


Figure 3.3: FB implementation of Figure 3.2 and reported in [8].

inactive during the remaining operating time. This will significantly improve efficiency, reducing energy consumption and workload of the communication channel. Notice that such a *function encapsulation* into FBs increases the possibility of reuse. IEC 61499 defines a general model and methodology for describing FBs in a format that is *independent of implementation*. For instance, algorithms can be written in *different programming languages*. Since a system can be defined in terms of *logically connected function blocks* that run on different processing resources, the methodology easily allow the design of DCSs. In other terms, this methodology is used throughout the system design life-cycle, i.e., from the analysis of the plant to the definition of the DCS.

As reported in Figure 3.5, the design of a control system typically starts with the analysis of the physical plant diagrams and documentation of the control system requirements [6]. The plant is analysed using standard tools, e.g., *Process and Instrumentation Diagrams* (P&ID), to create functional requirements. In many cases, as in P&ID, the diagram of the functional requirements and machinery physical design reports the location of actuators and sensors. For example, the previous example shows the presence of valves and pumps (*actuators*) and pressure and temperature instruments (*sensors*). In this phase, the main software components and their *interconnections* are defined.

For a DCS, not only *which is the algorithm* is relevant, but also *where* it is executed. In the *distribution phase*, the functionality are allocated to the *processing resources*. The IEC 61499 standard provides models and concepts for defining the *distribution of functionality* into interconnected FBs. In

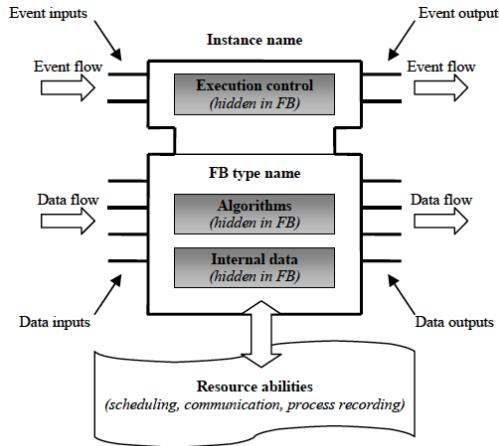


Figure 3.4: Function block model [9].

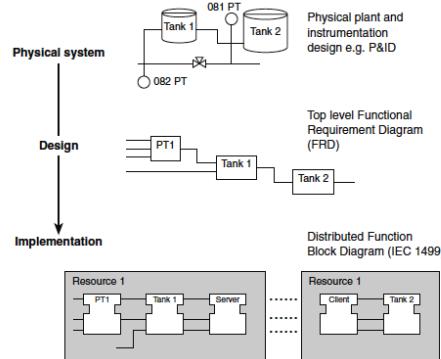


Figure 3.5: Design life-cycle [6].

other words, the functionality is split into various *function blocks* that are distributed on *various processing resources*. This idea enables the design of *intelligent devices*, such as smart valves that can provide software packaged as a FB. Each FB used can be designed from scratch or based on a an existing solution. Hence, the final phase results in mapping functionality into physical resources such as PLCs, instruments and controllers. The PID controller for the plant in Figure 3.5 is represented in Figure 3.6.

The standard is divided in two parts, each considering different aspects of the automation components. *Part 1* of the standard covers the architecture and concepts for designing and modelling FB oriented systems, in particular: general requirements, including an introduction, scope and normative references (i.e. to other standards), definitions and reference models; rules for the declaration of FB types, and rules for the behaviour of instances of

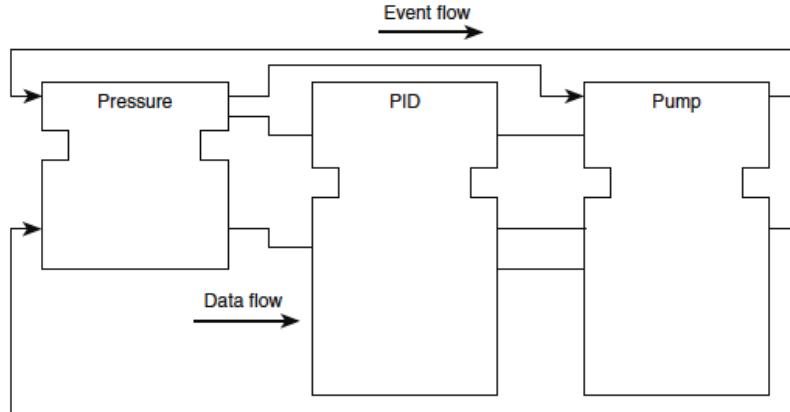


Figure 3.6: PID control of a pump [6].

FB types; rules for the use of FBs in the configuration of DCSs; rules for the use of FBs in meeting the communication requirements of DCSs; rules for the use of FBs in the management of applications, resources and devices in DCSs; requirements for compliant systems and standards.

Part 2 defines software tool requirements to create FB type definitions and manage FB libraries. It includes an extensive annex of *eXtended Markup Language* (XML) document definition types for the exchange of FB designs between software tools from different suppliers. The main focus of Part 2 is *engineering task support* and provides guidance on engineering tasks concerned with the design, implementation, operation and maintenance of DCSs constructed using the architecture and concepts defined in Part 1.

FBs also share most of the benefits introduced in programming by the *objects*, which are: the quantity of control software to be developed for an application is reduced by using FBs; the time required to develop control systems is reduced; control systems using the same types of FB will have more consistent behaviour; the quality of control systems will be improved. In particular, the *Object language embedded for Process Control* (OPC) is used to create *communication links* independently from the application.

3.1 Design views

No matter how hard people have tried, it is just *not possible* to convey all aspects of a distributed system design using one *design methodology*. For example, the software runs in different processing resources deployed in a large area. The solution is borrowed from object oriented software design,

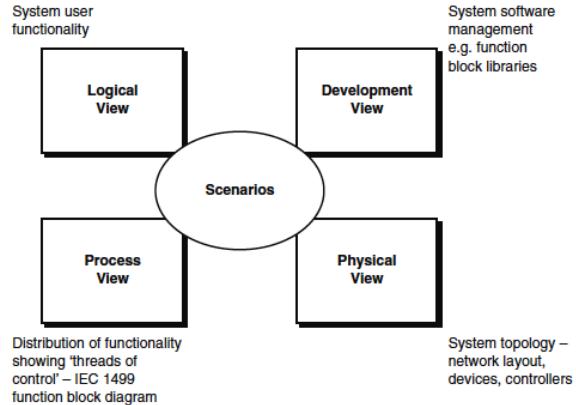


Figure 3.7: System development preview model [10].

developed by P. Kruchten in [10]. In fact, it is now recognised that most complex software designs can require *at least four different design views and a set of scenarios*. Some views will express the *abstract aspects* of the design while others are required to show the *physical structure* of the system or the way the software is organised. Hence, an aiding tool is represented by *previews*: some of the previews express abstract design aspects, some are needed to see the system's physical structure, or its software organisation. This idea can be mapped in the design phases of IEC 61499, as reported in Figure 3.7. The design previews are:

- *Logic preview*: This design view is used to show *function requirements* of the system. It expresses the software functionality requested by the system user. Major software FBs and major interfaces between them are reproduced in distributed system design.
- *Process preview*: It contains many *non-functional system requirements*, including performance or system distribution. During application of IEC 61499 standard, which provides the architecture showing the implementation preview of the distributed system, a network of interconnected function blocks is created.
- *Development preview*: This preview shows the *organisation of developed software* integrated into a larger system and relations between software components. Creating a large scale DCS includes a number of software libraries and modules. For example, consider a FB used for conveyor control; we would want to indicate which device types support it and we would like to show any constraints on other blocks that need to interact with it.

- *Physical preview*: It shows *physical devices* in distributed systems and controllers within the system and reflects different network communication connections between them.

According to Kruchten [10], the last component is the *scenario*: a scenario depicts the major *interactions* between units of software to provide the most important, key functionality of a system. Possible scenarios of a DCS may be: system start-up, device fault detection and recovery, recipe activation, system shutdown, etc. This methodology defines the classic computer science approach dubbed '*what if?*'. Nonetheless, there is currently *no methodology* defined by any IEC standard that can be used to define scenarios for DCSs.

IEC 61499 provides just *one* of the five design views required for DCSs. The other views have emerged as designers start to face the challenge of building large distributed systems. For example, many research works have been trying to introduce various trendy ideas *from computer science into industrial automation* context using IEC 61499 as a vehicle. In most cases, these research works demonstrate *the potential of IEC 61499 to implement those techniques* in a much more consistent way than any existing IT technology used in industrial automation.

3.2 Models

The primary purpose of IEC 61499 is *not* as a programming methodology but as an *architecture and model* for distributed systems. The IEC 61499 provides a set of models for describing distributed systems that *have been programmed* using *function blocks*. Hence, the IEC 61499 provides *terminology, models and concepts* to allow the implementation of a FB oriented DCS to be described in an *unambiguous and formal manner*. To this end, the standard uses few *models* which together form the *architecture of a DCS* oriented on FBs:

- *System model*: At the physical level, a DCS consists of a *set of devices interconnected by various networks* to form a set of *co-operating applications* (see Figure 3.8). Hence, the *system model* defines relations between communication devices and applications. Indeed, an application can be physically executed on a single device or can contain functions distributed for a larger number of devices. Hence, a *distributed application* is shown as a FB network.
- *Device model*: A purpose of a device is to provide *an infrastructure* to support one or more *resources* (see Figure 3.9). A resource provides *in-*

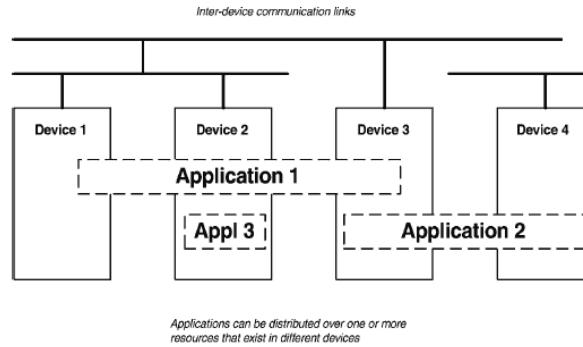


Figure 3.8: System model [6].

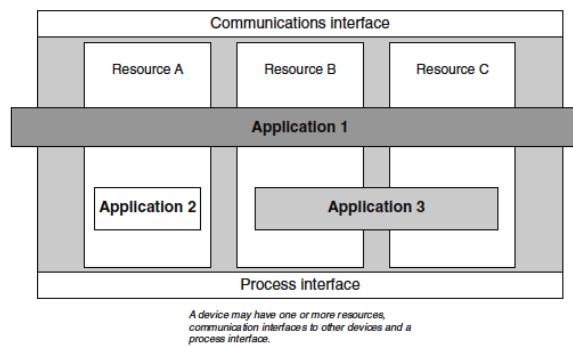


Figure 3.9: Device model [6].

dependent execution and control of (networks of) FBs. The model contains the *process interface* which provides services that allow resources to exchange data with input-output points on the physical device. Similarly, the *communication interface* provides connection through the external network, hence, allowing the existence of the *remote resources*.

- *Resource model:* The resource provides *equipments and services* to single or multiple FB *applications or application fragments*. FBs of distributed systems are placed into *interconnected device resources* (see Figure 3.10). The purpose of the model is to define the *behaviour of FBs within each resource*. For example, each resource will have an *interface to the communications system* to allow FBs to exchange data with blocks in remote resources and an *I/O interface* to read and write on the local device. An important feature of the resource is its support for *independent execution*: a resource can be recorded, configured, activated or stopped without influencing other resources in the same

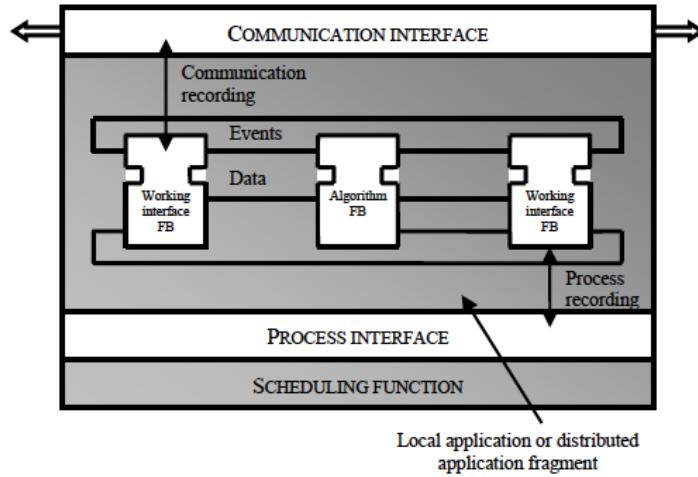


Figure 3.10: Resource model [9].

device, or network.

- **Application model:** An IEC 61499 application is defined as a *network of interconnected function blocks* linked by events or data flows (see Figure 3.11). An application can be *divided and distributed in multiple resources*. The resources on which the FBs are distributed must ensure that the events are used for particular *algorithm timing* in FBs with correct priority and time. In practice, *an application is the entire set of function blocks and interconnections to solve a particular automation control problem*. Examples might be: the set of FBs required to control a production line.

3.3 Function Block Types

At the core of the standard is the *function block model* that underpins the whole IEC 61499 architecture. A FB is described as a *functional unit of software* that has its own data structure which can be manipulated by one or more algorithms. A *function block type* definition provides a formal description of the data structure and the algorithms to be applied to the data that exists within the various instances. For example, the PID block may have *two instances* working on a different set of variables but still sharing the same algorithm. The execution model of a FB is reported in Figure 3.12. The main features of a FB are summarised as follows:

- Each FB type has a type name and an instance name;

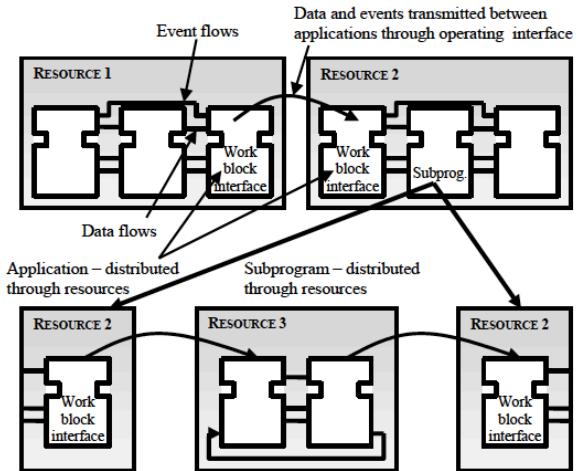


Figure 3.11: Application model [9].

- Each block has a set of event inputs, which can receive events from other blocks via event connections;
- There are one or more event outputs, which can be used to propagate events on to other blocks;
- There is a set of data inputs that allow data values to be passed in from other blocks;
- There is a set of data outputs to pass data values produced within the function block out to other blocks;
- Each block will have a set of internal variables that are used to hold values retained between algorithm invocations;
- The behaviour of the FB is defined in terms of algorithms and state information. Using the block states, various strategies can be modelled to define which algorithms are to execute in response to particular events.

In IEC 61499, a *function block type* defines *the behaviour and interfaces of FB instances* that can be created from the type. A function block type is defined by: type name; formal definitions for the block's input and output events; definitions for the input and output variables. In particular:

- *Basic FBs*: components containing *Execution Control Chart* (ECC) ruling the mapping of events on algorithms. The execution generate *output event signals*. Various programming languages can be adopted;

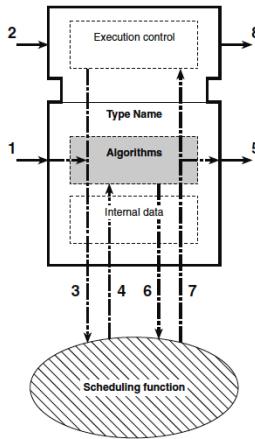


Figure 3.12: Execution model for a basic FB [6].

- **Composite FBs:** composition of basic or composite blocks. The definition therefore includes data and event connections that need to exist between the internal function block instances;
- **Service Interface FBs:** These FBs provide an interface between the FB domain and external services. Because a service interface function block type is primarily concerned with data transactions, it is defined using time sequence diagrams.

An example of a PID implementation by means of a FB is reported in Figure 3.13.

Several programming tools are used according to IEC 61499 standard:

- **FBDK** is the first programming tool which follows IEC 61499. It is written in the Java language and FBs are implemented as Java classes. FBDK is not used in the field, but rather it is mainly used in *academic environments and development communities*;
- **ISaGRAF**, by ICS Triplex ISaGRAF, is the first full-valued automation product that supports the entire design process.
- **FBench**, an open source project initiated by the Canadian nonprofit organisation OOONEIDA, is able to design, develop, debug, activate and verify the IEC 61499 application. FBench aims to be a complex tool, which supports programming languages of both IEC 61131 and IEC 61499 standards.

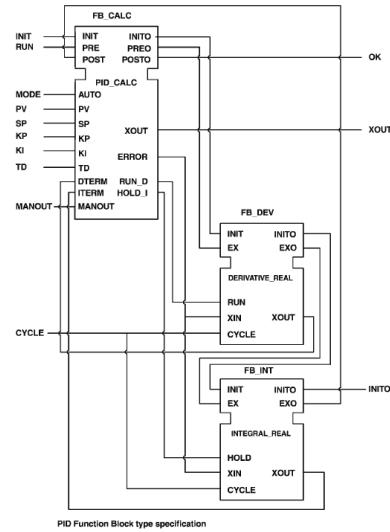


Figure 3.13: PID control implemented using a set of FBs [6].

Some solutions exist that tries to extend the application of the IEC 61499 to the world of traditional PLCs. The extension comes with existing communication standards for *industrial communication*. In particular, we will see that *PROFINET* is an open standard for automation based on industrial Ethernet. *PROFINET CBA* is a part of this standard and is based on IEC 61499-1: describes technologies for modular and distributed system implementation on the base of *pre-defined components*.

Chapter 4

Fieldbuses

The automation protocols are communication protocols conceived for automation. A list of possibilities is reported in Table 4.1 for reference. In the table, M/S = master/slave, P/S = publisher/subscriber, PtP = point to point or peer to peer, M-M = multi-master, C-S = client-server. It is of course obvious that so many solutions exist given the specific application that is intended to solve.

In this notes, we will see in some more details only a small subset of the proposed solutions. The choice is dictated by the *popularity* gained for industrial automation solutions. In particular, we will see:

- *Fieldbuses*: Foundation Fieldbus H1, CAN bus, DeviceNet, PROFIBUS;
- *Real-Time Ethernet-based industrial communication systems*: Foundation Fieldbus HSE, EtherCAT, Ethernet Powerlink, PROFINET IO, PROFINET IRT.

4.1 Fieldbus

A DCS is organised as a *hierarchy of controller systems*, as depicted in Figure 4.1. At the top of the hierarchy there is usually the *human supervisor* that interact with a *Human Machine Interface* (HMI) to monitor data in a meaningful way or operate the system. The second level of the hierarchy usually comprises a network of *programmable logic controllers* (PLCs) through a non-time-critical communication system (e.g. Ethernet). The lowest level of the hierarchical control chain there is the *fieldbus* connecting the PLCs with sensors, actuators, switches, etc. At the beginning of the industrial automation, components were connected through *point-to-point* links. For example,

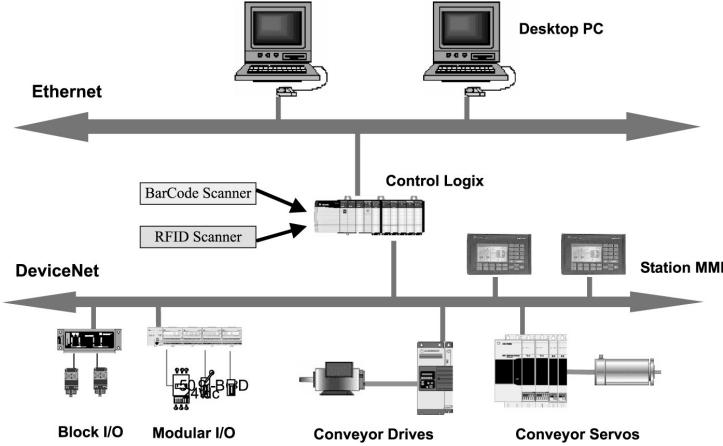


Figure 4.1: Example of hierarchical structure [11].

the classic RS-232 serial communication bus was adopted. The fieldbus allows to build up a *network of devices* with a LAN. Fieldbus is hence the name of a *family of industrial digital network protocols* used for real-time distributed control. Indeed, the generic term *fieldbus* refers to any bus that connects to field devices.

The fieldbus is standardised in the *IEC 61158* that defines 8 different *protocol sets*, called *type* (the majority of them reported in Table 4.1):

- *Type 1* Foundation Fieldbus H1;
- *Type 2* ControlNet;
- *Type 3* PROFIBUS;
- *Type 4* P-Net;
- *Type 5* FOUNDATION fieldbus HSE (High Speed Ethernet);
- *Type 6* SwiftNet (Boeing protocol, no more active);
- *Type 7* WorldFIP;
- *Type 8* Interbus.

There are different standards defined over the fieldbus, coming from different requirements and application domains. Allen-Bradley (US) developed industrial standards that eventually grew into *DeviceNet* and *ControlNet*.

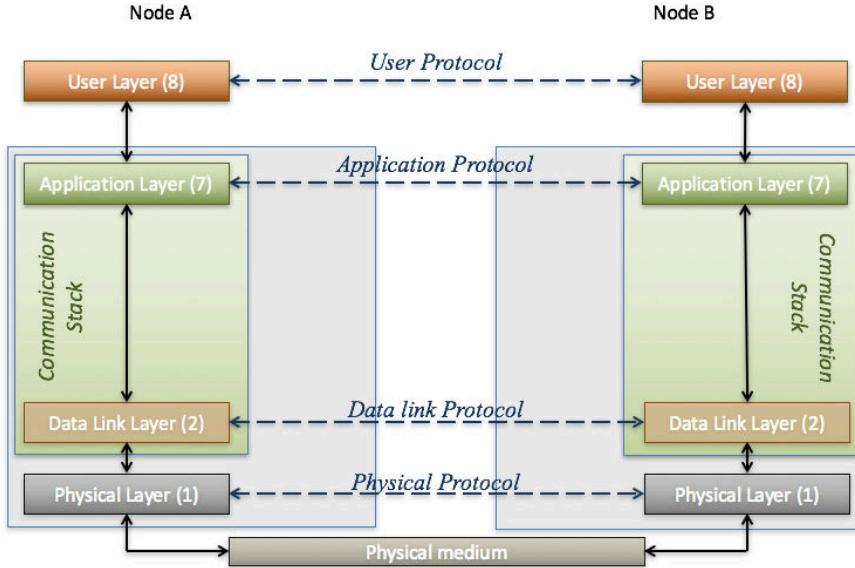


Figure 4.2: Typical OSI model implementation for a fieldbus.

Siemens (Germany) and other manufacturers developed an industrial protocol for automation which evolved into *PROFIBUS*. Bosch GmbH (Germany) first developed the *Controller Area Network* (CAN) to connect different control units in a car, which is now shifted in the automation domain. Despite the fact that they are all fieldbuses, the various solutions *are not* interchangeable.

With respect to the OSI model, fieldbus communication layers consists of only the *Physical layer*, the *Data Link layer*, the *Application layer*. The *Data Link layer* and the *Application layer* are usually referred to as the *Communication stack* (see Figure 4.2). Fieldbus does not implement layers three (*Network*), four (*Transport*), five (*Session*), and six (*Presentation*) of the OSI model because the services of these layers are not required in a process control application. A very important part of the fieldbus is the defined *User layer*, sometimes referred to as layer eight.

4.1.1 Foundation Fieldbuses: H1 Protocol

In light of the previous description, we will focus on a specific fieldbus, the *FOUNDATION fieldbus* (FF, see the logo in Figure 4.3) as reference to highlight the characteristics of a generic fieldbus. *Fieldbus Foundation*: responsible for the definition of the FF specification and of the certification of the compliant products www.fieldbus.org. The *Fieldbus Foundation* defines as fieldbus *A digital, two-way, multi-drop communication link among intelligent devices*.



Figure 4.3: Logo of Fieldbus FOUNDATION members.

gent measurement and control devices. It is one of the several possibilities available for *Local Area Networks* dedicated to industrial automation. The goal of the FF is to help *create products that use a robust industrial network based on existing standards and other proven technologies and to increase the standardisation of those products*. FF is an open standard, which allows to use FF products *from different vendors interchangeably* in process control and monitoring and able to implement *distributed control*.

The FF provides a broad spectrum of services and functions compared to other fieldbus systems:

- Intrinsic safety for use in hazardous environments;
- Bus-powered field devices;
- Bus or tree topology;
- Multi-master capable communication;
- Deterministic (predictable) dynamic behaviour;
- Distributed Data Transfer (DDT). With DDT, the single field devices can *execute automation tasks* so that they are no longer just sensors or actuators, but contain additional functions (implemented by *function blocks*);
- *Standardised block model* for uniform device interfaces (*interoperability and interchangeability*);
- Flexible extension options based on device descriptions.

FF is an all-digital, two-way, multi-drop communication system that *brings the control algorithms into instrumentation*. It is able to build a LAN for FF devices including process control sensors, actuators, and control devices and, unlike many traditional systems requiring a set of wires for

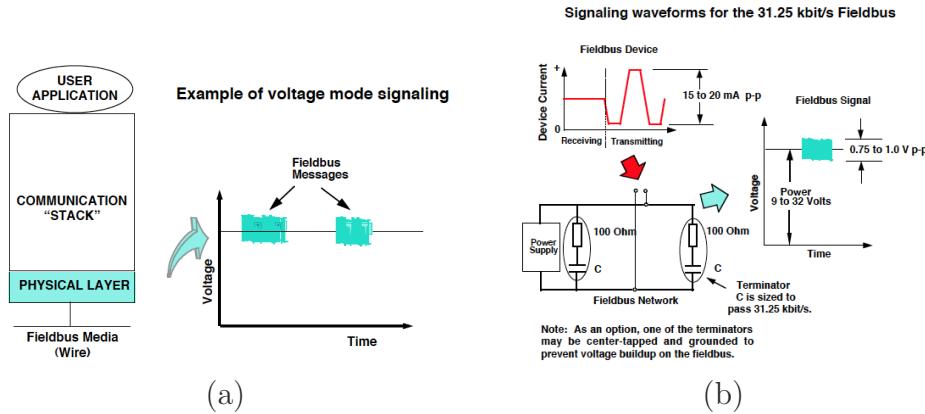


Figure 4.4: Example of voltage signalling (courtesy of FOUNDATION™ fieldbus).

each device, multiple devices can be connected to the *same set of wires*. FF overcomes some of the disadvantages of proprietary networks by offering a *standardised network* to connect systems and devices and defines two communication protocols: *H1* (*Type 1* on the *IEC 61158*) and *HSE* (*High Speed Ethernet, Type 5* on the *IEC 61158*).

H1 interconnects field equipments such as sensors, actuators and controllers and it is a Local Area Network (LAN) for instruments used in both process and manufacturing automation with built-in capability to distribute the control application across the network. In a H1 network, the devices are connected by a serial bus, called *link*, so as a network consists of one or more links. The devices can be *field devices* (e.g., sensors) or *host devices* (e.g., PCs, distributed control systems). Each physical device is configured with a physical and unique *device tag*, an address, and a unique *device ID*.

Physical layer

The *Physical layer* converts messages into physical signals according to IEC Standard 61158-2 and ISA Standard ISA S50.02. Among various devices with various speeds standardised by IEC, the Fieldbus Foundation chose a *low speed wire*, an *optical fibre* media, and *Ethernet*. The *Physical layer* of 31.25 kbps is the most common one since IEC and ISA approved it in 1992. 31.25 kbps transmission speed was chosen for those applications which demand for devices *with very low power consumption* and it is needed to satisfy various requirements by replacing traditional 4 to 20 mA analog transmissions.

An example of voltage signalling for a H1 fieldbus is reported in Figure 4.4. When a signal travels on a cable and encounters a discontinuity, such as

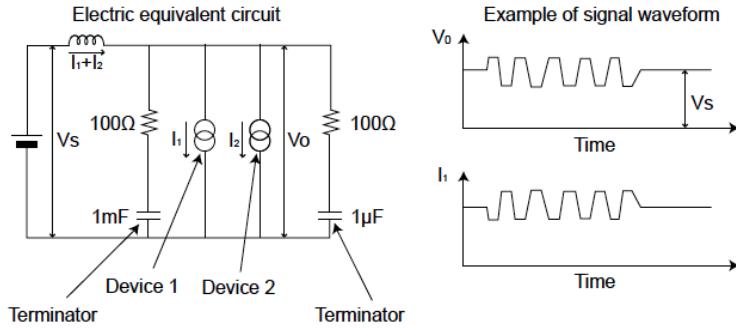


Figure 4.5: FF equivalent circuit of signal transmission (courtesy of Yokogawa Electric Corporation).

a wire short or open, it produces a *reflection*. This portion of the signal that *echoes from the discontinuity* travels in the opposite direction. The *terminator* is used to prevent a reflection at the ends of a Fieldbus cable. In most networks, the terminator is simply a *resistor* whose value is the same as the characteristic impedance of the wire. Since the Fieldbus cable also carries power, a simple resistor cannot be used *because it would use up the power intended for the devices*. A Fieldbus terminator has a *capacitor* in series with the resistor *to block the DC voltage but lets the signal, at 31.25 kbs, through to the resistor*.

Let us consider, a H1 bus with a supply voltage provided by a power supply through an *impedance conditioner* and ranging between 9 to 32 V DC at the devices (i.e. DC current through the impedance conditioner feeds devices), as reported in Figure 4.5. A 100Ω impedance terminator is installed at each cable terminal: it makes an instrumentation cable a *balanced transmission line* (relatively high frequency transmission). For a *bipolar device*, there is commonly a 10 mA that is *wasted*, i.e. absorbed without using it, hence when the device wants to transmit a high signal, it *turns off* this 10 mA. As per Figure 4.5, if the device 1 drops the I_1 current of 10 mA, the voltage between the cables is given by 10 mA times 50Ω (that is the resistor given by the parallel of two 100Ω resistors), thus yielding a voltage increase of 0.5 V. When the device 1 absorbs, i.e. wastes, $I_1 = 20$ mA, the voltage between the cables decreases by 1 V DC and a *low level* is transmitted. Thus the average voltage (V_s) is maintained at the same level and generates a modulated signal of 1 V amplitude. Notice that for unipolar devices, the current is not absorbed when there is no transmission: power saving but more complicated transmission. An example of fieldbus actual signalling on the cable can be found in Figure 4.6, while Figure 4.7 reports an example of signal encoding

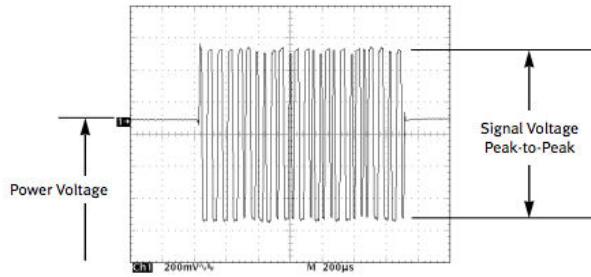


Figure 4.6: FF actual transmitted signal (courtesy of RELCOM, INC.).

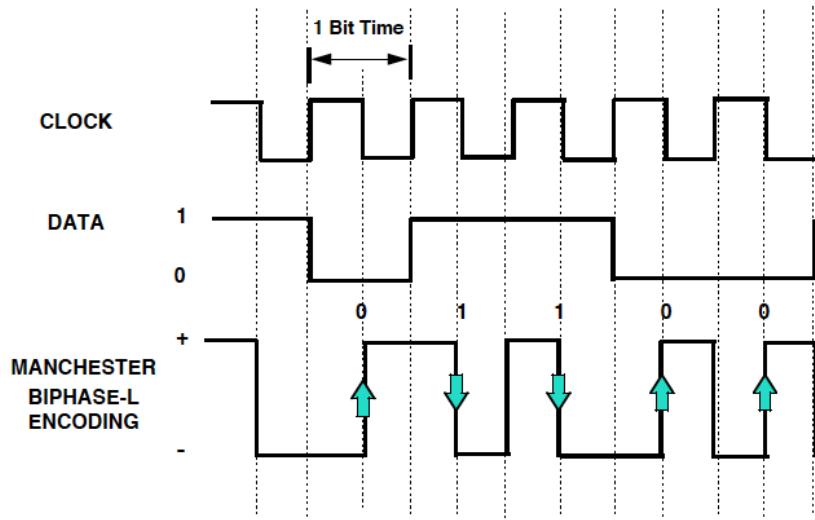


Figure 4.7: Example of signalling encoding (courtesy of FOUNDATION™ fieldbus). The bit time at 31.25 kbs is 32 ms.

with a bit time of 32 ms.

An example of a typical frame waveform can be found in Figure 4.8. Data is encoded as a voltage change in the middle of the one-bit time (32 μs at 31.25 kbps): 1 is encoded as a voltage drop in the middle of one bit time; the 0 is instead encoded as a voltage increase in the middle of one bit time. $N+$ and $N-$ are encoded as the constant voltage between the bit times (e.g., start/stop signals used for dividing frames to generate specifically identified signals separated from ordinary data). The receiving *Physical layer* retrieves the bit time using the preamble and the *octets* (bytes) of start delimiter signal, and the end delimiter indicates the end of the *Physical layer* signal. The length of the preamble can be increased when the signal goes through the repeaters.

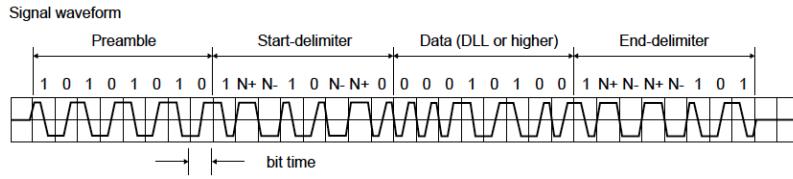


Figure 4.8: FF typical waveform of a *Physical layer* signal (courtesy of Yokogawa Electric Corporation).

In control systems, sensing instruments produce signals in the range of 4 – 20 mA. Similarly, control inputs in the same current amplitude range travel towards the actuators. The availability of low cost *Central Processing Unit* (CPU) paves the way to simplify the cabling. So instead of using a bulk of cables, the fieldbus allows multiple instruments to share the same medium, usually called a *trunk* or a *segment*. A fieldbus trunk is a single *twisted pair wire* carrying both a *digital signal* and *DC power* connecting up to 32 devices. Typical characteristic for a *twisted-pair cable* designed especially for Fieldbus should be:

- Voltages between 9 V and 32 V;
- Maximum cable length of 1900 m;
- Wire Size: 18 AWG (American Wire Gauge, around 1 mm of diameter for copper wires);
- Shield 90% coverage;
- Attenuation: 3 dB/km at 39 kHz;
- Characteristic Impedance: 100 Ohms $+/-20\%$ at 31.25 kHz.

In general, the number of devices that can fit on a fieldbus is limited by the current requirement of each device, the length of the segment, etc., as described in the following example.

Example 8. Consider a fieldbus having power supply and power conditioner output at 20 volts and a cable having a resistance of $22 \Omega/\text{km}$ for each conductor, hence 44Ω for the two wires. The cable is supposed to be of 1 km length and each device at the chicken-foot draws 20 mA.

Since the minimum voltage needed by a device is 9 V, there are $20 - 9 = 11$ V that can be used by the cable. The total current that can be supplied is

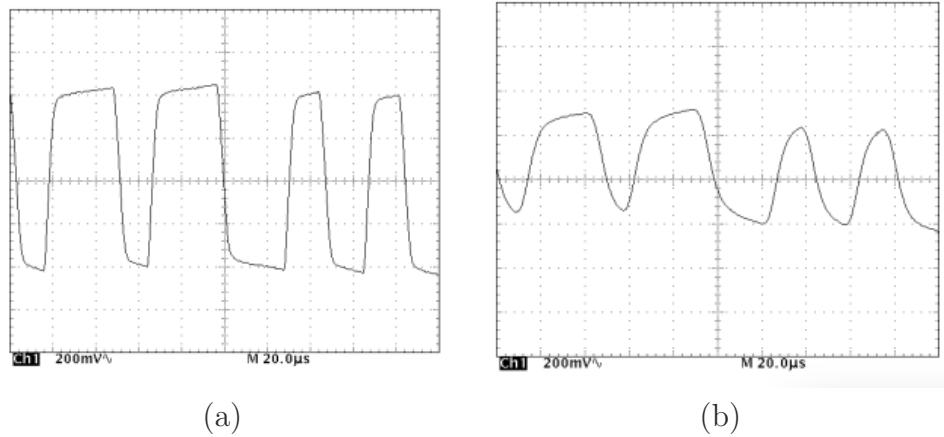


Figure 4.9: FF actual transmitted (a) and received (b) signals due to attenuation (courtesy of RELCOM, INC.).

$11/44 = 250 \text{ mA}$. Since each device draws 20 mA , the maximum number of devices is $250/20 = 12$ devices.

Notice that the power used by fieldbus devices varies by device type and one important issue to take into account is the **initial inrush current** and the **lift-off voltage**, hence the power consumption should be considered for the **worst case**. Moreover, when the network becomes more complicated, e.g. tree topology with several sprung, the power consumption computation becomes more involved.

The IEC standards define the *minimum amplitude* and the *worst waveform* that a received signal should have in order to be correctly received by the *Physical layer* receiver circuit of a fieldbus device. As signals travel on a cable, they become *attenuated* and *distorted*, that is, getting smaller. An example of an actual signal incurring attenuation can be found in Figure 4.9. Attenuation is measured in units called *dB* or *deci-Bell* and it is calculated as

$$dB = 20 \log_{10} \frac{\text{transmitted signal amplitude}}{\text{received signal amplitude}}.$$

Cables have *attenuation ratings* for a given frequency. Standard Fieldbus cables have 3 dB/km at 39 kHz. In other words, the percentage of the transmitted signal at the receiver side after one kilometre is

$$\frac{1}{10^{\frac{3}{20}}} 100 \approx 71\%.$$

For a shorter cable, say 500 m, we have

$$\frac{1}{10 \frac{1.5}{20}} 100 \approx 84\%.$$

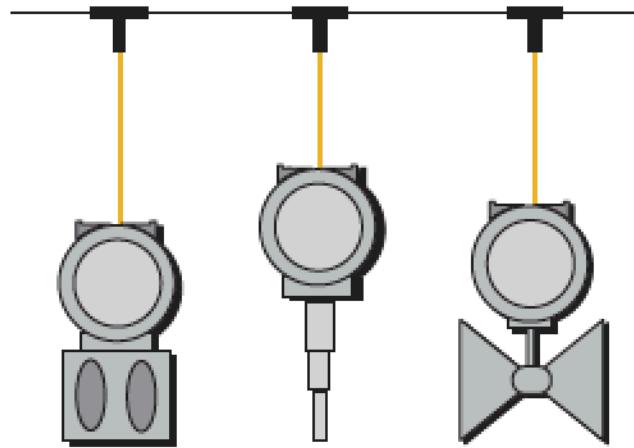


Figure 4.10: “T” configurations are the simplest fieldbus connection (courtesy of Moore Hawke, www.miinet.com/moorehawke).

Example 9. Assume that a **Fieldbus transmitter** can generate a signal as low as 0.6 V peak-to-peak in the worst case. Moreover, assume that a **Fieldbus receiver** is able to detect a signal as little as 0.2 V peak-to-peak. The cable can then attenuate the signal up to

$$20 \log_{10} \frac{0.6}{0.2} \approx 9.5dB.$$

Having an attenuation of 3 dB/km, the Fieldbus can be as long as $9.5/3 \approx 3.2$ km.

Each fieldbus device connects to the segment in *parallel*, called a *spur*, whose simplest form is a “*T*” connection, exemplified in Figure 4.10. With “T” connections, if any one of the devices or cables short out, it takes down the entire system. Therefore, *short-circuit protection* is an unavoidable requirement for proper fieldbus implementation.

A more complicate type of connection is the *device coupler*, connecting multiple devices in the same point of the fieldbus, depicted in Figure 4.11. In device couplers, short-circuit electronic spur protection are implemented via *current limiting* and *fold back*, thus having fault detection and correction capabilities.

A typical H1 fieldbus segment consists of the following components: H1 card (fieldbus interface card or multiple cards for redundancy); bulk power (Vdc) to Fieldbus Power Supply (FPS); FPS and signal conditioner; terminators (exactly 2 terminators are used per fieldbus segment: one at the FPS

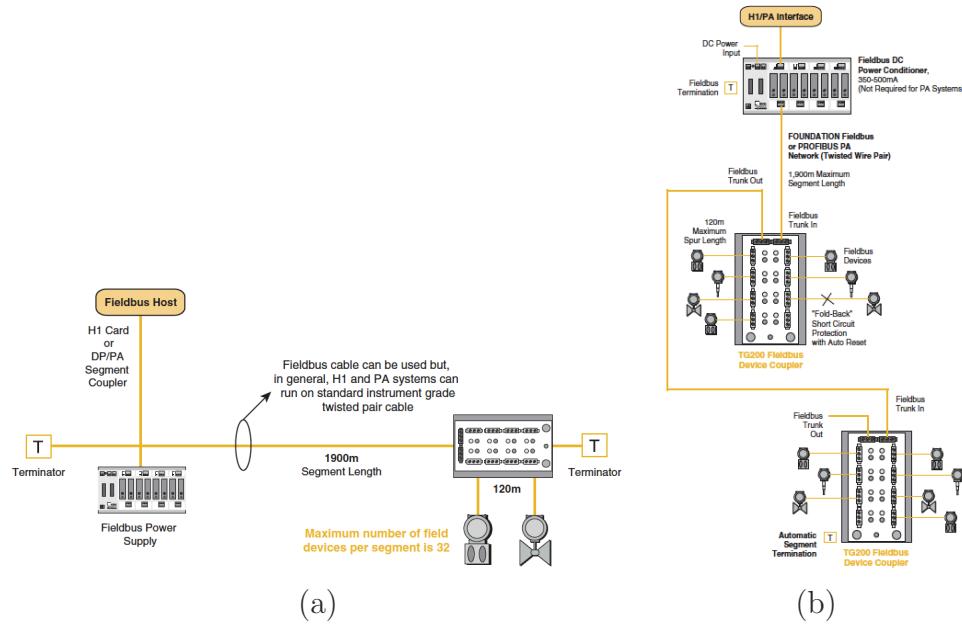


Figure 4.11: (a) A fieldbus segment starts with an H1 interface card and a power supply for FF or a segment coupler for PROFIBUS. (b) A device coupler permits multiple instruments to be connected to a fieldbus segment. Each “spur” has short-circuit protection. Courtesy of Moore Hawke (www.minet.com/moorehawke).

and one at the furthest point of a segment at the device coupler); linking devices, using a gateway to connect to a HSE backbone network; Fieldbus devices.

Needless to say, if a segment cable per-se fails, all the devices attached on it will fail. This problem is particularly severe for plant-critical segments. Even if the fieldbus standard *does not* consider this problem explicitly, many vendors have developed a redundant fieldbus scheme, as reported in Figure 4.12. To this end, a software-based *voting scheme* is needed to properly detect the faulty segment. This is a problem that is usually used in the literature of *consensus*, which will be covered in this course.

An important characteristics of fieldbuses is that they are suitable for hazardous areas, which require a higher degree of safety, i.e., *intrinsically safe* solutions. In this case, specific solutions should be defined to let areas with different levels of safety to coexist (see Figure 4.13). Intrinsically safe solutions are hard to be designed, however some solutions exist, as listed below:

- *Entity*: based on barriers limiting the amount of power entering in a

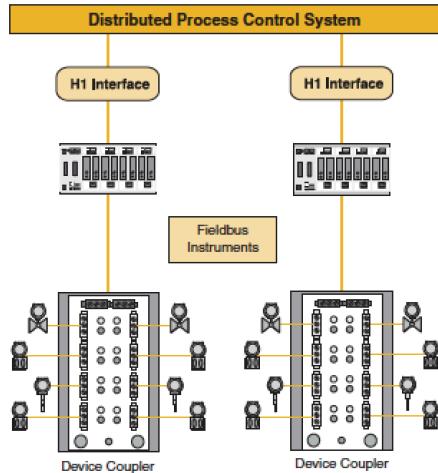


Figure 4.12: Example of fieldbus redundancy. Duplication of the segment may imply duplication of the field devices. In case of fault, one segment is substituted by the other one (courtesy of Moore Hawke, www.miinet.com/moorehawke).

hazardous area (see Figure 4.13-a);

- *Fieldbus Intrinsically Safe COncept* (FISCO): based on the physical tests of each of the devices attached to the fieldbus. The devices has to comply to very strict limits.
- Some vendor provides its own solution, usually based on a mixture of the previous two.

In general, the plant operator himself *must ensure* that intrinsic safety requirements are met when planning and installing the communications network. For instance, the capacitance and inductance of all line segments and devices must be calculated to ensure that the permissible limit values are observed (see for example Figure 4.14)

Data-Link layer

H1 links are interconnected physically only by Foundation Fieldbus H1 *Data Link bridges* (see Figure 4.15). Three different devices can be found in H1 networks. The *Link master* controls the communications traffic on a link by scheduling the communication on the network and at least one (or more) link masters for each H1 network should be placed. It can offer computational capabilities, e.g. for I/O, control, etc., and only one will take control of the

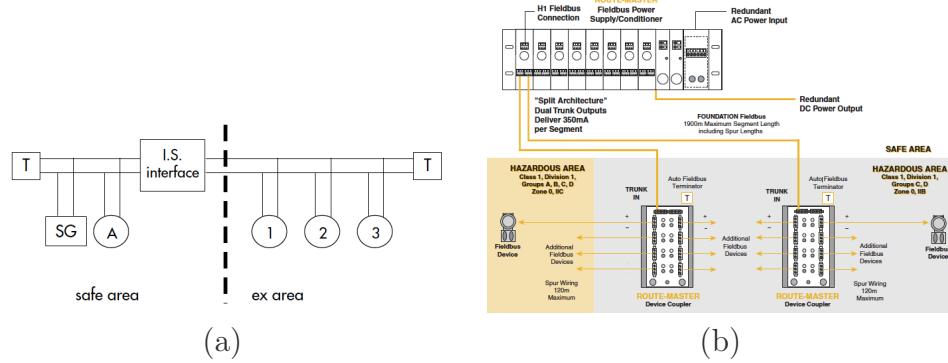


Figure 4.13: (a) Elements for the intrinsically safe H1 bus applying an Entity solution (courtesy of Samson AG). (b) Split architecture for two different safe areas (courtesy of Moore Hawke, www.miinet.com/moorehawke).

bus through a voting algorithm (see Figure 4.15-b), thus becoming the *Link Active Scheduler* (LAS). Having more than one link master ensures robustness against failures and allow the network to always have one LAS active. Apart the link master, the *basic devices* are standard devices that are not able to schedule communication, while the *H1 bridge* connects links together into a *spanning tree*. Such bridges are always link master devices and they must be the Link Active Scheduler.

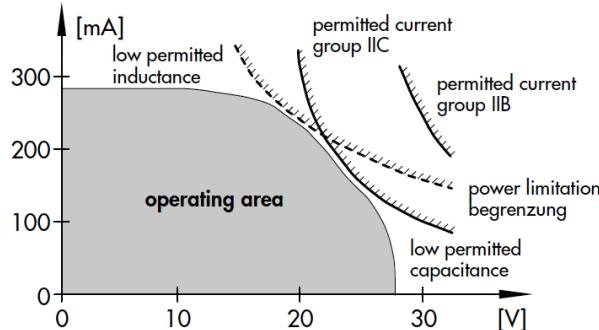
The communication services of the FF specification utilise *scheduled* and *unscheduled* data transmission. Time-critical tasks, such as the control of process variables, are exclusively performed by *scheduled services*, whereas parameterisation and diagnostic functions are carried out using *unscheduled communication* services. To solve communication tasks in time and without access conflicts, all time-critical tasks are based on a *strict transmission schedule* defined during the configuration of the FF system. Each field device receives a *separate schedule* in order to simplify the system management.

For *scheduled* communications, the communication is organised following the *publisher/subscriber* scheme. The *Link Active Scheduler* (LAS) uses the publisher/subscriber information to tell the publisher devices when to transmit their data over the bus. Data is transferred between a *given set* of devices (i.e., the subscribers) at the same time during each loop iteration. Hence, the transmissions are done in a *deterministic* manner, following the *schedule* developed with a configuration software and downloaded to the link masters. An example of a possible scheduling is reported in Figure 4.16. The LAS *cyclically* transmits the data for the *scheduled* transmission according to the scheduling list using a specified message (see Figure 4.17):

- If a device is prompted to publish its data (hence, to become a pub-

Group	$C_o (C_a)$	$L_o (L_a)$
IIC	165 nF	0.35 mH
IIB	1.14 μ F	1.04 mH

(a)



(b)

Figure 4.14: (a) Capacitance and inductance limit values for installation of intrinsically safe instrumentation. (b) Limited operating area for intrinsically safe IIB and IIC installations (including a safety factor of 1.5). Courtesy of Samson AG.

lisher), the LAS issues the *Compel Data* (CD) command to the device;

- Upon receipt of the CD, the device publishes the data in the buffer;
- The subscribers of this message can read and evaluate this data accordingly.

The LAS periodically broadcasts a *synchronisation signal*, i.e. *Time Distribution* (TD) approach, on the fieldbus to synchronise all the devices. This way, *determinism* of transmission is granted since every device knows the point in time in which transmission takes place. Based on this schedule, a *transmission list*, which resides in the LAS, is generated and defines when a specific field device is prompted to send its data.

Not all the data are *time critical* (e.g., device parameters and diagnostic data must be transmitted on request), therefore *unscheduled communication* are also considered in FF. Unscheduled communication takes place only if *no scheduled communication is foreseen*, i.e., in free time slots. Permission for a certain device to use the bus is granted by the LAS when it issues a *Pass Token* (PT) command to the device. Each device may use the bus as long

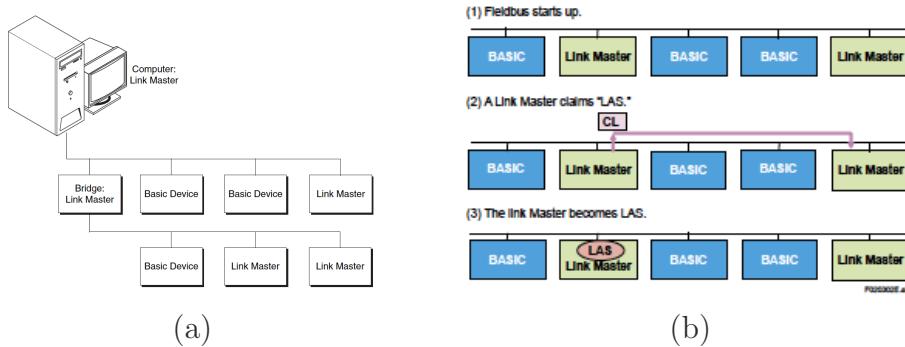


Figure 4.15: (a) H1 network example (courtesy of National Instruments). (b) The process of Link Master class device becoming LAS (courtesy of Yokogawa Electric Corporation).

as required until it: a) returns the token or b) the maximum granted time to use the token has elapsed. Hence: *Compel Data* command for scheduled operations, *Pass Token* command for unscheduled operations.

To correctly manage the unscheduled messages, the LAS has to know which are the devices on the fieldbus. To this end, a *live list* is continuously updated by the LAS using the transmission of special commands. Devices which *do not respond* to the *Pass Token* command or return the token after *three* successive tries are removed from the live list. The *Probe Node* command is sent to the addresses *not* in the live list, searching for newly added devices. If a device returns a *Probe Response* message, the LAS adds the device to the live list where it receives the pass token for unscheduled communication. To update the other *Link Masters*, all these information are broadcasted in the network.

To summarise, the LAS follows a *strict schedule* to ensure that unscheduled communication (token passing, CD or PN commands) do not interfere with the scheduled data transmission. To do so, the LAS refers to the transmission list to check for any scheduled data transmissions. In case of scheduled transmission, sends the Compel Data (CD) message to activate the operation. In case there are no scheduled transmissions and *sufficient time is available* for additional operations, the LAS sends one of the other commands. Following this, the sequence starts all over again with the above mentioned check of the transmission list entries.

Application layer

The *Application layer* is divided into the *Fieldbus Message Specification* (FMS) and the *Fieldbus Access Sublayer* (FAS), as reported in the full OSI

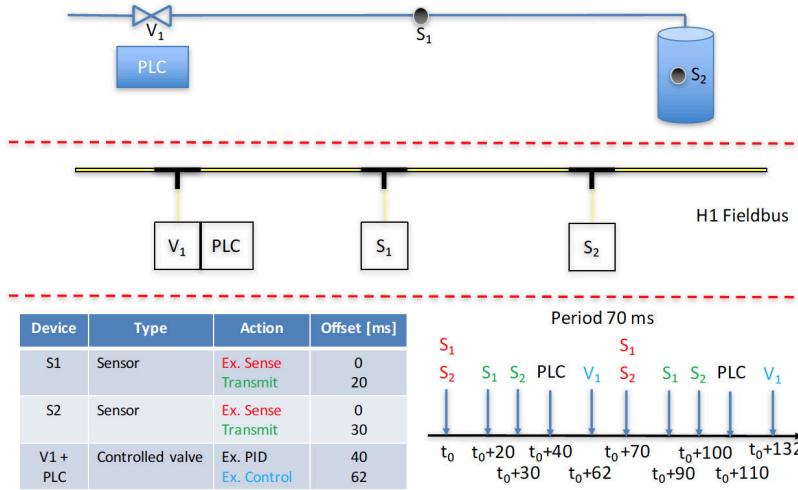


Figure 4.16: Scheduling example for a distributed plant.

model in Figure 4.18. The *Fieldbus Message Specification* (FMS) may or may not be implemented (for example, it is not implemented in *PROFIBUS-DP*). If it is implemented, it is based on two components: the *object dictionary* is a structure in a Fieldbus device that describes data that can be communicated; the *virtual field device* (VFD) is a model for remotely viewing data described in the object dictionary. From the application viewpoint, VFDs can be regarded as *different field devices*. Communication services guarantee VFDs independence.

The *Fieldbus Access Sublayer* (FAS) uses the *scheduled* and *unscheduled* communications to provide services to the FMS using the *Virtual Communication Relationships* (VCR). For a specific device, the VCR is basically a table with *logic connections* to the other fieldbus devices connected to it. The method implemented by the FAS using the VCR are the publisher/subscriber (for scheduled control messages), the client/server (for the human operator messages) and the report distribution (to acknowledge the other connected devices of warnings or problems).

In the OSI model structure (see Figure 4.18), each layer appends layer-control information called *Protocol Control Information* (PCI). Moreover, a data unit exchanged in between the same layers is called *Protocol Data Unit* (PDU), which may contain an optional data called *Service Data Unit* (SDU) (passed to and from a higher layer).

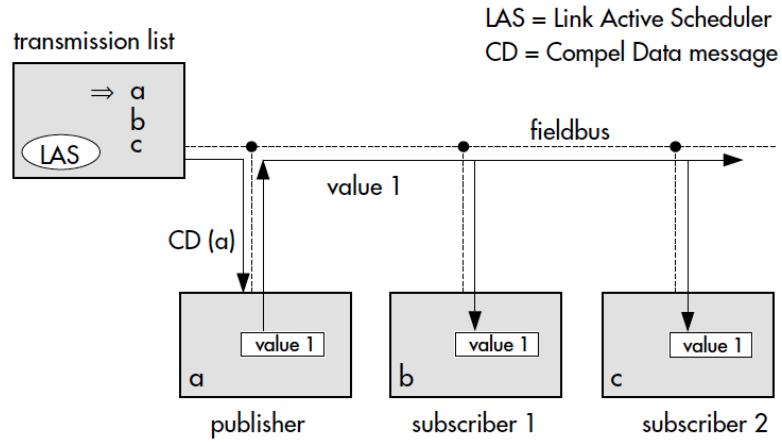


Figure 4.17: Scheduled data transmission according to the transmission list (courtesy of Samson AG).

User layer

The *User layer* provides the interface for user interaction with the distributed system. It uses the *device description*, i.e. a *configuration object*, to tell the host its inputs/outputs and the device capabilities, i.e. its *functionalities*. An example of configuration of devices is reported in Figure 4.19. Each device becomes a *function block* according to the IEC 61499. To do so, the device vendor supplies *device description files*, which describe the parameters of the function and transducer blocks contained in a device. The *linkages* connects such functionalities. The *control loops* are sets of function blocks connected by linkages executed with a desired rate. It is possible to have *multiple loops* with different rates hosted on the same link.

According to the *FOUNDATION Fieldbus specifications* for function blocks, the *FF function block* (FF-FB) parameters are accessed and monitored via communication services and the block behaviour depends on the parameter values. According to the standard IEC 61499, the FF-FB can be resided in *any device on the network* and a set of FF-FB connected to each application can be resided in *one device or distributed among devices*. The FF-FB is a *generalised model for control and measurement* (Figure 4.19-d). The *Fieldbus Foundation's system architecture document* describes it as follows:

The FB model has been specified within the architecture to support low level functions found in manufacturing and process control. Function Blocks model elementary field device functions, such as analog input (AI) functions and proportional integral derivative (PID) functions (see Figure 4.20 for a

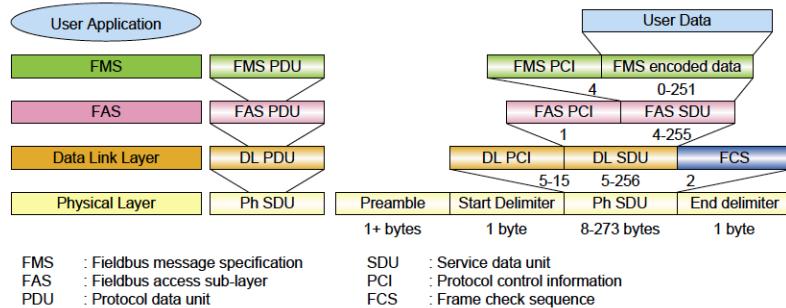


Figure 4.18: How a user data is transferred through the OSI Model of the Foundation Fieldbus (courtesy of Yokogawa Electric Corporation).

function block example of a PID controller).

The FB model has been supplemented by the transducer block model to decouple FBs from sensor and actuator specifics. Additional models are defined for remote input/output and programmable devices.

The FB model provides a common structure for defining FB inputs, outputs, algorithms and control parameters and combining them into an Application Process.

The FF-FB is categorised in three classes:

- *Standard block* as specified by the Fieldbus Foundation;
- *Enhanced block* with additional parameters and algorithm;
- *Open block* or a custom block designed by each manufacturer.

The *function blocks* implemented in a device provide information about the tasks the device can perform. For instance: sensors perform analog or discrete (i.e. digital) input; control valves refer to analog or discrete (i.e. digital) output; control processes implement proportional-derivative (PD) controller or proportional-integral-derivative (PID) controller. A complete list of function block for fieldbus is given in Figure 4.21-a, while a graphical representation with examples of linkages implementing different functionalities is offered in Figure 4.21-b. Examples of function blocks are:

- *AI (Analog Input):* The AI block reads data from a *single analog input channel*. This block performs simple filtering and scaling of the raw data to engineering units from the input channel and supports limit alarming;

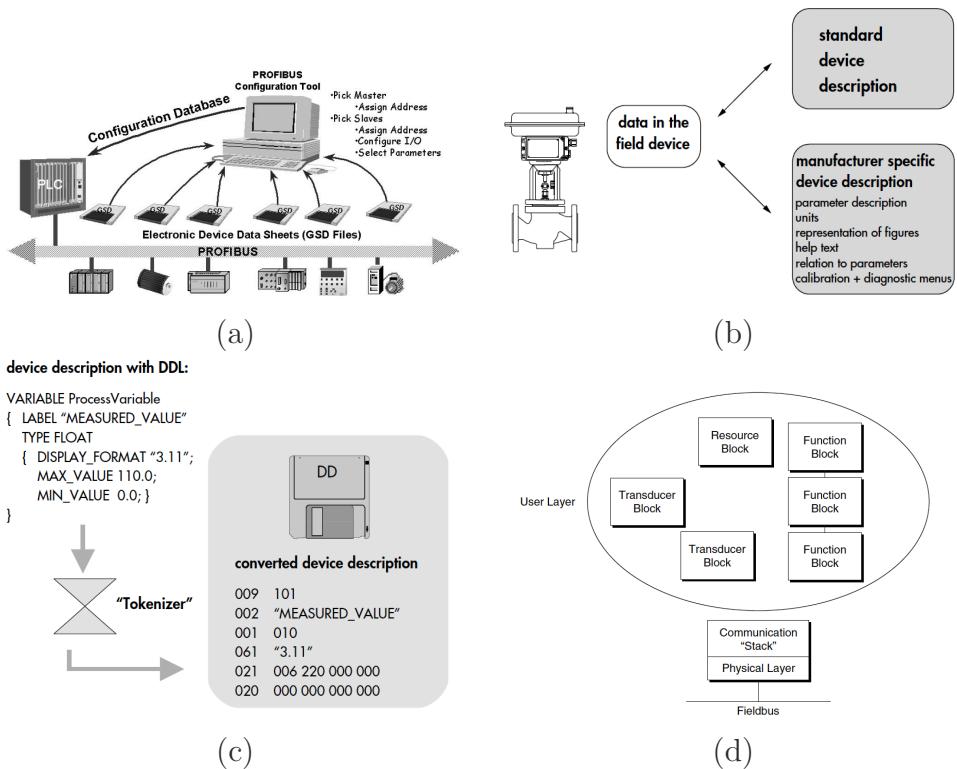


Figure 4.19: (a) Example of configuration of devices in PROFIBUS (courtesy of Samson AG). (b) Device description extends the description of all objects in the virtual field device (courtesy of Samson AG). (c) Creating a device description (courtesy of Samson AG). (d) Example of a user layer implementing *function blocks* (courtesy of National Instruments).

- **AO (Analog Output):** The AO block writes data to an *analog output channel*. This block supports cascade initialisation to allow upstream control blocks to switch smoothly from manual to automatic mode. It also has a *fault state behaviour* that allows the block to react if communications fail between itself and the upstream block;
- **PID (Proportional–Integral–Derivative):** The PID block implements a PID control algorithm. In Fieldbus, a *PID block must be connected* to an upstream block (such as an AI block) and a downstream block (such as an AO block) before it can be used for control. These *software connections* are established by using host *Fieldbus configuration software*;
- **DI (Discrete Input):** The DI block reads data from *discrete input channel*

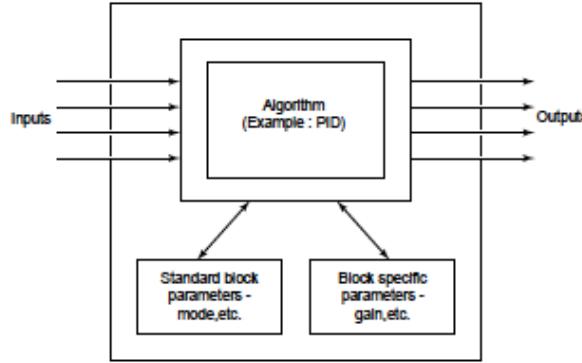


Figure 4.20: An example of the PID FF-FB (courtesy of Yokogawa Electric Corporation).

nels. This block performs simple filtering and processing of the raw data from the input channel and supports limit alarming;

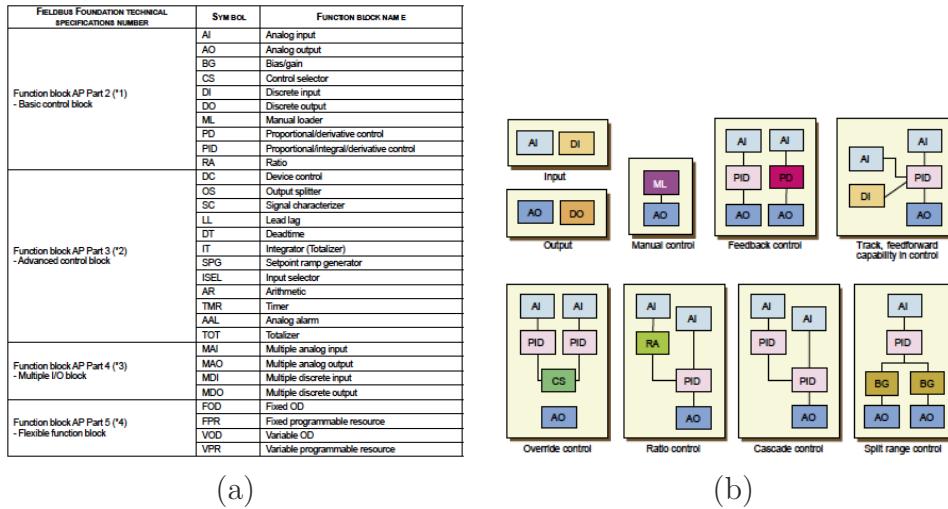
- *DO (Discrete Output)*: The DO block writes to a *discrete output channel*. This block supports cascade initialisation to allow upstream control blocks to determine the current state of the process before assuming control. It also has a fault state behaviour that allows the block to react if communications fail between itself and the upstream block.

The schedule can be divided into two parts: a *function block schedule* that determines when a block executes, and a *publishing schedule* that determines when data parameters are published over the Fieldbus, i.e. transmitted. The *function block schedule* is *downloaded to the particular device* that contains each function block, while the *publishing schedule* is downloaded to the devices having *link master capability* (hence also to the LAS).

4.1.2 Why fieldbuses?

Strictly speaking, the FF is designed to shift the automation tasks down to the *field* in order to have a flexible, distributed processing of control tasks using the ideas of *function blocks*. The main effect is to *reduce the load* on the central process control station, which can even be replaced entirely in small-scale installations. Therefore, an *entire control loop* can be implemented as the smallest unit, consisting only of one sensor and one control valve with integrated process controller which communicate over the FF, as depicted in Figure 4.22.

The fieldbus was firstly proposed to limit the amount of wiring and terminations in a plant (in the order of 90% less for particular plants). Next,



(a)

(b)

Figure 4.21: (a) FF-FB complete list. (b) Examples of function block linkage using basic control function blocks introduced in the FF specifications. Courtesy of Yokogawa Electric Corporation.

the adoption of the fieldbus was limited by the *cost* of the devices (and of their protection circuits) and by the need for cabling redundancy. The major benefits of the fieldbus are:

- **Planning:** The fieldbus integrates plant equipment into a *single plant automation system* through its digital communication network. It enables users to enhance information productivity and realise a small control room and a smaller cabinet by connecting devices from various suppliers without customised software;
- **Installation:** The fieldbus reduces installation and material cost by replacing the traditional one-to-one wiring scheme with *networking or multi-drop* connection, while intelligent field instruments shorten commissioning and saves plant startup cost;
- **Operational:** An integrated *Human-Machine Interface* (HMI) is provided for the plant operation. The fieldbus function blocks enable control functions to be installed on each field device, which enables to transfer the control function to the field;
- **Maintenance:** The fieldbus enables various notifications related to the *self-diagnostics, calibration*, and *environmental conditions* of field instruments without interrupting the plant control. By adopting intelligent field devices equipped with self-diagnostic function, the fieldbus

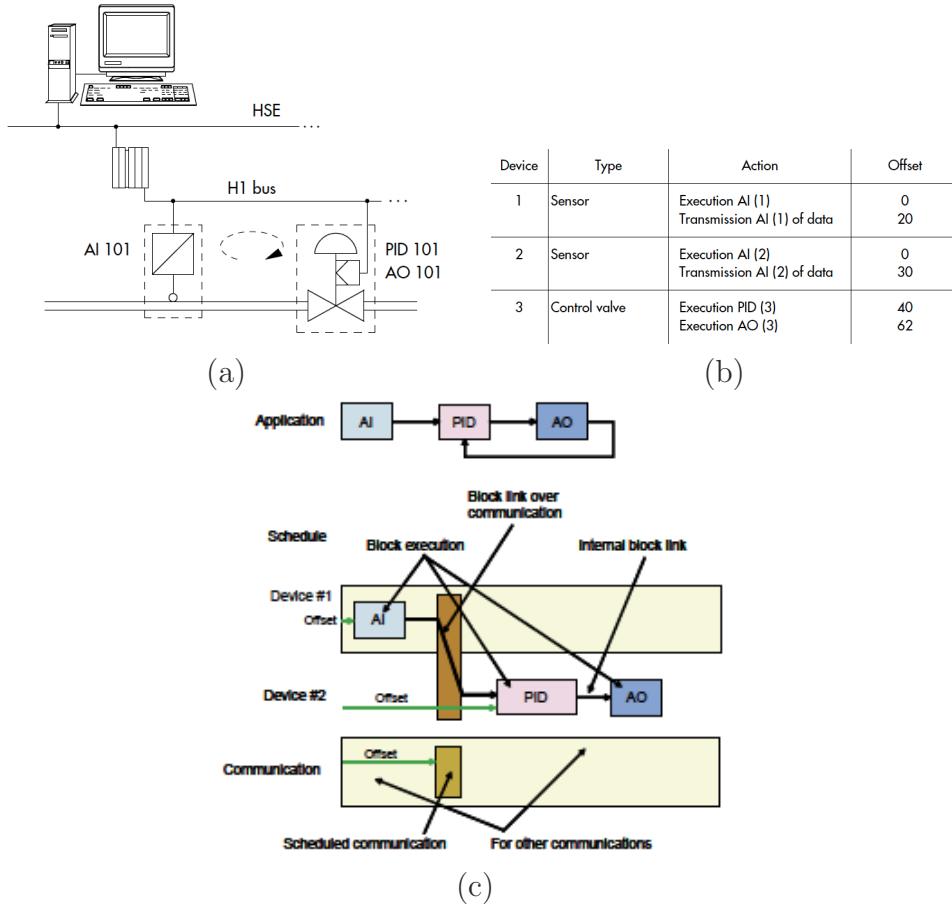


Figure 4.22: (a) Complete control loop based on the FF and (b) schedule for processing function blocks (courtesy of Samson AG). (c) Another example of schedule of FF-FBs and communications (courtesy of Yokogawa Electric Corporation).

saves a large part of field work by remote access to the device condition information and enable condition-based or proactive maintenance;

- **System modification:** Functionality of field instruments keeps enhancing endlessly. The fieldbus devices have become the de facto standard and off-the-shelf instruments, which help extend the plant's life cycle cost effectively and without difficulty. The new field devices bring users the benefits of the latest technology. Upgrading of the device firmware can be done *online* via fieldbus, which contributes in reducing the upgrading cost.

Network	Type	Users	Max. Speed	Max. Devices
Ethernet TCP/IP	RA/CD	78%	1 Gb/s	1024
Modbus	TDM/MS	48%	35 Mb/s	32
DeviceNet	RA/AMP	47%	500 kb/s	64
ControlNet	TDM/TP	39%	5 Mb/s	99
WiFi (IEEE 802.11b)	RA/CA	35%	11 Mb/s	??
Modbus TCP	TDM/MS	34%	1 Gb/s	256
PROFIBUS-DP	TDM/MS and TP	27%	12 Mb/s	127
AS-I	TDM/MS	17%	167 kb/s	31

Figure 4.23: Most popular fieldbuses. Maximum speed depends on the *Physical Layer*. Note that totals are more than 100% because most companies use more than one type of bus [12].

4.2 Fieldbuses examples

There are several different possible choices for fieldbuses and networks in an industrial environment. A list of the most popular is given in Figure 4.23. The networks are classified according to type: random access (RA) with collision detection (CD), collision avoidance (CA), or arbitration on message priority (AMP); or time-division multiplexed (TDM) using token-passing (TP) or master-slave (MS). Time-Division Multiplexing can be accomplished in one of two ways: *master-slave* or *token passing*. In a *master-slave* network, a single master polls multiple slaves. Slaves can only send data over the network *when requested by the master*; there are no collisions, since the data transmissions are carefully scheduled by the master. A *token-passing* network has *multiple masters*, or peers. The node that currently has the token is allowed to send data. When it is finished sending data, or the maximum token holding time has expired, it passes the token to the next *logical* node on the network. If a node has no message to send, it just passes the token to the successor node.

Collision of data frames does not occur, as only one node can transmit at a time. Most token-passing protocols guarantee a *maximum time* between network accesses for each node, and most also have provisions to regenerate the token if the token holder stops transmitting and does not pass the token to its successor. AS-I, Modbus, and Interbus-S are typical examples of master-slave networks, while PROFIBUS and ControlNet are typical examples of token-passing networks. The token bus protocol (e.g., IEEE 802.4) allows a linear, multi-drop, tree shaped, or segmented topology.

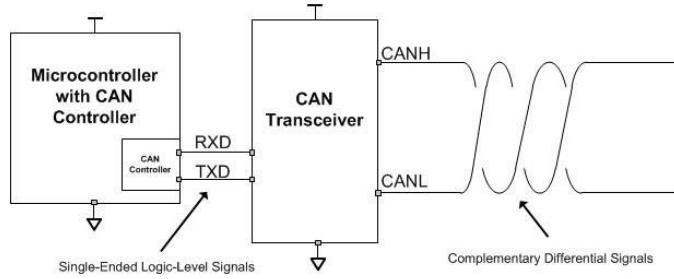


Figure 4.24: CAN Node (courtesy of Texas Instruments).

4.2.1 CAN bus

The *Controller Area Network* (CAN) is a *serial communication protocol* developed mainly for applications in the automotive industry but also capable of offering good performance in other time-critical industrial applications. The CAN protocol is optimised for *short messages* and uses a CSMA/CA (Random access with collision arbitration) on message priority as medium access method.

The history reads as follows:

1980-1983: Creation of CAN as an initiative by BOSCH to meet a requirement in the automotive industry.

1983-1987: The prices of drivers and micro-controllers featuring CAN become very attractive as they are used in high volume in the automotive industry.

1991: *CIA = CAN in Automation* promotes industrial applications.

1993: *CAL = CAN Application Layer* specifications published by CIA describing transmission mechanisms but not when and how to use them.

2001: CIA publishes DS-304 which can be used to integrate level 4 safety components on a standard CANopen bus (*CANsafe*).

The CAN bus comes in two different set-up: the *high speed CAN* (ISO 11898-2) uses a linear bus, while the *low speed or fault tolerant CAN* (ISO 11898-3) uses a linear bus, star bus or multiple star buses connected by a linear bus.

As depicted in Figure 4.24, each node on the CAN bus has:

- A *central processing unit, a microprocessor or host processor*: It is responsible to *understand* the received message content and *what message transmit*. Sensors, actuators and control devices can be connected to the host processor;
- *CAN controller*: It *stores* the received serial bits from the bus until an entire message is available and then it *fetches* it to the host processor,

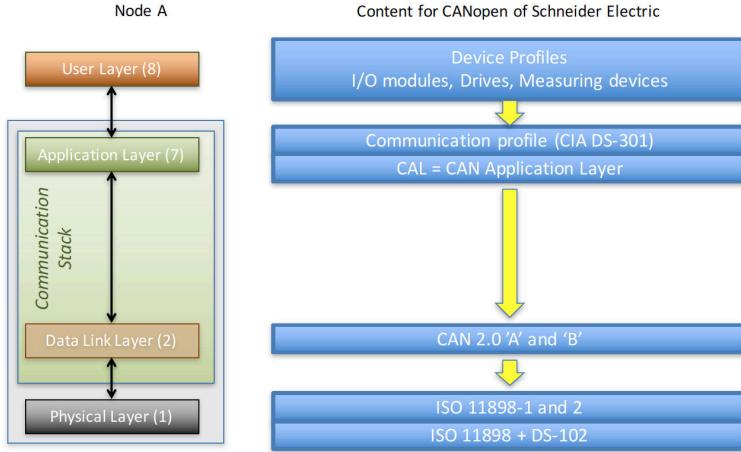


Figure 4.25: CAN and the OSI model.

usually using an interrupt. Moreover, it *sends* the message(s) bits, received by the host processor, serially onto the bus when the bus is free;

- *Transceiver* (ISO 11898-2/3): It *converts* the received data stream from the CAN bus levels to levels that the CAN controller uses. Moreover, it *converts* the data stream to send from the CAN controller to CAN bus levels.

As the majority of the fieldbuses, the CAN bus implements just three layers, plus the *User Layer*, as depicted in Figure 4.25.

Physical Layer

The Physical Layer defines the type of medium adopted, that is a shielded twisted pair 2 or 4-wire (if power supply). The adopted topology is the bus with short tap links and 120 ohm line termination resistor, usually of a maximum distance of 1000 m. The communication rate changes among 9 possible options, from 1 Mbps to 10 Kbps, which depends on bus length and cable type (e.g. 25 m at 1 Mbps, 1000 m at 10 Kbps). The maximum number of devices is 128 with 1 master and 127 slaves. The maximum size of useful data is 8 bytes per frame, while for the transmission security is enforced with numerous signalling and error detection devices so as the CAN is considered one of the best local industrial networks. At the Physical Layer also the type of signals is defined: two levels are consider, one *dominant* and one *recessive*, depicted in figure 4.26.

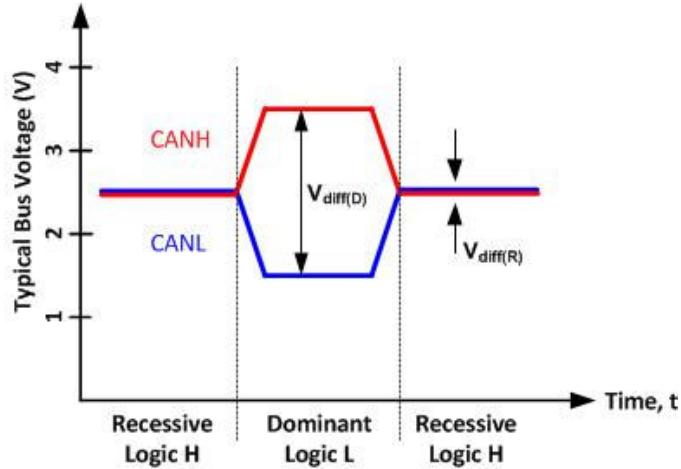


Figure 4.26: Signals at the *Physical layer* (courtesy of Texas Instruments).

Application Layer

At the Application Layer, there are four types of standardised services: *Network administration*, used for Parameter settings, start-up, monitoring, using a master-slave approach; *Process Data Object* (PDO), used for the transmission of low-volume process data (≤ 8 bytes) in real time, based on the producer-consumer; *Service Data Object* (SDO) used for the transmission of high-volume parameter data (> 8 bytes) by segmentation without time restrictions, and relying on the client-server paradigm; *Special Function Object* (SFO) for predefined synchronisation messages (SYNC, time-based references) and fatal errors management.

CAN profiles are based on the *object dictionary* concept. The CAN object dictionary is an ordered group of objects which can be accessed via an index of 16 bits and, if required, a sub-index of 8 bits. Each network node has an object dictionary in an ASCII format Electronic Data Sheet (EDS). This dictionary contains all the elements *describing the node* along with its *network characteristics*.

Data-Link Layer

At the Data-Link Layer the CAN relies on a random access with collision arbitration. The protocol is *message oriented*, and each message has a *specific priority* that is used to arbitrate access to the bus in case of simultaneous transmission. The bit stream of a transmission is *synchronised* on the start bit, and the *arbitration* is performed on the *following message identifier*, in which a logic zero is dominant over a logic one. A node that wants to transmit

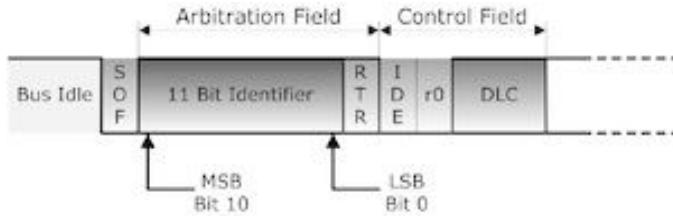


Figure 4.27: CAN frame with the Arbitration field [13]. *SOF*: Start of Frame; *RTR*: Remote Transmission Request.

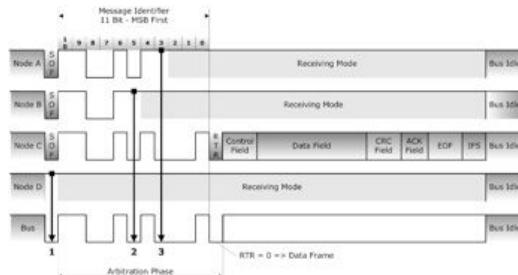


Figure 4.28: Arbitration phase [13].

a message *waits* until the bus is free and then starts to send the identifier of its message bit by bit. Conflicts for access to the bus are *solved during transmission* by an arbitration process at the bit level of the arbitration field, which is the initial part of each frame (see Figure 4.27). If two devices want to send messages at the same time, they *first continue to send* the message frames and *then listen to the network*. If one of them receives a bit different from the one it sends out, it *loses the right* to continue to send its message, and the other wins the arbitration. For example, consider the situation depicted in Figure 4.28, whose nodes have the following Arbitration Filed: *Node A* 1100101100 (32C hex); *Node B* 1100110000 (330 hex); *Node C* 1100101000 (328 hex); *Node D* no significance (not requesting bus access). Since the recessive bit is the level 1, after the arbitration phase node *C* can transmit safely. With this method, an ongoing transmission is *never corrupted*, and collisions are nondestructive. In essence, the priority of a message is indicated by the value of the identifier: the identifier with the *lowest value* has priority.

4.2.2 DeviceNet

DeviceNet is an example of a technology based on the CAN specification that has received considerable acceptance in device-level manufacturing ap-

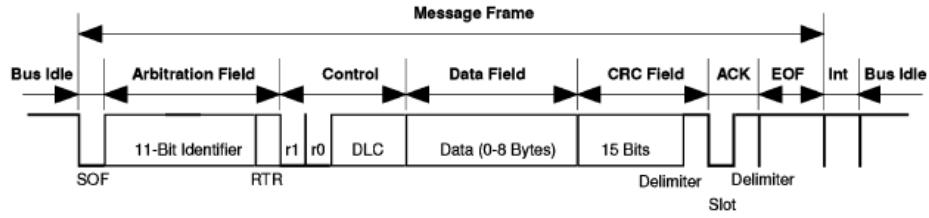


Figure 4.29: Message frame format of DeviceNet based on standard CAN format [12].

plications. *DeviceNet* utilises the *Common Industrial Protocol* (CIP) over a CAN media layer and defines an application layer to cover a range of device profiles. Typical applications are information exchange, safety devices, and large I/O control networks.

Common Industrial Protocol is an industrial protocol collecting an extensive set of messages and services for manufacturing automation applications. It allows users to *integrate* the manufacturing applications at a higher hierarchical level of the DCS. It is *media-independent* and *provides a unified communication architecture* throughout the manufacturing system. It is supported by *Open DeviceNet Vendors Association* (ODVA) and it is a *standard development organisation*.

An example of a DeviceNet frame is reported in Figure 4.29. The total overhead is 47 bits, which includes *Start Of Frame* (SOF), *Arbitration* (11-bit identifier), *Control*, *CRC*, *ACKnowledgment* (ACK), *End Of Frame* (EOF), and *INTermission* (INT) fields. The size of a data field is between 0 and 8 bytes. The DeviceNet protocol uses the arbitration field to provide *source and destination addressing* as well as message prioritisation.

The major disadvantage of a network based on CAN compared with the other solutions is the *slow data rate*, limited by the network length. DeviceNet has a maximum data rate of 500 kb/s for a network of 100 m. Thus, the *throughput* is limited compared with other control networks. CAN is also *not* suitable for transmission of messages of *large data sizes*, although it does support fragmentation of data that is more than 8 bytes into multiple messages.

DeviceNet uses *object type modelling* for:

- The list of available communication services;
- Device characteristics;
- A standard means of describing how to access the internal variables of a product.

Therefore, a DeviceNet node is modelled as a collection of objects. The object is defined by a *profile* written in an EDS file.

4.2.3 PROFIBUS

In 1987, the German federal minister for technological research and development creates a *Fieldbus* working group comprising 13 organisations including SIEMENS and 5 research institutes, which released *Profibus*, i.e., *PRO*cess *FIELD***BUS**. PROFIBUS is managed by a user group which includes manufacturers, users and researchers: the *PROFIBUS CLUB*. User clubs in 20 of the world's most industrialised countries provide support in native languages. These centres of competence are governed by the *PROFIBUS International* (PI) organisation, which has more than 750 members (<http://www.profibus.com/>).

There are three different PROFIBUS family:

1. *PROFIBUS-FMS*: Fieldbus Message Specification;
2. *PROFIBUS-DP*: Dezentrale Pheripherie;
3. *PROFIBUS-PA*: Process Automation.

An interesting feature of PROFIBUS is that the protocol is the same no matter what transmission medium or technology is used. In particular, the PROFIBUS-FMS is meant for data exchange between *high level components*, such as PLCs or PCs, so it is considered as a *master-master acyclic* communication standard. The other two solutions, together with *Profinet*, are the Siemens products for automation, as reported in Figure 4.30.

PROFIBUS-FMS

PROFIBUS-FMS it is the result of a joint collaboration with the German government and *BOSCH*, *Siemens* and *Moller*. It is based around DIN 19245-Part 1 and Part 2 since 1990 and uses typically RS-485. It implements all the layers of the OSI model as specified by the FF, i.e., *Application layer*, *Data-Link layer* and *Physical layer*. The *Application layer* comprises both the *Fieldbus Message Specification* (FMS, which exports explicitly the communication services and the details of the adopted protocol) and the *Fieldbus Access Sublayer* (FAS, the interface towards the FMS). The OSI model of PROFIBUS-FMS is reported in Figure 4.31

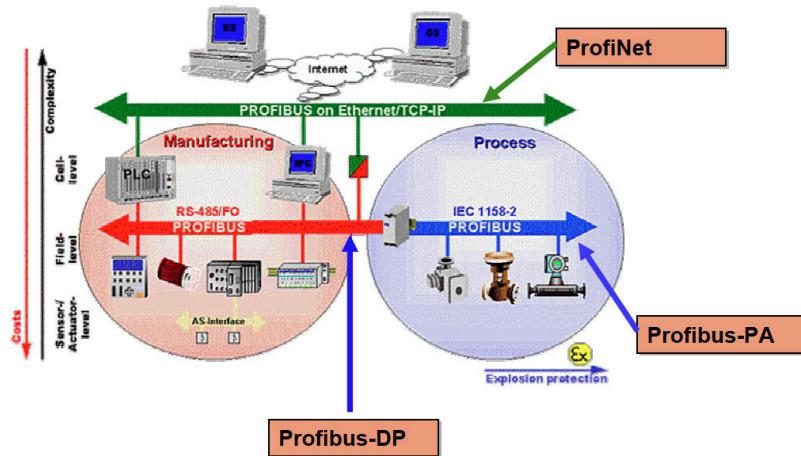


Figure 4.30: The three versions of PROFIBUS (courtesy of Schneider Electric).

PROFIBUS-DP

PROFIBUS-DP (utilising typically RS-485) is aimed at *time-critical communications* between automation and distributed peripherals and is based around DIN 19245-Part 3 since 1993. It is suitable as a replacement for the costly wiring of 24V and 4-20 mA measurement signals. PROFIBUS-DP is included in the European Fieldbus standard EN50170. PROFIBUS-DP and PROFIBUS-FMS adopt *the same transmission techniques and MAC*, hence they can work *simultaneously on the same bus*. It does not implement the *Application layer*, as reported in Figure 4.32. Therefore, at the *User layer* it describes the devices with profiles.

In Figure 4.32, MBP at the *Physical layer* stands for *Manchester Bus Powered* transmission technology, the same as FF H1. *Data and field bus power are fed through the same cable*. The power can be reduced in such a way that use in explosion-hazardous environments is possible. The bus topology can be up to 1900 m long and permits branching to field devices (max. 60 m branches). The bit rate here is 31.25 kbs. This technology was specially established for use in process automation for **PROFIBUS PA**.

PROFIBUS DP is a widespread fieldbus based on a *master-slave cyclic* approach, implementing *real-time* data exchange between controllers (masters) and field devices such as sensors and actuators (slaves) at the maximum speed of 12 Mb/s (for 100 m cables), slowing to 9.60 kbs for 1200 m cables. The protocol specifies also a master-master communication, typically meant for the *non real-time* exchange of diagnostic and/or parameterisation data

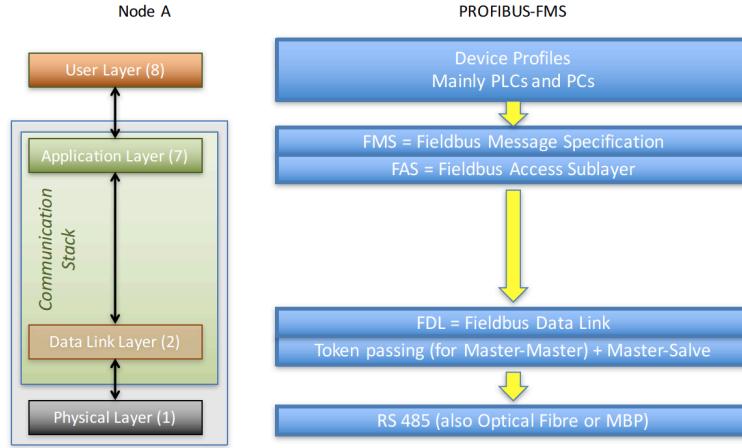


Figure 4.31: *PROFIBUS-FMS* and the OSI model.

and based on *token-passing*.

At the *Data-Link layer*, the medium access is granted exclusively to master stations via a token passing scheme. The most practical configurations make use of a single master (*mono-master networks*). More precisely, at the *Data-Link layer* a broad spectrum of services, which enables optimum communication for different applications: *DP-V0* is the cyclic *master-slave* polling, which is time deterministic; *DP-V1* adds acyclic communication for parameter assignment, configuration, and alarm handling. Used also for *master-master* communication; *DP-V2* isochronous slave mode and direct slave-to-slave communication.

With multiple masters, the arbitration between them is carried out with *token-passing*. For *token-passing* networks, the node with data to send must first wait to receive the token. The time it needs to wait can be computed by adding up the transmission times for all of the messages on nodes ahead of it in the logical ring. For example, in ControlNet, each node holds the token for a minimum of 22.4 μs and a maximum of 827.2 μs .

In *master-slave* networks, the master typically polls *all* slaves every cycle time. Slaves cannot transmit data until they are polled. After they are polled, there is no contention for the network so the waiting time is zero. The master will only wait for a response from a slave until a *timer has expired*. If the slave does not respond within the timeout value for several consecutive polls, it is assumed to have dropped off the network. Also, *every cycle time*, the master attempts to poll an inactive slave node (in a round-robin fashion). This way, new slaves *can be added* to the network and will be eventually noticed by the master. Slave devices can only respond when queried by a

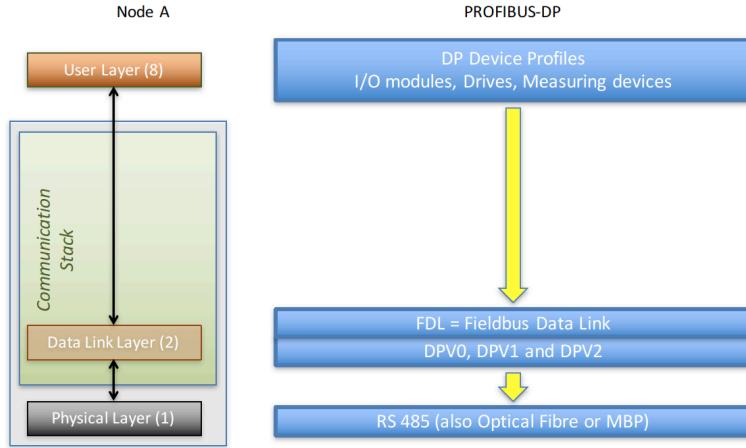


Figure 4.32: *PROFIBUS-DP* and the OSI model.

master. The operation of the network is based on a *polling cycle of slaves* that is repeated continuously by the master. At each query of a device, the master provides the slave with output data and, at the same time, fetches the input data. Suitable techniques are also available for the transmission of acyclic data.

The overall architecture comprising token-passing and master-slave is depicted in Figure 4.33.

The profiles in *PROFIBUS-DP* are of three types: *DP master class 1* (DPM1) is a central controller that *cyclically exchange information with the distributed slaves*. Typically they are programmable controllers such as PLCs, PCs, etc.; *DP master class 2* (DPM2) is a device for *configuring, maintenance and diagnostics*. Typically they are laptops equipped with software for configuration, maintenance and diagnostics; *DP slave*: Peripheral device performing cyclic exchanges with "its" active station. The type of each station implicitly defines the priority of the messages associated, with *DPM1* being the highest.

Application profiles complete the standard for a given area of application, e.g., *numerical controllers and robots* (based on sequential diagrams, movements and commands are described from the point of view of the control system), *encoders* (based on the connection of rotary, angle and linear encoders, and based on the definition of functions, such as scaling, diagnostics, etc.); *PROFIDRIVE variable speed drives* (based on the basic functions of the drive: drive commands and states are described); *process control and supervision* (HMI - specifies how control and supervision devices are linked with higher-level control system components and it uses the extended functions of

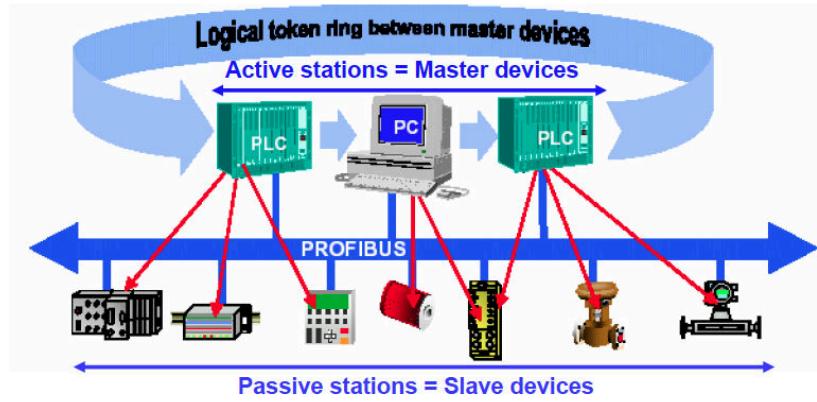


Figure 4.33: PROFIBUS: the token ring and master-slave (courtesy of Schneider Electric).

PROFIBUS DP relating to communication).

PROFIBUS-PA

PROFIBUS-PA (utilising MBP-IS, i.e., it operates at a single baud rate of 31.25kbit/s) is a version suitable for industrial automation developed by *Siemens* that guarantees *interchangeability*, the *intrinsic safety* of stations and also permits *bus-powered stations*. It utilises the transmission technique specified in IEC 61158-2 (as the FF H1) and/or RS-485. The standard IEC 61158-2 is related to *serial communication with intrinsic safety*. IEC 61158-2 is a bit-synchronous protocol with continuous *current-free transmission*: the transmission *does not* inject power on the bus when a station is sending. Every field device consumes a constant basic current at steady-state and, hence, acts as passive current sinks.

Its *Physical layer* matches up with that of the FF H1 *Physical layer* specification for the low speed standard, but still works much like the command and control features of **PROFIBUS**. It does not implement the *Application layer*, hence the *Communication Stack* only comprises the *Data-Link layer* (see Figure 4.34). Strictly speaking, it can be considered as a sub-net of **PROFIBUS-DP**, used to communicate directly with field process transmitters and hazardous areas (see Figure 4.30).

Comparison

The typical frame of **PROFIBUS DP & PA** comprises:

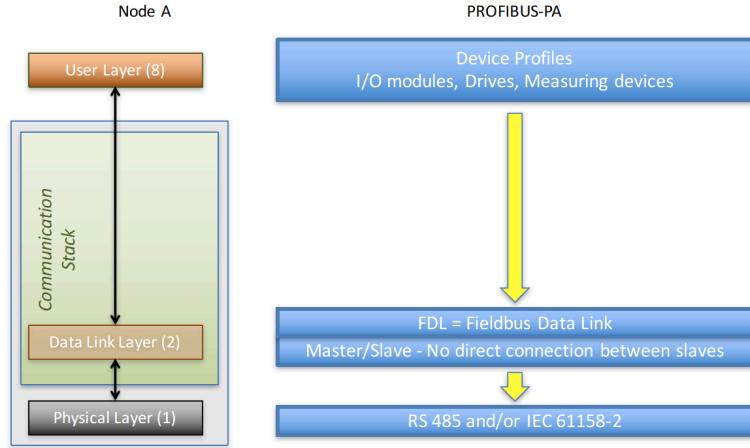


Figure 4.34: *PROFIBUS-PA* and the OSI model.

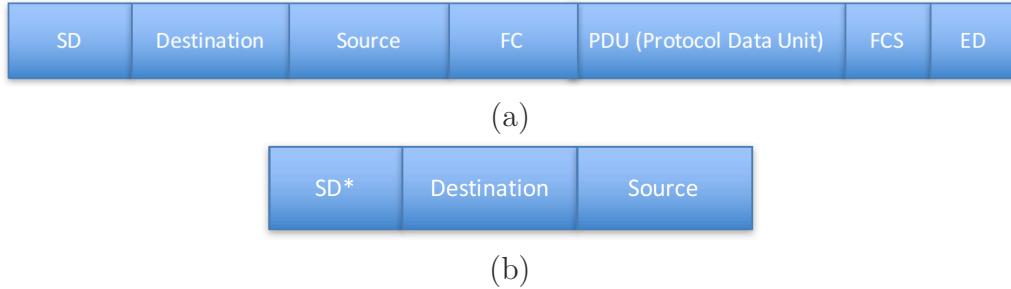


Figure 4.35: Typical data frame (a) and token frame (b) of *PROFIBUS DP & PA*.

- SD: *Start Delimiter*.
- FC: *Function Code*.
- FCS: *Frame Checking Sequence*.
- ED: *End Delimiter*.

The graphical representation of the typical message and token frames can be found in Figure 4.35. In a token frame, the SD* is a start delimiter different from SD.

Both the PROFIBUS-DP and FMS communicate with the RS-485 standard. The RS-485 standard defines the *electrical characteristics* of drivers and receivers for use in balanced digital multipoint systems. RS-485 can be used effectively over *long distances* and in *electrically noisy* environments, thus making it perfectly suitable for industrial environments. It offers data

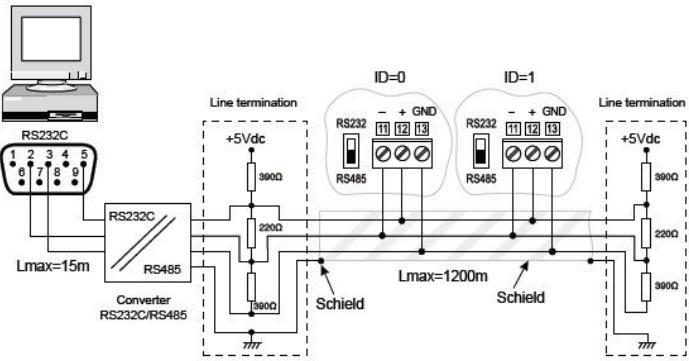


Figure 4.36: RS-485 connection.

transmission speeds of 35 Mbit/s up to 10 m and 100 kbit/s at 1200 m. It uses a differential balanced line over twisted pair (*two wires*), shown in Figure 4.36. The line terminators comprise one *pull-down* resistor towards the data-ground and one *pull-up* resistor towards the power supply voltage. This way the voltage in *idle state* is known. The typical RS-485 cables, connectors and data transmission levels are reported in Figure 4.37. To transmit a logical “1”, Conductor B has a positive level with respect to Conductor A. It implements an *asynchronous, half-duplex serial* communication, however full duplex cables also exists, as reported in Figure 4.38.

No more than 32 devices can be attached to the bus. For a higher number of devices, *repeaters* should be added. Notice that the repeater should be included as a device. Example: if you have 33 devices, you will have 31 devices on the first *segment*, hence the repeater and then other 2 devices on the second *segment*. The total number of stations, however, is *below 127*, which is the maximum number of different addresses.

Protocol	Developer	Year	Standard	Topology	Physical Media	Max Dev.	Max Dist.	Comm. methods
ARCNET	Datapoint SMC	1975	ANSI 878	Bus	Twisted pair Coax, fiber	255 nodes	6 km	PtP
AS-I	AS-I Cons.	1993	IEC CiA	Bus, Ring, star	Two wire Bus	31	100 m - 300m	M/S with polling
CAN open	Phillips	1992	Controlnet	Bus, tree, star	Twisted pair Coax, fiber	64 nodes	500 m	M/S
CONTROL NET	Allen Bradley	1997	International	Bus	Coax R6/U fiber	99 nodes	5 km coax 30 km fiber	M/S, M-M, PtP
DEVICE NET	Allen Bradley	1994	ISO 11898	Bus	Twisted pair with power	64	500 m	M/S, M-M
FOUNDATION FIELDBUS	Fieldbus Foundation	1995	ISO 11519	Multidrop	240 segments 6500 segments	1900m @31.25 kbps 500m @2.5 Mbps	1500m	C-S, P/S
INDUSTRIAL ETHERNET	Intel, Xeros, HP	1979-1998	IEEE 802.3	Star, bus	Twisted pair Fiber	247	400m x segment 12.8 km total	CSMA/CD
INTERBUS-S	Phoenix Contact	1984	DIN 19258	Segmented bus	Twisted-pair, Fiber	256	400m x segment 12.8 km total	M/S
LON Works	Echelon Corp	1991	AS-TRADE star	Bus, ring, fiber	Twisted-pair, @75kbps	3200	2000 m	M/S, PtP
MODBUS PLUS	AEG Modicon	1981	None	Network segment	RS-485 per segment	64 max	450m - 1800m	TP
PROFIBUS DP/PA	Profibus Org.	DP:1994 PA:1995	DIN 19245 part 3/4	Bus, star, ring	Twisted pair, fiber	127	24 km (fiber)	M/S, PtP
SDS	Honeywell	1994	ISO11989	Bus	Twisted pair with power	64	500m	M/S, PtP, M-M
SERIPLEX	APC Inc	1990	Seriplex Spec	Tree, ring, star	4-wire	500+ nodes	150+ m	M/S, PtP
WORLD FIP	World FIP	1988	IEC 1158-2	Bus	Twisted-pair, fiber	256 nodes	up to 40 km	PtP
Protocol	Data rate	Data Size	Arbitration	Error Check	Diagnostics	Cycle time	Cycle time	Transfer of 128 bytes
ARCNET	31.25 kbps - 10 Mbps	508 bytes	Token	16 bit CRC	Built in	< 2 ms @ 2.5 Mbps	< 2 ms @ 2.5 Mbps	< 2 ms @ 2.5 Mbps
AS-I	107 kbps	31 slaves	M/S polling	Manchester	Fault detection	4.7 ms	Not possible	Not possible
CAN open	125Mbps - 1Mbps	8 bytes variable	CSMA	CRC check	Bus monitor	Not available	Not available	Not available
CONTROL NET	5 Mbps	510 bytes	TDMA	16 bit CCL/T	Fault detection	2 ms	2 ms	2 ms
DEVICE NET	125 - 500 kbps	8-bytes variable	CSMA	CRC check	Bus monitoring	2.0 ms @ 500 M/S polling	10 ms @ 500 M/S polling	36 ms @ 31.25 kbps
FOUNDATION FIELDBUS	31.25 kbps, 1 Mbps, 2.5 Mbps	??	Scheduled	16 bit CRC	Remote diagnostics	100ms@31.25 kbps, <1ms@2.5 Mbps	600 ms @ 31.25 kbps, <8ms@2.5 Mbps	36 ms @ 31.25 kbps, 0.45ms@2.5 Mbps
INDUSTRIAL ETHERNET	10 - 100 Mbps	1500 bytes	CSMA/CD	32 bit CRC	Net, management	Network dependent	Network dependent	Network dependent
INTERBUS-S	500 kbps full duplex	512 bytes	None	16 bit CRC and cable break	CRC error location	1.8 ms	7.4 ms	140 ms
LON Works	1.125 Mbs full duplex	228 bytes	CSMA	16 bit CRC	Device error	20 ms	5 ms @ 1 Mbps	5 ms @ 1 Mbps
MODBUS PLUS	1 Mbps	256 bytes	PtP, TP	16 bit CRC	Local chip	Not available	Not available	Not available
PROFIBUS DP/PA	DP:12 Mbps, PA: 31.25 kbps	244 bytes	TP	HD4 CRC	Diagnostics	'Typical < 2 ms	'Typical < 2 ms	'Typical < 2 ms
SDS	125 kbps - 1 Mbps	8 bytes variable	CSMA	CRC check	Diagnostics	< 1ms event driven	< 1ms	2 ms @ 1 Mbps
SERIPLEX	200 Mbps	7680 transfer	Multiplexing	End of frame	Cabling problems	1.32 ms@200Mbps	10.4 ms	10.4 ms
WORLD FIP	31.25 kbps up to 6Mbps (fiber)	No limit	Arbitration	16 bit CRC and redundant cabling	Device time-out	2 ms @ 1 Mbps	5 ms @ 1 mMbps	5 ms @ 1 mMbps

Table 4.1: A list of automation protocols and their characteristics.

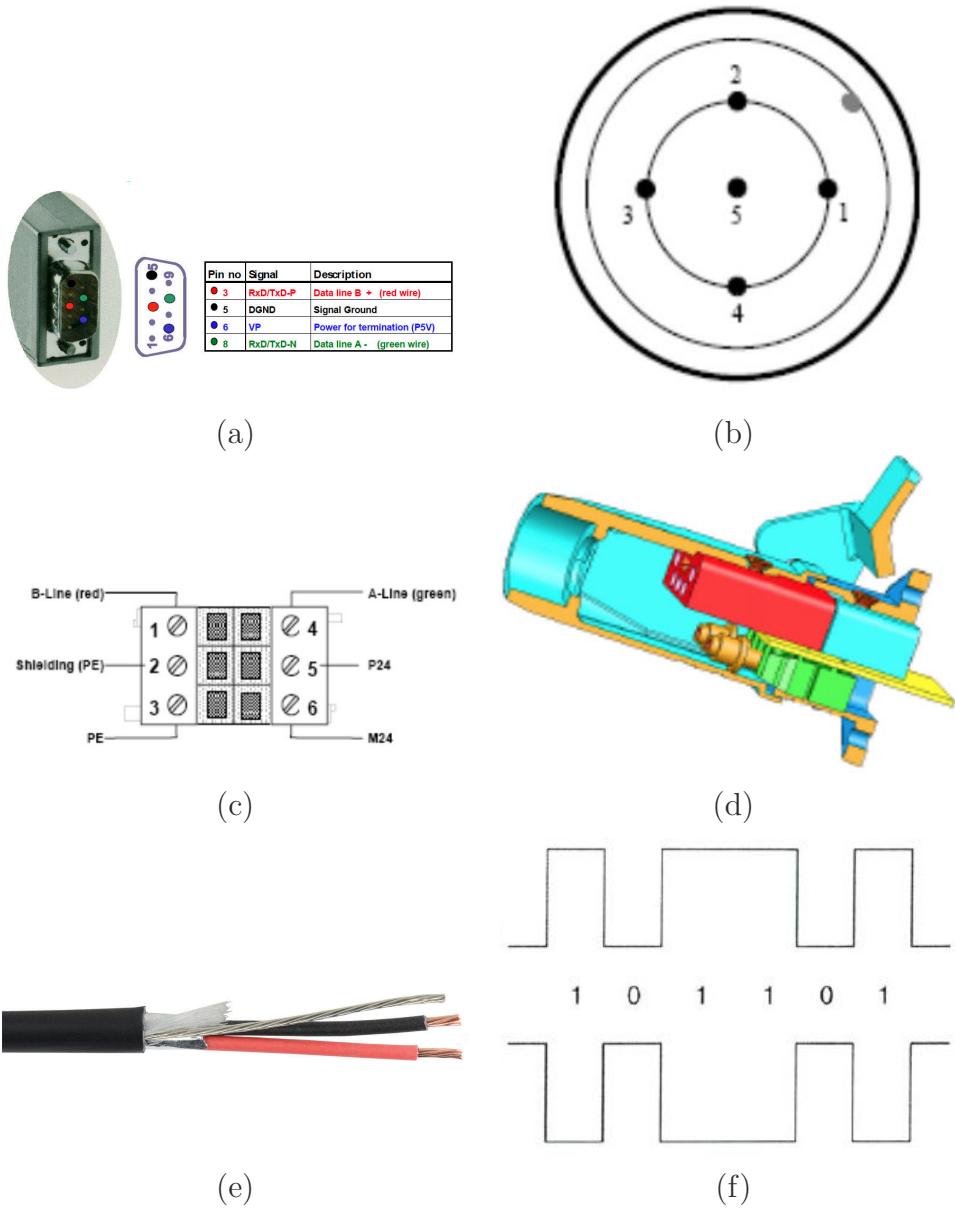


Figure 4.37: (a) RS-485 connector. (b) RS-485 connector: M12 circular connector according to IEC 947-5-2. Degree of protection IP65/67. (c) RS-485 connector: Siemens Hybrid Connector for transmission of 24 V power supply and PROFIBUS data via copper wires: degree of protection IP65/67. (d) RS-485 connector: HAN-BRID connector for fibre optic communication and 24 V power supply according to DESINA recommendations: degree of protection IP65/67. (e) RS-485 cable. (f) RS-485 data transmission.

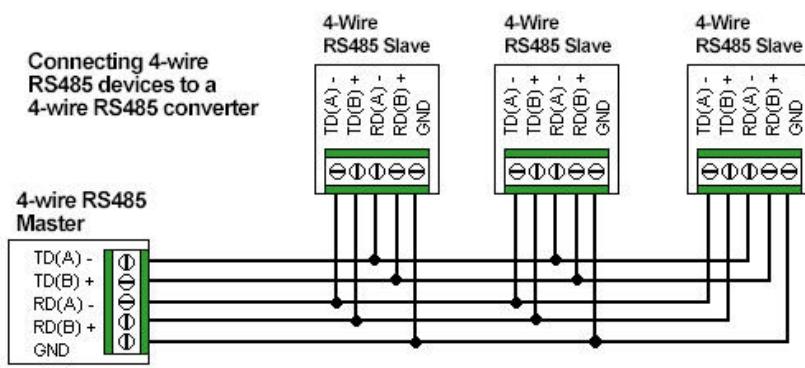


Figure 4.38: RS-485 full-duplex with four wires.

Chapter 5

Industrial Networks and QoS

Control networks are replacing traditional point-to-point wired systems while providing a number of advantages, such as fewer physical potential points of failure using reduced *wiring*. Networks enable *complex distributed control systems* to be realised: in *horizontal* direction having peer-to-peer coordinated control among sensors and actuators; in *vertical* direction adopting machine to cell to system level control. Another important point is the enhanced *interchangeability* and *interoperability* of devices, and improved *reconfigurability* of control systems.

One primary application of the *industrial networks* is the *Supervisory Control And Data Acquisition* (SCADA) systems. However, networks are being used at *all levels* of the manufacturing hierarchy, loosely defined as *device, machine, cell, subsystem, system, factory, and enterprise*. We can identify three different sub-domain of applications in industrial networks: *safety networks*, where the main important features are determinism, reliability and self diagnosis; *diagnostics networks*, meant for data collection and event based reactions without determinism constraints; *control networks*, that can be either time or event based and where determinism plays the lion's share.

At the industrial level, there is a strong intent in *minimising the cost* and *maximise interoperability and interchangeability*. Hence, there is a strong effort in consolidating around a single network technology at different levels of control and across different application domains. For example, *Ethernet*, which was widely regarded as a high level-only communication protocol in the past, is now being utilised as a lower level control network. Ethernet has achieved an undisputed *leading role* for *diagnostics networks*. Ethernet is *gaining relevance* for *control networks* (enhanced determinism using *switches*). However, it is not yet clear whether or not Ethernet is a candidate for *safety networks*.

5.1 Parametrisation of Industrial Networks

Although any network protocol can be used to send data, each network protocol has its pros and cons. The main questions to answer when selecting a network are:

- Will the network carry *many small packets* of data frequently or *large packets* of data infrequently?
- Must the data arrive before a given *deadline*?
- *How many nodes* will be competing for the bandwidth, and how will the contention be handled?

Two possible parameters that can answer to the previous questions are the *average data rate* and the *determinism*. The *average data rate* is function of the network *access time* and *bit transfer rate*, i.e., the *throughput*. The *determinism* is a measure of the ability to communicate data consistently from end-to-end within a guaranteed time. As we saw, parameters that affect determinism are the *Physical layer* adopted, e.g., wired or wireless, the *MAC sublayer* and, partially, all the other layers implemented in the ISO OSI model.

Another important feature for an industrial distributed application is the *device data rate*, in terms of *reaction* to events and *computation* times. Indeed, is becoming more and more evident in practical applications that the device data rate plays a *major* role in the plant effectiveness. The lowest between all *the different* network segments and the device performance is the *communication bottleneck*.

Quality of Service

Is there any unified vision about this topic? The *Quality of Service* (QoS) of a network is a *multidimensional parameterised measure* of how well the network exports its functionalities. Associated to the QoS there are the capability of *evaluating* and *controlling* its parameters. The QoS is critical since there is a *shared resource* that has to be mediated among the different network nodes in order to give guarantees about the time to deliver correctly the information. The parameters usually include:

- The *data rate* and *throughput*: how much data can be transmitted in a time interval;

- The *delay* and *jitter*: parameters associated with data transmission representing the time for a message to reach its destination, and the repeatability of this time interval;
- The *reliability* and *security*: parameter associated to the overall network infrastructure.

For *control networks*, it is of prominent importance to assess *determinism* as a QoS parameter.

5.2 Transmission times

Recall that the *data rate* of an industrial network is given in terms of the number of bits that can be transmitted per second. Industrial networks vary widely in data rate: the *CAN-based networks* have a maximum data rate of 1 Mb/s; the *Ethernet-based networks* can support data rates up to 1 Gb/s, rising up to 10 Gb/s for *optical fibre* medium; the *DeviceNet* is based on CAN and has a maximum data rate of 500 kb/s. A lot of networks currently used in the manufacturing industry are based on 10 and 100 Mb/s *Ethernet*.

The *time to transmit a bit* is the “inverse of the data rate”. For example: the time to transmit 1 bit of data over the network is $t_{bit} = 1 \mu\text{s}$ for 1 Mb/s CAN; the time to transmit 1 bit of data for a 10 Mb/s Ethernet is $t_{bit} = 100 \text{ ns}$. The data rate of a network must be considered together with the *packet size* and *overhead*. Indeed, data is *encapsulated into packets*, with headers specifying the source and destination addresses of the packet, and often a checksum for detecting transmission errors. All industrial networks have a *minimum packet size*, ranging from 47 bits for CAN to 84 bytes for Ethernet. Moreover, a minimum *interframe time*, which is specific to each protocol, has to elapse between two successive frames to ensure that each packet can be distinguished individually. The *overhead* is the set of bits used for *addressing* and *padding* the packet.

The *transmission time* t_{tx} is a quantity that can be measured over a network with a very high precision. t_{tx} is given by the sum of two components: the *frame time* t_{fr} and the *propagation time* t_{pr} , hence

$$t_{tx} = t_{fr} + t_{pr}.$$

The *frame time* t_{fr} obviously depends on the number of bits used in the packet encoding. Depending on the adopted protocol, the frame may contain start/stop bits, error checking mechanisms, preamble, identifier, data, etc.

Usually, *padding* bits are also used to meet the minimum size number of a certain network. The *frame time* is then given by

$$t_{fr} = N_{bits} t_{bit},$$

where N_{bits} is the overall number of bits contained in the frame. The *frame time* and the packet encoding should be carefully considered along with the network *data rate*. For example, to send one bit of data over a 500 kb/s CAN network, a 48 bit message is needed, hence a *frame time*:

$$t_{fr} = \frac{N_{bit}}{\text{data rate}} = \frac{48}{500\text{kb/s}} = 96\mu\text{s}.$$

To send the *same one bit* of data over 10 Mb/s Ethernet, an 84 byte message is needed (64 byte frame size plus 20 bytes for interframe separation), requiring a $67.2\ \mu\text{s}$. Thus, even though the raw network data rate is 20 times faster for Ethernet, the frame time is only 30% lower than CAN. This example shows that the network data rate is only one factor that must be considered when computing the effective data rate of a network.

The *propagation time* t_{pr} is usually negligible since the *transmission speed* of a standard medium is typically in between $2 \cdot 10^8$ m/s and $3 \cdot 10^8$ m/s (depending on the medium). For example, we usually have $t_{pr} = 67.2\ \mu\text{s}$ for a 2.5 km Ethernet cable or $t_{pr} = 1\ \mu\text{s}$ for a 100 m CAN bus. The *propagation time* is difficult to be characterised since it is dependent from the relative *distance between the devices*: it may be of some microseconds as the previous example or reach up to 280 ms for *geostationary satellite transmissions*. Moreover, if Ethernet is adopted, the presence of switches, bridges, routers and hubs on the network generate a delay that is largely more relevant than the t_{pr} .

Another important factor to consider is the time spent for *packet collisions*, which usually entails *packet retransmission* and depends on *traffic*. As already noticed, this is a function of the *Media Access Method* adopted in the MAC sublayer.

5.3 Delays and jitters

The *time delay* on a network is the total time between the data being available at the source node, e.g., sampled from the environment or computed by the controller, and the same data being available at the destination node, e.g., received and decoded. Even though, many control techniques have been developed for systems with constant time delays, with *variability* the

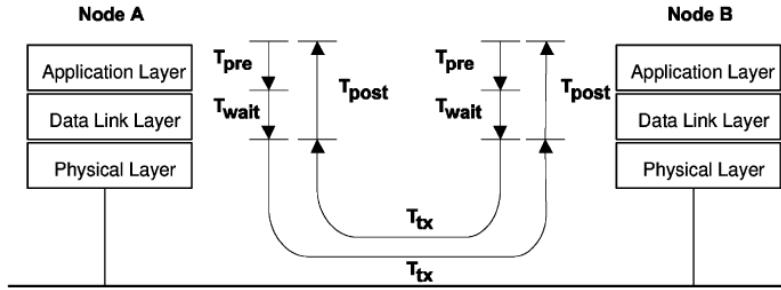


Figure 5.1: Timing diagram showing time spent sending a message from source node to destination node [12].

problems become worse. The *jitter* is the variability of the delay. Although time delay is an important factor to consider for control systems implemented over industrial networks, it has not been well defined or studied by standards organisations defining network protocols. As reported in Figure 5.1, as long as one node collects or computes the data of interest, it has to first *preprocess and encapsulate* the data for a time t_{pre} . According to the chosen media access method, there may be some time t_{wait} to wait to get the control of the shared resource. Then the message is transmitted within a time t_{tx} . Finally the message is received and post-processed by the destination node for a time t_{post} . Therefore

$$t_{delay} = t_{pre} + t_{wait} + t_{tx} + t_{post}.$$

The pre-processing t_{pre} and post-processing t_{post} times are very difficult to be estimated, and also to be measured. Moreover, they have usually a *large variability*. The t_{wait} expresses the fact that a message may spend time waiting in the *queue* at the sender's buffer and could be blocked from transmitting by other messages on the network. As recalled, this is related to the *traffic* and to the *MAC*. An example is reported by Figure 5.2, where DeviceNet (hence, endowed with a CAN protocol) devices are strobed. Even though the nodes with higher priority (lower node numbers) have on average a shorter waiting time (this is due to the mechanism of arbitration), not always the node with the highest priority takes the control of the bus: this is due to the post-processing time variability and the receiving node. Therefore, higher priority nodes experience also a larger variability. The opposite happens to lower priority nodes.

The wait time variability is of course quite relevant for *Ethernet TCP/IP* networks, but it may be detrimental also for Fieldbuses due to the adopted *MAC* solutions. Indeed, for TCP/IP-based networks, the wait time t_{wait}

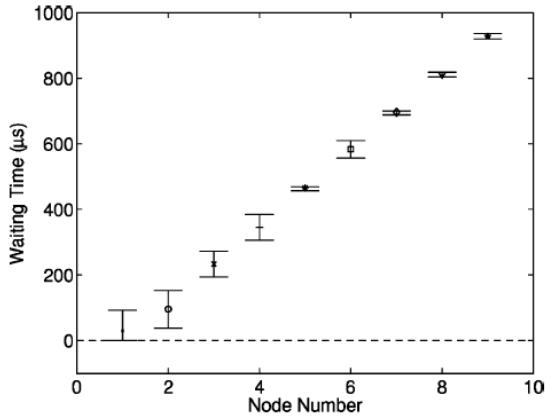


Figure 5.2: Nine identical devices on DeviceNet with strobed message connection: variability depends on the processing time [12].

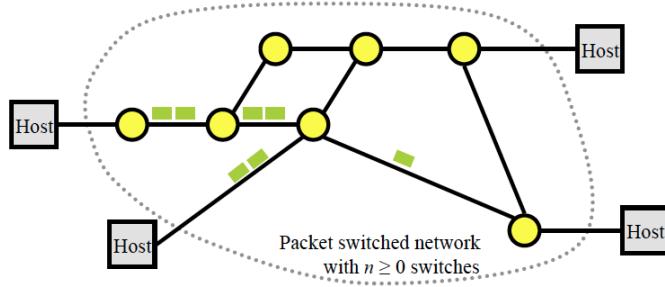


Figure 5.3: Packet delay is dependent from the particular route chosen (courtesy of University of Glasgow).

is still affected by the *traffic* and the *MAC*, but the traffic is usually *heavier* and the *MAC* protocol is based on *disruptive collision detection*. As a consequence, the messages, incoming and outgoing, are stored in message *queues* that are largely responsible of delay variability in senders and receivers. *Message queues* are widely studied in literature and are one of the main responsible of the presence of the *delay* for this type of networks. *Message queues* are implemented at the *Transport layer* and are part of the TCP.

When using large networks, especially if based on Ethernet, the packets are usually *routed* in different directions (see the picture in Figure 5.3). Since the network is usually different depending on the choice of the link made, the delay changes according to that choice. In *switched networks* the sources of these uncertainties are given by the presence of *switches* (or sometimes

hubs).

Example 10. To understand the effect of packet routing, consider a network with r routers and with **data rate** of m bs, where we assume $t_{pr} \approx 0$, i.e. $t_{tx} \approx t_{fr}$. Imagine that a node A transmit a message with p packets of N_{bit} each to a node B at time t_0 . Hence, at time $t_0 + t_{tx}$, where $t_{tx} = N_{bit}/m$ s, the packet have entirely reached the first **router** and can then be retransmitted to the **second router**, where it will be received at time $t_0 + 2t_{tx}$.

It then follows that at time $t_0 + (r+1)t_{tx}$ the **first packet** reached node B. In other words the **end-to-end delay** is $\delta_{end-to-end} = (r+1)t_{tx}$. To transmit the entire message, $p(r+1)t_{tx}$ seconds are needed.

To reduce this time, we can make use of **pipelining**: after the first packet reached the first router, we can start with the transmission of the second packet. Therefore, the entire transmission process becomes $(r+1+p-1)t_{tx} = (r+p)t_{tx}$ in place of $(pr+p)t_{tx}$. It has to be noted that pipelining has no effect on the end-to-end delay!

The effect of packet routing can be very relevant and highly detrimental: multiple links are connected together to the same switches and, hence, switches need **message queues**. In particular, each **switch** has an **output queue** for each link: if a message finds the desired link **busy**, it should be stored in the **output queue** and, hence, increase the delays. But it can be even worse: since the amount of buffer space is finite, an arriving packet may find that the buffer is completely full with other packets waiting for transmission: whatever the policy, the packet will be lost. Since the **traffic** has an immediate effect on the **message queue** status, it is evident that the more the traffic, the more the probability of loosing packets (either by **collision** or **message queues**).

Example 11. Suppose, for simplicity, we have a network where all the packets have length N_{bit} and the **data rate** is m bs and assume that the average arrival rate at the **message queue** in a switch is α s⁻¹. Moreover, assume for simplicity's sake that the **message queue** as an **ideal infinite length**. It turns out that if $\alpha N_{bit}/m > 1$, then the queue reaches an infinite length.

$\alpha N_{bit}/m$ is the traffic intensity, which leads us to the **golden rule**: always design your system to have $\alpha N_{bit}/m < 1$.

To bound the **delays and jitters**, either **buffering** is applied along the line to reconstruct timing (e.g., **isochronous communication**) and/or a suitable **scheduling of messages** is adopted. According to the **Oxford Dictionary**:

Definition 23. An **isochronous network** is a high-data rate network for which the latency can be easily predicted. Such networks are used for applications such as video on demand.

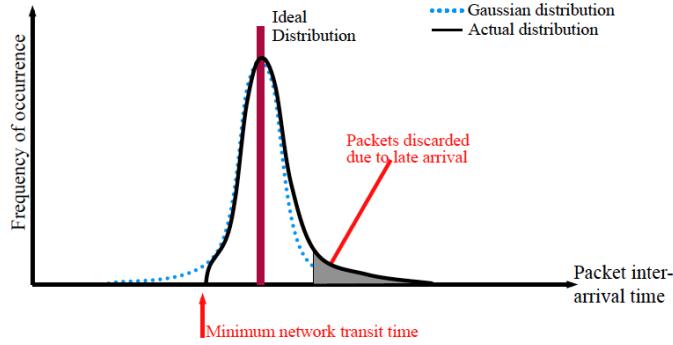


Figure 5.4: A measure of the jitter is usually given stochastically through direct measurements (courtesy of University of Glasgow).

Strictly speaking, in isochronous networks the correctness of the data depends on *both* the *data content* and the *time* in which it has arrived (the same concept of RT). Example: tracking a target in a distributed surveillance system. The main problem for distributed systems is that the internal clock of each device is *free-running* and usually *unsynchronised*. Hence, it results in a steady increase or decrease in the inter-packet spacing observed at the receiver. The solution is to adopt specifically conceived *synchronisation protocols* to synchronise the network clocks.

Since *delay* and the *jitter* are extremely relevant for the QoS and since they are related to *unpredictable fluctuations*, their behaviour is quantified using a *stochastic description*. The stochastic description is given in terms of the set of *measurements* that can be collected, as customary for every measurement system. An example of such description is given in Figure 5.4.

5.3.1 Additional QoS measures

Other measures of QoS that can be adopted are the reliability and the security. *Reliability of data transmission* deals with data corruption: some network adopt a *handshake* mechanism by sending *acknowledgment* packets (which introduce overhead). *Security* refers to malicious attack or viruses: for instance, fieldbuses prevent this problem by detaching the internal network from the outer network. In this case, the secure connection are used to *prevent misuses* of process data rather than preventing malicious attacks.

Notice that *Security* is not *safety*!

Definition 24. *Safety* is the state of being *safe*, i.e., the condition of being

protected against physical failure, damage, error, accidents, harm, or any other event that could be considered non-desirable.

Definition 25. *Security* is the degree of robustness/resistance to or protection from harm.

For networks, the term *security* refers to the policies adopted to prevent and monitor authorised access, misuse, modification, or malicious usage of network-accessible resources, which has been a major problem for Ethernet. Indeed, Ethernet adds up to an automation environment that is at once *open*, with the ability to *communicate with other networks* as needed, and can be *managed from anywhere*. When automation systems are attached to Ethernet networks, *the same sort of threats* that face any Internet-connected computer, including hackers, worms, Trojans, and various other forms of malware, should be managed. There are different possible solutions to face those threats, however for industrial networks the highly adopted tool kits are *firewalls*, *VPNs* (virtual private networks), *network address translation* (NAT) technology.

5.3.2 QoS vs QoC

The QoS offered by the network is not the only parameter to consider in a *control network*. Indeed, for control applications, the *Quality of Control* (QoC) measure is of major importance. The QoC can be measured using standard measures in control theory, e.g., *steady state value*, *settling time*, *rise time*, *mean square stability*, etc. In practice, the QoC is a function of the QoS offered by the network. Another important player for QoC on control networks is related to the *digital controller* implementation: the *continuous time* controller should be sampled. The digital controller is a function of the *sampling time* adopted. Ideally, the *shorter* is the sampling time, the *closer* is the continuous time behaviour of the plant. However, this represents a problem for *networked* and *distributed* control systems, since the number of data to be transmitted over the network increases as well, hence increasing latencies, delays, traffic, etc. Usually the choice of the sampling is *traded* between this two contrasting goals, as clearly reported in Figure 5.5.

5.4 Control networks: Final comments

Control networks in a factory are typically divided into *multiple levels* to correspond to the factory control distributed in a *multitiered hierarchical* fashion, as well exemplified in Figure 5.6. At the lowest level are the *device*

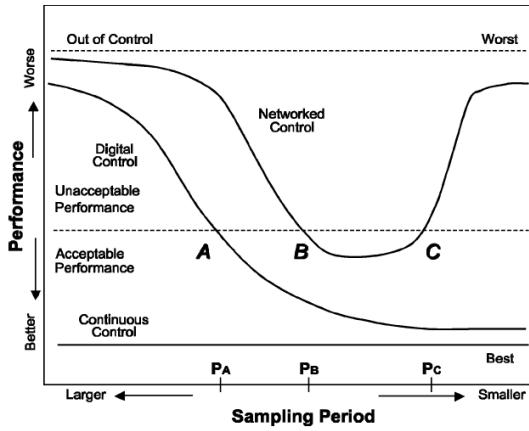


Figure 5.5: Performance comparison of continuous control, digital control, and networked control, as a function of sampling frequency [12].

networks, which are usually characterised by *small number* of nodes (e.g., less than 64), communicating *small data* packets at high sample frequencies and with a higher level of *determinism*. Example: *servo control*, where network delay and jitter requirements are very strict (see Figure 5.7). The *fieldbuses* are usually adopted at this level. Although seemingly non optimal for this level of control, Ethernet is becoming more common by adopting switches and adding features at the protocol (e.g., *Profinet*). The required *determinism* and *reduced jitter* for lower level networks are enhanced by utilising network contention, such as *master-slave* operation or *polling* techniques.

An intermediate level of network is the *cell* or *subsystem*, which includes *SCADA*. At this level, multiple *supervisory controllers* are connected to the network, instead of having devices directly connected to the network, thus forming the *distributed control system*. The controllers exchange both information and control signals, but the cells or subsystems are typically physically decoupled and, hence, the *timing requirements are not as strict* as they are at the lowest levels. These networks are also used for *maintenance*. *Token-passing* and *Ethernet-based* networks are commonly used at this level, with the ability to communicate larger amounts of data and support for network services generally *taking precedence over strict determinism*.

The highest level comprise networks at the *factory or enterprise level* that coordinate multiple cells and link the factory floor control infrastructure to the enterprise level systems (e.g., part ordering, supply chain integration, etc.). Large amounts of data travel over these networks, but real-time requirements are usually *nonexistent* (see Figure 5.6). *Ethernet* is the most popular choice here primarily because internet support at this level is usu-

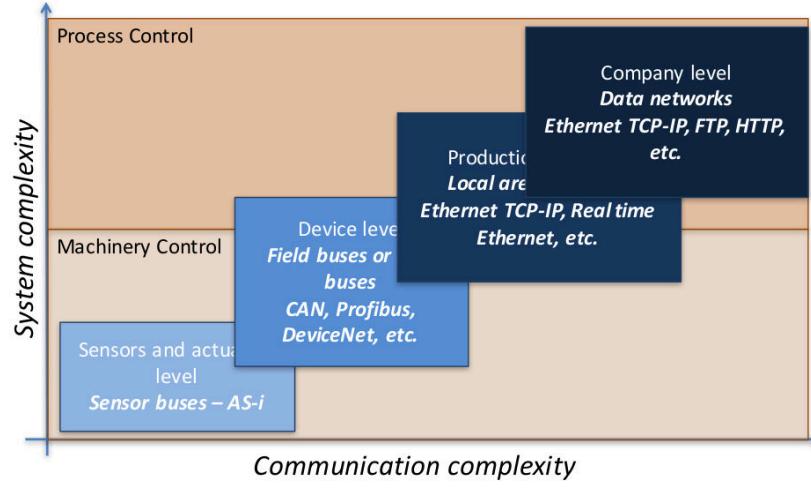


Figure 5.6: Main industrial networks and buses: company level.

ally critical, and Ethernet also brings attractive features to this environment such as support for high data volumes, network services, availability of tools, capability for wide area distribution, and low cost. An example of an actual plant automation structure can be found in Figure 5.8.

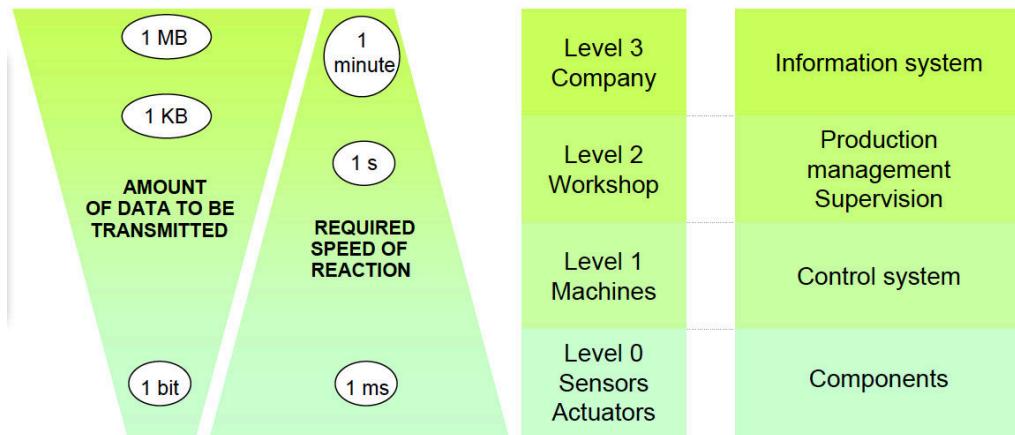


Figure 5.7: Example of required QoS (Courtesy of Schneider Electric).

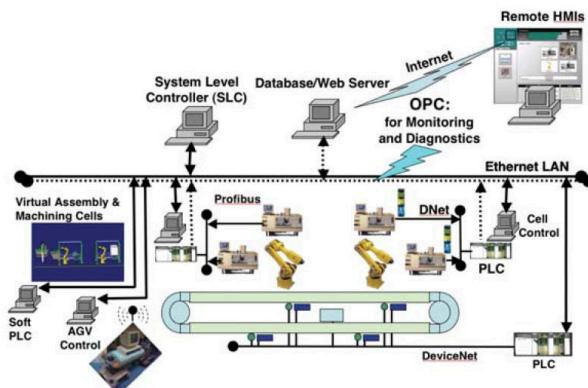


Figure 5.8: Control networks are indicated by solid lines, and diagnostics networks are indicated by dashed lines [12].

Chapter 6

Ethernet as Control Network

The proliferation of the internet has led to the pervasiveness of Ethernet in both homes and businesses. Ethernet has only 1 bit of difference from the standard *IEEE 802.3* (a subclass of the *IEEE 802* family, see Figure 6.1), that is the one concerning with CSMA/CD. Because of its low cost, widespread availability, and high communication rate, Ethernet has been proposed as the ideal network for industrial automation. Emerging real-time Industrial Ethernet solutions complement the fieldbus standards, e.g., by using common layers.

Standardisation started in the '80s with the *IEEE 802* bodies:

- 802.1: Local Area Network (LAN) Internetworking;
- 802.2: previously known as ISO/IEC 8802-2, defines the Logical Link Control (LLC) as the upper portion of the Data-Link layer of the OSI Model;
- 802.3: CSMA/CD (i.e. Ethernet);
- 802.4: Token Bus, which is a LAN in which the stations on the bus or tree form a logical ring;
- 802.5: Token Ring, which is a token passing scheme on a ring topology LAN used in place of CSMA/CD;
- 802.6: uses the Distributed Queue Dual Bus (DQDB) network form (i.e. a distributed multi-access network) used for Metropolitan Area Networks (MAN);
- 802.7: Broadband Technical Advisory Group;
- 802.8: Fibre-Optic Technical Advisory Group;

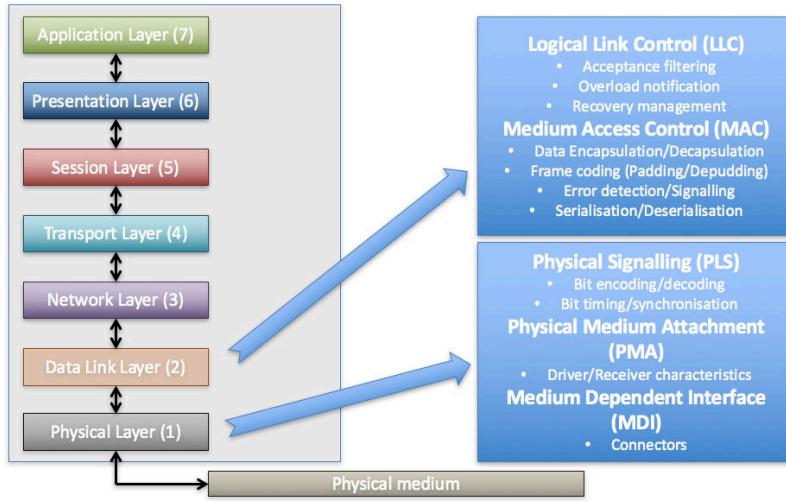


Figure 6.1: IEEE 802 family purposes.

- 802.9: Integrated Data and Voice Networks;
- 802.10: Network Security;
- 802.11: Wireless Networks (/a/b/g/h/f/s/n/p/ac/ax/...);
- 802.12: deals with the Data-Link layer. The IEEE 802.12 100 VG - AnyLAN is a high-speed network with a data rate of 100 megabits per second (Mb/s);
- 802.15: Personal Area Networks (PAN), such as Bluetooth (802.15.1) or ZigBee (802.15.4);
- 802.16: Wireless MAN (e.g. WiMax);
- Others up to 802.22.

Ethernet is only defined at the Physical layer and Data-Link layer of the OSI model and it has been conceived for standard non real-time traffic (200/300 ms of packet delays). At the Network layer and Transportation layer (the third and the fourth level of OSI), Ethernet is used by TCP/IP or UDP/IP. Notice that TCP and UDP are de facto the most used protocols, but they are not part of IEEE 802.3, and hence may or may not be used with Ethernet. The standard deals with Local and Metropolitan Area Networks (LANs and MANs), employing CSMA/CD as the shared media access method and the IEEE 802.3 (Ethernet) protocol and frame format for data

communication. The standard defines two distinct modes of operation: *half duplex* and *full duplex*. A given IEEE 802.3 instantiation operates in either half or full duplex mode at any one time, even though full duplex mode does not implement all the features of the protocol used for the shared medium (half-duplex) operations.

Standard Ethernet is not a deterministic protocol, and network QoS cannot be guaranteed. Non determinism is a natural consequence of the collision detection algorithm used at the MAC sublayer. To solve this problem, several solutions rely on adding layers on top of the TCP/IP protocol. Recently, the utilisation of switches to manage the Ethernet bandwidth can provide a TDM approach among time critical nodes that can be also very effective.

Ethernet is a random access network, also often referred to as Carrier Sense Multiple Access (CSMA). Once the carrier is clear, a node must wait for a specified amount of time, called the interframe time before sending a message. To reduce collisions on the network, nodes wait an additional random amount of time, called the backoff time, before they start transmitting. Priorities can be implemented by allowing for shorter interframe times for higher priority traffic. If a collision is detected, the data is corrupted and the message must be resent.

While transmitting, a node must also listen to detect a message collision. Basically, a collision is detected if the sender simultaneously receives a bit from another source: in such a case, a transmitting node continues the transmission by transmitting 32 jam bits to ensure that all receivers detect the collision. Finally, after a collision, the node waits a random length of time to retry its transmission.

The random time is determined by the standard Binary Exponential Backoff (BEB) algorithm. The retransmission time is randomly chosen between 0 and 2^k slot times, where k denotes the k -th collision event detected by the node. The slot time is the minimum time needed for a round-trip transmission. However, after ten collisions have been reached, the interval is fixed at a maximum of 1024 slots. After 16 collisions, the node stops attempting to transmit and reports failure back to the node microprocessor. Further recovery may be attempted in higher layers. The algorithm is reported in Figure 6.2.

The Ethernet standard states that valid frames must be at least 64 bytes long, reaching 72 bytes including preamble and start delimiter. There are two reasons for this minimum size limitation: a) it makes it easier to distinguish valid frames from garbage; b) it prevents a node from completing the transmission of a short frame before the first bit has reached the far end of the cable, where it may collide with another frame.

This size defines also the slot time: for a 10 Mb/s Ethernet with a max-

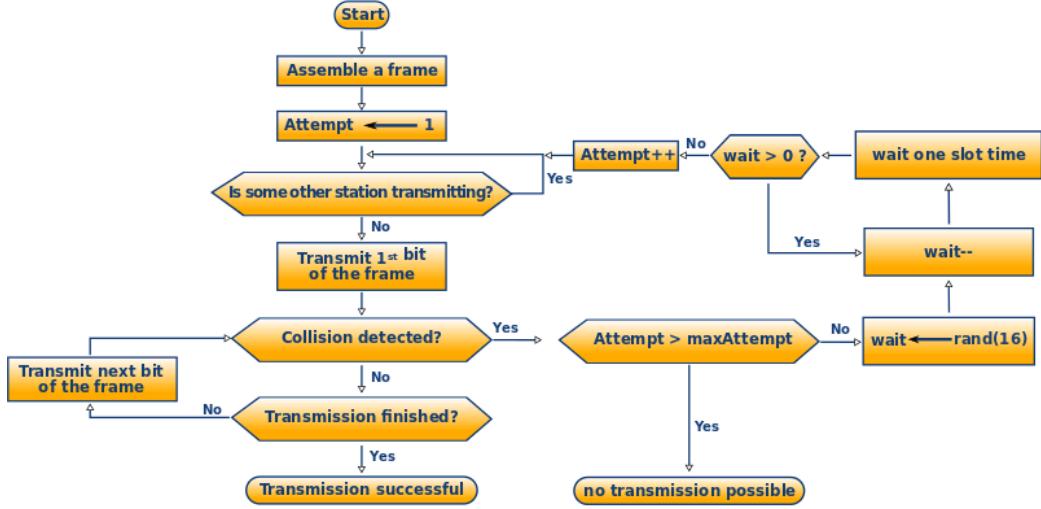


Figure 6.2: Simplified CSMA/CD algorithm.

imum length of 2500 m and four repeaters, the time to travel the end-to-end cable with a $t_{pr} = 2 \cdot 10^8$ is $12.5 \mu\text{s}$, which is greater than the minimum allowed slot time or frame time $t_{fr} = 51.2 \mu\text{s}$ (i.e. the time required to transmit 64 bytes at 10 Mb/s). A description of the dimension of the slot-time and of the jam-bits is reported in Figure 6.3. In the most common situation, the total overhead, i.e., data used to rules the transmission but not carrying any information, is 26 bytes, including preamble and start delimiter, as reported in typical frame of Figure 6.4. As a consequence, the minimum data packet frame size is $72 - 26 = 46$ bytes, while the maximum data size is 1500 bytes. If the data portion of a frame is less than 46 bytes, the pad field is used to fill out the frame to the minimum size.

Hub-based and Switched Ethernet

Hub-based Ethernet uses hubs to interconnect the devices on a network. When a packet comes into one hub interface, the hub simply broadcasts the packet to all other hub interfaces. All of the devices on the same network receive the same packet simultaneously, and message collisions are possible. Due to the presence of CSMA/CD algorithm, standard Ethernet gives some stochastic guarantees only if the network shows low traffic. With heavy traffic, no guarantee can be given. Moreover, due to the BEB algorithm, end-to-end communication is not deterministic as well.

Using standard Ethernet for control applications asks for alternative solutions:

Parameters	MAC data rate			
	Up to and including 100 Mb/s	1 Gb/s	2.5 Gb/s, 5 Gb/s, 25 Gb/s, 40 Gb/s, 100 Gb/s, 200 Gb/s, and 400 Gb/s	10 Gb/s
slotTime	512 bit times	4096 bit times	not applicable	not applicable
interPacketGap ^a	96 bits	96 bits	96 bits	96 bits
attemptLimit	16	16	not applicable	not applicable
backoffLimit	10	10	not applicable	not applicable
jamSize	32 bits	32 bits	not applicable	not applicable
maxBasicFrameSize	1518 octets	1518 octets	1518 octets	1518 octets
maxEnvelopeFrameSize	2000 octets	2000 octets	2000 octets	2000 octets
minFrameSize	512 bits (64 octets)	512 bits (64 octets)	512 bits (64 octets)	512 bits (64 octets)
burstLimit	not applicable	65 536 bits	not applicable	not applicable
jpgStretchRatio	not applicable	not applicable	not applicable	104 bits

Figure 6.3: Different lengths of the Ethernet frame components as function of the media adopted [14].

- Synchronisation: message can be time-stamped to reduce non determinism in communications. This solution can be adopted ensuring node clocks synchronisation, e.g., adopting Precision Time Protocol (IEEE 1588);
- Deterministic retransmission: collided packets are retransmitted with a deterministic delay, which ensures upper-bounded delays. The price is an under-exploitation of the CSMA/CD protocol and of the available bandwidth;
- Prioritised CSMA/CD: improvements in the response time for time-critical packets, e.g., the networking platform LonWorks.

The most effective solution is to use switches instead of hubs. Switched Ethernet utilizes switches to subdivide the network architecture, hence avoiding collisions, increasing network efficiency, and improving determinism. Switches “learn” the topology of the network and forward packets to the destination port only. Switched Ethernet usually relies on the star cluster layout to

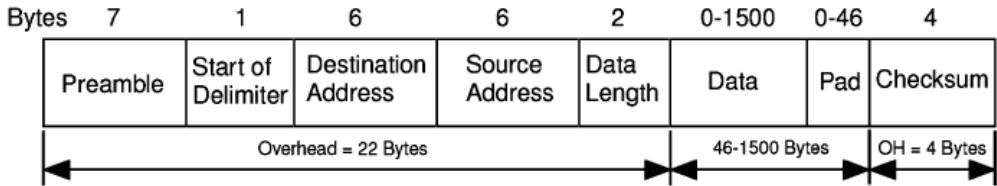


Figure 6.4: Ethernet (CSMA/CD) frame format; 20 byte interframe space not shown [12].

achieve this collision-free property. Switches use buffers to enqueue the messages on the output ports.

Two different solutions are adopted in switches:

- Cut-through: first read the destination address, i.e. the MAC address, and then forward the packet to the destination port according to the MAC address of the destination and the forwarding table on the switch.
- Store-and-forward: before applying the cut-through, these switches examine the complete packet analysing the Cyclic Redundancy Check (CRC) code. This way, even if a little bit slower, they do not forward corrupted packets.

To better compare the two methods, imagine that node A sends a packet to node B , that has N_{bit} number of bits over a link with a data rate of m bs. How long does it take from A to B if we assume that $t_{tx} \approx t_{fr}$ (i.e. ignoring the t_{pr})? With the Store-and-forward, the source begins to transmit at time t_0 and at time $t_0 + t_{tx}$, where $t_{tx} = N_{bit}/m$ s, the packet have entirely reached the switch and, hence, it can start the retransmission. Hence, at time $t_0 + 2t_{tx}$ the packet reached node B . With a Cut-through approach, the time would be $t_0 + t_{tx}$, but taking some risks. i.e. useless transmission of corrupted packets.

Now it should be easier to understand why in the Ethernet frame the destination address should be present. Imagine the work of a switch: when it receives the packet, it analyses the destination address and compare it to the stored forwarding table. Then, it routes the message towards the most efficient next switch (e.g., based on traffic, distance, other performance policies). Notice that the destination address is analysed with increasing level of details: this is exactly what a human being does when it has to reach a destination or asks for directions.

There are two different approaches for switches:

- Circuite switching: the resources needed along a path (e.g., buffers, links) to provide for communication between the nodes are reserved for the duration of the communication session (e.g., telephone lines);
- Packet switching: the resources are not reserved but, instead, used on demand, which asks for the use of message queues.

Again, circuit switching leads to under exploitation of resources (as for time based approaches), while packet switching uses the resources on demand (as for event based approaches).

Ethernet Full-duplex

As previously stated, even if the full-duplex operations implement just a subset of the CSMA/CD protocol, *CSMA/CD MAC* is intended in the standard as a synonymous of the *802.3 MAC*. Indeed, in the standard [14] it is reported that the:

“Full duplex operation allows simultaneous communication between a pair of stations using point-to-point media (dedicated channel). Full duplex operation does not require that transmitters defer, nor do they monitor or react to receive activity, as there is no contention for a shared medium in this mode. Full duplex mode can only be used when all of the following are true: a) The physical medium is capable of supporting simultaneous transmission and reception without interference. b) There are exactly two stations connected with a full duplex point-to-point link. Since there is no contention for use of a shared medium, the multiple access (i.e., CSMA/CD) algorithms are unnecessary. c) Both stations on the LAN are capable of, and have been configured to use, full duplex operation. The most common configuration envisioned for full duplex operation consists of a central bridge (also known as a switch) with a dedicated LAN connecting each bridge port to a single device. Repeaters as defined in this standard are outside the scope of full duplex operation. Full duplex operation constitutes a proper subset of the MAC functionality required for half duplex operation.”

In order to have a real-time communication system, the transmission of data needs determinism (i.e. the ability of serving a request in a limited and known time, that is the maximum latency is known upfront), isochronous behavior (i.e. the behavior is repetitive in time, that is low jitter) and synchronisation (e.g. of communication - TDM, of Input/Output (IO) operations, of the applications).

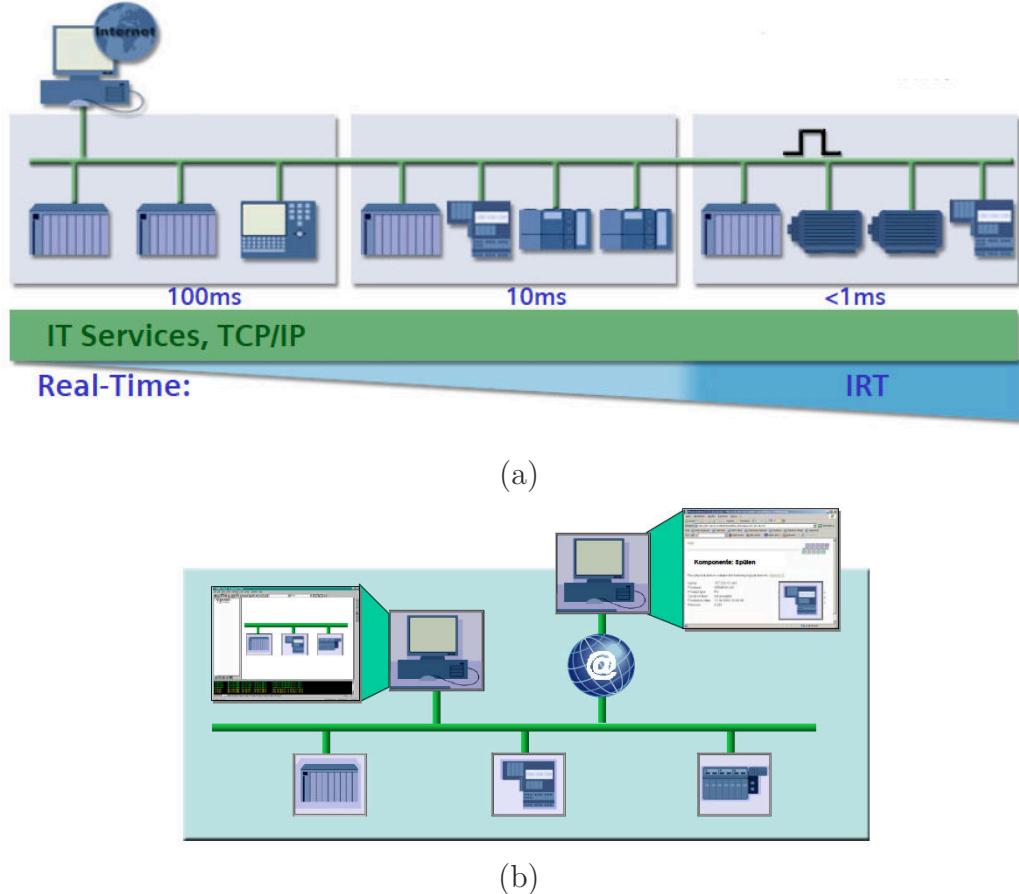


Figure 6.5: One solution offering different capabilities as a function of the desired traffic and with capabilities of being connected to the web (courtesy of Paolo Ferrari, Università di Brescia).

6.1 Profinet

PROFIBUS and Profinet (with the term Profinet we make explicit reference to Profinet IO) are the main communication solutions foreseen by the Profibus and Profinet International (PI) community for industrial environments, each of which is, in turn, made up of several profiles. Profinet is not only PROFIBUS over Ethernet. Implementing a fieldbus on Ethernet is simple but Profinet, in its developments, does not use this approach because it is ineffective and it does not introduce advantages with respect to classic fieldbuses. Profinet uses the advantages of Ethernet but it also improves it for the real-time part and it simultaneously transmits real-time and TCP/IP traffic (see Figure 6.5). It is organised by the PROFIBUS International (PI,

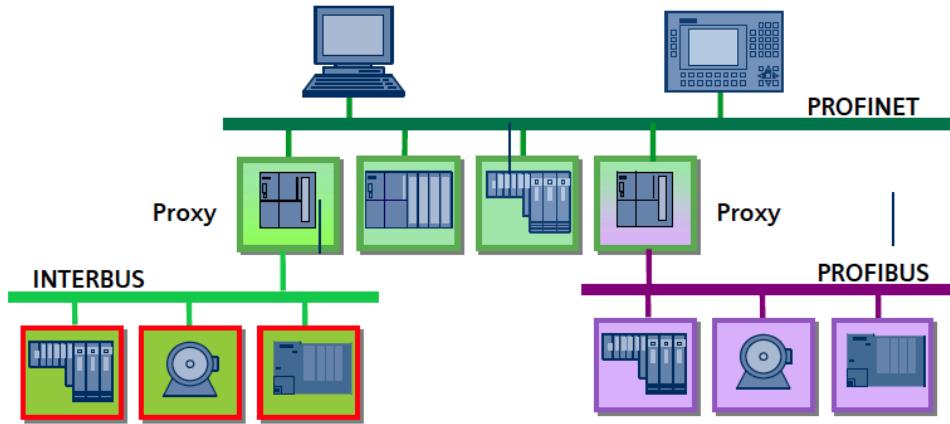


Figure 6.6: Integration of other fieldbuses (courtesy of Paolo Ferrari, Università di Brescia).

the same of PROFIBUS). Profinet IO is a Real-Time Ethernet (RTE) network included in Part 2 of the IEC 61784 International Standard. It can use mechanism for the network management (DHCP), network diagnostics (SNMP), web access (HTTP).

The components adopted in Profinet are:

- Profinet IO-Controller: manage the IO with sensors and actuators and execute the controller (e.g., PLCs and PCs);
- Profinet IO-Device: field device connected to the IO-Controller (e.g., actuator and sensors);
- Profinet IO-Supervisor: HMI and diagnostics.

An important feature is that the devices adopted for PROFIBUS can be used directly in Profinet, which relies on proxies (Figure 6.6) to integrate fieldbuses, such as Interbus or PROFIBUS.

Definition 26. *A proxy is the agency, function, or office of a deputy who acts as a substitute for another (Merriam-Webster dictionary).*

In computer networks, a proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers. Proxies were invented to add structure and encapsulation to distributed systems.

Profinet makes use of a Time Division Multiple Access (TDMA) technique to guarantee fast and precise access to the transmission medium, where

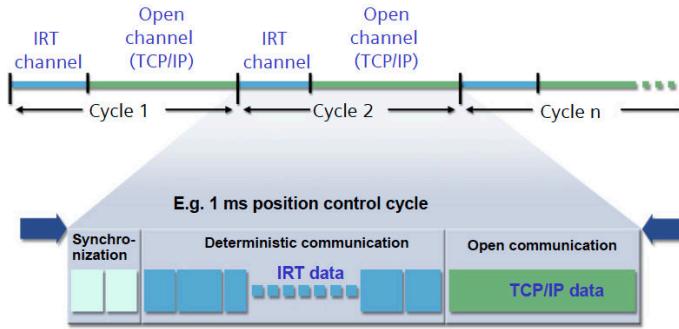


Figure 6.7: Time Division Multiple Access idea (courtesy of Paolo Ferrari, Università di Brescia).

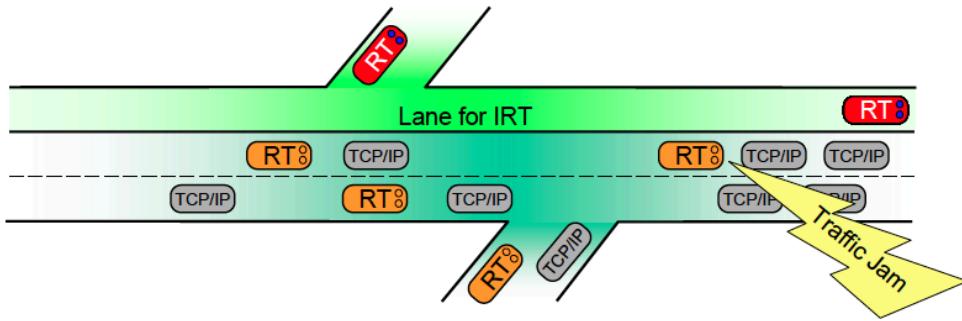


Figure 6.8: The basic idea (courtesy of Paolo Ferrari, Università di Brescia).

devices synchronisation is achieved by means of Precision Transparent Clock Protocol (PTCP). In practice, the traffic scheduling takes place according to a cycle timing (the different types of traffic are reported in Figure 6.7 with reference to the cyclic time). Messages are divided into three types, with increasing time criticality (as depicted in Figure 6.8): a) TCP/IP standard: used for devices set-up, diagnostics, start-up of the network; b) Real-Time (RT): used for high-performance data transfer, cyclic data, event-based data communication. Known as RT_CLASS 1; c) Real-Time channel (IRT): used for hi-performance data transfer with isochronous feature (i.e., jitter less than $1 \mu\text{s}$). Known as RT_CLASS 2 and RT_CLASS 3. In practice for RT_CLASS 3 (the most critical), the communication takes place over predefined physical paths enforced by a special kind of switches purposely developed for Profinet (similar to the circuit switches). For RT_CLASS 2 (that is also isochronous), physical paths are not predefined (uses packet switches). For RT_CLASS 1

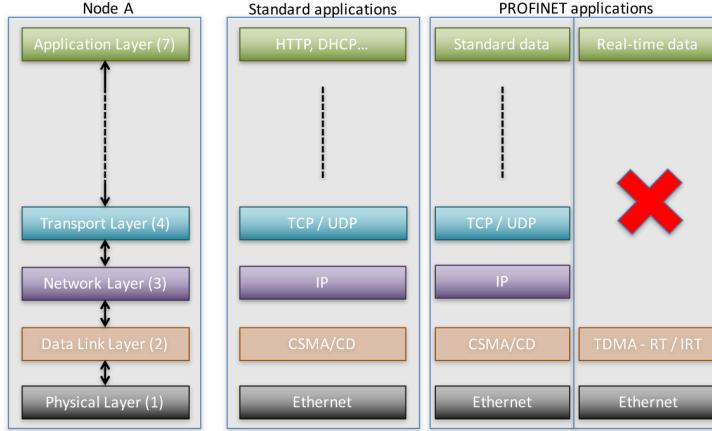


Figure 6.9: The implementation of the OSI model for Profinet.

(real-time communications), which is the only mandatory one, access to the physical medium is regulated by the priority assigned to the frames, according to IEEE 802.1Q.

To ensure a quick execution of the communication stack and in the spirit of fieldbuses, profinet implements just the Application, Data-Link and Physical layers, as depicted in Figure 6.9. The delay and jitter statistical characterisation for Profinet are depicted in Figure 6.10.

Similarly to other RTE networks defined by the IEC 61784 standard (such as PROFIBUS-FMS), the Application layer protocol of Profinet is based on the definition of communication objects that can be exchanged over the network via communication relationships established between IO controllers and IO devices, i.e. function blocks as defined in Profinet CBA. Profinet implements the PROFIsafe feature for hazardous areas.

6.2 Profinet CBA

A Profinet CBA (CBA = Component Based Automation) system consists of various automation components, comprising the mechanical, electrical and communication capabilities. A component can be generated using the standard programming tools, and it is usually described by means of the Profinet Component Description (PCD) file in XML. A planning tool loads these descriptions and enables the logical interconnections between the individual components to be generated for implementing a plant (see Section 4.2.3 for further details). The main element of PROFINET CBA is a component, an element having control functionalities and communicating with other components through its interface. In the PROFINET environment, each com-

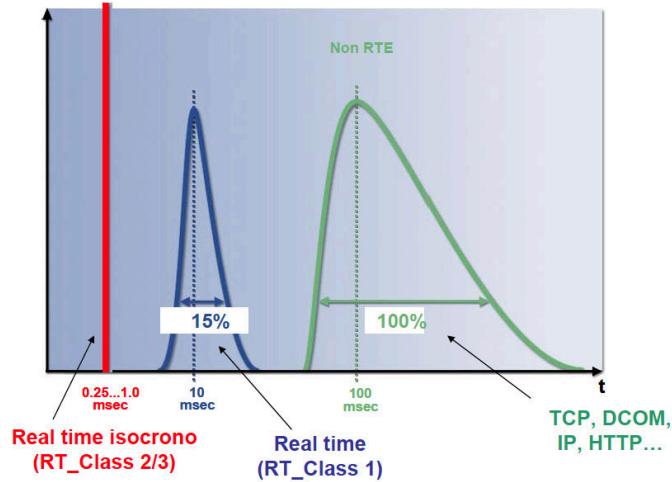


Figure 6.10: Performance achievable on the different channels (courtesy of Paolo Ferrari, Università di Brescia).

IEC 61499-1	PROFINET CBA
System	System
Device	Physical device
Resource	Logic device
Function block	Object/Control function
Application	Application
Conjunction	Connection

Table 6.1: Basic expressions in IEC 61499-1 and PROFINET CBA.

ponent equals the control device: the PROFINET CBA enables the creation of component interfaces and relations between them. Individual components are programmed with tools supplied by device manufacturers.

Distributed control systems in accordance with IEC 61499-1 standard have a hierarchical structure and are not based on components. Basic terms are "System", "Device", "Resource", "Application" and "Function block". Specification of PROFINET CBA is largely in line with the model described in IEC 61449-1, as reported in Table 6.1 and depicted for comparison in Figure 6.11.

6.3 EtherCAT

Beckhoff, a German automation company, developed a fieldbus system called Fast Lightbus to correct the low bandwidth utilisation problem present in

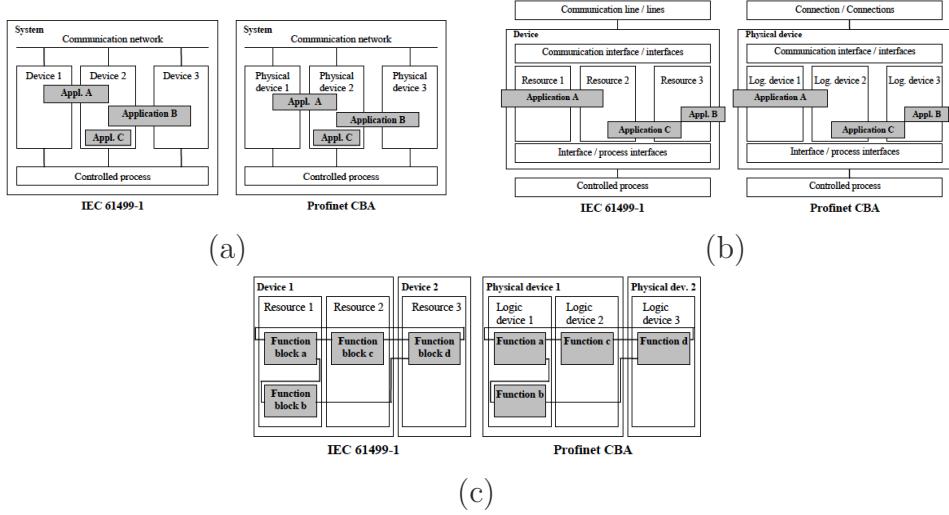


Figure 6.11: (a) System model according to IEC 61499-1 and PROFINET CBA [9]. (b) Device model according IEC 61499-1 and PROFINET CBA [9]. (c) Resource model according to IEC 61499-1 and PROFINET CBA [9].

other Ethernet protocols. This protocol led to Ethernet for Control Automation Technology (EtherCAT) which Beckhoff released in 2003. In 2004, Beckhoff helped to create a new group to promote the EtherCAT protocol. Their efforts led to the EtherCAT Technology Group (ETG). Beckhoff donated the rights to EtherCAT to the ETG. The ETG works with the International Electrotechnical Commission (IEC), specifically serving as a liaison for the digital communications working group. The partnership has led to standardisation throughout the EtherCAT protocol's history.

EtherCAT requires full-duplex on copper or fiber optic cables. It supports any network topology, including bus. It is based on master-slave and interoperates with TCP/IP or Profinet. The EtherCAT master processes the RT traffic via dedicated hardware and software. As noted previously, typical automation networks are characterised by short data length per node, typically less than the minimum payload of an Ethernet frame. Using one frame per node per cycle therefore leads to low bandwidth utilisation and thus to poor overall network performance. The idea is to use the same frame to transmit multiple messages. To overcome this limit, EtherCAT uses the processing on the fly approach: the Ethernet frame is no longer received, interpreted and copied as process data at every node, but rather uses the idea of *broadcasted telegram*, depicted in Figure 6.12. To this end, the EtherCAT slave devices read the data addressed to them while the telegram passes through the device. Similarly, input data are inserted while the tele-

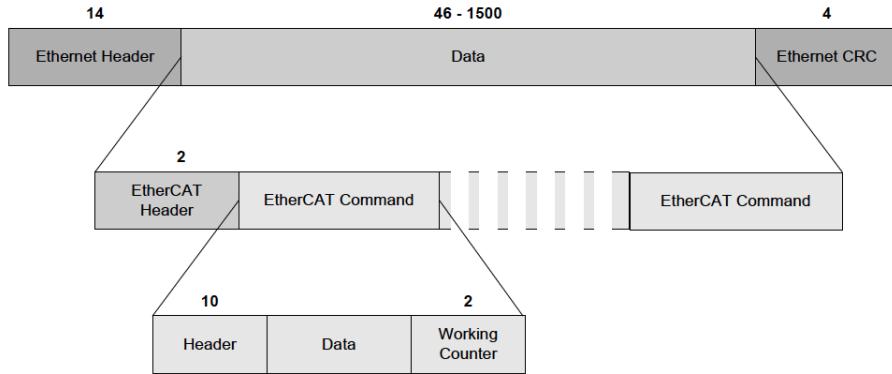


Figure 6.12: EtherCAT frame encapsulation [15].

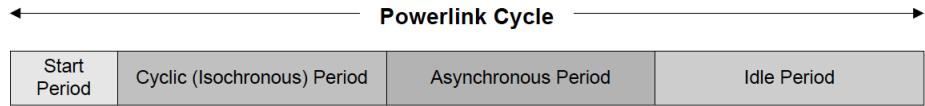


Figure 6.13: Powerlink cycle [15].

gram passes through. Both operations are computed in hardware, hence a delay of nanoseconds is generally introduced. Many nodes - typically the entire network - can be addressed with just one frame.

The EtherCAT uses special devices and also optionally adopts the E-bus, which is an EtherCAT physical layer for Ethernet offering Low Voltage Differential Signal (LVDS) scheme. EtherCAT is a fast RTE solution, offering determinism if not used with intermediate switches or routers between master and slave.

6.4 Ethernet Powerlink (EPL)

The Ethernet Powerlink (EPL) is a hard-RT protocol based on Ethernet. EPL devices use standard Ethernet devices without special hardware. EPL uses cyclic communication with TDMA over a cycle time of $200 \mu\text{s}$ and jitter $\leq 1\mu\text{s}$. It adopts a master-slave model: only one master schedules all transmissions for the network using a fixed time cycle (Figure 6.13). The first message of each cycle is the Start Period, during which the master broadcasts the Start-Of-Cycle (SOC) message to synchronise the slaves. This is the only time-based frame, since all the other are event-based. During the Cyclic Isochronous Period, the master polls all the slaves, that in case respond

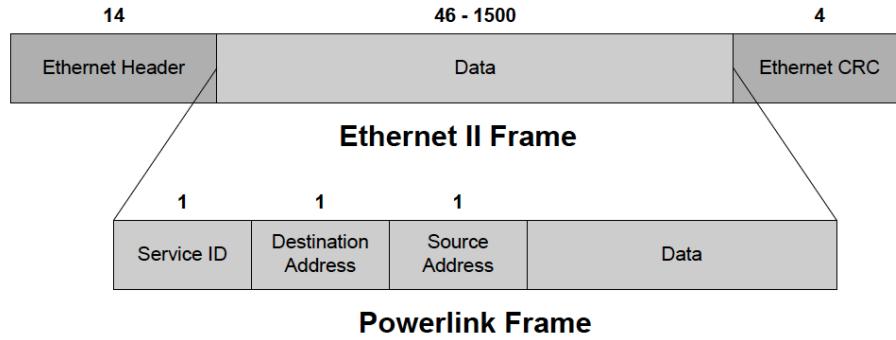


Figure 6.14: Powerlink frame encapsulation [15].

(Figure 6.13). At the end of this phase, the End-Of-Cycle (EOC) message is broadcasted by the master to notify all the devices that the cycle traffic has terminated. In the Asynchronous Period (Figure 6.13), non-cyclic data transfer is available under master control and standard IP packets can be optionally sent. The typical Powerlink frame is reported in Figure 6.14.

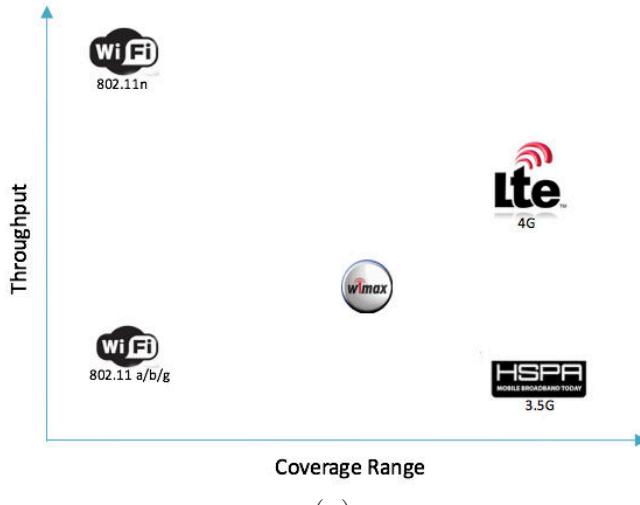
The message cyclic management is close to the one seen in Profinet. However, EPL does not employ switches to avoid collisions or to provide network synchronisation. In fact, the adoption of switches is strongly discouraged to increase determinism. Instead, Powerlink entirely rely on hubs, suggesting the adoption of one hub per slave. Up to ten hubs can be cascaded. Unlike Profinet, EPL RT-devices cannot coexist with non RT traffic, even though EPL devices can act as non RT devices.

Chapter 7

Wireless in the Industrial Domain

Thanks to their peculiar features, *wireless networks* are currently becoming more and more attractive for use in industrial automation and manufacturing scenarios. However, they cannot be thought as a total replacement for more traditional wired networks in such environments, at least at the moment. This means that solutions exist integrating the existing industrial wired solutions and wireless solutions, in order to achieve enhanced flexibility, efficiency, and performance for the overall networked system. In a wireless network, the network access is on a non-guided media (aka theterless channel). Cellular Network is a global network where the coverage is obtained with a set of cells, i.e. adjacent or overlapping areas. The mobile terminal can move from one cell to the other keeping the communication seamlessly active. The protocol of wireless connections is a subset of IEEE 802, in particular 802.11 and 802.15, hence its typical OSI model implementation is reported in Figure 6.1. The different solutions available for wireless connection as a function of the coverage range (i.e. the distance covered) and the throughput (i.e. the data rate) are reported in Figure 7.1.

In a wireless connection, transmission happens approximately at the speed of light, which is $c \approx 299792458$ m/s, while f [Hz] is the frequency of the transmission, which leads to the definition of the wave length $\lambda = c/f$. The relation between frequency and wavelength is reported in Table 7.1 A complete description and namin convention of the wireless transmission is reported in Figure 7.2. The VHF/UHF ranges are used for mobile radio: simple, small antenna for cars; deterministic propagation characteristics, reliable connections. SHF and higher for directed radio links and satellite communication: small antenna, large bandwidth available. Wireless LANs use frequencies in UHF to SHF range, even though some systems planned



(a)

Standard	Family	Downlink (Mbps)	Uplink (Mbps)	Coverage
WiFi	802.11	11/54/150/300		100m
WiMAX	802.16e	144	35	10km
UMTS (3G) /HSPA (3.5G)	3GPP	14.4	5.76	30km
LTE (4G)	3GPP	360	80	30km

(b)

Figure 7.1: (a) Different wireless solutions in IEEE 802. (b) Different features for wireless solutions.

up to EHF. Limitations due to absorption by water and oxygen molecules (resonance frequencies), hence weather dependent fading, signal loss caused by heavy rainfall etc.

Propagation in free space happens in line-of-sight, i.e. following a straight line from a source to a destination. The received power is influenced by several environmental characteristics, i.e. fading (frequency dependent), shadowing, reflection at large obstacles, refraction depending on the density of a medium, scattering at small obstacles and diffraction at edges. A pictorial description of the most relevant of those effects is reported in Figure 7.3. The net effect of such nuisances is the presence of multi-paths: the receiver receives multiple replica of the same message (see Figure 7.4), thus generating

System	Frequency	Wavelength
FM radio	100 MHz	3 m
Cellular	800 MHz	37.5 cm
Ka band satellite	20 GHz	15 mm
Ultraviolet light	10^{15} Hz	10^{-7} m

Table 7.1: Relation between the transmission speed and the wave length.

Classification Band	Initials	Frequency Range	Characteristics
Extremely low	ELF	< 300 Hz	Ground wave
Infra low	ILF	300 Hz - 3 kHz	
Very low	VLF	3 kHz - 30 kHz	
Low	LF	30 kHz - 300 kHz	
Medium	MF	300 kHz - 3 MHz	Ground/Sky wave
High	HF	3 MHz - 30 MHz	Sky wave
Very high	VHF	30 MHz - 300 MHz	Space wave
Ultra high	UHF	300 MHz - 3 GHz	
Super high	SHF	3 GHz - 30 GHz	
Extremely high	EHF	30 GHz - 300 GHz	
Tremendously high	THF	300 GHz - 3000 GHz	

Figure 7.2: Radio frequency bands (courtesy of John Hopkins University, Computer Science Department).

possible loss of packets due to interference and make direct measurements on distance practically very noisy.

7.1 WiFi: IEEE 802.11

The IEEE 802.11 family includes several standard specifications for wireless LAN, short WLAN. All devices belonging to such a family (also referred to as WiFi, i.e. Wireless fidelity) work in specific bands of the radio spectrum, centered around 2.54 GHz (for 802.11, 802.11b, 802.11g) and 5 GHz (802.11a). Very high data rates are foreseen in the 802.11 standard, ranging from 11 Mb/s for IEEE 802.11b to a (nominal) 54 Mb/s for IEEE 802.11g/a.

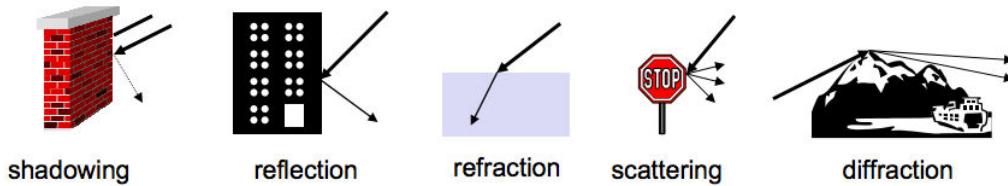


Figure 7.3: The effect of obstacles in wireless transmission (courtesy of John Hopkins University, Computer Science Department).

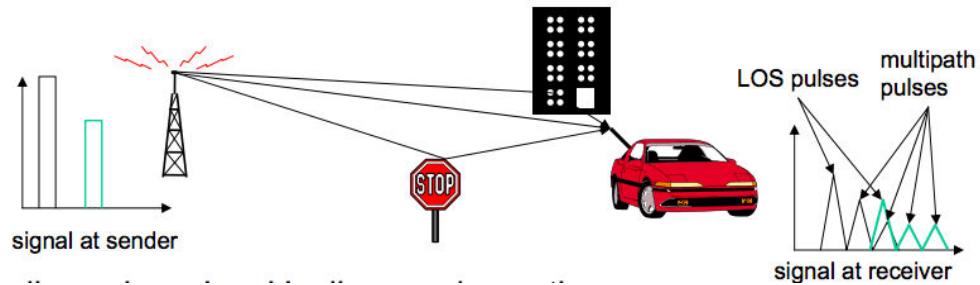


Figure 7.4: Multipath (courtesy of John Hopkins University, Computer Science Department).

Moreover, IEEE 802.11n, whose standard has been released in 2009, may reach up to 300 Mb/s, acting in dual band (both 2.54 and 5 GHz).

WiFi: Opportunistically use WiFi hotspots once they are available, thus generating different network topology:

- Point-to-Multipoint (Access Point);
- Point-to-Point (Ad hoc);
- Multipoint-to-Multipoint (Mesh Network).

Connection with almost any WiFi CERTIFIED device is possible, while the protocol is conceived for maximum flexibility, i.e. designed for portable and stationary devices.

The Basic Service Set (BSS) defines the set of nodes using the same coordination function to access the channel. Two BSS configuration modes:

- Ad hoc mode: stations can dynamically form a network without AP and communicate directly with each other (computer or laptop LANs, called Independent BSS - IBSS). Managed by the IETF MANET (Mobile Ad hoc Networks) working group;

- With infrastructure, i.e. the BSS is connected to a fixed infrastructure through a centralised controller, that is the AP.

The Basic Service Area (BSA) is the spatial area covered by a BSS, i.e. a WLAN. Interconnections among several BSSs is possible at the MAC sub-layer. Usually, there is a network of APs connected by a backbone, e.g. Ethernet. In general, to join a BSS three operations are needed: Scanning, Authentication (open system authentication, shared key authentication - WEP, per session authentication - WPA2) and Association (information exchange about the AP and node capabilities and roaming). Two different approaches: with APs, in which both authentication and association are necessary for joining a BSS; using an IBSS, where no authentication nor association procedures are required for joining an IBSS.

To “get in touch” with an AP, two different modalities exist. In passive scanning, the newcomer search for a Beacon frame, that is a particular frame sent by the AP every 100 ms. Instead, in active scanning, the newcomer sends a Probe Request frame on a given channel: all the receiving AP’s reply with a Probe Response frame.

Mac Sublayer

The IEEE 802.11 standard specifies two ways to access to the physical medium:

- Distributed Coordination Function (DCF): this is mandatory;
- Point Coordination Function (PCF): this is non mandatory. It is a centralized MAC protocol coordinated by a Point Coordinator which grants exclusive access to any wireless station, thus preventing collisions (very similar to the TDMA).

Although PCF is not currently supported by the vast majority of commercial WiFi boards, it nevertheless represents an interesting option for industrial applications. Basically, the DCF ensures an asynchronous data transfer for best-effort traffic. It is based on CSMA/CA (CA stand for Collision Avoidance), which is based on carrier sense and a back off algorithm (as in Ethernet). The main difference is that it is very difficult to detect a collision while transmitting, hence an ACK from the AP is expected: if it does not arrive, a collision took place. Instead, PCF is conceived for data transfer for real-time traffic. In this case it can also use the Request to Send/Clear to Send (RTS/CTS) handshake with the AP to ask for permissions to transmit.

The WiFi system is semi-synchronous: the coordination is maintained through Beacon frames (sent by the AP). The time is counted in intervals, called slots, i.e. the system unit time. The slot duration depends on the implementation of the Physical layer, e.g. it is 20 ms for 802.11b, 9 ms for 802.11a/h/g/n/ac. The Inter-frame space (IFS) is the time interval between frame transmissions and it is used to establish priority in accessing the channel. Devices complying with the IEEE 802.11e standard also support the concept of QoS by means of traffic prioritization/parameterization. As already noted, this feature makes these devices particularly suitable for industrial applications, since it may allow faster delivery of critical messages. This standard defines an additional Hybrid Coordination Function (HCF), which ensures contention-free access to the wireless medium by allocating stations Transmission Opportunities (TXOPs).

7.2 WiFi: IEEE 802.15.4

The IEEE 802.15.4 family deals with ad-hoc wireless interconnections of electronic devices within a limited transmission range (some tens of meters) at low data rates (up to 250 kb/s). It is a general purpose standard, whose applications can be found in home and building automation, simple cable replacement, smart tags, automotive sensing, etc. Devices of the IEEE 802.15.4 family typically operate in the band centered around 2.4 GHz .

Physical layer

The Physical layer (PHY) manages the physical Radio Frequency (RF) transceiver and performs channel selection and energy and signal management functions. It works in the range between about 800 MHz and 2.5 GHz . The standard IEEE 802.15.4e adopts channel hopping strategy to improve support for the industrial markets. This way, it increases robustness against external interference and persistent multi-path fading. Networks can be built as either peer-to-peer or star networks.

Data-link layer

Two types of MAC protocols are encompassed by the standard:

- Beacon-enabled: it is the most common approach and it is based on a network coordinator that periodically sends beacons.

- Non-beacon: a pure CSMA/CA protocol is adopted for channel access control. In this way, a totally decentralized and asynchronous MAC is obtained.

The beacon-enabled has a network coordinator that periodically issues superframes that are divided in two parts: Contention Access Period (CAP), taking place immediately after the beacon and is based on a distributed CSMA/CA; Contention-Free Period (CFP), in which the superframe is optional and it basically guarantees time slots that are exclusively allocated to the nodes.

A superframe is delimited by two beacons. The superframe is divided into time slots in which the device can transmit (it basically follows a TDM approach). The IEEE 802.15 standard does not exchange standard Ethernet frames. The physical frame-format is specified in IEEE802.15.4 and specifies that the Physical layer PHYs only support frames of up to 127 bytes. Adaptation layer protocols, such as 6LoWPAN, provide fragmentation schemes to support larger network layer packets.

7.3 Common Industrial Protocol

The Common Industrial Protocol (CIP), formerly known as the Control and Information Protocol, enables a high degree of interoperability among the different networks. The Common Industrial Protocol (CIPTM) encompasses a comprehensive suite of messages and services for the collection of manufacturing automation applications, which are: control, safety, synchronization, motion, configuration and information. Supported by many vendors, CIP provides users with a unified communication architecture throughout the manufacturing enterprise, which is media-independent.

The CIP technology is based on the CIP Object Model. CIP is strictly object-oriented. Each object is defined in terms of attributes, services, connections and behaviors. It has an object library to support network communication, network services (e.g., file transfer) and automation functions (e.g., I/O operations, HMI, motion control, feedback, etc.). It is based on a producer-consumer architecture that allows efficient use of network bandwidth. An example of the protocol stack of CIP is reported in Figure 7.5.

7.3.1 Hybrid networks

Using the CIP idea, it is possible to interconnect different types of networks, as exemplified in Figure 7.6 for a bridge or a gateway interconnection.

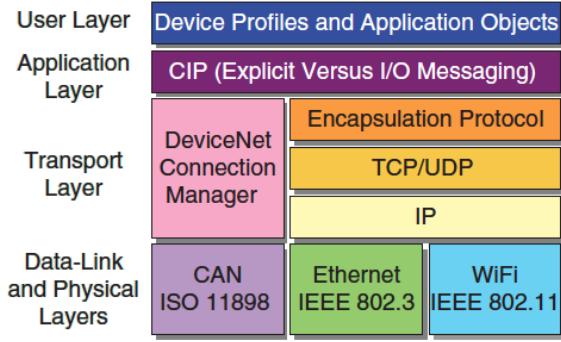


Figure 7.5: Protocol stack of CIP-based networks [16].

Although technically feasible, wireless extensions of industrial networks to be used in place of fieldbuses or RTE networks are not so straightforward. This is due by three main issues. First problem: the wireless medium is shared among all the nodes. Even operating at the maximum allowable speed (e.g., 54 Mb/s for currently available 802.11a/g/e networks), this may result in a low throughput compared to that of wired networks when the number of connected devices grows higher. Related to this problem, wireless connection requires the non negligible overheads to perform protocol control and for acknowledgement and reservation mechanisms. For example, transmit 8 bytes over a standard WLAN takes about 100 μ s, slightly shorter than a 1 Mb/s CAN bus. Additionally, there is no provision for full-duplex operations. To partially solve this problem, it is possible to either make use of more performant WLAN protocols, such as IEEE 802.11n, or make use of a set of small sub-networks, interconnected by means of a wired backbone in order to limit the packet rate on each of them.

The second problem is related to the random access techniques (e.g., CSMA/CA) employed by wireless networks. As already mentioned, this choice implies unpredictable (and unbounded) transmission delays and, even worse, network congestions could be experienced. In the presence of network congestions, the network may become temporarily unavailable to carry out timely data exchanges, which is not compatible with proper operation of distributed control systems. Finally, non negligible jitters may affect the transmission even in the case of lightly loaded networks, unless some form of prioritization scheme is suitably adopted. To alleviate this problem, it is possible to adopt TDMA based solution with either a central scheduler, such as the PCF, or as in IEEE 802.15.4, otherwise it is possible to adopt new prioritization features offered by the IEEE 802.11e standard, even though

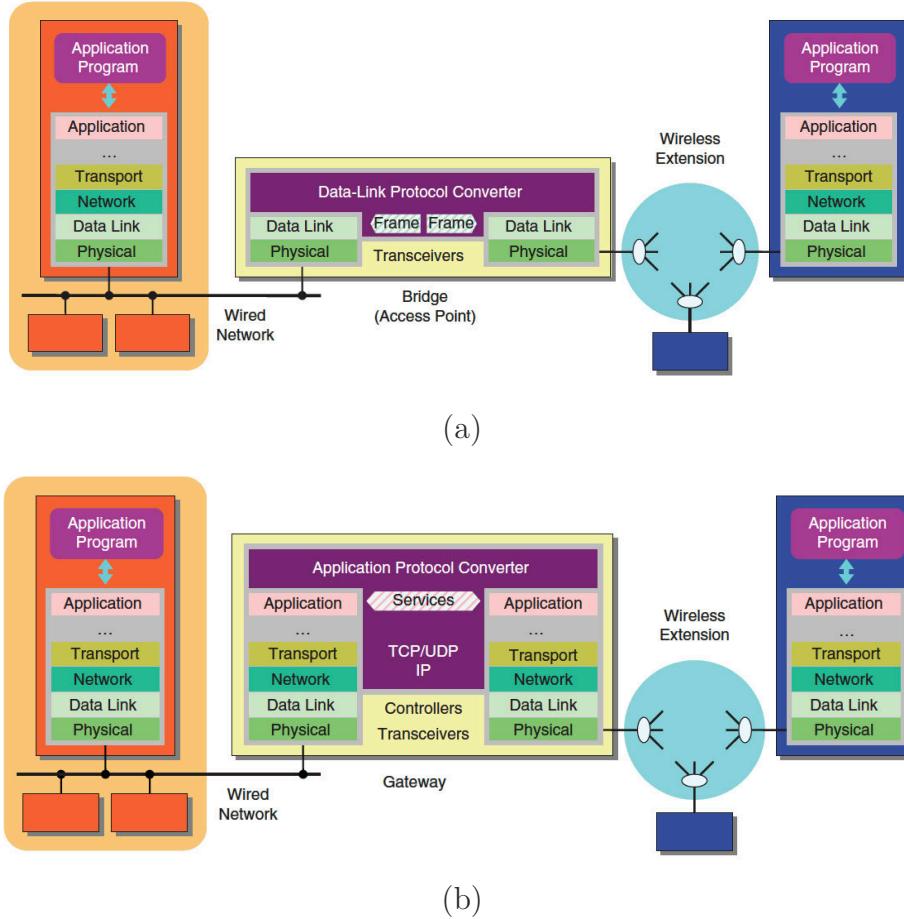


Figure 7.6: Interconnections via a bridge (a) or a gateway (b) [16].

strict determinism cannot be reached.

Finally, the third problem relates to wireless channels that are much more error prone than wired cabling, especially with high electromagnetic interference as in industrial environments. This problem generates communication latencies and jitters. Moreover they affect the network reliability and, consequently, the robustness of the overall system. There is a non negligible chance that messages sent over the air never reach the intended destination (e.g., consistency problems in multicast messages). In this case, the work-around is to use multiple antennas to increase message delivery reliability or adopt the 5 GHz band for communication, thus reducing interferences with other mobile phones, Bluetooth enabled equipment, notebooks with wireless connections.

Additional problems are related to the payload which in fieldbuses is

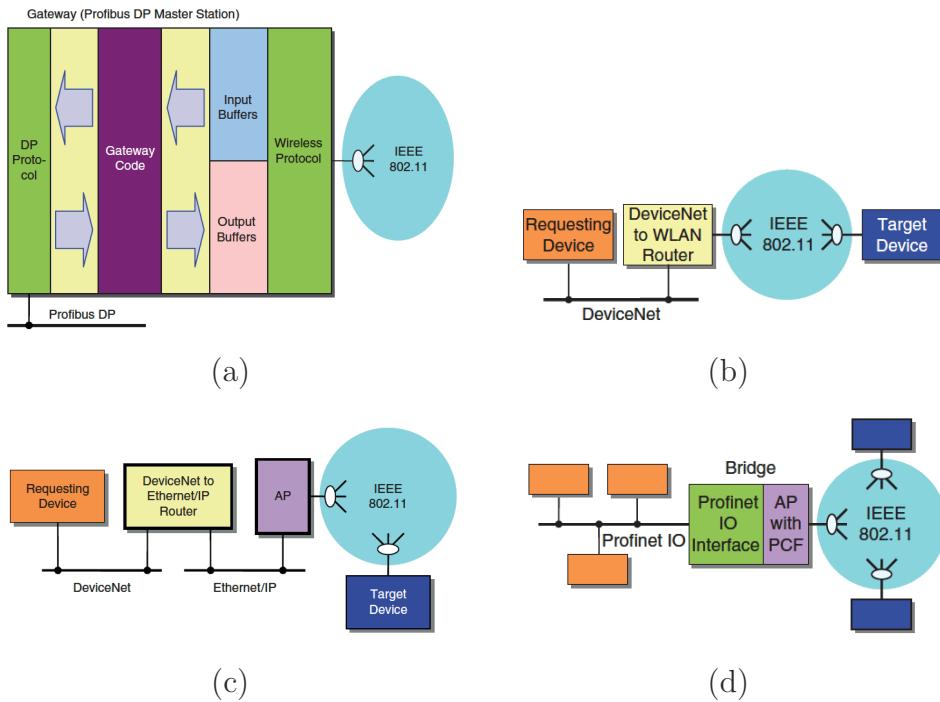


Figure 7.7: Gateway with proxy functionality for integration of 802.11 with PROFIBUS-DP (a), with DeviceNet via a WLAN-enabled CIP router (b), with DeviceNet via a CIP router and an access point (c), with a Profinet IO via a bridge (d) [16].

typically very small. Another problem is the addressing scheme, which is simplified in fieldbuses.

As an example, Figure 7.7 reports same examples of integration of the wireless domain with fieldbuses.

Chapter 8

PLC Standards

When creating the distributed control systems, two possibilities are available from the International Electrotechnical Commission (IEC):

- IEC 61131: Used for a long time in the PLC sphere, but it does not directly support the creation of extensive distributed systems;
- IEC 61499: Offers a direct solution for distributed control system creation.

While IEC 61499 is covered in Chapter 3, this chapter considers explicitly IEC 61131, which focuses on PLCs. It has to be noted that the two mentioned standards are close to each other, even though they are covering different aspect of industrial automation, as depicted in Figure 8.1. While IEC 61499 was developed especially as a methodology for distributed control system modelling, the standard IEC 61131 is not conceived for large automation systems. The IEC 61131 deals with the software development description Programmable Logic Controllers (PLCs). The standard comprises eight different parts:

- IEC 61131-1 Basic information: Basic terminology and terms are defined in this part;
- IEC 61131-2 Hardware requirements and its testing: Electronic and mechanical structure and verification tests;
- IEC 61131-3 Programming languages: Structure of PLC software, languages and program performance;
- IEC 61131-4 User instructions: Instructions for selecting, installation and maintenance of programmable controllers;

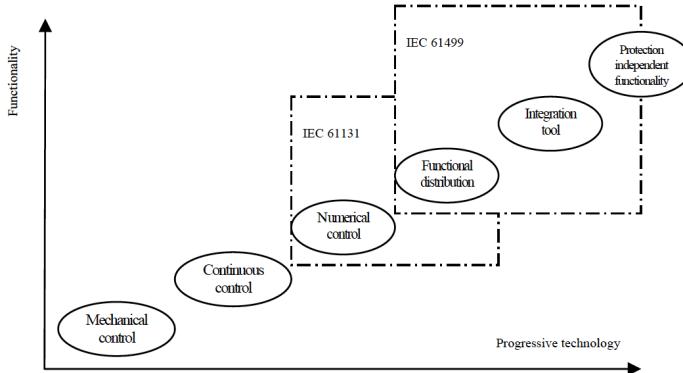


Figure 8.1: Technology development for industrial control [9].

- IEC 61131-5 Message specification: Software resources for communication between devices based on MAP (Manufacturing Messaging Services);
- IEC 61131-6 Communication through industrial networks: PLC software resources for communication using IEC Fieldbus;
- IEC 61131-7 Fuzzy control programming: Software resources including standard; function blocks for operation with fuzzy logic in PLC;
- IEC 61131-8 Directives for languages implementation used for PLC: Application and implementation directives for IEC 1131-3 languages.

In these book, we will give a description of the most interesting part of the standard, which is IEC 61131 part 3 - Programming languages. Therefore, we will investigate the structure of PLC software, languages and program performance.

Notice that some solutions exist that tries to extend the application of the IEC 61499 to the world of traditional PLCs. The extension comes with existing communication standards for industrial communication. In particular, we know that PROFINET is an open standard for automation based on industrial Ethernet. PROFINET CBA is a part of this standard and is based on IEC 61499-1: describes technologies for modular and distributed system implementation on the base of pre-defined components.

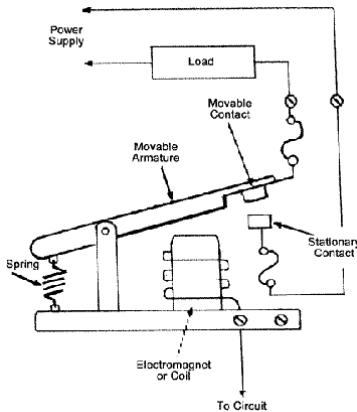


Figure 8.2: Electric scheme of a relay (courtesy of Galco Industrial Electronics).

8.1 Relay Logic

Originally developed in the late 60's to serve the automation needs of the automobile industry in USA, PLCs have grown much beyond this sector and today it is difficult to name an industry segment that does not use a PLC. The initial purpose was to replace hardwired relay based interlocking circuits by a more flexible device.

Definition 27. *Relay logic is a method for controlling industrial electronic circuits by using relays and contacts.*

Definition 28. *A relay is an electrically operated switch.*

In practice, a relay consists of a coil of wire wrapped around a soft iron core, a movable iron armature and one or more sets of contacts 8.2. When an electric current is passed through the coil it generates a magnetic field that activates the armature, and the consequent movement of the movable contact(s). When the current to the coil is switched off, the armature is returned by a force to its relaxed position. The force is usually provided by a spring, but in cases, as for electric industrial motors, the gravitational forces can be used.

The schematic diagrams for relay logic circuits are often called line diagrams, because the inputs and outputs are essentially drawn in a series of lines. A relay logic circuit is an electrical network consisting of lines, or rungs, in which each line must have continuity to enable the output device, as depicted in Figure 8.3-a. A typical circuit consists of a number of rungs,

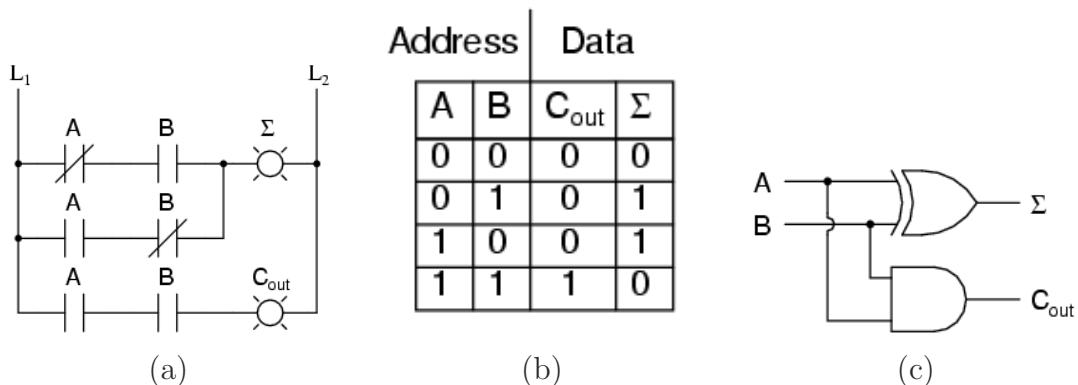


Figure 8.3: Graphical representation of the relay logic (a) representing the Half adder's truth table (b) and the logic circuit implementation (c).

with each rung controlling an output. This output is controlled by a combination of input or output conditions (IF-THEN statements), such as input switches and control relays. The conditions that represent the inputs are connected in series (logic AND), parallel (logic OR) or a combination thereof (see Figure 8.3-b,c). The relay logic circuit forms an electrical schematic diagram for the control of input and output devices and represents the physical interconnection of devices.

The basic format for relay logic diagrams is as follows:

- The two vertical lines that connect all devices on the relay logic diagram are labeled L₁ and L₂. The space between L₁ and L₂ represents the voltage of the control circuit.
- Output devices are always connected to L₂.
- Control devices are always shown between L₁ and the output device. Control devices may be connected either in series or in parallel with each other.
- Electrical devices are shown in their normal conditions (i.e., normally closed or normally open).
- All contacts associated with a device will change state when the device is energised.

8.2 PLC

With respect to the relay logic, PLCs add flexibility. Flexibility in PLCs comes through the programmability of the device, which made it possible to use the same basic hardware for any application as well as the ability to quickly change the program and modify the behaviour of a circuit. Originally the PLCs had inputs in the form of contacts (called digital inputs), a simple processor and a software to control its execution, and outputs in the form of contacts, referred as a digital outputs (or sometimes as voltages to drive external relays).

Another important point is how the complexity is handled by the software program. Indeed, the dimension of the PLC depends on the number of I/O. There exists micro PLC in which the number of I/O is less than 100, while there are large PLC, where that numbers is greater than 1000. Higher PLC dimensions imply higher complexity to be properly handled by the built in software. These needs have generated a plethora of possible programming languages. From this picture, it was immediately clear that the future growth of PLC and their widespread use in industry would not be possible unless the fundamental issue of program portability was addressed. In practice, this is the same technology pull we have seen for all the other standards: portability of code and interoperability. IEC 61131-3 is a result of this effort and has been evolved on a consensual basis by largely adopting the prevalent programming practices of major manufacturers in the PLC industry.

8.2.1 Ladder Diagram

The evolution of the simple connections reported in Figure 8.3 is shown in Figure 8.4, where additional functionalities are embedded in the system.

The Ladder Diagram (LAD, Figure 8.4) is the first graphical language for programming industrial control systems described in IEC-61131-3. Ladder Diagrams are graphical representations of Boolean expressions, but they can also include other information normally not possible with Boolean expressions. The diagram is used to describe the behaviour of: Functions, Function Blocks, Programs, Sequential Flow Charts. The PLC executes the program 1 rung at a time, starting with the first rung and then working down. Each individual rung is executed from the left to the right. The outputs at the right side of each rung is set to a Boolean condition that reflects the status of the permissive contacts in a particular rung.

Examples of possible LAD rungs are reported graphically in Figure 8.5. The rung in Figure 8.5-a is read as: “If the Start Button is on, turn the Motor on. If the Start Button is off, then turn the Motor off”. In this example, if

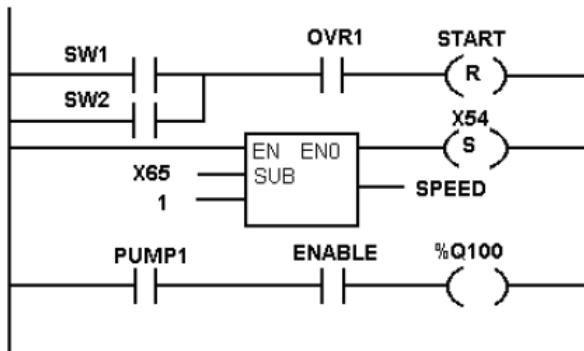


Figure 8.4: Ladder Diagram (LAD): graphic language based on relay logic, technique used for programming in present generation of PLC controllers [9].

the actual Start Button is on, then the value of all the eXamine If Closed (XIC) named “Start Button”, in the program will be true (also known as a “one”, or “closed”). The second instruction is known as an Output Energise (OTE) and it is an output instruction: it turns on if the logic to the left of the OTE is true. The text above the XIC and OTE is the address associated with the instruction. PLC addresses may appear in many different ways depending on the PLC being used (e.g. Start_Button, Local:2:I.Data[0].1, I:020/2, etc.) and are used by the PLC to determine exactly which input to read or which output to command.

I/O addresses are a means to tie a physical I/O point to a location in PLC memory. There are other addresses that do not connect to physical I/O, but are used to hold a value. Data addresses store a value used for functions like timers, counters, or calculations. Other important components are latches, storing the results and keep it on or off irrespective of the value of the input rung (Figure 8.5-b,c).

It is also possible to include functions as well as function blocks within Ladder Diagram networks provided the following basic conditions are fulfilled: the Functions/Function Blocks should have Boolean inputs and outputs; inputs can be directly driven from ladder rungs; outputs can drive coils/motors. Note how similar this diagram is to the conventional circuit diagram and how easy it is to follow its action. However, this programming approach has a number of limitations in the context of the modern PLC, such as lack of software structure (no sub-programs supported); problems in reusability (the same functionality available by copying the entire circuit); poor data structure definition (the same variable used in different places needs a copy); lack of support for describing sequential operations (a sequence of initial step - transition - operations - transition difficult to be

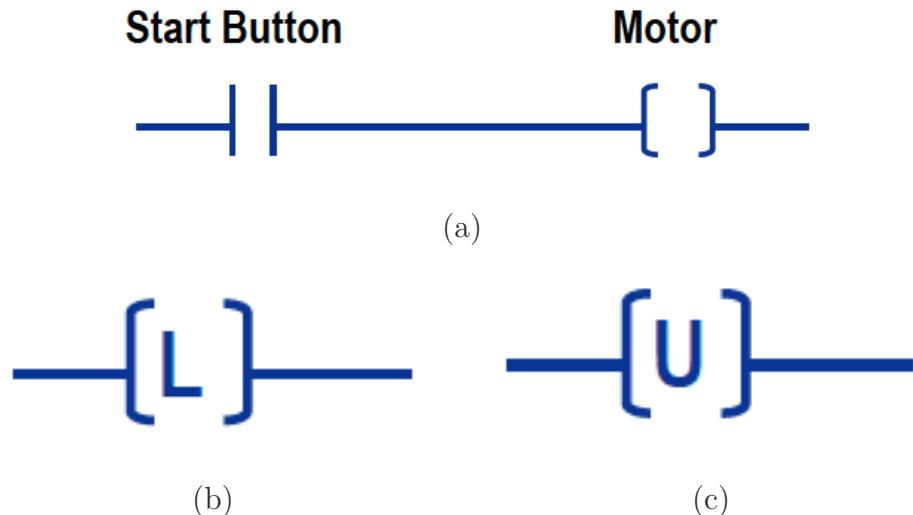


Figure 8.5: (a) A very simple example of ladder diagram. (b) OTL - Output Latch - turns on the output and keeps it even if the rung goes false. (c) OTU - Output Unlatch - turns off the output when the rung is true. (courtesy of Dr. Sunil Jha, Department of Mechanical Engineering, Indian Institute of Technology, Delhi).

modelled); difficulty in handling complex arithmetical operations. Another problem is the execution control. PLCs execute a program in a cyclic order, being the logical flow for a Ladder Program as:

- All input values are read via the I/O modules and the status stored in PLC memory;
- Next the ladder rungs are executed starting from the top and proceeding downwards till all rungs are covered;
- The output values are stored in the PLC memory as each rung is executed;
- Finally all the output values held in the memory are written to the physical outputs in a single operation.

The time of executing the entire program thus depends on the length of the program and the complexity of the logic, hence it is hard to ensure determinism of execution. One way to recover determinism is to divide the program into different sections and manage the execution in such a way that

critical sections are scanned at faster intervals. This kind of control is generally difficult to implement in the ladder diagram approach. For example, this may be the case of code implementing a PID with fast loop closure.

8.3 Software Issues

To overcome the previously highlighted limitations, the standard IEC 61131-3 defines a set of software structures. The software model suggested in IEC 61131-3 contains a number of elements arranged in a hierarchical order. To account for this additional requirements, the following definitions are needed.

Definition 29 (Configuration). *A language element corresponding to a programmable control system as defined in IEC 61131-1.*

Definition 30 (Resource). *A language element corresponding to signal processing function, man machine interface and sensor and actuator interface functions.*

A configuration can have one or more resources and a program can work only when it is loaded into a resource. A resource can exist in any device that includes a PLC.

Definition 31 (Task). *An execution control element providing for periodic or triggered execution of a group of associated program organisation units.*

A resource executes multiple programs. It does so using software elements called tasks. A task can invoke one or more programs or function blocks to be executed at a certain periodicity or by triggering them when a set of conditions is fulfilled. The programs or function blocks are executed once commanded to do so by the task.

Definition 32 (Program Organisation Unit [POU]). *A function, a function block or a program.*

Definition 33 (Function). *A POU which, when executed, yields exactly one data element (which may be multi valued, e.g., an array or structure) and whose invocation can be used in textual languages as an operand in an expression.*

Definition 34 (Function Block). *An instance of Function Block Type.*

Definition 35 (Function Block Type). *A PLC programming language element consisting of: a) Definition of a data structure partitioned into input, output and internal variables; b) Set of operations to be performed upon the elements of a data structure when an instance of the function block type is invoked.*

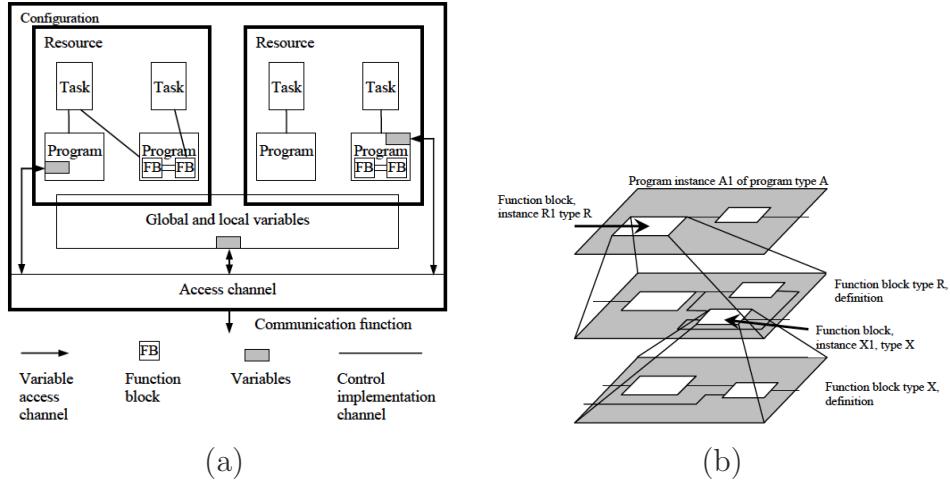


Figure 8.6: (a) IEC 61131-3 software model and (b) function block hierarchical structure [9].

Definition 36 (Program). *A program consists of a logical assembly of all programming elements and softwares necessary for the intended signal processing actions required for the control of a machine or a process by a programmable control system.*

A program contains instructions to be executed by the PLC hardware in a predetermined sequence. The instructions can be in the form of function blocks interconnected together to form a sequence of actions. Programs are initiated and controlled by tasks. A program has to be assigned to a specific task and the task must be configured to execute periodically or triggered upon certain conditions. Based on these definitions, the software model is depicted in Figure 8.6-a, while the function block hierarchical design is reported in Figure 8.6-b.

Though this is not formally defined in the standard, the concept of application is still an essential feature of the software of a large PLC system.

Definition 37 (Application). *An application is a solution for the control of a unit of plant.*

Application (Figure 8.7) must contain all control requirements for activation, control and finish. Control is necessary for physical sensors and actuator handling, event scheduling, and interaction with operating terminals. Because the standard allows the resources to be activated independently, large-scale PLC system can handle multiple independent applications simultaneously.

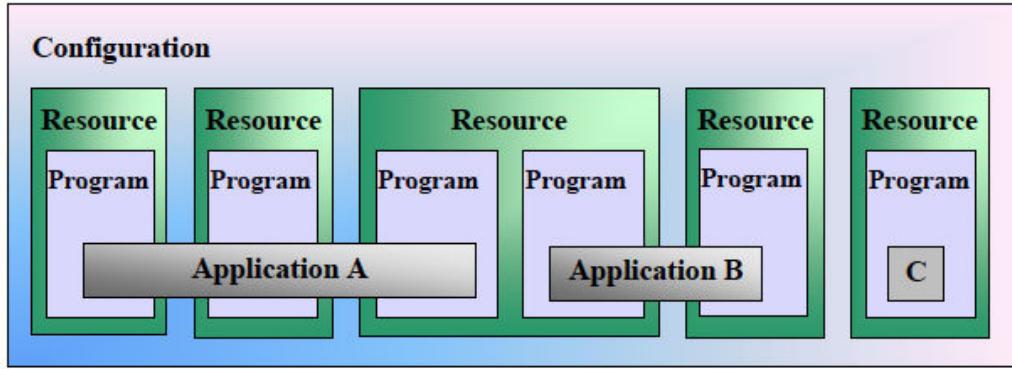


Figure 8.7: Applications: resource assignments [9].

8.3.1 Preemption

One of the main requirements in a large process control system is the ability to execute different parts of the control program at different rates depending on the process requirements. Tasks achieve this type of control by triggering programs and function blocks at specified time periods. When multiple tasks are declared within a resource, they need to be scheduled. The scheduler decides the exact moment that a task has to execute. Since two tasks cannot run concurrently, some form of arbitration is needed to resolve the sequence in which two waiting tasks have to be taken up.

There are two ways in which a PLC schedules tasks: by non-preemptive scheduling and by preemptive scheduling. The rules for non-preemptive scheduling are:

- Tasks once started are allowed to be completed before the next in line is taken up. Completion means that all programs and function blocks assigned to the task are executed in sequence;
- If two tasks are waiting to be executed, the one with the higher priority is taken up first;
- If two waiting tasks have equal priority, the one waiting for longer time is taken up first;
- A task, once completed, is taken up for execution only on completion of the assigned task interval.

The advantage of non-preemptive scheduling is that the design is more straightforward and easier. The disadvantage is that, it is difficult to predict the exact time interval with which a task will be executed. Moreover,

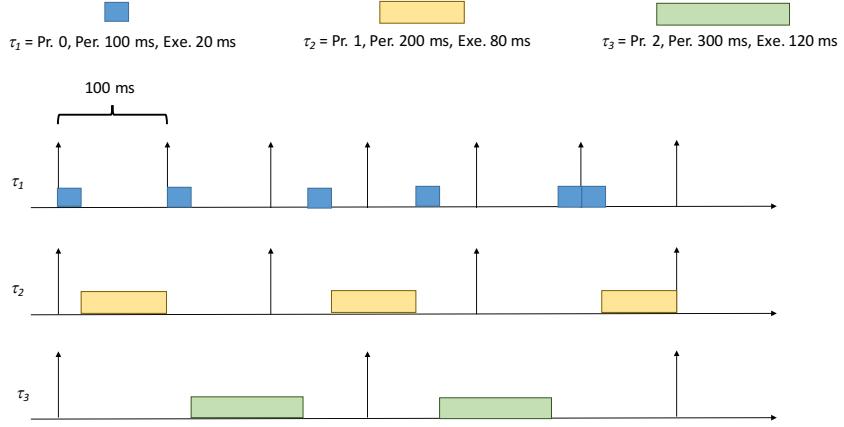


Figure 8.8: Example of non-preemptive scheduling.

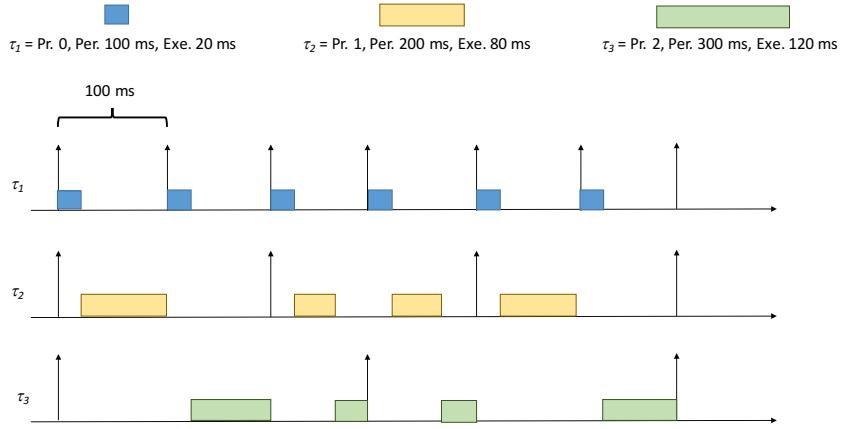


Figure 8.9: Example of preemptive scheduling.

a desired execution sequence of tasks cannot be ensured. This results in unpredictable system behaviour and is therefore non-deterministic in nature. A graphical comparison between a non-preemptive and a preemptive scheme is reported in Figure 8.8 and Figure 8.9, respectively.

When a system with deterministic behaviour is desirable, preemptive scheduling is to be used. Indeed, when a higher priority task is to be executed as per the specified interval, it is immediately scheduled and the currently active lower priority task is suspended and will be continued after the higher priority task terminates. This desired behaviour is clearly visible in Figure 8.9 with respect to the task τ_1 with highest priority: it is always executed at the beginning of the execution period, while this is not the case for the non-preemptive scheme of Figure 8.8.

From a practical perspective, the tasks are declared as follows:

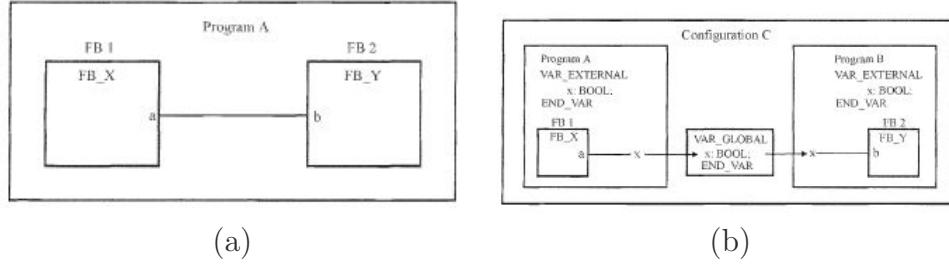


Figure 8.10: Example of internal communication within a program (a) and within a configuration (b) [17].

```
TASK INTERLOCK (INTERVAL := 50ms, PRIORITY := 1);  
TASK LOGRECORD (SINGLE :=LOG EVENT, PRIORITY := 2);
```

The task has an interval (if periodic or time-based) or not (event-based) depending on its nature. The priority is used by preemptive or non-preemptive schemes differently, but they are always needed.

8.3.2 Communication

The ability to communicate data between remote processors is the key to building large control systems. Communication can be grouped under two categories:

- Internal communications can either be communication between program elements or between different resources within a configuration. Internal communication takes place using variables within a program or using global variables within a configuration.
 - External communications between configurations use communication function blocks SEND and RCV or access paths with special VAR_ACCESS type of variables (i.e. shared memory approach).

A graphical representation of internal communications, within a program and within a configuration, is reported in Figure 8.10. Instead, an example of external communication, i.e. between different configurations using access paths or function blocks, are depicted in Figure 8.11.

Configurations and resources control the execution of Program organisation units contained in them. In particular, when a configuration starts, the following actions are performed: global variables are initialised; resources started; variables within the resource initialised; tasks enabled; program elements controlled by the task are executed. Instead, at shutdown, the con-

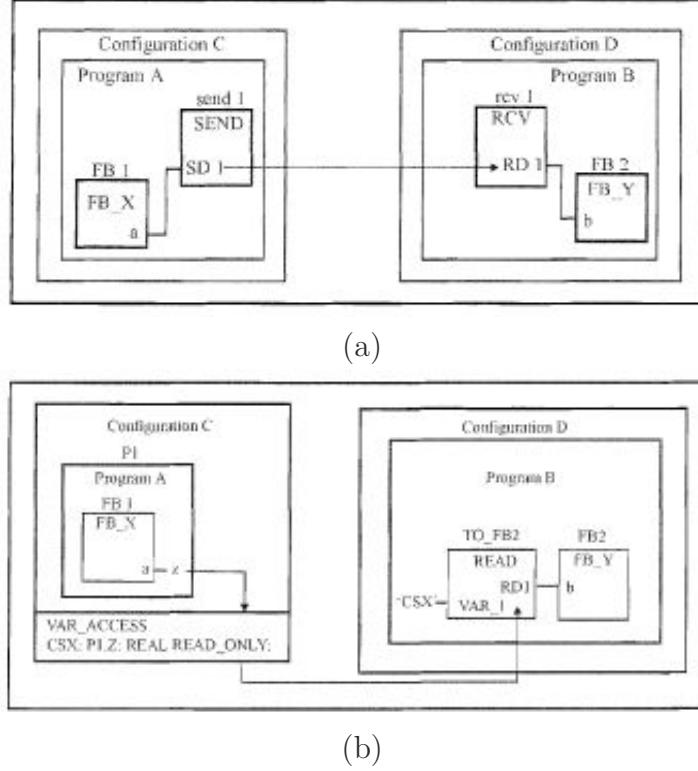


Figure 8.11: Example of external communication using function blocks (a) or access paths (b) [17].

figuration executes: stopping of all resources; tasks in these resources are disabled; programs and function blocks stop executing.

Simple systems with a single processor meant for controlling one machine or a system usually have one configuration, one resource and one program. However we noticed that larger systems having multiple processors may be represented by a single configuration. In general, each processor may correspond to a single resource and may control more than one program. Moreover, it is also possible to have a distributed control system spread across several PLCs connected through a high-speed communication network. This flexibility is summarised in Figure 8.12.

To recap, a configuration defines the software of a complete PLC. A configuration includes at least one, and usually several, resources. A resource in turn defines several tasks and programs to be executed. Global variables common to all resources are specified in the configuration and are accessible to all the programs within the resource. Access variables to be used for communication with other configurations are also specified at the configuration

Processor 1	Processor 2	Processor 3
Configuration		
Resource 1	Resource 2	Resource 3
Programs 1A and 1B	Programs 2A and 2B	Programs 3A and 31B

(a)

Processor 1	Processor 2	Processor 4	Processor 3	Processor 5
Configuration A			Configuration B	
Resource 1	Resource 2	Resource 3	Resource 4	Resource 5
Programs 1A and 1B	Programs 2A and 2B	Programs 3A and 3B	Programs 4A and 4B	Programs 5A and 5B

(b)

Figure 8.12: Multiple processors shared by a single configuration (a) or by multiple configurations (b) [17].

level. The configuration is defined using the following software construct:

```

CONFIGURATION <Identifier>
GLOBAL VARIABLE Declaration
ACCESS VARIABLE Declaration
RESOURCE Declaration
<Global variables within resource>
<Task definitions>
<Programs>
END CONFIGURATION

```

Variables used within a resource are identified hierarchically thus: <Resource Identifier>.<Program Identifier>.<Variable name>.

8.3.3 Ladder Diagram Revisited

Using the definitions comprised in the standard, the Ladder Diagram exports additional functionalities. The main advantage is the coherent use and integration of Function Blocks inside the Ladder Diagrams, as exemplified in Figure 8.13 and where SWITCH_A and SWITCH_B represent contacts of a button for starting and stopping a motor. Moreover, output MOTOR is a boolean variable which is used to set the variable MTR_B depending on the status of variable INHIB_2. Notice that the Function Blocks represented within a Ladder Diagram can have other non-Boolean inputs and outputs, as reported in the figure.

As defined in IEC 61131-3, standard programming languages for the PLC are: Structured Text (ST), Ladder Diagram (LAD), Instruction List (IL), Sequential Function Chart (SFC) and Function Block Diagram (FBD). Since

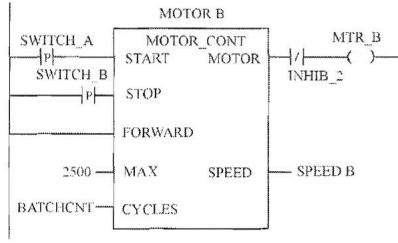


Figure 8.13: Function Block embedded within a Ladder Diagram [17].

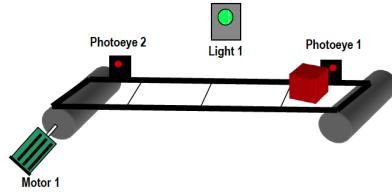


Figure 8.14: Example of the conveyor application (courtesy of Dr. Sunil Jha, Department of Mechanical Engineering, Indian Institute of Technology, Delhi).

PLC manufacturers implement programming languages on the base of IEC 61131-3 standard but with remarkable differences, compatibility problems arise. For example, a program in LAD language from a certain manufacturer, although similar, cannot be imported to PLC from other manufacturers with other programming languages.

Example 12. Consider a conveyor application in an industrial plant, depicted in Figure 8.14. When a box is placed on the conveyor in front of Photo-eye 1, Light 1 and Motor 1 will turn on, causing the box to move to the left. When the box passes in front of Photo-eye 2, Motor 1 and Light 1 will turn off, stopping the conveyor.

To solve this problem with the LAD, the conveyor solution encodes a timer: if the Motor 1 stays on for more than 30 seconds, the conveyor stops, the light is turned off and an alarm is issued. To implement the timer, we can make use of a function block inserted in a rung: when the rung is true the timer will run and will store the elapsed time in an accumulator (see Figure 8.15). As long as the rung remains true, it will continuously count until it reaches the preset value, when the DN bit will go on. When the rung goes false the timer will be reset. Using the Timer On Delay (TON), a possible LAD solution is represented in Figure 8.16.

TON	
Timer On Delay	timer1
timer	100
preset	
accum	0

Figure 8.15: TON - Timer On Delay (courtesy of Dr. Sunil Jha, Department of Mechanical Engineering, Indian Institute of Technology, Delhi).

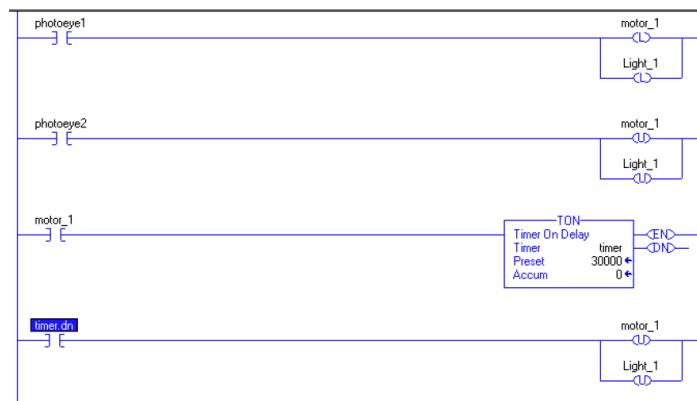


Figure 8.16: Example of the ladder diagram for the conveyor application (courtesy of Dr. Sunil Jha, Department of Mechanical Engineering, Indian Institute of Technology, Delhi).

8.4 The Future of PLCs

PLCs can be considered as a mature technology with little room for improvement as they've been around for nearly 50 years. But like their close counterparts in the world of consumer electronics, significant improvements continue with no end in sight, promising faster, smaller, and lower cost solutions. In the future, PLCs will continue to evolve while adapting technology improvements in hardware, communications, and software. Part of the evolution will include merging of PLC and programmable automation controller (PAC) functionality, i.e., modular industrial controller that uses a PC-based processor and allows programming options beyond the IEC 61131-3 languages. An example of a modern PLC by Siemens is reported in Figure 8.17.



Figure 8.17: A modern PLC.

Chapter 9

SCADA

One primary application at a higher level than the fieldbus level is the Supervisory Control And Data Acquisition (SCADA) systems. The networked SCADA systems often provide a supervisory-level factory-wide solution for coordination of machine and process diagnostics, along with other factory floor and operations information. SCADA has been around as long as there have been control systems. The first SCADA systems utilised data acquisition by means of panels of meters, lights and strip chart recorders. Supervisory control was exercised by the operator manually operating various control knobs. These devices were and still are used to do supervisory control and data acquisition on plants, factories and power generating facilities.

SCADA refers to the combination of telemetry and data acquisition. SCADA encompasses the collecting of the information, transferring it back to the central site, carrying out any necessary analysis and control and then displaying that information on a number of operator screens or displays. The required control actions are then conveyed back to the process.

SCADA is an acronym that stands for Supervisory Control And Data Acquisition, describing the three compounding components: Supervisory feature available to operators, engineers, etc.; Control which can be monitoring, telemetry and remote or local; Data Acquisition, that is access and acquire information or data from the equipment, both digital and analog, and sends it to different sites through telemetry.

9.1 A bird's eye view

SCADA is not a specific technology, but a type of application. In a very wide sense:

Definition 38. *Any application that gets data about a system in order to*

control that system is a SCADA application.

A SCADA application has two elements: the process/system/machinery to monitor and control, e.g., a power plant, a water system, a network, a system of traffic lights, etc.; A network of intelligent devices that interfaces with the first system through sensors and actuators. The network is the SCADA system since it gives the ability to measure and control specific elements of the considered plant and can be implemented with several different kinds of technologies and protocols. The simplest form of SCADA system is a warning/alarm light.

SCADA is used to manage any kind of equipment. Typically, SCADA systems are used to automate complex industrial processes where human control is impractical. For example, systems where there are more control factors, and more fast-moving control factors, than human beings can comfortably manage. Additional examples are power grids, water pressure control rooms (e.g. produced by ABB for energy optimisation of water supply in smart buildings), smart GRID control rooms (e.g. produced by ABB for Smart Grid SCADA systems) or water distribution monitoring (e.g. to monitor and regulate water flow, reservoir levels, pipe pressure and other factors), all represented in Figure 9.1. Other examples are building monitoring (e.g. to control HVAC, refrigeration units, lighting and entry systems), manufacturing systems (e.g. to manage parts inventories for just-in-time manufacturing, to regulate industrial automation and robots, and to monitor process and quality control), mass transit systems (used by the transit authorities to regulate electricity to subways, trams and trolley buses; to automate rail systems; to track and locate trains and buses; and to control railroad crossing gates) and traffic lights monitoring (e.g. to regulate traffic lights, controls traffic flow and detects out-of-order signals), all represented in Figure 9.2.

The potential applications for SCADA systems are almost unlimited: SCADA is used in nearly every industry and public infrastructure project, i.e., anywhere automation increases efficiency. SCADA operations are deep and complex. In every industrial plant, managers need to control multiple factors and the interactions between those factors: SCADA systems provide the sensing capabilities and the computational power to track everything that's relevant to the plant operations.

9.1.1 SCADA Components

SCADA encompasses the transfer of data between a number of remote sites (Remote Terminal Units or RTUs), connected through data multiplexing (MUX), a central host computer and the operator terminals. Basically, the

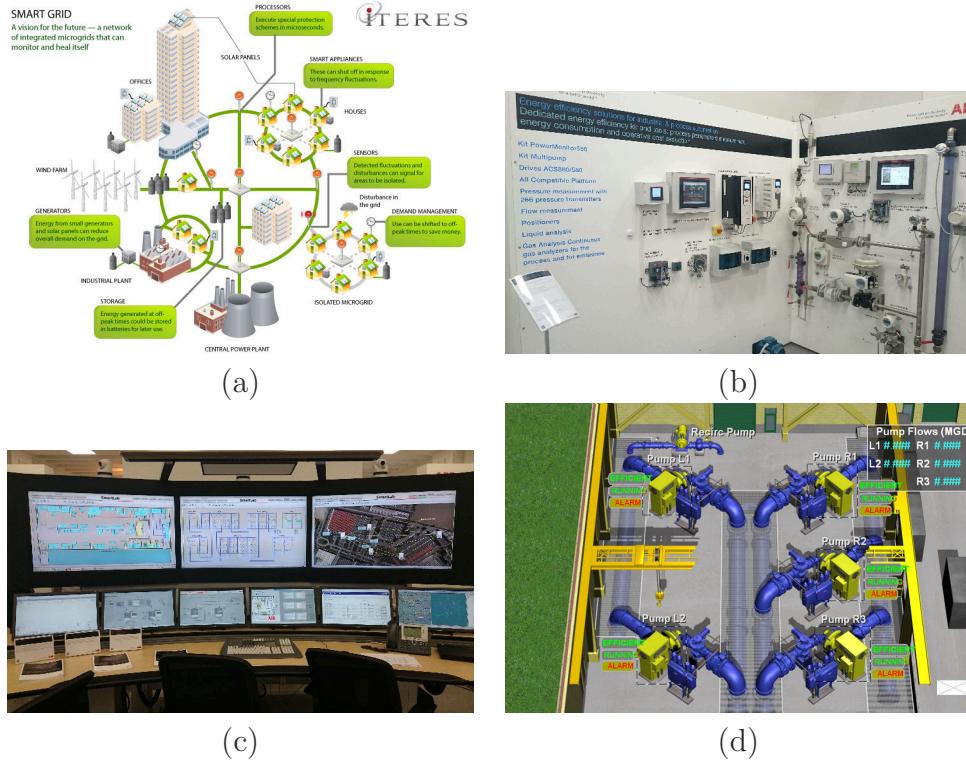


Figure 9.1: (a) Power grid, (b) ABB solution for water pressure control, Sesto San Giovanni, Milano, Italy, (c) Smart Grid SCADA system in ABB, Sesto San Giovanni, Milano, Italy, (d) water distribution monitor.

MUXs serve to route data to and from a (relatively high) number of RTUs on a local network. Very few physical links on a Wide Area Network (WAN) backbone are instead used to pass data back to the central host computer, as depicted in Figure 9.3.

SCADA systems components are:

- One or more field data interface devices, usually called Remote Stations, Remote Terminal Units (RTUs), or Programmable Logic Controllers (PLCs), which interface to field sensing devices and local control switch-boxes and valve actuators.
- A communication system used to transfer data between field data interface devices and control units and the computers in the SCADA central host. The system can be radio, telephone, cable, satellite, and so on, or any combination of these.
- A central host computer server or servers (sometimes called a SCADA

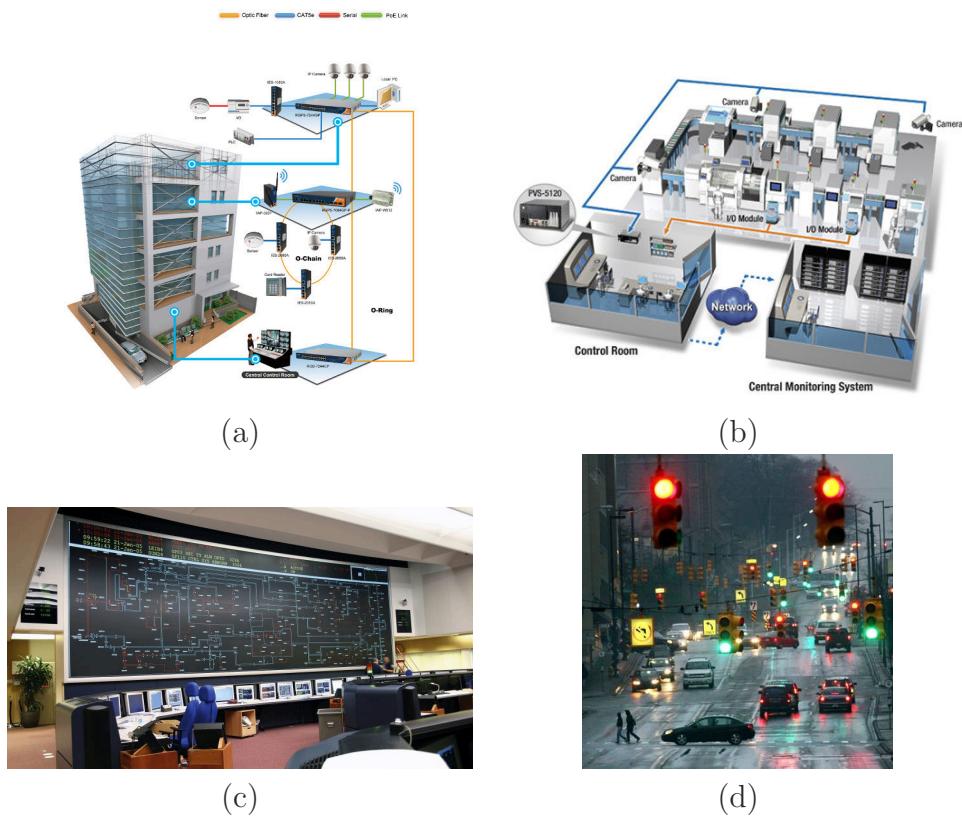


Figure 9.2: (a) Building monitor, (b) manufacturing system, (c) mass transit SCADA system, (d) traffic lights monitoring and control.

Center, master station, Master Terminal Unit or MTU).

- Another communication system to support the use of operator workstations that may be geographically remote from the central host computer.
- A collection of standard and/or custom software, usually called Human Machine Interface (HMI) software or Man Machine Interface (MMI) software, which are used to provide applications and support at the operator terminal.

Field data interface devices form the “eyes and ears” of a SCADA system. For example, a water distribution system comprises reservoir level meters, water flow meters, valve position transmitters, temperature transmitters, etc. In addition, the interface devices comprise the “hands” of the SCADA that build up the automated process, e.g., electric valve actuators, motor control switchboards, pumps, etc. The information that is passed to and from

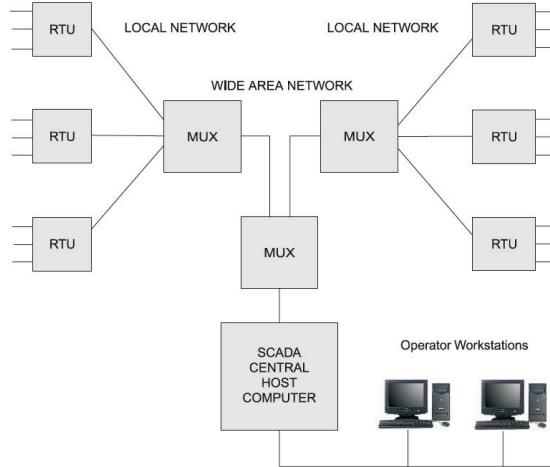


Figure 9.3: SCADA system with data multiplexing (MUXs) between the central host and the RTUs [18].

the field data interface devices must be converted by the Remote Terminal Units (RTUs), aka Remote Telemetry Units, to the adopted communication protocol. The instructions for the automation of field data interface devices, such as pump control logic, are usually stored locally in Programmable Logic Controllers (PLCs). This is largely due to the limited bandwidth typical of communications links between the SCADA central host computer and the field data interface devices. PLCs connect directly to field data interface devices and incorporate programmed intelligence in the form of logical procedures that will be executed in the event of certain field conditions. There exist SCADA systems with no PLCs. In this case, the local control logic is held within the RTU or in relay logic in the local switchboard.

SCADA communication links

The SCADA communication links generally offer less bandwidth and lower reliability than that offered for process plant, where fiber optic LAN infrastructures may be employed. On SCADA links, where a combination of data radio, telephone line, and satellite link technology may be employed, the availability may be as low as 99.0 percent (compared to 99.9) and bit error rates of 10^{-6} (instead of 10^{-9}) or greater. Communication outages typically result from equipment and power supply failures and human interference. Better availability (at a higher cost) is possible through the use of redundant communication paths to outstations. SCADA communication protocols are designed to provide secure transmission of data. The protocols employ error

detection and message retry techniques usually by receive/transmit handshaking established through the use of headers and footers attached to the raw data under transmission. As you should know, the extra information introduces an overhead to the transmission of data that should be traded with the speed of data transmission. Therefore, the speed of data communications associated with SCADA is further limited and then slower than that typical of a communication backbone commonly used on a process plant.

Users of SCADA systems and the resulting data do not need to be aware of the communications protocols used, even though they should know the problems affecting the communication, e.g., link failure, overhead, etc. For example, a control command could be sent to the wrong destination. SCADA systems often request confirmation from an operator to confirm that a control action is required. This approach provides also an additional level of protection against human operator errors.

Some communication protocols often adopted in SCADA systems:

- Distributed Network Protocol (DNP 3.0): a vendor independent protocol that incorporates multiple layers of error detection and correction and allows a select/confirm regime for control actions;
- Modbus is another widely used protocol for SCADA, but it does not offer the same level of data transmission security as DNP 3.0;
- PROFIBUS, Fieldbus, Controller Area Network (CAN), TCP/IP, etc.

The communications media are instead quite widespread: licensed radio links (UHF and VHF); unlicensed “spread spectrum” radio links; public switched telephone networks; mobile telephony; microwave; cable TV networks; dedicated satellite links; dedicated cable, including fiber optics (for very short distance communication); corporate WAN computer communications systems. Selection of the preferred communication media depends on several important factors:

- The remoteness of the field equipment site;
- The required reliability of the communications media (primarily determined by the perceived operational importance of the remote site);
- Availability of communications options;
- Cost of each option for the particular application;
- Availability of power (power company, battery, solar, or other).

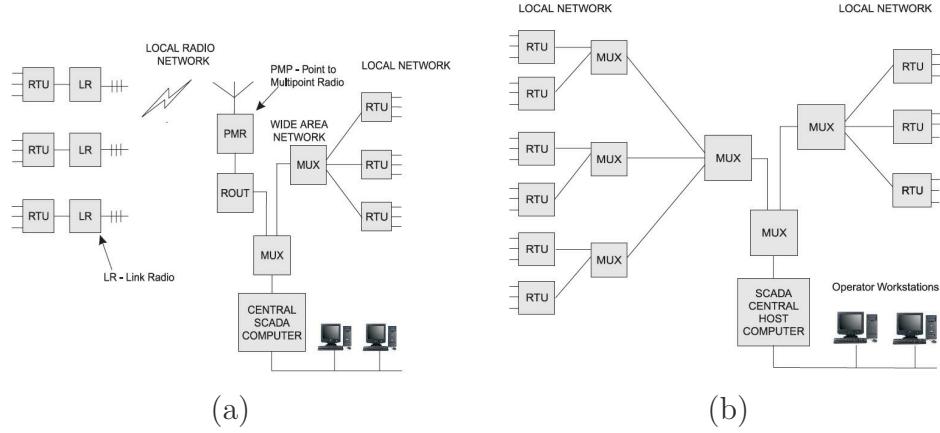


Figure 9.4: Different SCADA architectures: (a) data router with point-to-multipoint radio. LR refers to Local Radio, PMR to Point to Multipoint Radio and ROUT to data router; (b) tree network with multiple hierarchical levels of multiplexer processors [18].

The architecture of the communication system is indeed very flexible, whilst the most popular is reported in Figure 9.3. The WAN connects the central host computer to the multiplexers. It may comprise cable, radio, or satellite data communications links depending on the geographic distribution of the SCADA system. The WAN links are generally full duplex and may be configured in a star or loop topology. The loop configuration links adjacent multiplexers and provides alternative communications paths for redundancy, therefore providing higher reliability w.r.t. star topology. Looped WANs require data traffic routers. In simple SCADA system all RTUs are connected directly to the central host computer via a single multi-drop communications link. These systems therefore effectively only contain an RTU local area network.

The MUXs allow different data streams to share a single data link. Multiplexers combine communications paths to and from many RTUs into a single bit stream, usually using Time Division Multiplexing (TDM) or other such bit stream manipulation techniques. The MUXs must be able to combine the traffic to and from tens or sometimes hundreds of RTUs for transmission over the SCADA WAN. The multiplexer may itself be a SCADA processing device that manages the local network and not only combines the data, but also reduces the amount of data that must be interchanged with the central host. A simple form of multiplexer is to use a data traffic router together with a point-to-multipoint radio, as depicted in Figure 9.4-a. Another possible solution can be the adoption of a stratified, hierarchical collection of

multiplexers, as in Figure 9.4.

Local networks connect the RTUs to the multiplexers, or directly to the SCADA central host computer if there is no need for a WAN connection. Like the WAN, the local network may comprise cable, radio, or satellite data communications links depending on the geographic distribution of the SCADA system. In a common local network, the radio links are generally half duplex or simplex. Most local networks use a logical bus topology, hence the protocol usually includes ordered polling of each RTU, token passing, and data packet collision detection and avoidance mechanisms.

Central Host Computer

The central host computer, or master station, is most often a single computer or a network of computer servers that provide a man-machine operator interface to the SCADA system. The computers process the information received from and sent to the RTU sites and present it to human operators in a form that the operators can work with. Operator terminals are connected to the central host computer by a computer network so that the viewing screens and associated data can be displayed for the operators. For many SCADA systems, several operators may require simultaneous access to the SCADA central host computer to view the performance of the system. Operator workstations are most often computer terminals that are networked with the SCADA central host computer. The central host computer acts as a server for the SCADA application and the operator terminals are clients. The communication system in place between the central host computer and the operator terminals is a LAN.

The most obvious software component is the operator interface or MMI/HMI package. The primary interface to the operator from the operator terminal is a graphical user interface (GUI) display that shows a representation of the plant or equipment in graphical form. Data from the field are processed to detect alarm conditions, and if an alarm is present, it will be displayed on dedicated alarm lists on the application software running on the central host computer. In case of an alarm, operators can then investigate the cause of the alarm by using the SCADA system. Where variables in the field have been changing over time, the SCADA system usually offers a trending system whereby the behaviour of a particular variable can be plotted on a graphical user interface screen.

9.1.2 Acquisition, Processing, Control and Failures

Data acquisition within SCADA systems is accomplished by letting the RTUs scanning the field data interface devices connected to them. The time to perform this task is called the scanning interval. The central host computer scans the RTUs to access the data in a process referred to as polling the RTUs. Some systems allow also the RTU to transmit field values and alarms to the central host without being polled by the central host, unsolicited messaging. Unsolicited messages are usually only transmitted when the field data has deviated by a pre-specified percentage, so as to minimize the use of the communications channels, or when a suitably urgent alarm indicating some site abnormality exists, i.e., event-based. Control actions that are performed by using the central host are generally treated as data that are sent to the RTU. As such, any control actions by an operator logged into the central host will initiate a communication link with the RTU to allow the control command to be sent to the field data interface device under control.

Remark 2. *SCADA systems usually employ several layers of checking mechanisms to ensure that the transmitted command is received by the intended target.*

In contrast with DCS, SCADA systems generally cover large geographic areas and rely on a variety of communication systems that are normally less reliable than a LAN associated with a DCS. Loop control based in the central host computer is therefore less desirable. Instead, the controller application, that can be adjusted by the central host, is housed in the RTU. If communication to the remote site is lost, it is desirable that the local automatic control system continue to operate; therefore, the RTU is an autonomous unit which could control the valve without constant direction from the central host computer.

Different SCADA systems cope differently with data handling during a failure event. One possibility is to store data in the RTUs directly. In fact, some SCADA systems rely on the capacity of the RTU to store data collected from the field under normal operation and then periodically transmit that data as an unsolicited message or when polled by the central host. In times of SCADA system failure, the capacity of the RTU is used to archive information until a backup central host is brought online or the original system has recovered.

Another possibility is to use system redundancy. Most SCADA systems incorporate some form of redundancy in their design, such as dual communications channels, dual RTUs, or dual central host computers. Such systems may be designed for such redundant equipment to be online (hot standby)

to ensure a seamless transfer upon SCADA system failure, or offline (cold standby) where the backup mechanism must be manually brought online to operational capacity. Of course, the majority of the SCADA systems employ a combination of the two previously presented mechanisms.

9.1.3 Evaluate SCADA

Evaluating complex systems as a SCADA can be very tricky. However, there are some guidelines to what look for in a SCADA system. One first mean of evaluation is given by the employed RTUs, which are requested to communicate with all the on-site equipments and survive under the harsh conditions of an industrial environment. To this end the RTUs should offer sufficient capacity to support the necessary equipment (but not more capacity than can actually be used), should have a rugged construction and ability to withstand extremes of temperature and humidity (since the SCADA system needs to be the most reliable element in your facility), should have a secure, redundant power supply (the RTU should support battery power and, ideally, two power inputs, since it has to work 24/7.), should be endowed with a real-time clock for accurate date/time stamping of reports and should also have a watchdog timer to ensure that the RTU restarts after a power failure. Moreover, the RTUS should have redundant communication ports since network connectivity is as important to SCADA operations as a power supply (hence, usually a secondary serial port or internal modem can keep the RTU online even if the LAN fails and ensures easy support for LAN migration strategies, should have a nonvolatile memory for storing software and/or firmware (this memory retains data even when power is lost and stores new device firmware, usually provided via LAN) and should be able to implement intelligent control according to programmed responses to sensor inputs (very important for almost any application).

The central host is another important component and should be able to offer flexible, programmable tools to define response to sensor inputs (i.e. the system has to provide easy tools for programming soft alarms - reports of complex events that track combinations of sensor inputs and date/time statements - and soft controls - programmed control responses to sensor inputs), should offer 24/7, automatic pager and email notification (to efficiently warn the human operators or the repair technicians), should be able to display detailed information and reports in natural language (with a complete description of what activity is happening and how can be managed), should filter out nuisance alarm filtering that otherwise reduce the sensibility to alarm reports, should be easily expansible (it usually lasts for as long as 10 to 15 years), should be able to perform scure backups (using redundant and

geodiverse approaches) and should support multiple protocols and equipment types (i.e. support for multiple open protocols safeguards the SCADA system against unplanned obsolescence).

9.1.4 The Four SCADA Levels

The SCADA system is divided into four main levels. The Level IV - Enterprise - comprises the corporate WAN/LAN, the World Wide Web, the Virtual Private Network and the firewalls for external users. The Level III - SCADA - Central host - comprises the operator workstations, the supervisory control abilities, the engineering workstations and the servers and data logging systems. The Level II - Telecommunication - comprises the adopted communication media and the communication protocols. Finally, the Level I - Field - comprises the field devices, the RTUs, the PLCs and the sensors.

9.2 SCADA, DCS and PLCs

SCADA vs DCS

There is a considerable confusion nowadays about the difference between a SCADA system and a Distributed Control System (DCS). As by the name, the DCS is focused on control issues, while a SCADA system comprises the data acquisition part and the supervisory control. The difference between the supervisory control and the control is that the first one is executed sporadically and comprises high level behaviours of the control law, while the second is more close to the classic low-level dynamic control of machineries.

Historically, computer networks had very low bandwidth, thus making impossible to efficiently close the loop using such communication media. As a consequence, the SCADA was the higher-level controller for a set of lower level controllers. In this framework, the DCS, closer to the field, was in charge for the accurate control of specific components. The DCS then reports the information to the SCADA system, that in its turn generates high level control behaviours for the DCS components. Today, due to the efficiency of computer networks, the difference is becoming more and more blurred. The naming convention largely depends on the region in which the system is deployed. Nonetheless, a summary of the differences is reported in the next list:

- DCS is process oriented, while SCADA is data acquisition oriented.
- DCS is process state driven, while SCADA is event driven.

- DCS is usually used to handle local production processes, while SCADA is more related to large geographic areas.
- DCS operator stations are always connected to field devices I/O, while SCADA is expected to operate despite failure or field communications.

SCADA and PLC

The Programmable Logic Controller (PLC) is still one of the most widely used control systems in industry. As needs grew, the PLCs were distributed and the systems became more intelligent and smaller in size. To mark the difference between the PLCs and the RTUs, we can recall the history of their developments. PLCs have their origins in the automation industry and therefore are often used in manufacturing and process plant applications. At the beginning of the automation era, the need for PLCs to connect to communication channels was not great in these applications, as they often were only required to replace traditional relay logic systems or pneumatic controllers. SCADA systems, on the other hand, have origins in early telemetry applications, where it was only necessary to know basic information from a remote source. The RTUs connected to these systems had no need for control programming because the local control algorithm was held in the relay switching logic. As PLCs were used more often to replace relay switching logic control systems, telemetry was used more and more with PLCs at the remote sites. Hence, it became desirable to influence the program within the PLC through the use of a remote signal. This is the effect of the Supervisory Control of SCADA systems. Where only a simple local control program was required, it became possible to store this program within the RTU and perform the control within that device. At the same time, traditional PLCs included communications modules that would allow PLCs to report the state of the control program to a computer plugged into the PLC or to a remote computer via a telephone line.

As a consequence, PLC and RTU manufacturers compete for the same market. Therefore, the line between PLCs and RTUs has blurred and the terminology is virtually interchangeable. In general, the term RTU can be used to refer to a remote field data interface device. If such a device could include automation programming that traditionally would have been classified as a PLC.

Chapter 10

Measurement Processes

We already saw that there is an undisputed need of *analysing the measured data* to extract meaningful information from a production site, a plant, an urban area, etc. Perhaps the most relevant application in this case is a *SCADA* system, which is critically related to the collection of measurements, to their analysis and to the representation of those data to a human operator with a HMI. *Data analysis* is also of major importance for DCS in industrial environments, but it is becoming widespread with the use of the *Internet of Things*, well exemplified in Figure 10.1. In the IoT context we have an *ensemble of embedded low cost and low power devices*, going beyond the single device/single application paradigm using the many-to-many IP web paradigm. An IoT system basically comprises the same components of a DCS or a SCADA system: sensors; actuators; a communication infrastructure with possible different protocols and technology; storage; processing capability; a user interface. Usually *data collection analysis and transformation*, *data storage* and *data aggregation* are responsible for data management and are at the mid level of the layered semantic data flow for IoT, as reported in Figure 10.2. In [20], the layered approach is described and it is explained that data usually flows back and forth from the lower layer (i.e. Level 1 - “Physical Devices and Controllers”) to the upper layer (i.e. Level 7 - “Collaboration and Processes”). The exception is the monitoring applications, where data usually flows from the bottom to the top, where they are analysed. This scheme follows the same philosophy of the OSI model for communication, except that in this case the subject is not the message to be communicated but the data and its meaning. To be more precise, an entire OSI model (see Section 2.2) is adopted at Level 2 - “Connectivity”. Instead, Level 3 - “Edge (Fog) Computing” is responsible to analyse elementary data (i.e. the packets that arrive from Level 2) and transform them (e.g. encoding/decoding) in a standardised way. This set of operations can be seen as

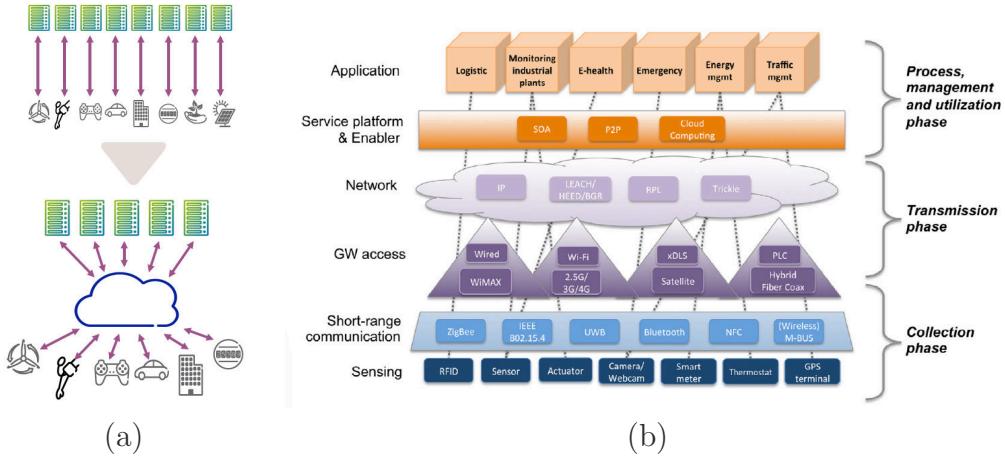


Figure 10.1: (a) The IoT paradigm (courtesy of A. Zanella, IEEE SPS Italy Chapter Summer School on Signal Processing (S3P)). (b) A representation of technological solutions for IoT applications [19].

a collection of estimation algorithms acting on, e.g., sensor data (the focus of the next chapters). At Level 4 - “Data Accumulation” the data is stored and it is determined if it is of interest to higher levels, if it has to be permanently stored to form a knowledge base, the type and the organisation of the data storage. The information are then abstracted (that is represented in a standard common way and verified for their completeness) to be easily handled by upper layers in Level 5 - “Data Abstraction”. At Level 6 - “Application”, information are interpreted and, since data are stored, it does not have to operate at network speeds, and, finally at Level 7 - “Collaboration and Processes” the data are shared among multiple users that can be located in different areas, to make the most out of the information collected.

For any possible application, the main issues related to the *data management* (i.e. occurring at Level 4, 5 and 6) are *data protection* (mainly related to *privacy*, that is how to control owned information: *certification* and *data anonymising*), *data processing* (mainly related to the execution of data analysis: *big data* and *data ownership*) and *data analysis*, which is the *main focus for our purposes*. There are three main goals for *data analysis*:

- *Extract meaningful information from data*: this is usually related to *model estimation*, in order to illustrate the behaviour of a given complex system and to predict its future evolution.
- *Resources optimisation*: mainly bandwidth, communication QoS and components lifetime.

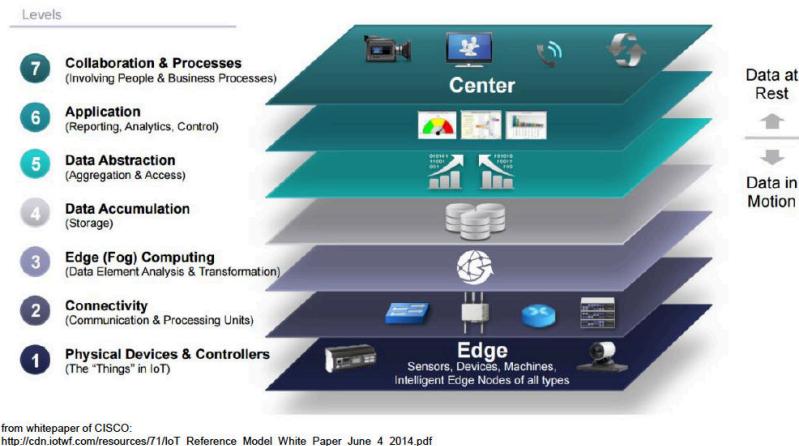


Figure 10.2: The Internet of Things Reference Model [20].

- *Estimate actual values from measurements*: this is the classic problem of *estimation*, *main focus in this chapter*.

Depending on the type of information to be extracted from the data, different approaches are available. The identification of possible problem in production systems is carried out using *process mining*: *what are the possible point of failures inside the production process?*. If aggregated information is needed, *statistics* is adopted: *what is the mean number of users of a service (e.g. electric power, communication network, etc.) in the day-time?* To derive relations in the datasets, *machine learning* is usually adopted. In this latter case, different problems can be tackled:

- The relation, if any, between *dependent variables* from *independent variables* given a data set can be retrieved using *regression*: *what is the relationship between weather condition and power consumption?*
- The automatic definition of *categories* (aka *clusters*) in a data set is the goal of *clustering* algorithms: *what is the typical behavioural pattern of the users of a certain service?*
- Identifying to which *category* a certain variable belongs to is instead a *classification* problem (a subset of *pattern recognition*): *is this power grid state leading to a possible incoming black-out?*

In this notes we will mainly consider only the *estimation problems*, which are of major concerns for *distributed autonomous systems*.

10.1 Measurement processes

Any state estimation or localisation problem relies on the availability of *measurements*.

Definition 39 (*International Vocabulary of Metrology* (VIM) [21]). *The measurement is the process of experimentally obtaining one or more quantity values that can reasonably be attributed to a quantity.*

Definition 40 (VIM). *The quantity intended to be measured is called the measurand.*

The result of the *measurement process* is the *measurement result*.

Definition 41 (VIM). *The quantity is the property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference.*

The *reference* can be a *measurement unit* (e.g. the metre), a *measurement procedure* or a *reference material* (e.g. in chemical reactions). The main issue is that the measurement result *cannot be* expressed by just a number. Moreover, the measurement result can only provide a *finite amount of information* about the measurand. So what is a *measurement result*?

Definition 42 (VIM). *The measurement result is the set of quantity values being attributed to a measurand together with any other available relevant information.*

The *main* attribute is called *uncertainty*. Therefore:

Definition 43 (VIM). *The measurement result is generally expressed as a single measured quantity value and a measurement uncertainty.*

All the previous definitions pertain to the field of science called *metrology*. In particular:

“The *measurement process* maps a single manifestation q of a property (for instance weight, length,...) of an object (that is a phenomenon, a body, or a substance) belonging to the empirical world onto a symbol x that belongs to the symbolical world.” [22].

Formally, let us define the set of possible manifestations of q as \mathcal{Q} and let \mathcal{Z} be the class of all the symbols z , than a *measurement process* is

$$\eta : \mathcal{Q} \rightarrow \mathcal{Z},$$

while

$$z = \eta(q).$$

is the *measurement result*. According to the *Guide to the Expression of Uncertainty in Measurement* (GUM) [23], the word *uncertainty* means *doubt* about the information provided by the measurement result. Notice that *uncertainty* expresses also the *quantitative measure* of the doubt. There are different origins of the *uncertainty*:

- *Definitional uncertainty*: it is related to the modelling errors (e.g. assuming that a surface is actually planar);
- *Interaction uncertainty*: physical interaction of the measurement system with the measurand (e.g. a thermometer measuring the temperature of a fluid);
- *Instrumental uncertainty*: the imperfect behaviour of the measurement system (e.g. the use of reference quantities for comparison, which are actually uncertain).

The *uncertainty* has different effects on the *measurement results*, which can be grouped in two broad sets: *systematic* and *random* effects. *Systematic effects* are constant in time and very dangerous. A standard way to detect them is to compare the measurement results obtained by using *two independent measuring systems*, where one of the two is known to be with *negligible* systematic effects. In general, the *systematic effects* depend on some *influence quantities* (e.g. the temperature). In the GUM [23] it is recognised the *high relevance* of such uncertainties: “*It is assumed that the result of a measurement has been corrected for all recognised significant systematic effects and that every effort has been made to identify such effects.*” Usually the systematic effects are characterised by the *manufacturer* or by the *calibration certificates*.

Let us clarify how the systematic effect can be mitigated using *regression techniques* by assuming we are interested in finding the relationship between the *systematic offset*, say o , and an *influence quantity*, say d . The *systematic effect* can be collected as difference between the *measurement quantity* z and the measurement quantity collected with a sensor with negligible *systematic effect* \bar{z} , i.e. $o = z - \bar{z}$, taken with *n different values of d*. For a *supervised learning* approach, aka *white or grey-box model*, one can start from a given model, which can be linear or not, i.e. for the i -th value of d

$$o_i = h_\theta(d_i) = \theta_0 + \theta_1 d_i \text{ or } o_i = h_\theta(d_i) = \theta_0 + \theta_1 d_i + \theta_2 d_i^2.$$

Then, the solution for the *systematic effect* parameters $\theta = [\theta_0, \theta_1, \theta_2]$ is given by the *Least Squares* solution, i.e.

$$\hat{\theta} = \arg \min_{\theta} \sum_{i=1}^n (h_{\theta}(d_i) - o_i)^2$$

10.1.1 The sensor

The term *sensor* defines the set of all the possible (usually) electro–mechanical systems that convert physical phenomena to measurable signals, typically voltages or currents. Sensors are also called *transducers* and implements the mapping $\eta : \mathcal{Q} \rightarrow \mathcal{Z}$ implementing the *measurement process*. Consider a simple pressure measuring device: the output voltage proportionally increases as the pressure increases. A digital device could measure the voltage, and convert it to a pressure. Each of this processes introduces *uncertainties*. Most of these sensors are based on subtle electrical properties of materials and devices. As a result the signals often require *signal conditioners* (e.g., amplifiers). The basic physical phenomena typically measured with sensors include angular or linear position, acceleration, temperature, pressure or flow rates, stress, strain or force, light intensity and sound.

Applications using sensors have major issues to cope with, since real sensors convert an input phenomena to an output of a different type. This transformation relies upon a manufactured device, with limitations and imperfection. Therefore real sensors are *uncertain*. Real sensors returns a *partial description of the environment*, i.e., only finite and well precise phenomenon can be directly observed. Real sensors are *difficult to be modelled*. As a consequence, data coming from real sensors should be managed carefully, taking into account their *uncertainties* and adopting model–based reconstruction algorithms.

Sensors are mainly characterised by three parameters, directly derived from *metrology*:

- *Accuracy*: closeness of agreement between a measured quantity value and a true quantity value of a *measurand*. A measurement is said to be more accurate when it offers a smaller *measurement error*.
- *Precision*: closeness of agreement between *measurement results* obtained by replicate measurements on the same or similar objects under specified conditions. Measurement precision is usually expressed numerically by measures of *uncertainty*, such as *standard deviation*, *variance*, or coefficient of variation under the specified conditions of

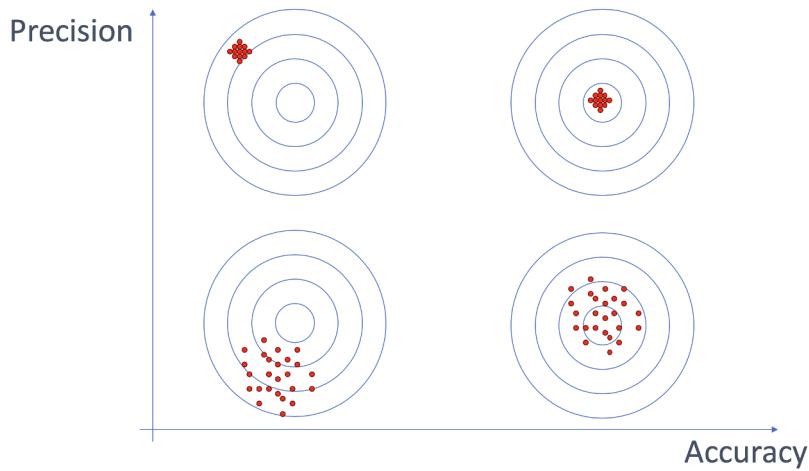


Figure 10.3: Precision and accuracy of measurements.

measurement. Measurement precision is used to define *measurement repeatability*, *intermediate measurement precision*, and *measurement reproducibility*. A pictorial difference between accuracy and precision can be found in Figure 10.3.

- *Repeatability*: measurement precision under a set of *repeatability conditions of measurement*, e.g., the condition of measurement, out of a set of conditions that includes the same measurement procedure, same operators, same measuring system, same operating conditions and same location, and replicate measurements on the same or similar objects over a short period of time. Roughly speaking, this concept deals with *time invariance of the measurement results*.
- *Resolution*: the smallest increment that the sensor can detect (due to *quantisation*).
- *Linearity*: a linear sensor has a linear relationship between the input phenomenon and the output signal. This way, it is easier to derive the measured quantity from the sensor readings.
- *Range*: the range of the measurable inputs and the range of the associated outputs.
- *Dynamic Response*: the frequency range for nominal operation of the sensor (e.g., cameras or microphones). Typically, there is an upper operation frequency, while a lower frequency is occasional.

- *Robustness*: tolerance with respect to deviations from nominal operational conditions, e.g., changes in temperature, humidity, dirt/oil, pressure, mechanical vibrations.
- *Calibration*: many sensors need some calibration to determine or set the relationship between the input phenomena and the output. Calibration is usually performed when they are manufactured or installed. Nevertheless, mainly due to environmental characteristics, it may be needed frequently (e.g. due to *systematic effects*).
- *Speed of operation*: the rate at which the sensor returns the measurements in continuous mode and/or the delay until the measurement is returned when it is requested intermittently.
- *Error rate*: the time of inter-arrival of erroneous measurements, e.g., outliers or missed measurements.
- *Computational cost*: the amount of data processing needed to achieve the desired information. For example, a switch may not require any computation, while a vision processing algorithm can be very time consuming.
- *Cost*: more precision costs more. In the sensor cost, should be also considered the cost of the conditioning equipment. For example, some sensors are very inexpensive, but the signal conditioning equipment costs are significant.
- *Power, weight and size*: the physical encumbrance and the power requested for the operation (some sensors can be switched on and off).

The typical quasi-static response of a sensor is shown in Figure 10.4. The *sensor response* relates the quantity to be measured u , i.e., input phenomenon, with the measured value z , i.e., the sensor output.

In order to *calibrate* the sensor, the curve of the sensor response has to be estimated. Suppose the sensor response can be described as

$$z = a_0 + a_1 u + a_2 u^2 + \cdots + a_q u^q \Rightarrow z = f(u),$$

and suppose that a sufficient number of measurements $z_i = f(u_i)$, $i = 1, \dots, n$ and $n \geq q + 1$, is collected and stored in following matrix form

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^q \\ 1 & u_2 & u_2^2 & \cdots & u_2^q \\ \vdots & \vdots & \ddots & & \vdots \\ 1 & u_n & u_n^2 & \cdots & u_n^q \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_q \end{bmatrix} = Ua,$$

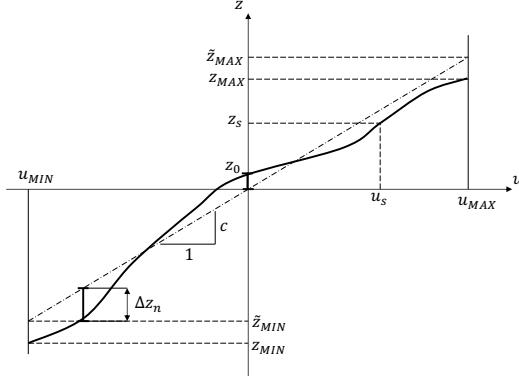


Figure 10.4: Classic quasi-static sensor response.

where of course $z \in \mathbb{R}^n$, $U \in \mathbb{R}^{n \times q+1}$ and $a \in \mathbb{R}^{q+1}$. To find the best *estimated* model $\hat{f}(u)$ it is possible to minimise the *mean squared error* between the measurements and $\hat{f}(u)$, i.e.

$$\arg \min_{\hat{f}(\cdot)} \frac{1}{n} \sum_{i=1}^n (z_i - \hat{f}(u_i))^2,$$

the mean of the sum of squares of the error $z_i - \hat{f}(u_i)$, $\forall i$. Graphically speaking, we are trying to fit a function in the data that better approximates *all* the available data. Hence, the problem is turned into a *parametric estimation* problem since we are searching for the set of *calibration parameters* $a = [a_0, a_1, \dots, a_q]^T$ that minimises

$$J(\hat{a}) = \sum_{i=1}^n [z_i - (\hat{a}_0 + \hat{a}_1 u_i + \hat{a}_2 u_i^2 + \dots + \hat{a}_q u_i^q)]^2,$$

(notice that the constant $\frac{1}{n}$ does not play any role in the minimisation problem, thus it is removed) or equivalently in matrix form

$$J(\hat{a}) = (z - U\hat{a})^T (z - U\hat{a}) = z^T z - \hat{a}^T U^T z - z^T U\hat{a} + \hat{a}^T U^T U\hat{a}.$$

The optimal choice of the parameters is then given by

$$\hat{a} = \arg \min_{\hat{a} \in \mathbb{R}^{q+1}} J(\hat{a}).$$

This is an *unconstrained minimisation problem*, i.e. a *regression*. More precisely a *polynomial regression*, since we are searching for a polynomial model.

It is then sufficient to impose that the *gradient* of the cost index is equal to zero for the optimal value, i.e.,

$$\frac{dJ(\hat{a})}{d\hat{a}} = 0.$$

To compute the derivative we first notice to properties of the matrix product derivatives (obtained by simple algebra):

- Given $x, y \in \mathbb{R}^n$, $\frac{dx^T y}{dx} = \frac{dy^T x}{dx} = y$;
- Given $x \in \mathbb{R}^n$ and $B \in \mathbb{R}^{n \times n}$, $\frac{dx^T Bx}{dx} = Bx + B^T x$. If B is also *symmetric* (i.e., $B = B^T$), it follows that $\frac{dx^T Bx}{dx} = 2Bx$.

Therefore

$$\frac{dJ(\hat{a})}{d\hat{a}} = -2U^T z + 2U^T U \hat{a}.$$

Since the *Hessian* is given by

$$\frac{d^2 J(\hat{a})}{d\hat{a}^2} = 2U^T U,$$

which is positive definite, the minimum is given by imposing

$$\frac{dJ(\hat{a})}{d\hat{a}} = 0 \Rightarrow \hat{a} = (U^T U)^{-1} U^T z = U^\dagger z,$$

where U^\dagger is the *pseudo-inverse* of U .

Notice that U^\dagger exists whenever $U^T U$ is *full rank*, i.e., a number of *independent* measures $n \geq q$. Moreover, $U^\dagger = U^{-1}$ if $n = q$.

The solution thus found is *optimal* with respect to the cost index $J(\hat{a})$ chosen, which is the sum of the *squared residuals* between the measurements and the model, i.e., the sum of the squares of the errors made in the results of every single equation.

Hence, we have solved the *Least Squares* (LS) problem, which is a standard approach to the approximate solution of overdetermined systems. In practice, calibration is used to reduce at most the *measurement error*, which affects the *accuracy*.

With the previous Figure 10.3 in mind, we can draw some remarks:

- We *cannot* express a measurement result *with a single value*, because the same value will be barely obtained by a new measurement!
- The single measurement value *cannot* be used in a comparison.

- The *measurement result* is represented by the whole set of values!
- It is clear that the *measurement uncertainty*, i.e. the *random effects*, can be characterised by the *dispersion* of the data around the *measured quantity*.

10.1.2 Random effects

As reported in the VIM [21], it is now clear why a *measurement result* cannot be expressed by a single number. According to the GUM [23], to apply a *correct measurement* we have to ensure that: QUI

- The method to express the uncertainty should be *universal*, i.e. applicable to any situation and measurand;
- The expression of the uncertainty should be *internally consistent*, i.e. directly derivable from the components that contribute to it, and *transferable*, i.e. the uncertainty of a measurement should be derived by the components uncertainty.

Moreover the GUM states that the uncertainty should be provided by an *interval* with an associated *level of confidence* or *coverage probability*.

The GUM suggests to *model the measured values as realisations of a random variable*. Provided that the *systematic effects* are negligible, the GUM assumes that *the best estimate of the measurand is the mean value of the different realisations*. Hence, the distribution of the measured variables is represented by a *probability density function* (pdf) and the related *standard deviation* provides a quantitative estimate of its dispersion.

Definition 44 (GUM). *The standard uncertainty is the uncertainty of the result of a measurement expressed as a standard deviation.*

Usually, the *standard uncertainty* of the measured value z is denoted with $u(z)$, even though we will use interchangeably the classic notation σ_z . The *coverage probability* (or *level of confidence*) is expressed with respect to an interval about its mean value μ_z , i.e. $[\mu_z - u(z), \mu_z + u(z)]$. For example, if the pdf is Gaussian, the *coverage probability* is given by the well known 68.27%. Since the GUM requires that the *coverage interval* comprises a *large* fraction of the possible values of the measurand:

Definition 45 (GUM). *The expanded uncertainty is the quantity defining an interval about the result of a measurement that may be expected to encompass a large fraction of the distribution of values that could reasonably be attributed to the measurand.*

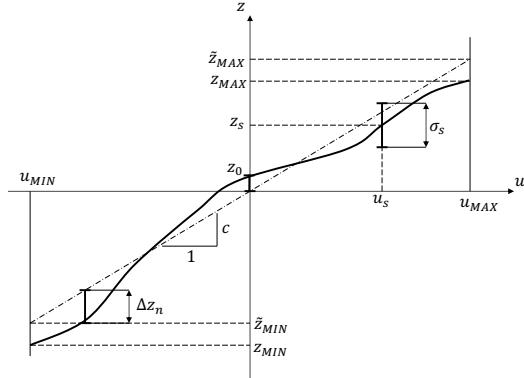


Figure 10.5: Classic quasi-static sensor response with *uncertainties*.

The *large fraction* can be viewed as the *coverage probability*, hence the *expanded uncertainty* $U(z)$ exists if and only if the pdf is *known*. The *expanded uncertainty* is given with a *coverage factor*. For the previous example, $[\mu_z - U(z), \mu_z + U(z)]$ with 99.73% if the *coverage factor* is 3.

In the GUM, there are two ways in which the uncertainty can be evaluated: *Type A* and *Type B* evaluation.

Definition 46 (GUM). *Type A evaluation method of uncertainty uses the statistical analysis of a series of repeated observations.*

When a measure is carried out, the result is represented by a *numeric array* or a *function*. To measure the *uncertainty* one may use the *measurement error*

$$w_i = z_i - z_a,$$

being z_i the i -th value that is measured and z_a the actual value. For example, $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures $z_i = 24.8^\circ$ ($w_i = -0.2^\circ$). It has to be noted that this approach cannot be actually pursued since the *actual value* of a measure is *ideal* and is not accessible, i.e. the actual value of the *measurand* is *unavailable*.

By computing n times the measure, we obtain a series of *random values*, generated by the time-varying nature of any measurement process, i.e. the *random effects*. These are due by the limited *resolution*, *precision* and *accuracy* of the measurements. For example, $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of $Z = [24.8^\circ, 23.9^\circ, 25.1^\circ, 24.9^\circ, 25.3^\circ]$.

To the sequence of *measurement results* it is possible to associate the *statistical description*. With respect to the sensor response in Figure 10.4, we now have to add an additional component, as depicted in Figure 10.5.

To better explain the Type A analysis, suppose we have a sequence of n measures belonging to a certain set $\mathcal{Z} \subset \mathbb{R}$. The n sampled values will

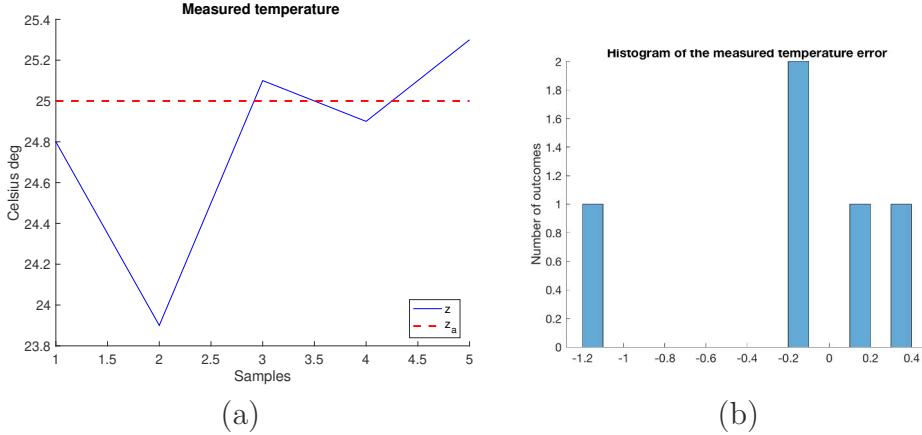


Figure 10.6: Examples of the temperatures: $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of $Z = [24.8^\circ, 23.9^\circ, 25.1^\circ, 24.9^\circ, 25.3^\circ]$. (a) Time evolution, (b) Histogram.

be ordered along one axis, e.g., the time axis, and then *collected* according to a constant interval Δ_z . For any $z \in \mathcal{Z}$, the function $N_\Delta(z)$ returns the number of samples belonging to the interval Δ_z to which z belongs. As a consequence it is possible to define the *histogram* as the following function:

$$P_\Delta(z) = \frac{1}{\Delta_z} \frac{N_\Delta(z)}{n}.$$

Notice how $\Delta_z P_\Delta(z)$ is *numerically equivalent* to the probability of having z in a certain interval Δ_z . An example of an histogram for the example of the thermometer is reported in Figure 10.6. With a larger data set, the histogram becomes more visible in its shape, as reported in Figure 10.7 and Figure 10.8 for two different distributions of the uncertainties. In the three cases, we can identify two different situations. In the case of Figure 10.6, there are *too few* observations to clearly identify the underlying statistical behaviour, hence the *Type A evaluation* cannot be carried out (more on this in the next slides). Instead, in Figure 10.7 and Figure 10.8, the mean value approaches the actual value, i.e. no *systematic effects* are present. Hence, a *Type A evaluation* is feasible. Recall that if the *systematic effects* are still there, the analysis of the *Type A evaluation* cannot be carried out, as in the case of Figure 10.9, where a bias is clearly visible.

Let us clarify this issue with an example. As stated previously and according to the GUM, the *the best estimate of the measurand is the mean value* provided that the *systematic effects* are negligible. Hence, recalling the definition of the *measurement error*

$$w_i = z_i - z_a,$$

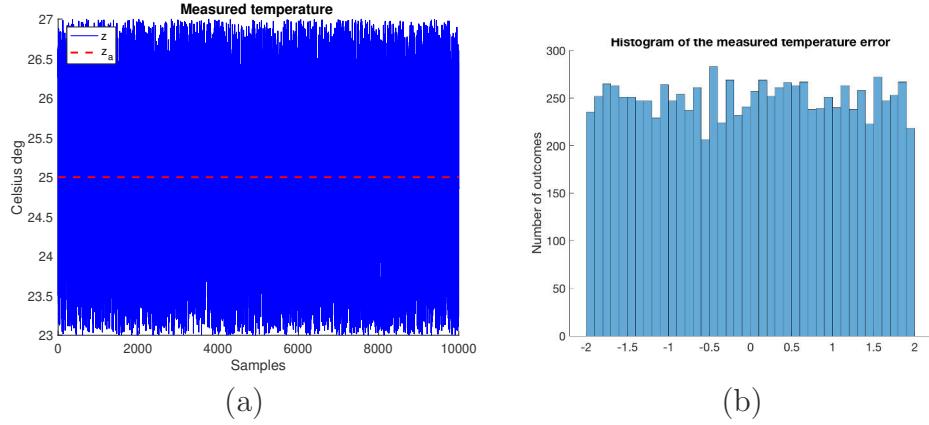


Figure 10.7: Examples of the temperatures: $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of 10^4 samples $z \in [23^\circ, 27^\circ]$. *Confidence level* 100%. (a) Time evolution, (b) Histogram.

we can state that the *systematic effects* o can be given by

$$o = \frac{1}{n} \sum_{i=1}^n w_i.$$

Indeed, defining $\bar{z}_i = z_i - o$ as the *measurement with compensated and negligible systematic effects*, we have

$$\frac{1}{n} \sum_{i=1}^n \bar{z}_i = z_a.$$

To further analyse the choice of the *arithmetic mean*, i.e. the *mean value*, in the *Type A evaluation*, let us make an example. Consider the *mean squared error* (MSE)

$$m^2 = \frac{\sum_{i=1}^n (z_i - z_a)^2}{n},$$

from which we can also compute the *root mean squared error* (RMSE), i.e. the square root of the MSE, also known as *mean error*. The question now is: *what is a good estimate of z_a , i.e., \hat{z}_a or simply \hat{z} , obtained using the measurements and how to compute it?* Notice that whenever an estimate \hat{z} is available, an estimate of the quantity of interest \hat{x} is available due to *sensor calibration*. Therefore, the more $\hat{z} \rightarrow z_a$, the more $\hat{x} \rightarrow x_a$. Hence, how to find \hat{z} using z_i is the first example of an *estimator design*!

An *estimate* \hat{z} can be the value that *minimises* the the MSE $m_{\hat{z}}^2$ of the estimated value. In this case, this is called the *Minimum MSE* (MMSE)

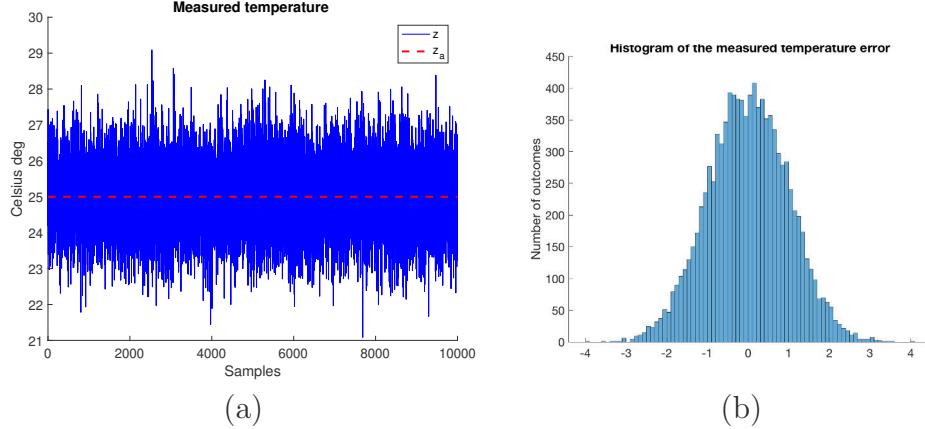


Figure 10.8: Examples of the temperatures: $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of 10^4 samples $z \in [20^\circ, 30^\circ]$. *Confidence level* 99.9%. (a) Time evolution, (b) Histogram.

estimator, i.e. the one that maximises the *precision* of the estimates. In analytical terms:

$$\hat{z} = \arg \min_{\hat{z}} \sum_{i=1}^n (z_i - \hat{z})^2.$$

The solution to this problem is given exactly by the *arithmetic mean*, i.e.,

$$\hat{z} = \frac{\sum_{i=1}^n z_i}{n},$$

computed nullifying the first derivative with respect to the estimates \hat{z} .

Definition 47. We will denote with

$$\hat{z}[z_1, \dots, z_n] = \frac{\sum_{i=1}^n z_i}{n},$$

the *estimator*, i.e. the function or set of operations generating the estimates.

In this case the *estimator* is the *arithmetic mean*.

Definition 48. We will denote with \hat{z} the *estimates* of z_a , i.e. the result of the *estimator* applied to the available measurements.

Remark 3. This solution has been obtained finding a solution to a maximum likelihood (ML) criterium, since the values around \hat{z} have the maximum probability to be measured.

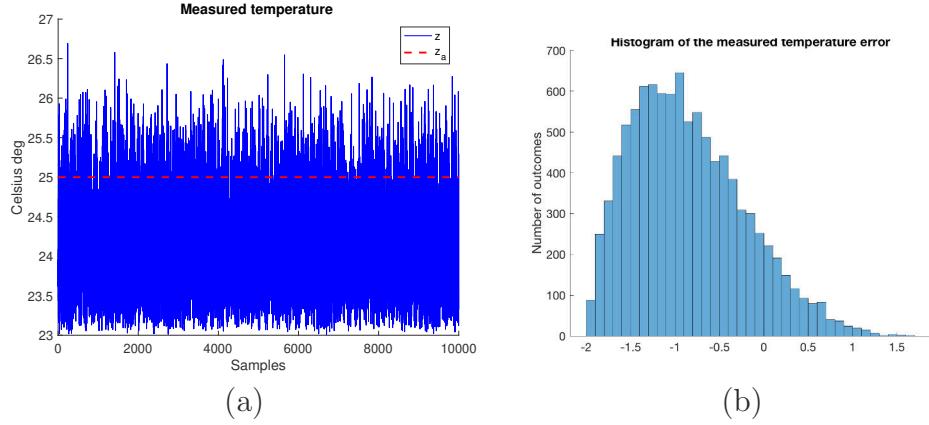


Figure 10.9: Examples of the temperatures: $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of 10^4 samples $z \in [23^\circ, 27^\circ]$. *Confidence level* 100%. (a) Time evolution, (b) Histogram. This sensor *has systematic effects*.

Remark 4. *This estimate is centred since if the set of the infinite measures is split into m subsets upon which the mean is computed, the mean of all the mean would be exactly the value z_a of interest.*

We have proved that the *arithmetic mean* is the optimal estimator with respect to the MSE when the *systematic effects* can be *neglected* (recall the GUM definition). We are now in a position to establish what is the *precision* of this estimator, i.e., the MSE attained by the *arithmetic mean*. The objective is to compute the MSE of \hat{z} , which we will denote with $m_{\hat{z}}^2$. First, let us notice that \hat{z} depends on a set of *measurement values* z_i that are affected by *uncertainties*, i.e.

$$\hat{z} = \arg \min_{\hat{z}} \sum_{i=1}^n (z_i - \hat{z})^2.$$

The previous observation, and what follows, is valid if we consider \hat{z} either as an *estimate*, i.e. the result of an *estimator*, or an *indirect measurement*, i.e. a measurement as a function or combination of other measurements, in general $\hat{z} = f(z_1, z_2, \dots, z_n)$.

If $f(z_1, z_2, \dots, z_n)$ is a function of the observed quantities z_1, z_2, \dots, z_n , whose *actual values* are $z_{a1}, z_{a2}, \dots, z_{an}$, it is possible to consider its Taylor expansion around those points, i.e.,

$$\begin{aligned} f(z_1, z_2, \dots, z_n) &= f_a + \frac{\partial f}{\partial z_1}(z_1 - z_{a1}) + \frac{\partial f}{\partial z_2}(z_2 - z_{a2}) + \dots + \\ &+ \frac{1}{2} \frac{\partial^2 f}{\partial z_1^2}(z_1 - z_{a1})^2 + \frac{1}{2} \frac{\partial^2 f}{\partial z_2^2}(z_2 - z_{a2})^2 + \dots \end{aligned}$$

where $f_a = f(z_{a_1}, z_{a_2}, \dots, z_{a_n})$. Assuming that the errors $z_i - z_{a_i}$ are small enough, we can limit the Taylor expansion up *to the first order*. Therefore

$$f(z_1, z_2, \dots, z_n) - f_a = \sum_{i=1}^n \frac{\partial f}{\partial z_i} (z_i - z_{a_i})$$

and applying the squares

$$\begin{aligned} (f(z_1, z_2, \dots, z_n) - f_a)^2 &= \sum_{i=1}^n \left(\frac{\partial f}{\partial z_i} \right)^2 (z_i - z_{a_i})^2 + \\ &+ 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{\partial f}{\partial z_i} \frac{\partial f}{\partial z_j} (z_i - z_{a_i})(z_j - z_{a_j}) \end{aligned}$$

Assuming that the measurement processes generating z_i and z_j , $\forall i \neq j$, are *independent*, we can neglect all the mixed terms. We can now compute the MSE of $f(\cdot)$ using the MSE of each measurement, i.e.

$$m_f^2 = \sum_{i=1}^n \left(\frac{\partial f}{\partial z_i} \right)^2 m_{z_i}^2$$

where $m_{z_i}^2$ is the MSE of the estimate related to the i -th measure. In the particular case of the mean function

$$f(z_1, z_2, \dots, z_n) = \hat{z} = \frac{\sum_{i=1}^n z_i}{n} \Rightarrow \frac{\partial f}{\partial z_1} = \frac{\partial f}{\partial z_2} = \dots = \frac{1}{n}.$$

Moreover, we have the same MSE for all the measures, i.e., $m_{z_1}^2 = m_{z_2}^2 = \dots = m^2$. Therefore,

$$m_f^2 = m_{\hat{z}}^2 = \sum_{i=1}^n \left(\frac{\partial f}{\partial z_i} \right)^2 m_{z_i}^2 = \sum_{i=1}^n \frac{m^2}{n^2} = \frac{m^2}{n},$$

where we have to remember that m^2 is the MSE of the *measurement results*. Notice that \hat{z} , that is the *estimate* of z_a , is a *random variable* and, more importantly, The more measurement values we have, the less is the MSE!

Recall that the MSE of all the measures m^2 is given by

$$m^2 = \frac{\sum_{i=1}^n (z_i - z_a)^2}{n},$$

which *involves the knowledge of z_a* that is *not available*. This is the reason why the concept of *measurement error*, i.e. $w_i = z_i - z_a$, cannot be used *when*

the observations are considered, because it is *philosophically incorrect*: the *actual value* of a quantity is an abstract representation of reality! To solve this problem, we first notice that $m_{\hat{z}}^2 = \frac{m^2}{n}$ can be computed by definition as $m_{\hat{z}}^2 = \frac{1}{n} \sum_{i=1}^n (\hat{z} - z_a)^2 = (\hat{z} - z_a)^2$, hence

$$\begin{aligned} nm^2 &= \sum_{i=1}^n (z_i - z_a)^2 = \sum_{i=1}^n (z_i - \hat{z} + \hat{z} - z_a)^2 = \sum_{i=1}^n (z_i - \hat{z} + m_{\hat{z}})^2 \\ &= \sum_{i=1}^n (z_i - \hat{z})^2 + m_{\hat{z}}^2 + 2m_{\hat{z}}(z_i - \hat{z}). \end{aligned}$$

Since $\sum_{i=1}^n (z_i - \hat{z}) = 0$ by definition, one has

$$nm^2 = \sum_{i=1}^n (z_i - \hat{z})^2 + \sum_{i=1}^n m_{\hat{z}}^2 = \sum_{i=1}^n (z_i - \hat{z})^2 + nm_{\hat{z}}^2 = \sum_{i=1}^n (z_i - \hat{z})^2 + m^2.$$

Finally

$$m^2 = \frac{1}{n-1} \sum_{i=1}^n (z_i - \hat{z})^2.$$

Notice that the MSE of the *measurement values* can be computed *without* the knowledge of the actual value. This equation explains why, with *Type A evaluation*, the *arithmetic mean* is considered as the central value to compare all the measurements.

Let us go back to the *histogram* definition:

$$P_{\Delta}(z) = \frac{1}{\Delta_z} \frac{N_{\Delta}(z)}{n}.$$

and recall that $\Delta_z P_{\Delta}(z)$ is *numerically equivalent* to the probability of having z in a certain interval Δ_z . Defining the infinite set $\mathcal{D} = \cup_i \Delta_{z_i} = \mathbb{R}$, we get that $\Delta_{z_i} P_{\Delta}(z)$ is always nonnegative and $\sum_i \Delta_{z_i} P_{\Delta}(z) = 1$, i.e., the *overall area covered by the histogram* sums up to 1. In practice, if $n \rightarrow +\infty$, it is possible to let $\Delta_z \rightarrow 0$ and hence the *discrete function* $P_{\Delta}(z)$ turns to be a *continuous function* named *probability density function* (pdf) $p(z)$. The *distribution function*, homologous of $\sum_i \Delta_{z_i} P_{\Delta}(z)$, is instead the integral of the pdf and, hence, expresses the probability, i.e.,

$$P(\bar{z}) = \Pr [z < \bar{z}] = \int_{-\infty}^{\bar{z}} p(z) dz,$$

or, in general,

$$P(\bar{z}'') - P(\bar{z}') = \Pr [\bar{z}' < z < \bar{z}''] = \int_{\bar{z}'}^{\bar{z}''} p(z) dz.$$

Assuming that *the actual value exists*, we can consider $w = z - z_a$. Assume the following assumptions: a) the *measurement error* is *symmetric*, i.e., $p(w)$ is symmetric; b) *small errors* are more frequent than *large errors*; c) the *greater* is the number of small errors, the *smaller* is the number of large errors. In such a case, the pdf turns to be a *Gaussian* (or *Normal*) pdf (see histogram Figure 10.8 for an example of this case), i.e.,

$$p(w) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{w^2}{2\sigma^2}}.$$

σ is the *standard deviation* and it is a function of the *precision* of the system, i.e. *standard uncertainty*. Again, it is evident that the *smaller* is the standard deviation, the *more precise* will be the measurement value. In particular, the square of the standard deviation, i.e., the *variance*, is given by

$$\sigma^2 = \lim_{n \rightarrow +\infty} \frac{\sum_{i=1}^n (z_i - z_a)^2}{n}.$$

Notice that σ^2 can be known *up to a certain accuracy* since n is finite.

The MSE

$$m^2 = \frac{1}{n-1} \sum_{i=1}^n (z_i - \hat{z})^2$$

is a *statistically consistent* estimate of the *variance* σ^2 when the number of measurements n is finite and the *mean value* is null. In other words, for $n \rightarrow +\infty$, the *probability that m^2 differs from σ^2 tends towards zero*. Indeed, for $n \rightarrow +\infty$, $\hat{z} \rightarrow z_a$ (recall that $m_{\hat{z}}^2 = \frac{m^2}{n}$). The fact that $n \rightarrow +\infty$ exemplifies why for the example in Figure 10.6 we cannot carry out the *Type A Evaluation*: there is an *insufficient* number of samples.

Sometimes it happens that the *systematic errors* cannot be recognised or compensated. Moreover, it may happen that *it is not possible* to get access to observations. In those cases, we can resort to *Type B evaluations*:

Definition 49 (GUM). *Type B evaluation method of uncertainty by means other than the statistical analysis of a series of repeated observations.*

According to the GUM, the possible sources of information are:

- Previous measurement data;
- Experience or previous knowledge;
- Manufacturer's specifications;
- Data provided in calibration and other certificates;

- Uncertainties assigned to reference data taken from handbooks.

Moreover, assumptions on a) the pdf of the *random effects* and b) the *coverage probability* are needed. With those information, the *standard uncertainty* can be computed. For example, assume that a thermometer measure $T_m = 24.845^\circ$ Celsius and that the manufacturer reports a range of possible values within 0.04% of the reading. What is the *standard uncertainty*? We first have that $\tau = 24.845 \cdot 4 \cdot 10^{-4} = 9.9 \cdot 10^{-3}$. Then, $T_m \pm \tau$ encompasses all the possible values with *coverage probability* 100%. Then, if no assumption can be made on the pdf, the maximum entropy assumption is taken (i.e., *uniform pdf*), finally having $u(T_m) = \frac{\tau}{\sqrt{3}} = 5.7 \cdot 10^{-3}$.

When an estimate of both types of uncertainty is available, say $u_a(z)$ and $u_b(z)$, the combined uncertainty is given by:

$$u(z) = \sqrt{u_a(z)^2 + u_b(z)^2}.$$

In the previous example of the MSE, we noticed that when one *measurement value* is a nonlinear combination of other measurements $z = f(x_1, \dots, x_n)$ (i.e. *indirect measurement*), $u(z)$ can be determined using the Taylor expansion (e.g. the velocity inferred from position measurements). However, the *coverage probability* is difficult to be computed since the pdf of z can be *hardly computed*. This is a crucial problem also for *estimators*, as we will see in the next classes. In those cases, *Monte Carlo simulations* should be used, unless the *Central Limit theorem* can be applied.

10.1.3 Modelling

Let us talk about the *modelling* of the measurement process. By computing n times the measure, we obtain a series of *random values*, generated by the time-varying nature of any measurement process, i.e. the *random effects*. This is due by the limited *resolution*, *precision* and *accuracy* of the measurements, which are affected by a *random measurement noise*, e.g. ε . For example, $z_a = 25^\circ$ Celsius is the actual temperature, while the thermometer measures a sequence of $z = [24.8^\circ, 23.9^\circ, 25.1^\circ, 24.9^\circ, 25.3^\circ]$. In other words

$$z = z_a + \varepsilon = 25^\circ + [-0.2^\circ, -1.1^\circ, 0.1^\circ, -0.1^\circ, 0.3^\circ],$$

where ε is the sequence of *measurement noises*. To the sequence of *measurement noises* ε it is possible to associate a *statistical description*. This fact is exemplified in Figure 10.10, subsuming the differences of the interpretations of the measurement process.

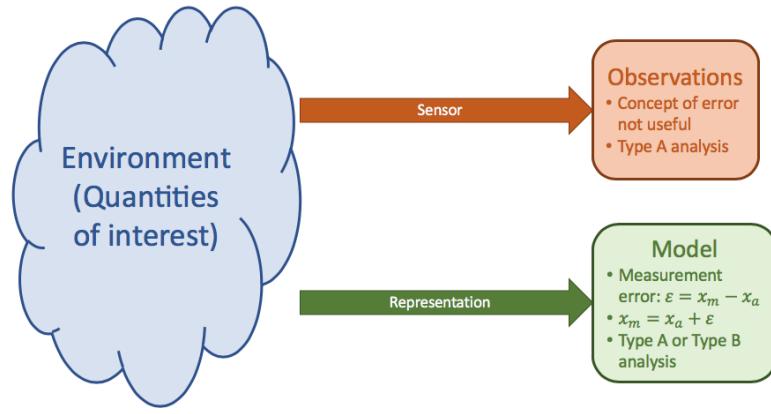


Figure 10.10: Two possible views of the environment using *observations* or using a suitable *model*.

10.2 Estimation Algorithms

In practice, *accuracy* is related to the *closeness of the mean value of repeated measurements* to the actual value, while *precision* is related to the *dispersion of repeated measurements* with respect to the actual value. Roughly speaking, while *accuracy* can be improved by calibration, *precision* asks for *estimation algorithms* on a *calibrated* sensor (recall Figure 10.3). Let us assume that we want to *estimate* (that is to *increase our knowledge*) about a certain quantity α . A *measurement process* can be formally defined as

$$z = g(\alpha, w, t),$$

where $g(\cdot)$ can represent either a *direct* measurement process, e.g. the thermometer measuring the temperature, or an *indirect* measurement, e.g. the velocity inferred from position measurements. The data $z \in \mathbb{R}^m$ are the *measurements*. The vector $w \in \mathbb{R}^m$ is the *measurement noise*. t expresses the *time variability* of the measurement process. Given the *measurement process* $g(\cdot)$ we usually define a *model* for it as well as for the quantities α . The model of the quantities α is usually represented by a vector $x \in \mathbb{R}^n$. An example is the representation by means of position and velocity. Notice that the model is crucial, since there might be models that *better fits the specific system at hand*, e.g. change of coordinates or Euler angles. Therefore, we need always to think carefully to the quantities to be measured and to their representation.

For the *measurement model*, we usually adopt a *static map*

$$z(k) = h(x(k)),$$

where the index k represents the k -th measurement (e.g., the sample taken at time $t = k\Delta_t$, $k \in \mathbb{N}$, with *sampling time* Δ_t). For example, assume that you are able to measure the *distance* (i.e. the quantity α) of an object in the plane with coordinates (x, y) from a wireless anchor of position (x_a, y_a) using the *Time of Flight*. In such a case the forward map is given by

$$z(k) = \sqrt{(x - x_a)^2 + (y - y_a)^2} = h(x, y, x_a, y_a).$$

How does the *static map model* $h(x)$ relates to the *measurement process* $g(\alpha)$? Whenever we have a *model* we have a *modelling error*! The problem of determining the *modelling error* is a complex problem per se. In the ideal case (no measurement error), the *modelling error* can be represented as

$$e_h = h(x(\alpha)) - g(\alpha).$$

Usually, the measurement error is *larger* than the modelling error, and this is what we will assume in the following. The model should be simple enough and in the *easiest case* is *linear*, i.e.

$$z(k) = Hx(k).$$

If this is not the case, usually a *Taylor approximation* is employed. Usually in the linear model $H \in \mathbb{R}^{m \times n}$. The objective of the *estimator* is to retrieve $x \in \mathbb{R}^n$ from $z \in \mathbb{R}^m$. If $n < m$ the problem is *over-determined* and the problem is usually solvable in the linear and nonlinear cases; if $n = m$ the problem is more involved since it may happens that it does not have solutions for the linear and the nonlinear cases; finally, if $n > m$ the problem is not solvable unless *prior knowledge* is available.

The problem of finding an *estimation algorithm* to determine the value of a certain (scalar or vector) unknown parameter x depends on the *nature* of x . If the parameter x is *time-invariant*, then *static estimation algorithms* (such as the WLS) are adopted, otherwise *dynamic estimation algorithm* (such as the Kalman Filter). In more strict terms, the problem of estimation of x given the set of measures

$$z(j) = h(j, x, w(j)), \quad j = 1, \dots, k, \tag{10.1}$$

where $w(j)$ is a sequence of measurement noises (or *disturbances*) amounts to find the *estimates*

$$\hat{x}(k) = \hat{x}[k, Z^k],$$

being Z^k the set of all measures up to time k , i.e.,

$$Z^k = \{z(j)\}_{j=1,\dots,k}.$$

Recall that the $\hat{x}(k)$ are the *estimates* of x , the function $\hat{x}[k, Z^k]$ is instead the *estimator*.

Example 13. Suppose

$$z(j) = x + w(j), \quad j = 1, \dots, k,$$

where $w(j)$ is the measurement noise.

Some possible estimators are:

- $\hat{x}(k) = \frac{1}{k} \sum_{j=1}^k z(j);$
- $\hat{x}(k) = \frac{\max(Z^k) + \min(Z^k)}{2};$
- $\hat{x}(k) = z(1).$

Each of the previous estimators works, but is there a systematic way to design an estimator? Using just one value it is impossible to judge: by chance we can derive the incorrect answer! A statistical analysis is needed (recall the definition of measurement result).

The *measurement error* w is the error affecting the measures at time k

$$z(k) = x(k) + w(k), \quad (10.2)$$

where $w(k)$ is a *random variable*. The *estimation error* at time k is instead given by

$$\tilde{x}(k) = x(k) - \hat{x}(k), \quad (10.3)$$

where $\hat{x}(k)$ is a function of the measurements $z(k), \forall k$.

To design an *estimator* we resort to the notice that the role of an estimator is to retrieve a *correct estimate* $\hat{x}(k)$, possibly the *best estimate*, i.e. the one with the smallest *estimation error* \tilde{x} . The information available are: a) the *measurements* $z(k)$, which are *noisy*; b) an idea about the *system output model*, i.e. the function $h(\cdot)$; c) maybe, some *additional knowledge*, e.g. the quantity is always positive. Moreover, *all* the available knowledge *must be exploited*, so the estimator needs to incorporate it properly and we can treat or not the quantity to estimate as a *random variable*.

Chapter 11

Background on Statistics

11.1 Probability

Consider an *experiment* with random outcomes.

Definition 50 (Event). *An event is a set of the outcomes of the experiment. The event occurs if the outcome is one of the element of the set.*

Definition 51 (Probability (Kolmogorov)). *The probability $Pr[\cdot]$ of an event A is a measure that satisfies the three axioms of probability:*

1. $Pr[A] \geq 0$;
2. $Pr[S] = 1$ if S is the **sure event**;
3. If $A \cap B = \emptyset$, then $Pr[A \cup B] = Pr[A + B] = Pr[A] + Pr[B]$, i.e., additivity with respect to **mutually exclusive events**.

As a consequence, $Pr[\bar{A}] = 1 - Pr[A] \leq 1$, where \bar{A} is the **complementary event**. Moreover, the **impossible event** is the complementary of S , i.e., $Pr[\emptyset] = 0$.

Remark 5. A possible event is a subset of S .

Remark 6. The additivity of mutually exclusive events, i.e., **countable additivity**, is necessary when the experiments has an infinite number of outcomes.

There are usually two different views of probability:

- The **frequentist** interpretation: the probability of something corresponds to what fraction of the time it happens “in the long run”. Although engineering sound, there are some systems in which this approach fails. For example, it is not clear how to apply the frequency approach to the Earth’s gravitational field since there is only one field;

- The *Bayesian* interpretation: the probability of something corresponds to how likely we “think” it is to happen.

The *Relative-frequency* concept, aka *measure-of-belief*, is a *non-rigorous* way to introduce the concept of probability, which are widely used in engineering systems. For example, we can explain the probability of each face of the dice by means of empirical measures.

Let us consider the joint experiment of *tossing a coin* and *extract a red card from a card deck*. Each event can be decomposed into the following sets: $C = \text{head}$, $\bar{C} = \text{tail}$ and $R = \text{red card}$, $\bar{R} = \text{black card}$. The sure event can be decomposed into the following set: $\{CR, C\bar{R}, \bar{C}R, \bar{C}\bar{R}\}$. After n experiments, we would have for the relative frequency $f(\cdot)$:

$$f(CR) = \frac{n_1}{n}, f(C\bar{R}) = \frac{n_2}{n}, f(\bar{C}R) = \frac{n_3}{n}, \text{ and } f(\bar{C}\bar{R}) = \frac{n_4}{n}.$$

From the empirical definition, the event *head or red card* is given by

$$f(C+R) = f(CR) + f(C\bar{R}) + f(\bar{C}R) = \frac{n_1 + n_2 + n_3}{n} = \frac{n_1 + n_2 + n_3 + n_1 - n_1}{n}.$$

Noticing that we have a head if either CR or $C\bar{R}$ and a red card if either CR or $\bar{C}R$, we get

$$f(C) = f(CR) + f(C\bar{R}) = \frac{n_1 + n_2}{n}, \text{ and } f(R) = f(CR) + f(\bar{C}R) = \frac{n_1 + n_3}{n}.$$

Substituting into the previous yields to

$$f(C+R) = \frac{n_1 + n_2 + n_3 + n_1 - n_1}{n} = f(C) + f(R) - f(CR).$$

The same results obtained using the empirical relative frequencies can be derived using Axiom 3. Indeed,

$$C = (C \cap R) \cup (C \cap \bar{R}), R = (C \cap R) \cup (\bar{C} \cap R), C \cup R = (C \cap R) \cup (C \cap \bar{R}) \cup (\bar{C} \cap R).$$

Notice that are all *mutually exclusive*, hence we have (by simple summation and subtraction)

$$\Pr [C \cup R] = \Pr [C] + \Pr [R] - \Pr [C \cap R],$$

which is the *total probability law*, where $\Pr [C \cap R]$ is the *joint probability*.

The probability of an event can be affected by the *a-priori* knowledge of some results of the experiment. For example, let us consider the probability of the event C knowing that the event R *has been already verified*, i.e.,

tossing the coin after that the red card has been selected. Using the relative frequency, we have

$$f(C|R) = \frac{n_1}{n_1 + n_3} = \frac{n_1}{n} \frac{n}{n_1 + n_3} = \frac{f(CR)}{f(R)}.$$

Similarly

$$f(R|C) = \frac{f(CR)}{f(C)}.$$

Hence, using the axioms of probability, we can define the *conditional probability* as

$$\Pr [C|R] = \frac{\Pr [C \cap R]}{\Pr [R]}.$$

Indeed, $\Pr [A|B]$ satisfies Axiom 1, since $\Pr [A \cap B] \geq 0$ and $\Pr [B] > 0$ (since it happened). Since $\Pr [S|A] = \Pr [A]/\Pr [A] = 1$, it also satisfies Axiom 2.

Finally, if A_1, A_2, \dots are *pair-wise mutually exclusive* events, one has

$$\begin{aligned} \Pr [A_1 \cup A_2 \cup \dots | B] &= \frac{\Pr [(A_1 \cup A_2 \cup \dots) \cap B]}{\Pr [B]} = \frac{\Pr [(A_1 \cap B) \cup (A_2 \cap B) \cup \dots]}{\Pr [B]} = \\ &= \frac{\Pr [(A_1 \cap B)]}{\Pr [B]} + \frac{\Pr [(A_2 \cap B)]}{\Pr [B]} + \dots = \Pr [A_1|B] + \Pr [A_2|B] + \dots \end{aligned}$$

which satisfies Axiom 3.

In general, since

$$\Pr [A|B] = \frac{\Pr [A \cap B]}{\Pr [B]},$$

one has

$$\Pr [A \cap B] = \Pr [A|B] \Pr [B] = \Pr [B|A] \Pr [A].$$

As a consequence, if A and B are *statistically independent*, we get

$$\Pr [A \cap B] = \Pr [A] \Pr [B].$$

Moreover, we can rewrite

$$\Pr [A \cap B] = \Pr [A|B] \Pr [B] = \Pr [B|A] \Pr [A],$$

as

$$\Pr [A|B] = \frac{\Pr [B|A] \Pr [A]}{\Pr [B]}.$$

Generalising to n *disjoint* events A_i , $i = 1, \dots, n$, we have

$$\Pr [A_i|B] = \frac{\Pr [B|A_i] \Pr [A_i]}{\Pr [B]} \equiv \frac{\Pr [B|A_i] \Pr [A_i]}{\sum_{j=1}^n \Pr [B|A_j] \Pr [A_j]},$$

which is the *Bayes Theorem*, having three different actors playing a crucial role:

- $\Pr[A_i]$ is the *prior probability* of A_i (the probability of A_i known up-front);
- $\Pr[B|A_i]$ is the *likelihood* of B given A_i (how likely B happens when A_i happens);
- $\Pr[A_i|B]$ is the *posterior probability* of A_i given B (the probability of A_i knowing that B has actually happened).

This is the basis of the *Bayesian inference*.

11.1.1 Bayesian inference example: The Monty Hall Problem

Let us consider the *Monty Hall problem*, a tv show of the 70s. There are 3 closed doors. Behind one door there is one million dollars, behind the other two doors there are goats. After our first choice, the game-master opens one door with a goat behind. In the actual tv show, Monty Hall did it just for increasing the pathos. However, in the version of the statistician *Steve Selvin*, after that the game-master opens the door with a goat, he asks us if we want to *switch the door*. The problem by Steve Selvin can be interpreted in light of the *Bayes theorem*.

Indeed, let us define the doors with D_i , $i = 1, 2, 3$. Let us then define the event “The million dollar is behind door D_i ” with $\$_i$, and the event “Monty opens the door D_i ” with M_i . We assume that we select door D_1 and that Monty opens door D_2 . Then, he asks us if *we want to change*, i.e. switch to door D_3 . Using the *Bayes theorem*, we want to compute what is the most convenient choice, i.e. the *highest* between $\Pr[\$_1|M_2]$ and $\Pr[\$_3|M_2]$. We have that

$$\Pr[\$1|M_2] = \frac{\Pr[M_2|\$1] \Pr[\$1]}{\Pr[M_2]},$$

and

$$\Pr[\$3|M_2] = \frac{\Pr[M_2|\$3] \Pr[\$3]}{\Pr[M_2]}.$$

For the priors, i.e. *prior to making our choice*, we have that $\Pr[\$1] = \Pr[\$2] = \Pr[\$3] = \frac{1}{3}$. Similarly, before Monty makes its move and considering we have selected the door D_1 , we have $\Pr[M_2] = \Pr[M_3] = \frac{1}{2}$. We know make use of the *likelihood*, i.e. the *evidence* we can collect from Monty choice. The question is, how likely Monty would have opened door D_2 knowing that *we selected D_1* if the one million dollar *is behind D_1* ? More precisely, what is $\Pr[M_2|\$1]$? It is obvious that $\Pr[M_2|\$1] = \frac{1}{2}$. Similarly, how likely

Monty would have opened door D_2 knowing that *we selected D_1* if the one million dollar *is behind D_3* ? More precisely, what is $\Pr[M_2|\$_3]$?

Following the same rationale, $\Pr[M_2|\$_3] = 1$. Therefore, we have that

$$\Pr[\$_1|M_2] = \frac{\Pr[M_2|\$_1]\Pr[\$_1]}{\Pr[M_2]} = \frac{\frac{1}{2}\frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}.$$

and

$$\Pr[\$_3|M_2] = \frac{\Pr[M_2|\$_3]\Pr[\$_3]}{\Pr[M_2]} = \frac{\frac{1}{2}\frac{1}{3}}{\frac{1}{2}} = \frac{2}{3}.$$

Notice that

$$\Pr[\$_2|M_2] = \frac{\Pr[M_2|\$_2]\Pr[\$_2]}{\Pr[M_2]} = \frac{0\frac{1}{3}}{\frac{1}{2}} = 0 \Rightarrow \Pr[\$_3|M_2] = 1 - \Pr[\$_1|M_2].$$

It is now evident that, when Monty asks us to switch, *we have to* in order to improve our chances of victory! Notice that we can find out the same solution using the frequency approach (*prove it!*).

In 1713 *Bernoulli* was facing the problem of *deductive logic* in games versus *inductive logic*, similar to what appears in real life. In a nutshell, he was facing the problem of reason in situations where *it is not possible to argue with certainty*. In a work that was published posthumously by a friend in 1763 *An Essay Towards Solving a Problem in Doctrine of Chances*, *Thomas Bayes* provided the answer. We can consider the variables as *random variables* and we can assign probabilities to them based on *our knowledge*: *Probability is something like a partial belief, rather than a frequency*.

This concept of *degree of belief* has been then rediscovered by *Laplace* in 1812. Laplace adopted the *Bayesian philosophy* in combination with observations and the law of mechanics to compute *the mass of Saturn*. Incredibly, after 150 years his results were changed less than 1%!

11.1.2 Random variables

In the majority of the engineering cases the variables are not described by sets but by *numbers*

Definition 52 (Random variable). A *random variable* is a (real-valued) function that assumes a certain *value* according to the outcome of a certain random experiment.

A random variable is *not* a *conventional variable*. For example, the number we got *after* rolling a dice is a variable, but the outcome we *expect* from

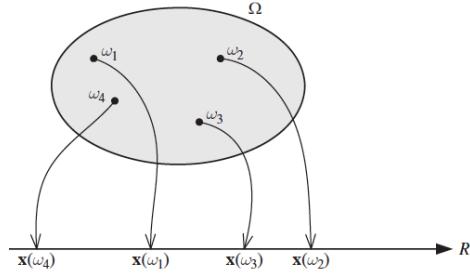


Figure 11.1: A random variable can be seen as a mapping from a sample space Ω to a continuous (discrete) set, e.g., the set \mathbb{R} (\mathbb{Z}) of real (integer) numbers (Courtesy of Ha H. Nguyen Ed Shwedyk - A First Course in Digital Communications, Cambridge University Press, 2009).

the rolling is a random variable (**rv**). More precisely, a **rv** maps the set containing all the possible outcomes of a certain experiment, i.e., the *event space* or *sample space* Ω , to the space of numbers. If the mapping is onto some continuous set (e.g., the \mathbb{R} set), the **rv** is continuous. If the mapping is onto some discrete set (e.g., the \mathbb{Z} set), the **rv** is discrete. In practice $\forall \omega \in \Omega$, we get that $x(\omega)$ is an element of the arriving set (e.g., a number). An example of the mapping of a **rv** is reported in Figure 11.1. To be completely described a **rv** needs a *probabilistic description*, that is given in terms of the *probability density function*.

Definition 53 (Probability density function). *The probability density function (pdf) of a scalar random variable x at a certain value $x = a$ is*

$$p(a) = \lim_{\delta_a \rightarrow 0} \frac{\Pr[a - \delta_a < x \leq a]}{\delta_a} \geq 0.$$

Notice the similarity with the *ratio of the increments* as the approximation of the derivative.

Since by the third axiom of probability

$$\Pr[a < x \leq b] = \int_a^b p(x)dx,$$

we have

Definition 54 (Cumulative distribution function). *The cumulative distribution function (cdf) of x at b is given by*

$$P(b) = \Pr[x \leq b] = \int_{-\infty}^b p(x)dx.$$

Therefore, in order to be a proper pdf, we have for the second axiom

$$\Pr [x \leq \infty] = \int_{-\infty}^{\infty} p(x)dx = 1.$$

Notice how

$$p(x) = \frac{dP(a)}{da} \Big|_{a=x}$$

Moreover,

$$\Pr [a < x \leq a + \delta_a] = \int_a^{a+\delta_a} p(x)dx,$$

that for sufficiently small δ_a means that

$$\Pr [a < x \leq a + \delta_a] = p(a)\delta_a + O(\delta_a^2),$$

so $p(x)$ represents the *density of probability* per unit value of a in the neighbourhood of a .

Hence, $p(x) \geq 0, \forall x$ (first axiom of probability). Furthermore, for a continuous **rv**

$$\Pr [x = a] = 0.$$

For the *cumulative distribution function*, we have that

$$\Pr [a < x \leq a + \delta_a] = P(a + \delta_a) - P(a),$$

which is verified by

$$P(a) = \Pr [x \leq a] = \int_{-\infty}^a p(x)dx.$$

The cdf has the following properties:

- $0 \leq P(a) \leq 1$;
- $\lim_{a \rightarrow -\infty} P(a) = 0$ and $\lim_{a \rightarrow +\infty} P(a) = 1$;
- $P(\cdot)$ is non-decreasing;
- $P(\cdot)$ is right continuous, i.e., $\lim_{\delta_a \rightarrow 0^+} P(a + \delta_a) = P(a)$.

Usually, all the pdfs of continuous **rvs** have the following form:

$$p(x) = \frac{1}{cN(s)} p\left(\frac{x-l}{c}\right),$$

where:

- l is a *location parameter*, that has the role to translate the pdf;
- c is a *scale parameter*, having the role of expanding or contracting the pdf. For normalisation purposes, its reciprocal multiplies the pdf;
- s is a *shape parameter*, that governs the shape of the pdf, and it is also an argument of the normalising function $N(s)$.

Example 14 (Uniform pdf). $x \sim \mathcal{U}(a, b)$ defines x as a uniformly distributed random variable in the interval $[a, b]$.

The pdf is given by

$$p(x) = \mathcal{U}(x; a, b) \triangleq \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b], \\ 0 & \text{otherwise.} \end{cases}$$

The cdf is instead given by

$$P(c) = \begin{cases} 0 & \text{if } c < a, \\ \frac{c-a}{b-a} & \text{if } c \in [a, b], \\ 1 & \text{if } c > b. \end{cases}$$

Example 15 (Gaussian pdf). $x \sim \mathcal{N}(l, c, N(s))$ defines x as a Gaussian (or normal) random variable with parameters $N(s)$, l and c .

The pdf is given by

$$p(x) = \mathcal{N}(x; l, c, N(s)) \triangleq \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-l)^2}{2c^2}},$$

where $N(s) = \sqrt{2\pi}$ comes from the normalisation of the pdf.

The cdf is instead given by

$$P(z) = \int_{-\infty}^z \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-l)^2}{2c^2}} dx = \int_{-\infty}^{\frac{z-l}{c}} \frac{1}{\sqrt{2\pi}} e^{-y^2} dy,$$

which is a widely studied integral, tabulated numerically, called cumulative standard Gaussian distribution.

Let us consider the discrete version of the continuous pdf:

Definition 55 (Probability mass function). The probability mass function (pmf) of a discrete-valued random variable x taking values in the set $\{a_i, i = 1, \dots, n\}$ is

$$\pi(a_i) = Pr[x = a_i] = \pi_i,$$

where π_i are the point masses. In order to be proper

$$\sum_{i=1}^n \pi_i = 1.$$

Using the property of the *Dirac delta function*, i.e.,

$$\delta(x) = 0, \quad \forall x \neq 0, \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) dx = 1,$$

the pdf corresponding to the pmf can be written as

$$p(x) = \sum_{i=1}^n \pi_i \delta(x - a_i).$$

Notice that the previous relation holds whenever the pdf is considered as a *generalised function*. If standard functions are used, the *Lebesgue's decomposition theorem* should be considered.

Moreover:

Definition 56 (Cumulative probability mass function). *The cumulative probability mass function (cmf) is a staircase function, i.e.,*

$$\Pr[x \leq a] = \sum_{i=1}^n \pi_i \mathbf{1}(a - a_i),$$

where $\mathbf{1}(\cdot)$ is the unitary step function.

Example 16 (Poisson pmf). $x \sim \mathcal{P}(\lambda)$ defines x as a Poisson distributed random variable with rate λ .

The Poisson distribution describes the probability of a certain number of random points in an interval T .

The pmf is given by

$$\pi(a) = \Pr[x = a] = e^{-\lambda T} \frac{(\lambda T)^a}{a!},$$

where $a \in \mathbb{N}_0$.

Moments

It is sometimes needed to characterise a **rv** with certain attributes describing, e.g., the *typical* value, the *spread* or variability, etc. These attributes are computed on the pdf and *not* on the data.

For example we can be interested on the *central value*.

r -th moment	Discrete	Continuous
about the origin	$\mu'_r = \sum_i a_i^r \pi_i$	$\mu'_r = \int_{-\infty}^{+\infty} x^r p(x) dx$
about the mean	$c'_r = \sum_i (a_i - \mu'_1)^r \pi_i$	$c'_r = \int_{-\infty}^{+\infty} (x - \mu'_1)^r p(x) dx$

Table 11.1: Definitions of moments.

r -th moment	Discrete	Continuous
raw	$\mu'_r = E\{x^r\}$	$\mu'_r = E\{x^r\}$
central	$c'_r = E\{(x - \mu'_1)^r\}$	$c'_r = E\{(x - \mu'_1)^r\}$

Table 11.2: Definitions of moments using the expected operator $E\{\cdot\}$.

Definition 57 (Mode). *The mode is the value in which the pdf attains its maximum value.*

This is not a very good measure of the central value since the pdf can have multiple peaks.

A good measure is instead given by the *moments*, a concept derived from standard mechanics. The moments are defined with respect the origin (or *raw statistical moments*) or about the mean (or *central statistical moments*), and are subsumed in Table 11.1. The r -th *moment* about the origin is called *r -th order raw statistical moment* of a **rv**. The r -th *moment* about the mean is called *r -th order central statistical moment* of a **rv**.

The *moments* are usually defined using the *expected value* operator:

$$E\{x\} \triangleq \int_{-\infty}^{+\infty} xp(x) dx \text{ or } E\{x\} \triangleq \sum_i a_i \pi_i,$$

which are reported in Table 11.2. Notice that:

- For $r = 0$, $\mu'_0 = 1, \forall x$. Hence no information can be retrieved about x . The same for c'_0 ;
- For $r = 1$, $\mu'_1 \triangleq \mu$ is called the *mean*, i.e. the long run average;
- For $r = 1$, $c'_1 = 0, \forall x$. Hence no information can be retrieved about x ;
- For $r = 2$, $c'_2 \triangleq \sigma^2$ is called the *variance*.

The definition of the mean and the variance is reported in Table 11.3. Moments of order greater than the second are called the *skewness* (third moment) and the *kurtosis* or *flatness* (fourth moment).

Moments	Discrete	Continuous
<i>mean</i>	$\mu = \sum_i a_i \pi_i$	$\mu = \int_{-\infty}^{+\infty} xp(x)dx$
<i>variance</i>	$\sigma^2 = \sum_i (a_i - \mu)^2 \pi_i$	$\sigma^2 = \int_{-\infty}^{+\infty} (x - \mu)^2 p(x)dx$

Table 11.3: Definitions of mean and variance.

Expected operator

Let us suppose that $y = g(x)$, where x is a **rv** and $g(\cdot)$ is a certain function. By definition, the *expected value* of $y = g(x)$ is defined as

$$E\{y\} \triangleq \int_{-\infty}^{+\infty} g(x)p(x)dx.$$

We further define:

$$V\{x\} \triangleq E\{(x - \mu)^2\} = \sigma^2,$$

where $V\{\cdot\}$ stands for “the variance of x ”. For the *expected value*, we further know that:

- If α is a deterministic value

$$E\{\alpha\} = \alpha.$$

(You can prove it by assuming that a deterministic value can be considered a **rv** with pdf equals to a Dirac δ function);

- Moreover, if α_1 and α_2 are two deterministic values and x_1 and x_2 two **rvs**, we have

$$E\{\alpha_1 x_1 + \alpha_2 x_2\} = \alpha_1 E\{x_1\} + \alpha_2 E\{x_2\},$$

i.e. the *expected value* is a *linear operator*.

Example 17 (Uniform pdf). For a **Uniform** pdf $\mathcal{U}(a, b)$, we have:

$$\mu = E\{x\} = \frac{b+a}{2},$$

and

$$\sigma^2 = V\{x\} = E\{(x - \mu)^2\} = \frac{(b-a)^2}{12}.$$

Example 18 (Gaussian pdf). For a Gaussian pdf $\mathcal{N}(l, c)$, we have:

$$\begin{aligned}
E\{x\} &= \int_{-\infty}^{+\infty} x \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-l)^2}{2c^2}} dx = \\
&= \int_{-\infty}^{+\infty} (x-l) \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-l)^2}{2c^2}} dx + l \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-l)^2}{2c^2}} dx = \\
&= \int_{-\infty}^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = \\
&= \int_{-\infty}^0 y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + \int_0^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = \\
&= \int_{-\infty}^0 y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + \int_0^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = \\
&= - \int_0^{-\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + \int_0^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = \\
&= - \int_0^{+\infty} (-z) \frac{1}{\sqrt{2\pi c}} e^{-\frac{(-z)^2}{2c^2}} d(-z) + \int_0^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = \\
&= - \int_0^{+\infty} z \frac{1}{\sqrt{2\pi c}} e^{-\frac{z^2}{2c^2}} dz + \int_0^{+\infty} y \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy + l = l = \mu = E\{x\}
\end{aligned}$$

For a Gaussian pdf $\mathcal{N}(\mu, c)$, we have:

$$\begin{aligned}
V\{x\} = E\{(x-\mu)^2\} &= \int_{-\infty}^{+\infty} (x-\mu)^2 \frac{1}{\sqrt{2\pi c}} e^{-\frac{(x-\mu)^2}{2c^2}} dx = \\
&= \int_{-\infty}^{+\infty} y^2 \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy.
\end{aligned}$$

To solve this integral we first notice that

$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy = 1 \Rightarrow \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2c^2}} dy = c.$$

Differentiating both sides with respect to c , we get

$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2c^2}} dy = c \Rightarrow \int_{-\infty}^{+\infty} y^2 \frac{1}{\sqrt{2\pi} c^3} e^{-\frac{y^2}{2c^2}} dy = 1.$$

Therefore

$$\int_{-\infty}^{+\infty} y^2 \frac{1}{\sqrt{2\pi c}} e^{-\frac{y^2}{2c^2}} dy = c^2 \Rightarrow V\{x\} = c^2 = \sigma^2.$$

For a Gaussian pdf, $\sigma = \sqrt{\sigma^2}$ is called the **standard deviation**. It has to be noted that **only** the first two moments are sufficient to characterise a Gaussian pdf, i.e. $\mathcal{N}(\mu, \sigma)$.

Example 19 (Poisson pmf). *The mean of the Poisson discrete random variable is given by*

$$\begin{aligned}\mu = E\{x\} &= \int_{-\infty}^{+\infty} x \sum_{n=0}^{+\infty} e^{-\lambda T} \frac{(\lambda T)^n}{n!} \delta(x - n) dx = \\ &= \sum_{n=0}^{+\infty} n e^{-\lambda T} \frac{(\lambda T)^n}{n!} = e^{-\lambda T} \sum_{n=0}^{+\infty} n \frac{(\lambda T)^n}{n!}.\end{aligned}$$

Since for $n = 0$ the last term is 0, we can write

$$e^{-\lambda T} \sum_{n=1}^{+\infty} n \frac{(\lambda T)^n}{n!} = e^{-\lambda T} \lambda T \sum_{n=1}^{+\infty} \frac{(\lambda T)^{(n-1)}}{(n-1)!}.$$

By changing the variable $n - 1 = m$, we have the Taylor expansion of the exponential function, i.e.,

$$e^{-\lambda T} \lambda T \sum_{m=0}^{+\infty} \frac{(\lambda T)^m}{m!} = e^{-\lambda T} \lambda T e^{\lambda T} \Rightarrow \mu = \lambda T.$$

Interestingly, $V\{x\} = \lambda T = \mu$.

The *mean* value gives an idea of the central value of the **rv**, while the *variance* gives an idea about the spread. Additional measures are given by the *quantiles*, which are based on the fact that the cdf is always *invertible*.

Definition 58 (Quantile). *The r -th quantile is the value x_r such that $x_r = P^{-1}(r)$.*

Usually, the quartiles $r = 0.25$ and $r = 0.75$ are considered. The difference $x_{0.75} - x_{0.25}$ is the *interquartile range*.

Finally, we can define the *median*.

Definition 59 (Median). *The median $\tilde{\mu}$ is the quantile for $r = \frac{1}{2}$.*

In practice:

$$\Pr[x \leq \tilde{\mu}] = \Pr[x \geq \tilde{\mu}] \Rightarrow \int_{-\infty}^{\tilde{\mu}} p(x) dx = \int_{\tilde{\mu}}^{+\infty} p(x) dx.$$

The median is in general preferable to the mean and the variance as a measure of the pdf because the latter are more influenced by the pdf tails or when the pdf is multimodal. For this reason, the median is defined as a *robust* attribute of the pdf.

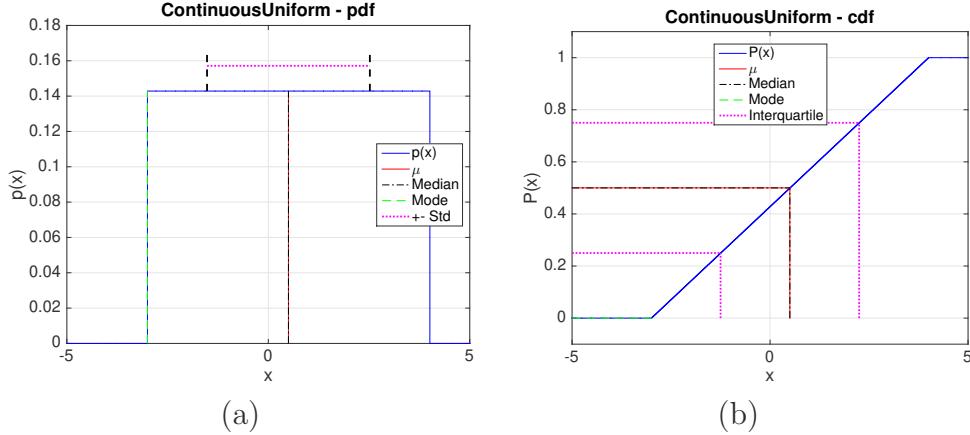


Figure 11.2: Uniform continuous pdf (a) and cdf (b) with attributes.

Example 20. Yesterday, 8 individuals payed these amounts of Euros to have a dinner in a certain restaurant:

$$x = [10 \ 10 \ 20 \ 360 \ 30 \ 10 \ 40 \ 30]$$

What is the typical amount of money you are expecting to pay for a dinner in the same restaurant tonight?

If you compute the **mean**, it is 60 Euros.

If you compute the **mode**, it is 10 Euros.

If you compute the **median**, it is 25 Euros.

What is the most appropriate?

Since the **median** robustly replaces the **mean**, the **mean deviation** replaces the **variance**:

$$\bar{\mu} = \int_{-\infty}^{+\infty} |x - \mu| p(x) dx.$$

Graphical representations of the pdfs and pmfs presented previously are reported in Figure 11.2, Figure 11.3, Figure 11.4 and Figure 11.5.

11.1.3 Multivariate Pdfs

Suppose we have a vector of random variables

$$x = [x_1, x_2, \dots, x_n]^T,$$

the **rvs** are then distributed according to a **joint probability density function**

$$p(x) \triangleq p(x_1, x_2, \dots, x_n).$$

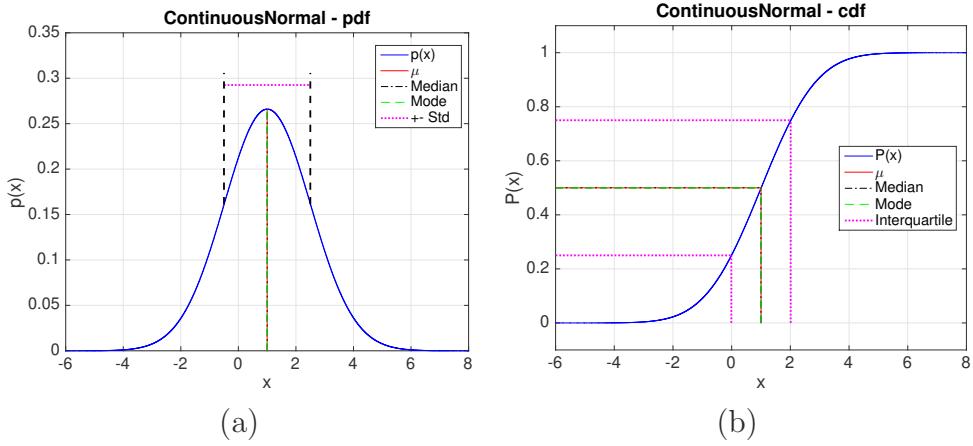


Figure 11.3: Gaussian continuous pdf (a) and cdf (b) with attributes.

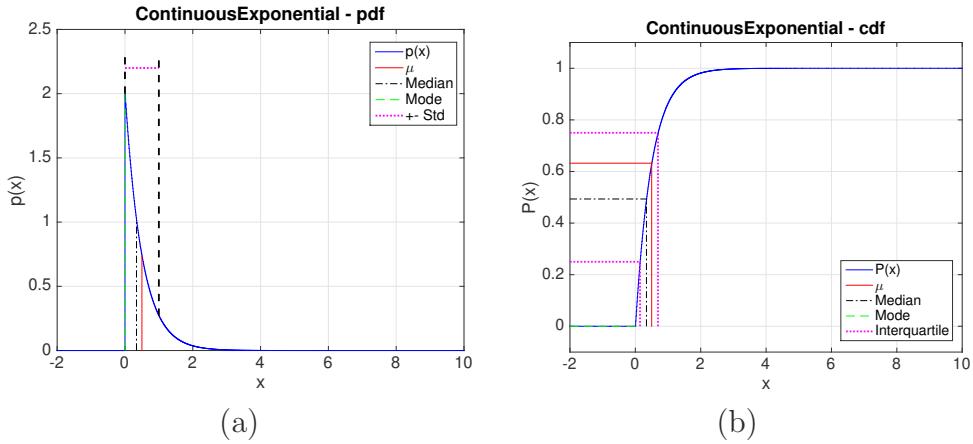


Figure 11.4: Exponential continuous pdf (a) and cdf (b) with attributes.

Moreover, if the domain of $p(x)$ is all the n -dimensional space (if it is not, the integrals should be computed with the appropriate limits)

$$P(a) = P(a_1, a_2, \dots, a_n) = \int_{-\infty}^{a_1} \int_{-\infty}^{a_2} \cdots \int_{-\infty}^{a_n} p(x) d^n x,$$

that is the *multivariate cumulative distribution function*.

A particular case of the multivariate pdfs are the *bivariate* pdfs, for which the graphical representation is possible. The main concept around the *bivariate* pdfs is that the probability of x_1 of falling in a certain range is not unrelated to the probability of x_2 of falling in a certain (possibly different) range. The main point here is that the multivariate pdfs allow to express *relationships* between variables. For a bivariate pdf is also practical writing

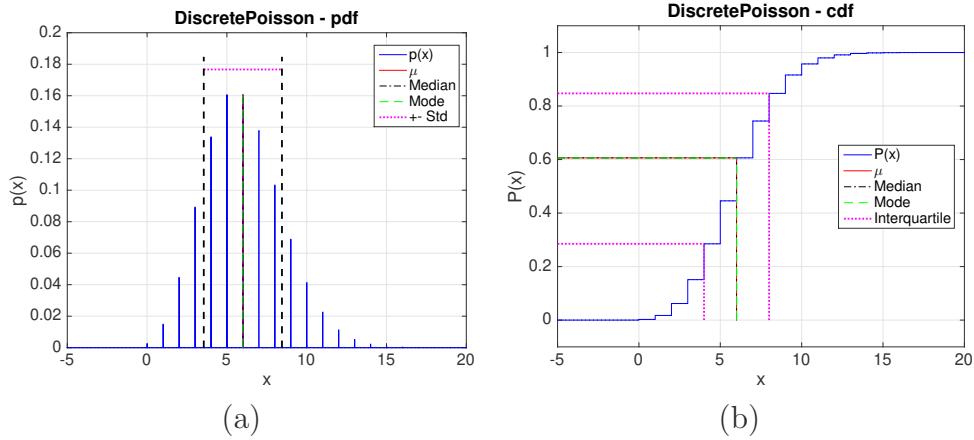


Figure 11.5: Poisson discrete pdf (a) and cdf (b) with attributes.

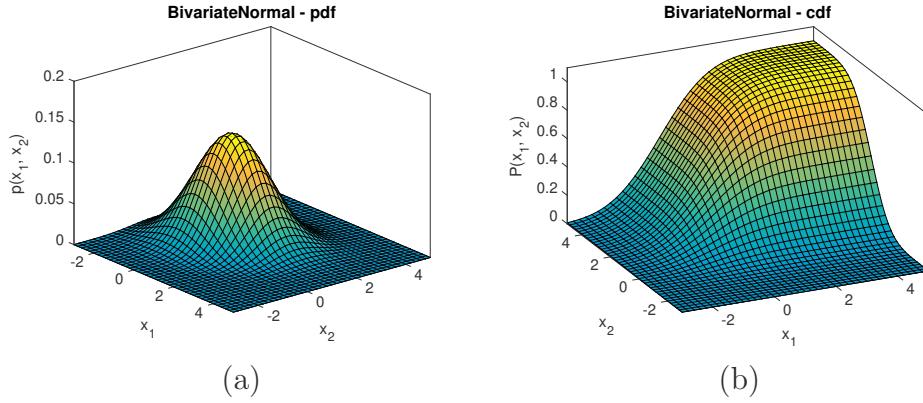


Figure 11.6: Bivariate Gaussian pdf (a) and cdf (b).

the definition of the *joint pdf* as

$$p(x_1, x_2) \triangleq \lim_{\delta_a \rightarrow 0, \delta_b \rightarrow 0} \frac{\Pr [(a \leq x_1 \leq a + \delta_a) \cap (b \leq x_2 \leq b + \delta_b)]}{\delta_a \delta_b}.$$

An example of a bivariate Gaussian pdf is reported in Figure 11.6, with the joint pdf and cdf reported.

Moments

The *joint statistical moments* of order $r + k$ of two discrete or continuous **rvs** x_1 and x_2 are reported in Table 11.4. Again, the sue of the expected value operator simplifies the formulation, as subsumed in Table 11.5.

Notice that:

Moments	Discrete
raw	$\mu'_{r,k}(x_1, x_2) = \sum_i \sum_j a_i^r a_j^k \pi_{x_1, x_2}$
central	$c'_{r,k}(x_1, x_2) = \sum_i \sum_j (a_i - \mu_1)^r (a_j - \mu_2)^k \pi_{x_1, x_2}$
	Continuous
raw	$\mu'_{r,k}(x_1, x_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1^r x_2^k p(x_1, x_2) dx_1 dx_2$
central	$c'_{r,k}(x_1, x_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_1 - \mu_1)^r (x_2 - \mu_2)^k p(x_1, x_2) dx_1 dx_2$

Table 11.4: Joint statistical moments for bivariate pdfs and pmfs.

Moments	Discrete
raw	$\mu'_{r,k}(x_1, x_2) = E\{x_1^r x_2^k\}$
central	$c'_{r,k}(x_1, x_2) = E\{(x_1 - \mu_1)^r (x_2 - \mu_2)^k\}$
	Continuous
raw	$\mu'_{r,k}(x_1, x_2) = E\{x_1^r x_2^k\}$
central	$c'_{r,k}(x_1, x_2) = E\{(x_1 - \mu_1)^r (x_2 - \mu_2)^k\}$

Table 11.5: Joint statistical moments for bivariate pdfs and pmfs expressed with the expected operator $E\{\cdot\}$.

- For $r = k = 0$, $\mu'_{0,0} = 1$, $\forall x_1, x_2$. Hence no information can be retrieved about x_1 and x_2 . The same for $c'_{0,0}$;
- For $r = 1$ and $k = 0$ or $r = 0$ and $k = 1$, $\mu'_{1,0} \triangleq \mu_1$ or $\mu'_{0,1} \triangleq \mu_2$, respectively, i.e. the *mean* of the single **rv**;
- For $r = 1$ and $k = 0$ or $r = 0$ and $k = 1$, $c'_{1,0} = c'_{0,1} = 0$, $\forall x_1, x_2$. Hence no information can be retrieved about x_1 and x_2 ;
- For $r = 1$ and $k = 1$, $\mu'_{1,1} \triangleq \phi_{x_1, x_2}$ is the *correlation*;
- For $r = 1$ and $k = 1$, $c'_{1,1} \triangleq C\{x_1, x_2\}$ is the *covariance*.

Marginals

Given a multivariate pdf it is possible to infer a certain set of *lower dimensional* pdfs. Consider a *bivariate* cdf. Basically, the *marginal* cdf consists of considering the probability only in one direction or for *one variable*, i.e.,

$$P(a_1, \infty) = \Pr[x_1 \leq a_1, x_2 \leq +\infty] = \Pr[x_1 \leq a_1].$$

So from the *bivariate cumulative distribution function* it is possible to obtain the *marginal cumulative distribution function* of x_1 . The same can be

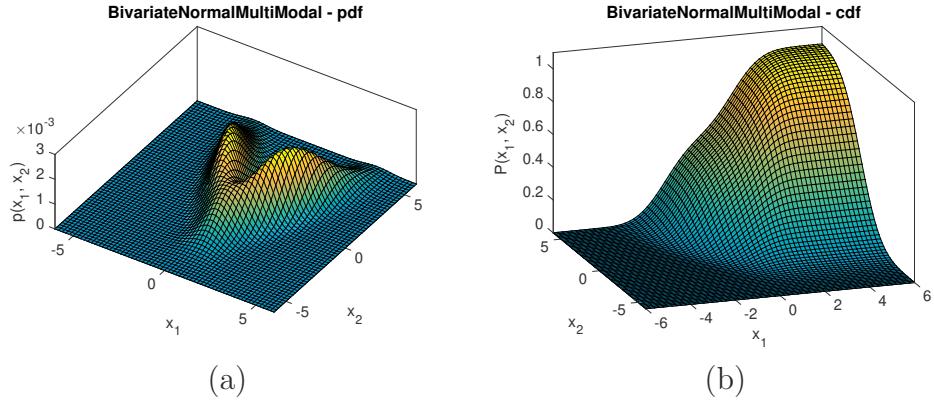


Figure 11.7: An example of a bivariate multimodal Gaussian pdf (a) and cdf (b).

obtained using the pdfs, and hence the *marginal probability density function* of x_1 is obtained by integrating the bivariate density function over all the possible values of x_2 , i.e.

$$p_1(x_1) = \int_{-\infty}^{+\infty} p(x_1, x_2) dx_2.$$

This operation corresponds in collapsing the density on one axis. For example, for the bivariate Gaussian pdfs in Figure 11.7, the two marginals are reported in Figure 11.8. Notice that this bivariate pdf is radically different from the one presented in Figure 11.6. Indeed, we can easily notice that in this case the bivariate pdf has two peaks: since the mode identifies the maximum of a pdf, this type of pdfs are called *multimodal*. This fact can also be noted in the marginal in Figure 11.8-a.

Conditionals

As the conditional probability gives the probability of A given B , i.e., $\Pr [A|B]$, the *conditional* pdf gives the pdf of x_1 given that $x_2 = a_2$. Let us recall that

$$\Pr [A|B] = \frac{\Pr [B|A] \Pr [A]}{\Pr [B]} = \frac{\Pr [A \cap B]}{\Pr [B]}.$$

For a cdf it becomes

$$P_1(a_1|B) = \frac{\Pr [(x_1 \leq a_1) \cap B]}{\Pr [B]}.$$

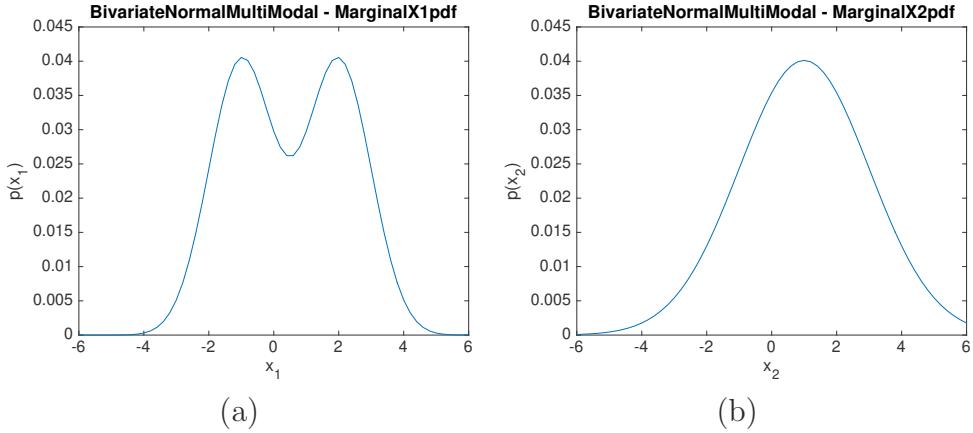


Figure 11.8: Marginals of the Bivariate multimodal Gaussian pdf in Figure 11.7 with respect to x_2 (a) and x_1 (b).

Let us suppose that the event B is $\{a_2 < x_2 \leq a_2 + \delta_{a_2}\}$. Hence, by substitution

$$P_1(a_1|a_2 < x_2 \leq a_2 + \delta_{a_2}) = \frac{\Pr[(x_1 \leq a_1) \cap (a_2 < x_2 \leq a_2 + \delta_{a_2})]}{\Pr[a_2 < x_2 \leq a_2 + \delta_{a_2}]},$$

and therefore, since $\Pr[a_2 < x_2 \leq a_2 + \delta_{a_2}] = P_2(a_2 + \delta_{a_2}) - P_2(a_2)$, we have

$$P_1(a_1|a_2 < x_2 \leq a_2 + \delta_{a_2}) = \frac{P(a_1, a_2 + \delta_{a_2}) - P(a_1, a_2)}{P_2(a_2 + \delta_{a_2}) - P_2(a_2)}.$$

Recall that the cdf is the integral of the pdf, hence

$$P_1(a_1|a_2 < x_2 \leq a_2 + \delta_{a_2}) = \frac{\int_{-\infty}^{a_1} \int_{a_2}^{a_2 + \delta_{a_2}} p(x_1, x_2) dx_2 dx_1}{\int_{a_2}^{a_2 + \delta_{a_2}} p_2(x_2) dx_2}.$$

Assuming δ_{a_2} small enough, we get

$$P_1(a_1|a_2 < x_2 \leq a_2 + \delta_{a_2}) \approx \frac{\int_{-\infty}^{a_1} p(x_1, x_2 = a_2) dx_1 \delta_{a_2}}{p_2(x_2 = a_2) \delta_{a_2}},$$

and in the limit

$$\begin{aligned} P_1(a_1|x_2 = a_2) &= \lim_{\delta_{a_2} \rightarrow 0} P_1(a_1|a_2 < x_2 \leq a_2 + \delta_{a_2}) = \\ &= \frac{\int_{-\infty}^{a_1} p(x_1, x_2 = a_2) dx_1}{p_2(x_2 = a_2)}. \end{aligned}$$

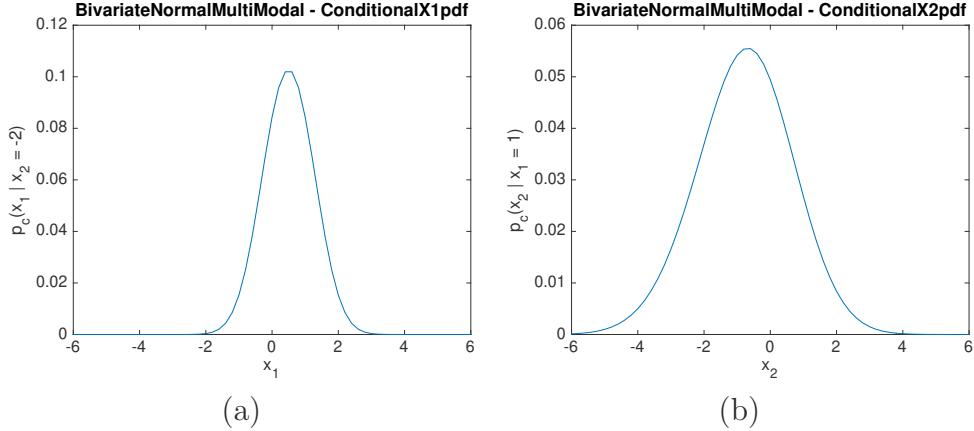


Figure 11.9: Conditional pdf with respect to $x_2 = -2$ (a) and with respect to $x_1 = 1$ (b) for the bivariate multimodal Gaussian pdf in Figure 11.7-a.

Finally, by differentiating both sides with respect to x_1 , we finally have:

$$p_{c_1}(x_1|x_2 = a_2) = \frac{p(x_1, x_2 = a_2)}{p_2(x_2 = a_2)} = \frac{p(x_1, x_2 = a_2)}{\int_{-\infty}^{+\infty} p(x_1, x_2 = a_2) dx_1},$$

which has the marginal at the denominator. In practice, the *conditional pdf* $p_c(x_1|x_2 = a)$ is a *slice* of the joint pdf $p(x_1, x_2)$ with x_2 held constant and equal to a . The conditional pdf can be computed for *any arbitrary slice* or closed curve on the domain of $p(x_1, x_2)$. Of course, the conditional pdf and the marginal pdf can be extended to *any dimension* for the joint pdf. Figure 11.9 reports an example of two conditional pdfs given the joint pdf in Figure 11.7-a.

Let us show that the *conditional pdf* is actually a well posed pdf. To this end, we first notice that the conditioning value can be left unspecified, i.e., for an arbitrary value or, equivalently, for all the values of $x_2 = a$. In this case, the conditional pdf represents a set of pdfs function of x_2 , i.e.

$$p_{c_1}(x_1|x_2) = \frac{p(x_1, x_2)}{p_2(x_2)} \geq 0.$$

Moreover, since

$$p_{c_1}(x_1|x_2) = \frac{p_{c_2}(x_2|x_1)p_1(x_1)}{\int_{-\infty}^{+\infty} p_{c_2}(x_2|x_1)p_1(x_1) dx_1},$$

it is obvious that

$$\int_{-\infty}^{+\infty} p_{c_1}(x_1|x_2) dx_1 = 1,$$

i.e. the denominator turns to be a *normalisation factor*.

Example 21. Given

$$p(x_1, x_2) = \begin{cases} k, & 0 < x_1 < x_2 < 1 \\ 0, & \text{otherwise} \end{cases}$$

let us compute the conditional pdfs. First, we have to determine the value of the constant k , which is a function of the normalisation of the pdf, i.e.

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x_1, x_2) dx_1 dx_2 = 1.$$

Therefore

$$\int_0^1 \int_0^{x_2} k dx_1 dx_2 = \int_0^1 k x_2 dx_2 = \frac{k}{2} = 1 \Rightarrow k = 2.$$

For the marginals we have:

$$p_1(x_1) = \int_{-\infty}^{+\infty} p(x_1, x_2) dx_2 = \int_{x_1}^1 k dx_2 = k(1 - x_1), \quad 0 < x_1 < 1,$$

and

$$p_2(x_2) = \int_{-\infty}^{+\infty} p(x_1, x_2) dx_1 = \int_0^{x_2} k dx_1 = k x_2, \quad 0 < x_2 < 1.$$

As a consequence:

$$p_{c_1}(x_1|x_2) = \frac{p(x_1, x_2)}{p_2(x_2)} = \frac{1}{x_2}, \quad 0 < x_1 < x_2 < 1,$$

and

$$p_{c_2}(x_2|x_1) = \frac{p(x_1, x_2)}{p_1(x_1)} = \frac{1}{1 - x_1}, \quad 0 < x_1 < x_2 < 1.$$

Bayes Theorem for pdfs

From

$$p_{c_1}(x_1|x_2) = \frac{p(x_1, x_2)}{p_2(x_2)} = \frac{p(x_1, x_2)}{\int_{-\infty}^{+\infty} p(x_1, x_2) dx_1},$$

we can derive that:

- $p(x_1, x_2) = p_{c_1}(x_1|x_2)p_2(x_2) = p_{c_2}(x_2|x_1)p_1(x_1);$
- $p_1(x_1) = \int_{-\infty}^{+\infty} p(x_1, x_2) dx_2 = \int_{-\infty}^{+\infty} p_{c_1}(x_1|x_2)p_2(x_2) dx_2 .$

Therefore:

$$p_{c_1}(x_1|x_2) = \frac{p(x_1, x_2)}{\int_{-\infty}^{+\infty} p(x_1, x_2) dx_1} = \frac{p_{c_2}(x_2|x_1)p_1(x_1)}{\int_{-\infty}^{+\infty} p_{c_2}(x_2|x_1)p_1(x_1) dx_1},$$

which is the *Bayes theorem* for pdfs, i.e., the basis of the *Bayesian inference*. The *prior* $p_1(x_1)$ reflects the *degree of a-priori belief* about x_1 . The *likelihood function* $p_{c_2}(x_2|x_1)$ is the *evidence from the data*, i.e., the *measurements*. The *posterior* $p_{c_1}(x_1|x_2)$ is the combination of the prior with the likelihood function by means of the *Bayes theorem*.

Example 22. Let us consider the problem of estimating the angular position of a motor shaft, with angle $\theta \in [0, 2\pi]$, using a measure $\theta_m = \theta + \varepsilon$, with sensing error noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. This problem can be solved with conditional pdfs and Bayes theorem. Indeed, the pdf of the estimate $\hat{\theta}$ of the actual value θ can be derived if the posterior $p_{c_1}(\theta|\theta_m)$ is known. The posterior gives a stochastic description of the uncertainty associated with the estimates $\hat{\theta}$.

To this end, we apply the Bayes theorem and we write

$$p_{c_1}(\theta|\theta_m) = \frac{p_{c_2}(\theta_m|\theta)p(\theta)}{\int_{-\infty}^{+\infty} p_{c_2}(\theta_m|\theta)p(\theta)d\theta}.$$

Since before the first measure, there is no knowledge about the angle position, define $p(\theta)$ is a $\mathcal{U}(0, 2\pi)$, i.e. the prior.

Therefore

$$p_{c_1}(\theta|\theta_m) = \frac{p_{c_2}(\theta_m|\theta)\frac{1}{2\pi}}{\int_0^{2\pi} p_{c_2}(\theta_m|\theta)\frac{1}{2\pi}d\theta}.$$

Since $\theta_m = \theta + \varepsilon$, we immediately have that the measurement takes place when θ is deterministic (i.e. a value), hence

$$E\{\theta_m\} = E\{\theta + \varepsilon\} = E\{\theta\} + E\{\varepsilon\} = \theta,$$

and

$$V\{\theta_m\} = E\{(\theta_m - E\{\theta_m\})^2\} = E\{\varepsilon^2\} = V\{\varepsilon\} = \sigma^2.$$

Therefore, since θ is a value

$$p_{c_2}(\theta_m|\theta) \sim \mathcal{N}(\theta, \sigma^2),$$

which is the likelihood function, or the evidence from the data or, more succinctly, the measurements. Therefore:

$$\begin{aligned} p_{c_1}(\theta|\theta_m) &= \frac{\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta_m-\theta)^2}{2\sigma^2}}\frac{1}{2\pi}}{\int_0^{2\pi} \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\theta_m-\theta)^2}{2\sigma^2}}\frac{1}{2\pi}d\theta} = \frac{e^{-\frac{(\theta_m-\theta)^2}{2\sigma^2}}}{\int_0^{2\pi} e^{-\frac{(\theta_m-\theta)^2}{2\sigma^2}}d\theta} = \\ &= f(\theta_m)e^{-\frac{(\theta-\theta_m)^2}{2\sigma^2}} \end{aligned}$$

which is a **unimodal pdf** defined in the set $[0, 2\pi]$ (because of the presence of $p(\theta)$ at numerator) and with a mean value in θ_m . The **posterior** gives more information than the **prior**!

Marginal mean, conditional mean and variance

Given the existence of the **conditional pdf**, we may be interested in computing the **mean of the conditional pdf**. The mean of the conditional pdf is called the **conditional mean** and, given the slice $y = a$, it is given by

$$\mu_{x|y=a} = E\{x|y=a\} = \int_{-\infty}^{+\infty} xp_c(x|y=a)dx,$$

which is a function of a . In general, being $A \Leftrightarrow y = a$ the conditioning event and $g(\cdot)$ a generic function of the **rv** x , we have

$$E\{g(x)|A\} = \int_{-\infty}^{+\infty} g(x)p_c(x|A)dx.$$

As a consequence, the **conditional mean** given $y = a$ for a generic function $g(x)$ is given by

$$E\{g(x)|y=a\} = \int_{-\infty}^{+\infty} g(x)p_c(x|y=a)dx.$$

Notice that in this case, $E\{g(x)|y=a\}$ is a number, i.e. a **deterministic value**. Indeed, the conditional pdf $p_c(x|y=a)$ is a specific function (a specific pdf), hence its mean is a number. However, it is a **rv** if we **do not restrict** to a specific value of y , since now $p_c(x|y)$ is a set of pdfs that are function of y . Therefore

$$E\{g(x)|y\} = \int_{-\infty}^{+\infty} g(x)p_c(x|y)dx,$$

is a **rv** related to y or, better, to the pdf $p_y(y)$ of y . Now, we can compute the **mean** of this random variable, by definition given by

$$E\{E\{g(x)|y\}\} = \int_{-\infty}^{+\infty} E\{g(x)|y\} p_y(y)dy.$$

We will denote $\mu_{x|y} = E\{x|y\}$. In other words, this is the average value of all the possible average values of the conditional pdfs when y assumes the

different possible values. Therefore:

$$\begin{aligned}
E\{E\{g(x)|y\}\} &= \int_{-\infty}^{+\infty} E\{g(x)|y\} p_y(y) dy = \\
&= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x) p_c(x|y) dx p_y(y) dy = \\
&= \int_{-\infty}^{+\infty} g(x) \int_{-\infty}^{+\infty} p_c(x|y) p_y(y) dy dx = \\
&= \int_{-\infty}^{+\infty} g(x) \int_{-\infty}^{+\infty} p(x,y) dy dx = \int_{-\infty}^{+\infty} g(x) p_x(x) dx = \\
&= E\{g(x)\},
\end{aligned}$$

which is also called *marginal mean* because it is obtained through *marginalisation* of the *joint pdf*.

Example 23. Consider a population of n_m males and n_w females. The **rv** x represents the **height** of the population, while the **rv** y is the **sex**. The **conditional mean** is given by the average height among the men or among the women. Denoting with h_{m_i} the **height** of the i -th man and with h_{w_i} the **height** of the i -th woman, we get

$$E\{x|y=m\} = \frac{\sum_i h_{m_i}}{n_m} \text{ and } E\{x|y=f\} = \frac{\sum_i h_{w_i}}{n_w}.$$

For the **marginal mean**, start from

$$\sum_i h_{m_i} = n_m E\{x|y=m\} \text{ and } \sum_i h_{w_i} = n_w E\{x|y=f\}.$$

Hence, the **marginal mean**, i.e. the **mean height**, is given by

$$E\{x\} = \frac{\sum_i h_{m_i} + \sum_i h_{w_i}}{n_m + n_w} = \frac{n_m}{n_m + n_w} E\{x|y=m\} + \frac{n_w}{n_m + n_w} E\{x|y=f\},$$

or, in other words:

$$E\{x\} = Pr[y=m] E\{x|y=m\} + Pr[y=f] E\{x|y=f\}.$$

This relation can be generalised: for example if y is the **rv age**, we have

$$E\{x\} = \sum_{i=0}^{120} Pr[y=i] E\{x|y=i\} = \sum_{i=0}^{120} \pi_{y_i} E\{x|y=i\}.$$

It is now clear that:

$$E\{g(x)\} = E\{E\{g(x)|y\}\} = \int_{-\infty}^{+\infty} E\{g(x)|y\} p_y(y) dy.$$

Similarly, we may be interested in computing the *variance of the conditional pdf*. The variance of the conditional pdf is called the *conditional variance* and, given $y = a$, it is given by

$$\sigma_{x|y=a}^2 = V\{x|y = a\} = E\{(x - \mu_{x|y})^2|y = a\}.$$

We first notice that in general, for a **rv** x we have

$$\begin{aligned}\sigma_x^2 &= V\{x\} = E\{(x - \mu_x)^2\} = \\ &= E\{x^2 + \mu_x^2 - 2x\mu_x\} = E\{x^2\} - \mu_x^2.\end{aligned}$$

The *conditional variance*, given $y = a$, it is then given by

$$\begin{aligned}\sigma_{x|y=a}^2 &= V\{x|y = a\} = E\{(x - \mu_{x|y=a})^2|y = a\} = \\ &= E\{x^2 + \mu_{x|y=a}^2 - 2x\mu_{x|y=a}|y = a\} = E\{x^2|y = a\} - \mu_{x|y=a}^2.\end{aligned}$$

As we did for the conditional mean, the *conditional variance* $\sigma_{x|y}^2$ is a **rv** if we let y to be unspecified, i.e.

$$\sigma_{x|y}^2 = V\{x|y\} = E\{(x - \mu_{x|y})^2|y\} = E\{x^2|y\} - \mu_{x|y}^2.$$

We have then the *expected value of the conditional variance*

$$E\{V\{x|y\}\} = E\{E\{x^2|y\}\} - E\{\mu_{x|y}^2\} = E\{x^2\} - E\{\mu_{x|y}^2\},$$

where we make use of $E\{E\{g(x)|y\}\} = E\{g(x)\}$.

Moreover, we can also compute the *variance* of the *conditional mean*

$$V\{E\{x|y\}\} = E\{E\{x|y\}^2\} - E\{E\{x|y\}\}^2 = E\{\mu_{x|y}^2\} - E\{x\}^2,$$

again using $E\{E\{x|y\}\} = E\{x\}$. Summing the previous two relations, we have

$$V\{E\{x|y\}\} + E\{V\{x|y\}\} = E\{x^2\} - E\{x\}^2 = V\{x\},$$

where the rightmost relation follows by definition. We have then the expression of the *marginal variance*.

For pmfs, the same derivations can be adapted to the discrete case.

Multidimensional mean and variance

Recall that, as for *univariate* pdfs, it is possible to compute the moments for *multivariate* pdfs using similar definitions. In practice, for the *mean* of x_i , which is *jointly distributed* with $x = [x_1, x_2, \dots, x_n]^T$, we have:

$$\mu_i = E\{x_i\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_i p(x) d^n x.$$

For the second moments (about the mean), the matter is a little bit more complicated since we have n^2 moments, one for *each possible combination of variables*. Hence,

$$\begin{aligned}\text{Cov}\{x_i, x_j\} &= \text{E}\{(x_i - \mu_i)(x_j - \mu_j)\} = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} (x_i - \mu_i)(x_j - \mu_j)p(x)d^n x.\end{aligned}$$

Moreover, if we select the same i -th element:

$$\sigma_i^2 = \text{E}\{(x_i - \mu_i)^2\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} (x_i - \mu_i)^2 p(x) d^n x.$$

Since

$$\sigma_i^2 = \text{E}\{(x_i - \mu_i)^2\} = \text{E}\{x_i^2\} - \mu_i^2,$$

we have immediately

$$\begin{aligned}\text{Cov}\{x_i, x_j\} &= \text{E}\{(x_i - \mu_i)(x_j - \mu_j)\} = \text{E}\{x_i x_j\} - \mu_i \mu_j = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_i x_j p(x) d^n x - \mu_i \mu_j,\end{aligned}$$

that is the difference between the *correlation* $\text{E}\{x_i x_j\}$ and the product of the *means* $\mu_i \mu_j$.

We are now in a position to draw the following definitions:

- The term $\text{Cov}\{x_i, x_j\} = \text{Cov}\{x_j, x_i\}$ is called the *covariance* between x_i and x_j ;
- The term σ_i^2 is the *variance* of x_i , which is a special case of the *covariance*.

Notice that while $\sigma_i^2 \geq 0$, the *covariance* $\text{Cov}\{x_i, x_j\}$ can be positive or negative. For example, if $\mu_x = 0$ and $y = -x$ (hence $\mu_y = 0$), we have that

$$\text{Cov}\{x, y\} = \text{E}\{xy\} = \text{E}\{x(-x)\} = -\text{E}\{x^2\} = -\sigma_x^2.$$

Covariances play a fundamental role for joint pdfs since they express the degree of *linear dependence* between the variables of the multidimensional **rv** x . In practice:

- If x_j tends to *increase linearly* away from μ_j when x_i does so with respect to μ_i , the covariance $\text{Cov}\{x_i, x_j\}$ will be large and positive;

- If x_j tends to *decrease linearly* away from μ_j when x_i *increases linearly* away with respect to μ_i , the covariance $\text{Cov}\{x_i, x_j\}$ will be large and negative;
- If there is very little linear dependency between x_i and x_j , then $\text{Cov}\{x_i, x_j\}$ will be small.

Usually, the *variance* $\sigma_i^2 \geq 0$ is normalised taking the square root, which yields the *standard deviation* $\sigma_i \geq 0$. This solution is not viable for the *covariance* $\text{Cov}\{x_i, x_j\}$, which is undefined in sign. Moreover, we know from the *Hölder inequality* that

$$|\text{Cov}\{x_i, x_j\}| \leq \sqrt{\text{V}\{x_i\} \text{V}\{x_j\}}.$$

This yields to the *correlation coefficient*:

$$\rho(x_i, x_j) = \frac{\text{Cov}\{x_i, x_j\}}{\sqrt{\text{V}\{x_i\} \text{V}\{x_j\}}} = \frac{\text{Cov}\{x_i, x_j\}}{\sigma_i \sigma_j} \in [-1, 1].$$

Hence:

- If x_j tends to *increase linearly* away from μ_j when x_i does so with respect to μ_i , then $\rho(x_i, x_j) \approx 1$;
- If x_j tends to *decrease linearly* away from μ_j when x_i *increases linearly* away from μ_i , then $\rho(x_i, x_j) \approx -1$;
- If there is very little linear dependency between x_i and x_j , then $\rho(x_i, x_j) \approx 0$ (the variables are *uncorrelated*).

It is easy to prove the following properties:

- The *covariance* and the correlation coefficient are symmetric, i.e. $\text{Cov}\{x, y\} = \text{Cov}\{y, x\}$ and $\rho(x, y) = \rho(y, x)$;
- $\text{Cov}\{ax, by\} = ab\text{Cov}\{x, y\}$, $a, b \in \mathbb{R}$;
- $\rho(ax, by) = \frac{ab}{|ab|}\rho(x, y)$, $a, b \in \mathbb{R}$;
- $\text{Cov}\{a_1x_1 + a_2x_2 + a_0, x_3\} = a_1\text{Cov}\{x_1, x_3\} + a_2\text{Cov}\{x_2, x_3\}$.

Independent pdfs

We saw that the joint probability of independent events is given by the product of the probabilities. Similarly, for $x = [x_1, x_2, \dots, x_n]^T$ where x_1, x_2, \dots, x_n are *independent random variables*:

$$p(x) \triangleq p_1(x_1)p_2(x_2)p_3(x_3)\dots p_n(x_n).$$

Alternatively, we can say that the n **rvs** in x are independent if *their joint pdf equals the product of the corresponding marginal densities*. A set of random variables is called *independent and identically distributed* (or *i.i.d.*) if they are *independent* and *the marginal densities are identical*.

Recall that the conditional pdf is

$$p_{c1}(x_1|x_2) = \frac{p(x_1, x_2)}{p_2(x_2)}.$$

If the **rvs** are independent

$$p_{c1}(x_1|x_2) = \frac{p(x_1, x_2)}{p_2(x_2)} = \frac{p_1(x_1)p_2(x_2)}{p_2(x_2)} = p_1(x_1),$$

or equivalently

$$p_{c2}(x_2|x_1) = \frac{p(x_1, x_2)}{p_1(x_1)} = \frac{p_1(x_1)p_2(x_2)}{p_1(x_1)} = p_2(x_2).$$

Suppose we have two **rvs**, we can say that

$$\begin{aligned} & \Pr [(a_1 \leq x_1 \leq a_1 + \delta_{a_1}) \cap (a_2 \leq x_2 \leq a_2 + \delta_{a_2})] = \\ &= \int_{a_1}^{a_1 + \delta_{a_1}} \int_{a_2}^{a_2 + \delta_{a_2}} p(x_1, x_2) dx_1 dx_2, \end{aligned}$$

hence $x_1, x_2 \sim p(x_1, x_2)$, i.e., x_1 and x_2 are *jointly distributed* with pdf $p(x_1, x_2)$. Let us suppose that we want to compute the pdf of $y = x_1 + x_2$. Then

$$P_y(a) = \Pr [y \leq a] = \Pr [x_1 + x_2 \leq a] = \int_{x_1+x_2 \leq a} p(\xi_1, \xi_2) d\xi_1 d\xi_2,$$

which defines a half plane. For the *joint* pdf of two *independent rvs* we have

$$\Pr [y \leq a] = \int_{x_1+x_2 \leq a} p(\xi_1, \xi_2) d\xi_1 d\xi_2 = \int_{x_1+x_2 \leq a} p_1(\xi_1)p_2(\xi_2) d\xi_1 d\xi_2.$$

Letting $z = \xi_1 + \xi_2$, we have

$$P_y(a) = \Pr [y \leq a] = \int_{-\infty}^a \int_{-\infty}^{+\infty} p_1(\xi_1)p_2(z - \xi_1)d\xi_1 dz.$$

Since $P_y(a)$ is the cdf of y , for its pdf we have (by differentiation)

$$p_y(y) = \frac{dP_y(a)}{da} \Big|_{a=y} = \int_{-\infty}^{+\infty} p_1(\xi_1)p_2(z - \xi_1)d\xi_1 = p_1 * p_2,$$

that is $x_1 + x_2 \sim p_1 * p_2$. For a generic number of random variables we have

$$x_1 + x_2 + x_3 + \cdots + x_n \sim p_1 * p_2 * p_3 * \cdots * p_n,$$

that is: the pdf of the sum of n *statistically independent* rvs is the *convolution* of the n pdfs.

We know that the convolution integral in the Fourier domain is given by the product of the *Fourier transform*. The Fourier transform of a pdf is given by

$$\bar{p}(f) \triangleq \int_{-\infty}^{+\infty} p(x)e^{-j2\pi fx}dx \leftrightarrow p(x) \triangleq \int_{-\infty}^{+\infty} \bar{p}(f)e^{j2\pi fx}df,$$

that is the *characteristic function* of the pdf $p(x)$. Notice that

$$\bar{p}(0) = \int_{-\infty}^{+\infty} p(x)dx = 1.$$

Furthermore

$$\bar{p}(f) = \mathbb{E} \{ e^{-j2\pi fx} \}.$$

If two random variables of a *bivariate* pdf are independent then:

$$\begin{aligned} \text{Cov} \{x_1, x_2\} &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_1 - \mu_1)(x_2 - \mu_2)p(x_1, x_2)dx_1 dx_2 = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_1 - \mu_1)(x_2 - \mu_2)p_1(x_1)p_2(x_2)dx_1 dx_2 = \\ &= \int_{-\infty}^{+\infty} (x_1 - \mu_1)p_1(x_1)dx_1 \int_{-\infty}^{+\infty} (x_2 - \mu_2)p_2(x_2)dx_2 = 0. \end{aligned}$$

which states that the two variables are *uncorrelated*, i.e., $\rho = 0$. This is true for any dimension of the **rv** vector.

So, *two independent* rvs are also *uncorrelated*. The converse is *not* true! There exists variables that are *uncorrelated but not independent*. For example, consider $x_1 \sim \mathcal{N}(0, 1)$ and $x_2 = x_1^2$. They cannot be independent by definition. However:

$$\text{Cov} \{x_1, x_2\} = \mathbb{E} \{ (x_1 - \mathbb{E} \{x_1\})(x_2 - \mathbb{E} \{x_2\}) \} = 0,$$

(prove for exercise - Hint: use integration by parts). This apparent incongruence simply state that there is *no linear dependence* between the two variables (indeed, the dependence is quadratic).

It turns out that for two *uncorrelated rvs*, we have

$$E\{x_1, x_2\} = E\{x_1\} E\{x_2\}$$

Indeed, since they are *uncorrelated* their *covariance* is

$$\begin{aligned} & \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_1 - \mu_1)(x_2 - \mu_2) p(x_1, x_2) dx_1 dx_2 = 0, \text{ hence} \\ & \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 p(x_1, x_2) dx_1 dx_2 = \mu_2 \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 p(x_1, x_2) dx_1 dx_2 + \\ & + \mu_1 \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_2 p(x_1, x_2) dx_1 dx_2 - \mu_1 \mu_2 \end{aligned}$$

Since the first term is the *correlation* $E\{x_1, x_2\}$, it follows, using the marginals that

$$E\{x_1, x_2\} = \mu_1 \mu_2$$

as desired. Alternatively, we can simply notice that

$$\text{Cov}\{x_1, x_2\} = E\{x_1, x_2\} - \mu_1 \mu_2,$$

and that $\text{Cov}\{x_1, x_2\} = 0$.

Example 24. Let $z = ax + by$, where x and y are two *uncorrelated rvs*.

The mean of z is

$$E\{z\} = E\{ax + by\} = aE\{x\} + bE\{y\} = a\mu_x + b\mu_y = \mu_z.$$

The variance of z is

$$\begin{aligned} V\{z\} &= E\{(z - \mu_z)^2\} = E\{(ax + by - a\mu_x - b\mu_y)^2\} = \\ &= E\{[a(x - \mu_x) + b(y - \mu_y)]^2\} = \\ &= a^2 E\{(x - \mu_x)^2\} + b^2 E\{(y - \mu_y)^2\} + 2abE\{(x - \mu_x)(y - \mu_y)\} = \\ &= a^2 \sigma_x^2 + b^2 \sigma_y^2 \end{aligned}$$

Furthermore, the covariance of $z = ax + by$ and x is

$$\begin{aligned} E\{(z - \mu_z)(x - \mu_x)\} &= E\{[a(x - \mu_x) + b(y - \mu_y)](x - \mu_x)\} = \\ &= aE\{(x - \mu_x)^2\} + bE\{(x - \mu_x)(y - \mu_y)\} = \\ &= a\sigma_x^2 \end{aligned}$$

hence a full linear dependence.

Total Probability Law

We finally present the extension to pdfs of the *Total Probability Law*, i.e.,

$$\Pr[B] = \sum_{i=1}^n \Pr[B \cap A_i] = \sum_{i=1}^n \Pr[B|A_i] \Pr[A_i]$$

where key is that the A_i are *mutually exclusive* and exhaustive with respect to the event set Ω . The extension is based on the *marginal* and the *conditional* pdfs for bivariate, i.e.

$$p_x(x) = \int_{-\infty}^{+\infty} p(x,y) dy = \int_{-\infty}^{+\infty} p_c(x|y)p_y(y) dy.$$

The generalisation to an arbitrary number of pdfs is given by

$$p_{x|z}(x|z) = \int_{-\infty}^{+\infty} p(x,y|z) dy = \int_{-\infty}^{+\infty} p_c(x|y,z)p_{y|z}(y|z) dy.$$

Compact representations

For multivariate pdfs, there is a compact form in which we can represent the mean and the covariances. Consider a vector of random variables

$$x = [x_1, x_2, \dots, x_n]^T.$$

The *mean* can be efficiently represented by

$$\mu = E\{x\} = \begin{bmatrix} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_1 p(x) d^n x \\ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_2 p(x) d^n x \\ \vdots \\ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} x_n p(x) d^n x \end{bmatrix}$$

For the *covariance* can be efficiently represented by

$$E\{(x - \mu)(x - \mu)^T\} = E\left\{ \begin{bmatrix} (x_1 - \mu_1)^2 & (x_1 - \mu_1)(x_2 - \mu_2) & \dots & (x_1 - \mu_1)(x_n - \mu_n) \\ (x_2 - \mu_2)(x_1 - \mu_1) & (x_2 - \mu_2)^2 & \dots & (x_2 - \mu_2)(x_n - \mu_n) \\ \vdots & \vdots & \ddots & \vdots \\ (x_n - \mu_n)(x_1 - \mu_1) & (x_n - \mu_n)(x_2 - \mu_2) & \dots & (x_n - \mu_n)^2 \end{bmatrix} \right\}$$

Such a matrix is *symmetric* and *positive definite*, unless there is a linear dependence among the elements of x .

Example 25. For example, let $z = ax + by$, where x and y are two uncorrelated rvs. We already know that:

$$E\{z\} = E\{ax + by\} = aE\{x\} + bE\{y\} = a\mu_x + b\mu_y = \mu_z,$$

$$V\{z\} = a^2\sigma_x^2 + b^2\sigma_y^2,$$

and

$$E\{(z - \mu_z)(x - \mu_x)\} = a\sigma_x^2 \text{ and } E\{(z - \mu_z)(y - \mu_y)\} = b\sigma_y^2$$

hence a full linear dependence.

In such a case the covariance matrix of the vector $w = [x, y, z]^T$ is given by

$$\begin{aligned} E\{(w - \mu_w)(w - \mu_w)^T\} &= \\ &\begin{bmatrix} \sigma_x^2 & 0 & a\sigma_x^2 \\ 0 & \sigma_y^2 & b\sigma_y^2 \\ a\sigma_x^2 & b\sigma_y^2 & a^2\sigma_x^2 + b^2\sigma_y^2 \end{bmatrix} \end{aligned}$$

which is symmetric but positive semidefinite.

The covariance matrix is given by

$$\begin{aligned} C\{x\} &= \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{bmatrix} = \\ &= \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \dots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \dots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n1}\sigma_n\sigma_1 & \rho_{n2}\sigma_n\sigma_2 & \dots & \sigma_n^2 \end{bmatrix} \end{aligned}$$

where we make use of the notation $\sigma_{ij} = \text{Cov}\{x_i, x_j\}$.

Example 26. For a multivariate normally distributed pdf, the expression of a the vector-valued Gaussian density is given by

$$\begin{aligned} p(x) &= \frac{1}{\sqrt{(2\pi)^n |C\{x\}|}} e^{-\frac{1}{2}(x-\mu)^T C\{x\}^{-1}(x-\mu)} = \\ &= \frac{1}{\sqrt{|2\pi C\{x\}|}} e^{-\frac{1}{2}(x-\mu)^T C\{x\}^{-1}(x-\mu)} \end{aligned}$$

The components of the vector x are then said to be jointly Gaussian. As for the univariate pdfs, the vector of means and the covariance matrix completely describe the pdf.

Notation: $|M|$ for a generic matrix M stands for the determinant of M .

The following properties hold for the Gaussian density function:

- All *marginals* distributions are Normal;
- The *conditional* distribution is still Normal;
- If all the variables are mutually *uncorrelated*, so that $C\{x\}$ is diagonal, the variables are also *independent*. In other words, *independence* and *zero correlation* are equivalent.

The last property can be shown by substituting a diagonal $C\{x\}$ in the Gaussian pdf definition.

It is evident how the Gaussian pdf has a set of relevant properties. Interestingly, the summation of **rvs** tends towards a Gaussian pdf, as stated by the *central limit theorem*:

Definition 60 (Central limit theorem). *If a random variable y is the sum of a sufficiently large number of rvs, then y will be approximately distributed as a Gaussian, irrespective of the distribution of the components.*

Linear combinations

In general, a *linear combination* of random variables can be expressed as

$$y = Lx.$$

We therefore have

$$E\{y\} = E\{Lx\} = LE\{x\} = L\mu.$$

Moreover

$$\begin{aligned} C\{y\} &= E\{(y - L\mu)(y - L\mu)^T\} = E\{(Lx - L\mu)(Lx - L\mu)^T\} = \\ &= E\{L(x - \mu)(x - \mu)^T L^T\} = LE\{(x - \mu)(x - \mu)^T\} L^T = \\ &= LC\{x\} L^T = L\Sigma_x L^T \end{aligned}$$

11.1.4 Propagation of errors

Sometimes we assume, for simplicity, that the **rvs** are distributed according to a Normal pdf. Sometimes this is implicit, by propagating *only* the mean and the variance. This is what happens when we have the following situation: suppose that sensor readings $y \in \mathbb{R}^n$ of certain quantities $x \in \mathbb{R}^n$ are corrupted by a **rv** $\varepsilon \in \mathbb{R}^n$, i.e.,

$$y = x + \varepsilon.$$

It follows that y is a **rv** having:

$$\mu_y = \text{E}\{y\} = \text{E}\{x + \varepsilon\} = x + \text{E}\{\varepsilon\} = x + \mu_\varepsilon$$

and

$$\text{C}\{y\} = \text{C}\{x + \varepsilon\} = \text{E}\{(\varepsilon - \mu_\varepsilon)(\varepsilon - \mu_\varepsilon)^T\} = \text{C}\{\varepsilon\} = \Sigma_\varepsilon.$$

Suppose now that

$$y = g(x) + f(\varepsilon),$$

where $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ are *nonlinear* vector functions. Now $y \in \mathbb{R}^m$, where in general $m \leq n$. It is hard to understand what would be the pdf of y for a generic nonlinear function, even in the case of a normal pdf for ε . However, we can approximate *locally* the propagation of the *mean* and the *covariance* through the nonlinear transformation $f(\cdot)$ using the so called *propagation of errors*.

This is possible using the *Taylor expansion* of $f(\cdot)$ about the pdf *mean*, i.e.,

$$f(\varepsilon) = f(\mu_\varepsilon) + \frac{df(\varepsilon)}{d\varepsilon} \Big|_{\varepsilon=\mu_\varepsilon} (\varepsilon - \mu_\varepsilon) + \mathcal{O}(\varepsilon^2)$$

Using the expansion up to the first order, we have

$$f(\varepsilon) \approx f(\mu_\varepsilon) + J(\varepsilon - \mu_\varepsilon)$$

where $J \in \mathbb{R}^{m \times n}$ is the so called *Jacobian* of $f(\cdot)$ about ε . Therefore

$$y \approx g(x) + f(\mu_\varepsilon) + J(\varepsilon - \mu_\varepsilon).$$

It then follows that

$$\begin{aligned} \mu_y &= \text{E}\{y\} = \text{E}\{g(x) + f(\mu_\varepsilon) + J(\varepsilon - \mu_\varepsilon)\} = \\ &= g(x) + f(\mu_\varepsilon) + J\text{E}\{(\varepsilon - \mu_\varepsilon)\} = g(x) + f(\mu_\varepsilon). \end{aligned}$$

For the covariance matrix

$$\begin{aligned} \Sigma_y &= \text{C}\{y\} = \\ &= \text{E}\{(g(x) + f(\mu_\varepsilon) + J(\varepsilon - \mu_\varepsilon) - \mu_y)(g(x) + f(\mu_\varepsilon) + J(\varepsilon - \mu_\varepsilon) - \mu_y)^T\} = \\ &= \text{E}\{J(\varepsilon - \mu_\varepsilon)(\varepsilon - \mu_\varepsilon)^T J^T\} = J\Sigma_\varepsilon J^T, \end{aligned}$$

that is, in practice, equal to the linear transformation. Notice that this is an *approximated relation*.

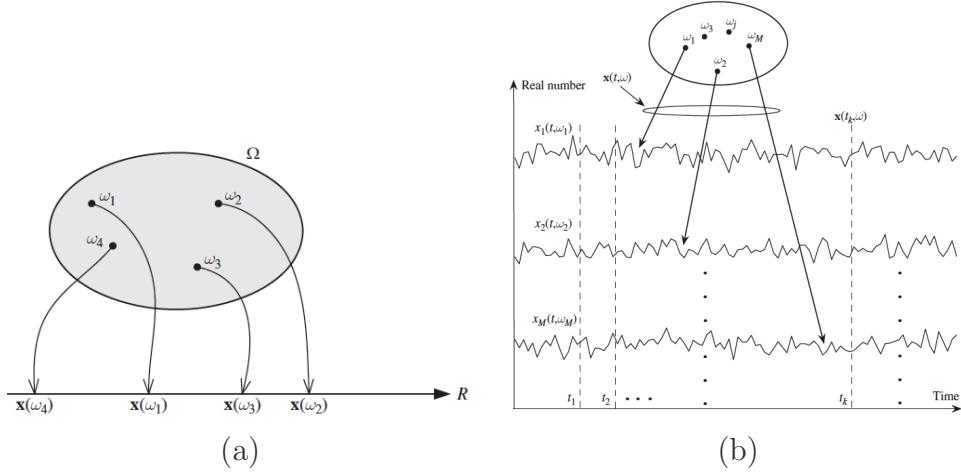


Figure 11.10: (a) A random variable can be seen as a mapping from a sample space Ω to a continuous (discrete) set, e.g., the set \mathbb{R} (\mathbb{Z}) of real (integer) numbers [4]. (b) A stochastic process can be seen as a mapping from a sample space Ω to a set of time varying functions [4].

11.2 Stochastic Processes

A *stochastic process* S is a collection of evolving random variables, i.e., $\{s_t, t \in T\}$, where T is an ordered set, in particular T can be the time [24], as depicted in Figure 11.10-b. More precisely

Definition 61 (Stochastic process). A *stochastic process*, aka *random process*, is a collection of random variables representing the evolution of some system of random values.

For example, meteorological phenomena such as the random fluctuations in air temperature and air pressure are functions of time. When dealing with *stochastic processes* is not of interest the knowledge of the precise mapping between the event set Ω and the function set; what instead is of interest is the *ensemble of functions*. Indeed, let $x(t)$ be the set of functions $x(\omega_1, t), x(\omega_2, t), \dots, x(\omega_n, t)$. Fixing a specific time, say t_0 , we have that $x(t_0)$ is a **rv**: therefore a *stochastic process can be interpreted as a time varying random variable*. In particular, given the process $x(\omega, t)$, we have that:

- $x(\omega_j, t)$ is the *sample function*, i.e., one of the possible time varying functions (i.e. a *realisation*);
- $x(\omega, t_i)$ is a **rv**;

- $x(\omega_j, t_i)$ is a *number*.
- A *discrete stochastic process* describes a process whose **rv** $x(\omega, t_i)$ are discrete;
- A *continuous stochastic process* describes a process whose **rv** $x(\omega, t_i)$ are continuous;
- A *discrete time process* describes a process whose sample functions $x(\omega_j, t)$ are discrete-time;
- A *continuous time process* describes a process whose sample functions $x(\omega_j, t)$ are continuous-time.

Since given a time instant t_i we have a **rv**, it is possible to give a description of it in terms of its cdf, i.e.,

$$P_{x(t_i)}(a) = \Pr [x(t_i) \leq a].$$

Notice that this amounts to take a slice for $t = t_i$ through the *ensemble* of functions (see the vertical dashed line in Figure 11.10-b). It is therefore defined the pdf as

$$p_{x(t_i)}(x) \triangleq \lim_{\delta_a \rightarrow 0} \frac{\Pr [a \leq x(t_i) \leq \delta_a + a]}{\delta_a}$$

where the suffix $x(t_i)$ explicitly state that the pdf is possibly *time varying*. In general, $\Pr [x(t_i) \leq a] \neq \Pr [x(t_j) \leq a]$, hence the knowledge of

$$P_{x(t_i), x(t_j)}(a, b) = \Pr [x(t_i) \leq a, x(t_j) \leq b]$$

asks for the *joint probability distribution*, which yields to

$$p_{x(t_i), x(t_j)}(x) = \lim_{\delta_a, \delta_b \rightarrow 0} \frac{\Pr [(a \leq x(t_i) \leq a + \delta_a) \cap (b \leq x(t_j) \leq b + \delta_b)]}{\delta_a \delta_b},$$

which is the *joint probability density function*. Hence, by selecting a set m of different time instants, we have that the **rvs** can be described by a *joint* pdf:

$$x(t_0), x(t_1), \dots, x(t_m) \sim p_{x(t_0), x(t_1), \dots, x(t_m)}(x).$$

Usually, the most important pdfs are given by $m = 1$ and 2 . Whatever is the order of the joint pdf we want to compute, we can get an estimate of it using the following frequency:

$$\frac{\text{Number of times the event is verified}}{\text{Number of trials}},$$

i.e., the pdf is estimated with a *histogram*.

The *mean* of a *continuous stochastic process* for a given t_i is then computed as

$$\mu_{x(t_i)} = \mathbb{E} \{x(t_i)\} = \int_{-\infty}^{+\infty} \xi p_{x(t_i)}(\xi) d\xi,$$

which is usually referred to as the *ensemble average*. Of course, for a *discrete stochastic process* for a given t_i is then computed as

$$\mu_{x(t_i)} = \mathbb{E} \{x(t_i)\} = \sum_j x_j \Pr [x(t_i) = x_j] = \sum_j x_j \pi_j(t_i).$$

The *autocorrelation* of a *scalar real-valued continuous stochastic process* for a t_i and t_j is then computed as

$$R \{t_i, t_j\} = \mathbb{E} \{x(t_i)x(t_j)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \xi \eta p_{x(t_i), x(t_j)}(\xi, \eta) d\xi d\eta.$$

The *autocovariance* of a *scalar real-valued continuous stochastic process* for a t_i and t_j is then computed as

$$V \{t_i, t_j\} \triangleq \mathbb{E} \{(x(t_i) - \mu_{x(t_i)})(x(t_j) - \mu_{x(t_j)})\} = R \{t_i, t_j\} - \mu_{x(t_i)} \mu_{x(t_j)}.$$

The extension to *vector-valued stochastic processes* follows the same steps carried out previously for the covariance of a vector-valued **rv**. Notice that the *autocovariance* corresponds to the *variance* if $t_i = t_j = t$. Indeed, for the *autocorrelation*

$$R \{t, t\} = \mathbb{E} \{x(t)^2\},$$

while for the standard variance of a **rv** y , we have:

$$\begin{aligned} V \{y\} &= \mathbb{E} \{(y - \mu_y)^2\} = \mathbb{E} \{y^2\} + \mathbb{E} \{\mu_y^2\} - \mathbb{E} \{2y\mu_y\} = \\ &= \mathbb{E} \{y^2\} + \mu_y^2 - 2\mu_y \mathbb{E} \{y\} = \mathbb{E} \{y^2\} - \mu_y^2. \end{aligned}$$

Therefore

$$V \{t, t\} = R \{t, t\} - \mu_{x(t)} \mu_{x(t)} = \mathbb{E} \{x(t)^2\} - \mu_{x(t)}^2.$$

The term $\mathbb{E} \{x(t)^2\}$ is called the *mean squared value*.

11.2.1 Properties of Stochastic Processes

Each stochastic process can be described by its properties, which basically describe the behaviour of the sequences of **rvs** in time. The first important property is *stationarity*.

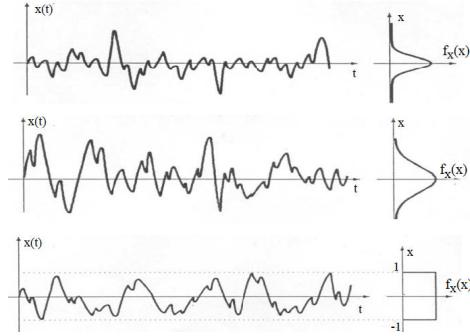


Figure 11.11: Three different pdfs $p_{x(t)}$ for three different zero-mean wide sense stationary processes [4].

Definition 62 (Wide sense stationarity). A **wide sense stationary process** (often called **stationary process**) is a stochastic process having a **time invariant mean**, i.e., $\mu_{x(t)} = \mu_x$, and the **autocorrelation** $R\{t_i, t_j\} = R\{\tau\}$, where $\tau = t_j - t_i$ (hence, $V\{t_i, t_j\} = R\{\tau\} - \mu_x = V\{\tau\}$).

Definition 63 (Strict stationarity). A **strict stationary process** is a stochastic process having all the pdfs time-invariant with respect to time shift, i.e.,

$$p_{x(t_0), x(t_1), \dots, x(t_m)}(x) = p_{x(t_0+\tau), x(t_1+\tau), \dots, x(t_m+\tau)}(x), \forall \tau.$$

In practice, **strict stationarity** (which is almost impossible to be verified in reality) requires that **all the moments** are stationary (not only the first two). Hence, in a stationary process, each random variable is drawn from the same pdf, as exemplified in Figure 11.11 with respect to the pdfs on the rightmost part of the figure.

The **autocorrelation function** $R\{\tau\}$ of a **wide-sense stationary process** has several properties:

- $R\{\tau\} = R\{-\tau\}$: It is an even function of τ because the same set of product values is averaged across the ensemble, regardless of the direction of translation;
- $|R\{\tau\}| \leq R\{0\}$: The maximum always occurs at $\tau = 0$. Further, $R\{0\}$ is the **mean squared value** of the random process;
- If $\mu_x \neq 0$, then $R\{\tau\}$ tends asymptotically to μ_x^2 ;
- The **power spectral density** is the Fourier transform of its **autocorrelation**

$$R\{\tau\} = \frac{1}{2\pi} \int_{-\infty}^{+\infty} S(\omega) e^{j\omega t} d\omega \leftrightarrow S(\omega) = \int_{-\infty}^{+\infty} R\{\tau\} e^{-j\omega\tau} d\tau,$$

and has the dimension of a power/hertz, that's why it is called like that.

Let $f(t)$ be a generic function of time, it is possible to define the *autocorrelation function* as

$$R\{\tau\} = \int_{-\infty}^{+\infty} f(t + \tau) f^*(t) dt = \int_{-\infty}^{+\infty} f(t) f^*(t - \tau) dt,$$

where f^* stands for the complex conjugate of $f(\cdot)$. In practice it is the *convolution* of a function by itself and, hence, measures the *similarity* of a function by itself in order to detect repeating patterns or similarities *in time*.

If $f(k)$ is a generic discrete time function, it is possible to redefine the *autocorrelation function* with

$$R\{k\} = \sum_{i \in \mathbb{Z}} f(i) f^*(i - k).$$

This definition applies to *finite energy* and *square summable* signals.

Remark 7. Since a signal $f(t)$ can be seen as a map between \mathbb{R} and \mathbb{R} , we have that the *instantaneous power* of $f(t)$ is $f(t)^2$, i.e. the amount of *energy* consumed per unit time. Instead, the *energy* of $f(t)$ for $t \in [0, T]$ is instead given by

$$\int_0^T f(t)^2 dt,$$

while the *average power* of $f(t)$ in $t \in [0, T]$ is instead given by

$$\frac{1}{T} \int_0^T f(t)^2 dt.$$

All these quantities can be obtained also in terms of norms.

A *square integrable* signal $f(t)$ is defined by

$$\sqrt{\int_{-\infty}^{+\infty} f(t)^2 dt} = \sqrt{\frac{1}{2\pi} \int_{-\infty}^{+\infty} |f(j\omega)|^2 d\omega} < +\infty,$$

which is often referred to as the $L_2 = \|f(t)\|_2$ (i.e., norm 2) measure of the signal. Instead, a *bounded* signal $f(t)$ is defined by

$$\sup_{t \geq 0} |f(t)| < +\infty,$$

which is often referred to as the $L_\infty = \|f(t)\|_\infty$ (i.e., norm infinity) measure of the signal.

Of course both concepts can be extended to vector-valued signals, i.e.

$$\begin{aligned} L_2 &= \|f(t)\|_2 = \sqrt{\int_{-\infty}^{+\infty} f(t)^T f(t) dt} = \\ &= \sqrt{\frac{1}{2\pi} \int_{-\infty}^{+\infty} f(j\omega)^* f(j\omega) d\omega} < +\infty, \end{aligned}$$

and

$$L_\infty = \|f(t)\|_\infty = \sup_{t \geq 0} \max_i |f_i(t)| < +\infty.$$

If the signal has infinite energy, then it is only possible to talk about the average power over the time, i.e.

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)^2 dt.$$

In such a case, the square root of the average power, i.e.

$$\sqrt{\lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)^2 dt},$$

defines the root mean square of the signal. \square

Assuming that $f(t)$ and $g(t)$ are two real functions of time, it is possible to generalise the autocorrelation function to the cross-correlation:

$$R\{\tau\}_{fg} = \int_{-\infty}^{+\infty} f(t + \tau) g(t) dt = \int_{-\infty}^{+\infty} f(t) g(t + \tau) dt,$$

which is a measure of the similarity of the two functions $f(t)$ and $g(t)$.

Another important property of a stochastic signal is related to its ergodicity.

Theorem 1 (Ergodicity). *For an ergodic stationary process, the time average of some sample functions of the process is the same as the ensemble average.*

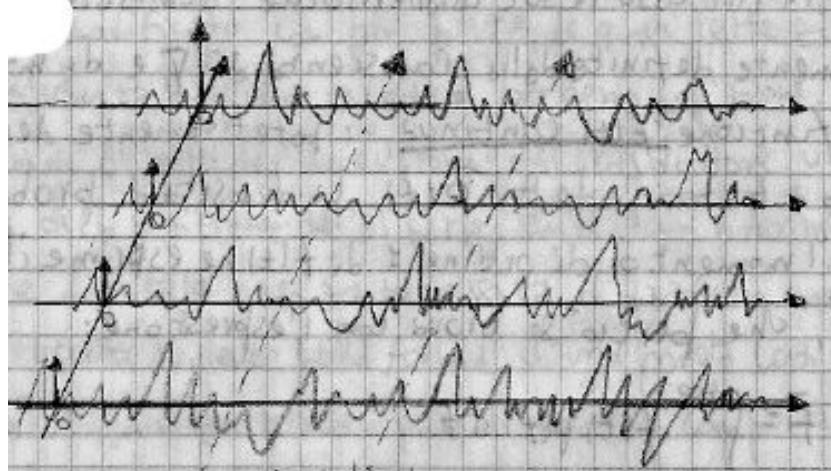


Figure 11.12: Ergodicity property.

Hence, for any given realisation of the stochastic process, the *time average* are equal to the *ensemble average with probability 1*. Therefore, any realisation of the stochastic process gives a good estimate of the process *statistical description*. It follows that for an *ergodic process*

$$E \{x^r(t)\} = \int_{-\infty}^{+\infty} x^r(t) p_{x(t)}(x) dx = \lim_{\tau \rightarrow +\infty} \frac{1}{2\tau} \int_{-\tau}^{\tau} x^r(\tau, \omega_i) d\tau, \forall i$$

where the factor $1/2\tau$ is adopted for normalisation. As for the stationarity property, the process is *wide sense ergodic* if the previous relation holds only for the *mean* and *autocorrelation* ($r = 1, 2$). If it holds $\forall r$, the process is *strictly ergodic*. An example of the ergodicity property is reported in Figure 11.12. Recalling the Definition 61, the horizontal axis represents the sample function $f(\omega_j, t)$, i.e. the function of time for a particular realisation ω_j . Instead, moving along the vertical axis, different realisations are reported, i.e. different values for the **rv** $\omega \in \{\omega_1, \omega_2, \dots\}$. Hence, computing the mean using the expected operator is possible once the time t is fixed to a specific time instant, i.e. $f(\omega, t_i)$, that is as a function of the pdf of ω . In this case the mean is computed along the vertical axis. Instead, if we fix the **rv** ω_j and we compute the time average, i.e. using $f(\omega_j, t)$, we move along the horizontal axis. Being ergodic means that no difference between the two directions exists. From a practical view-point, it means that only one realisation of the process, i.e. with just one ω_j , is necessary to compute the moments.

11.2.2 White processes

One of the most important properties of a process is of being *white*.

Definition 64 (White noise). *A (not necessarily stationary) stochastic process having the autocovariance null for any two different times is called a white noise.*

In such a case:

$$V\{t_i, t_j\} = \sigma^2(t_i)\delta(t_i - t_j)$$

where $\sigma^2(t_i)$ is the *instantaneous variance*. Recall that if the *covariance* of two **rvs** is null, than the **rvs** are *uncorrelated*. It then follows that the time *uncorrelatedness* defines a *wide-sense whiteness* property. There is also a *strict-sense whiteness* property that requests for the time *independence* rather than the time *uncorrelatedness* of the stochastic process.

By the definition of the *autocovariance* it follows that a *stationary zero-mean white* process has

$$R\{\tau\} = E\{x(t + \tau)x(t)\} = \sigma^2\delta(\tau)$$

It then follows that for a *stationary zero-mean white* process, the *power spectral density* is

$$S(\omega) = \sigma^2,$$

hence constant across the frequency spectrum. For *non-stationary zero-mean white* processes we have

$$V\{t_i, t_j\} = R\{t_i, t_j\} = E\{x(t_i)x(t_j)\} = \sigma^2(t_i)\delta(t_i - t_j)$$

where the meaning of $\sigma^2(t_i)$ *is not* instantaneous variance but *time-varying spectral density*. It can be noticed the same behaviour that we have discovered about the correlation and the covariance for standard random variables defined in Section 11.1.3.

Remark 8. *In general, the results related to the theory of estimators requests the strict-sense stationarity. However, this condition is hardly verified in practice, since only the first two moments of a stochastic process are available. Therefore, the estimators are applied to wide-sense stationarity, hence the results hold only approximately.*

Definition 65. *An independent and identically distributed (i.i.d.) random sequence corresponds to a stationary and white stochastic process.*

If the stochastic process defines vectors, i.e. $x(t) \in \mathbb{R}^n$, we have immediately for a generic *stochastic process* that

$$R\{t_i, t_j\} = E\{x(t_i)x(t_j)^T\} = Q(t_i, t_j)$$

and

$$V\{t_i, t_j\} = R\{t_i, t_j\} = E\{(x(t_i) - \mu_{x(t_i)})(x(t_j) - \mu_{x(t_j)})^T\} = Q(t_i, t_j).$$

It then follows that for a *zero-mean white* stochastic process

$$R\{t_i, t_j\} = V\{t_i, t_j\} = Q(t_i)\delta(t_i - t_j),$$

and, for a *stationary zero-mean white* process

$$R\{t_i, t_j\} = V\{t_i, t_j\} = Q\delta(t_i - t_j) = Q\delta(\tau).$$

11.2.3 Auto and Cross-correlation characteristics

By definition, the *autocorrelation* of a real function is *symmetric, real, even* and *with maximum in the origin*. The *cross-correlation* is *real*, but *not necessarily* symmetric, and with maximum in the origin.

Theorem 2. *For stationary stochastic processes generating $f(t)$, the Fourier transform of the autocorrelation function is the power spectral density, which expresses the squared amplitudes of every harmonics that are presents in the Fourier series expansion of $f(t)$.*

In other words:

$$\phi_f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \Phi_f(\omega) e^{j\omega t} d\omega \leftrightarrow \Phi_f(\omega) = \int_{-\infty}^{+\infty} \phi_f(t) e^{-j\omega t} dt.$$

By definition, it follows that $|F(\omega)|^2 = \Phi_f(\omega)$, where $F(\omega)$ is the Fourier transform of the signal $f(t)$. As previously mentioned, in the analysis of measurement and control systems, it is often considered the presence of *white noise*. The white noise is *ideal* and represents a noise *with power spectral density that is constant for any possible frequency*. The *value of the constant power spectral density* for the white noise of normally distributed stationary and ergodic processes can be computed, since

$$\begin{aligned} \sigma_f^2 &= \lim_{t \rightarrow +\infty} \frac{1}{2t} \int_{-t}^t f(\tau)^2 d\tau = \\ &= \lim_{t \rightarrow +\infty} \frac{1}{2t} \int_{-t}^t f(\tau) \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{j\omega\tau} d\omega d\tau = \\ &= \frac{1}{2\pi} \lim_{t \rightarrow +\infty} \frac{1}{2t} \int_{-\infty}^{+\infty} F(\omega) \int_{-t}^t f(\tau) e^{j\omega\tau} d\tau d\omega. \end{aligned}$$

The previous equation yields to the *Parseval Theorem*, i.e.,

$$\sigma_f^2 = \frac{1}{2\pi} \lim_{t \rightarrow +\infty} \frac{1}{2t} \int_{-\infty}^{+\infty} |F(\omega)|^2 d\omega.$$

In the case of white noise, $|F(\omega)|^2 = \Phi_f(\omega) = A$ is constant, hence the limit above is a *mean on an infinitesimal interval of a constant value*, i.e., *the* constant value A . Therefore,

$$\sigma_f^2 = \frac{1}{2\pi} A \Rightarrow A = 2\pi\sigma_f^2.$$

In other words, the power *spectral density of a white noise is a scaled version of the variance of the process generating the noise*.

Using similar arguments, it is interesting to derive the autocorrelation function of the white noise, i.e.

$$\begin{aligned} \phi_f(t) &= \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-T}^T f(t + \tau) f(\tau) d\tau = \\ &= \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-T}^T \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{j\omega(\tau+t)} d\omega f(\tau) d\tau = \\ &= \frac{1}{2\pi} \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-\infty}^{+\infty} F(\omega) \int_{-T}^T f(\tau) e^{j\omega\tau} d\tau e^{j\omega t} d\omega = \\ &= \frac{1}{2\pi} \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-\infty}^{+\infty} F(\omega) F(-\omega) e^{j\omega t} d\omega, \end{aligned}$$

that, using the Parseval theorem and the property of the Dirac delta function, can be finally rewritten as

$$\begin{aligned} \phi_f(t) &= \frac{1}{2\pi} \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-\infty}^{+\infty} F(\omega) F(-\omega) e^{j\omega t} d\omega = \\ &= \frac{1}{2\pi} \lim_{T \rightarrow +\infty} \frac{1}{2T} \int_{-\infty}^{+\infty} |F(\omega)|^2 e^{j\omega t} d\omega = \\ &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} 2\pi\sigma_f^2 e^{j\omega t} d\omega = \sigma_f^2 \delta(t). \end{aligned}$$

Hence, in the white noise there is *not* any *correlation between successive values*, i.e., the values are definitely random. This is a very important result. Indeed, if $\nu(t)$ is a white zero-mean noise, we have immediately that

$$\phi_\nu(t, \tau) = E\{\nu(t)\nu(\tau)\} = \sigma_\nu^2 \delta(t - \tau).$$

Hence, in the white noise there is *not* any *correlation between successive values*, i.e., the values are definitely random.

11.2.4 Markovian processes

The *Markov processes* are defined by the following *Markov property*:

$$p(x(t)|x(\tau), \tau \leq t_1) = p(x(t)|x(t_1)), \forall t > t_1$$

So, the past up to any t_1 is *fully characterised* by the value of the process at time t_1 . To state it with the words of Bar-Shalom in [24]: “the future is independent from the past if the present is known”. This is very important, since the state of a possibly *time varying dynamic system* driven by *white noise* $\nu(t)$, i.e.,

$$\dot{x}(t) = f(x(t), \nu(t), t),$$

is indeed a *Markov process*.

Example 27. As an example, let us consider the Wiener stochastic process. Being $\nu(t)$ a zero-mean white noise, the Wiener stochastic process is given by

$$w(t) = \int_0^t \nu(\tau) d\tau,$$

which is a form of the *random walk*. An alternative way of writing it is

$$dw(t) = \nu(t) dt,$$

hence an independent increment process. Notice that, more precisely, the white noise is the Wiener stochastic process derivative. However, this is not rigorous, since the Wiener process is nowhere differentiable, otherwise the variance of the derivative would be unlimited. Nevertheless, the Wiener stochastic process is Markovian, indeed

$$w(t) = \int_0^t \nu(\tau) d\tau = w(t_1) + \int_{t_1}^t \nu(\tau) d\tau,$$

and $\nu(\tau)$ is uncorrelated from $w(t_1)$. The Wiener stochastic process and the white noise are used to model unknown inputs for linear systems.

Prewhitening

One important fact from *Markov processes* that relates *white noise* to linear systems is the following. Consider a stable linear system excited by *white noise*:

$$\dot{x}(t) = Ax(t) + B\nu(t)$$

The state $x(t)$ at steady state is a *stationary Markov process* with *power spectral density* given by

$$S(\omega) = H(j\omega)QH^*(j\omega), \text{ with } H(j\omega) = (j\omega I - A)^{-1}B,$$

where $H(j\omega)$ is the *input to state transfer function matrix*, which is a *rational spectrum*, i.e., it can be expressed as a ratio of polynomials. The *autocorrelation function* of the input is in general a matrix given by

$$R\{t_1, t_2\} = E\{\nu(t_1)\nu(t_2)^T\} = Q\delta(t_1 - t_2),$$

where Q is the *power spectral density* of the white noise input, usually (and improperly) referred to as the *covariance matrix* of $\nu(t)$. It then follows that: *every Markov process with a rational spectrum can be represented as a linear time-invariant system excited by white noise.*

This fact is of paramount importance in our set-up. Indeed, given a system that is driven by a *stationary and not white* noise $\eta(t)$

$$\dot{x}(t) = Ax(t) + B\eta(t).$$

Since $\eta(t)$ is not white, the state $x(t)$ is *not* a *Markov process*. However, if $\eta(t)$ has a *rational spectrum*, it can be considered as the output of a linear system driven by *white noise* $\nu(t)$, i.e.,

$$\begin{aligned}\dot{z}(t) &= A_z z(t) + B_z \nu(t) \\ \eta(t) &= C_z z(t)\end{aligned}$$

Basically in this operation we consider the *stationary and not white* noise $\eta(t)$ as a Markov process. This operation is called the *prewhitening*, where the system A_z, B_z, C_z is called the *prewhitening system* or *shaping filter*. In the *prewhitening*, we consider an overall system driven by the *white noise* $\nu(t)$ and comprising the series of two linear systems. In fact, the overall system state

$$y(t) = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}$$

is now a *Markovian state*.

Remark 9. A non white noise is also called an autocorrelated noise or a coloured noise.

Discrete time Markov Chains

A *discrete-time stochastic process* is a time indexed sequence of random variables:

$$X_k = \{x(i)\}_{i=1}^k$$

Similarly to the continuous time, a *random sequence is Markovian* if

$$\Pr [x(k)|X_j] = \Pr [x(k)|x(j)], \forall k > j.$$

The *zero-mean sequence* $\nu(j)$, $j = 1, \dots$ is a *discrete-time white noise sequence* if

$$\mathbb{E} \{v(k)v(j)\} = Q(k)\delta_{jk},$$

where δ_{jk} is the *Kronecker delta function* and $Q(k)$ is the *covariance matrix*.

If the discrete-time process generates a *stationary white sequence*, $Q(k) = Q$ is the time invariant covariance matrix. A *discrete-time Markov process* is given by a time invariant system as

$$x(k+1) = f(k, x(k), \nu(k)).$$

The state of a linear time-invariant system excited by a *white Gaussian noise*

$$x(k+1) = Ax(k) + \nu(k),$$

is a *Gauss-Markov sequence*. If the system is scalar with $A = 1$

$$x(k+1) = x(k) + \nu(k),$$

hence $x(k)$ is the integral of the $\nu(k)$ and, hence, it is called a *discrete-time Wiener process*.

A *Markov chain* is a special case of the *Markov sequence* in which the state is *discrete and finite*. A *Markov chain* is fully characterised by the *transition probabilities*

$$\Pr [x(k) = x_i | x(k-1) = x_j] \triangleq p_{ij}.$$

Therefore the probability of being in the state x_i at time k , i.e., π_i , given the probabilities of being in the state x_j at time $k-1$ is then given by

$$\pi_i = \sum_{j=1}^n p_{ij}\pi_j.$$

By collecting all the probabilities in a single row vector $\pi = [\pi_1, \pi_2, \dots, \pi_n]$, one has

$$\pi = \pi P,$$

where $P = (p_{ij})_{ij}$ is the *transition probability matrix*.

Chapter 12

Estimation Algorithms

Let us recall the already introduced concepts about *estimators*. Let us assume that we want to *estimate* (that is to *increase our knowledge*) about a certain quantity x . Recalling the definitions of Chapter 10 in (10.1), a *measurement process* can be defined as

$$z(j) = h(j, x, w(j)), \quad j = 1, \dots, k,$$

where $w(j)$ is a sequence of measurement noises (or *disturbances*) and the estimation algorithm amounts to find the *estimates*

$$\hat{x}(k) = \hat{x}[k, Z^k].$$

For the *measurement model*, we usually adopt a simplified *forward map*

$$z(k) = h(x(k)).$$

The model in the easiest case is linear, i.e.

$$z(k) = Hx(k).$$

If this is not the case, usually a *Taylor approximation* is employed. Usually in the linear model $H \in \mathbb{R}^{m \times n}$. The objective of the *estimator* is to retrieve $x \in \mathbb{R}^n$ from $z \in \mathbb{R}^m$.

- If $n < m$ the problem is *over-determined* and the problem is usually solvable in the linear and nonlinear cases;
- If $n = m$ the problem is more involved since it may happens that it does not have solutions for the linear and the nonlinear cases;
- If $n > m$ the problem is not solvable unless *prior knowledge* is available.

In a very simple linear case, the *measurement error* w is the error affecting the measures at time k reported in (10.2), while the *estimation error* at time k is instead given by $\tilde{x}(k)$ as defined in (10.3).

To design an *estimator* we resort to the following description:

- The role of an estimator is to retrieve a *correct estimate* $\hat{x}(k)$, possibly the *best estimate*, i.e. the one with the smallest *estimation error* \tilde{x} ;
- The information available are: a) the *measurements* $z(k)$, which are *noisy*; b) an idea about the *system output model*, i.e. the function $h(\cdot)$; c) maybe, some *additional knowledge*, e.g. the quantity is always positive;
- *All* the available knowledge *must be exploited*, so the estimator needs to incorporate it properly;
- We can treat or not the quantity to estimate as a *random variable*.

To this end, let us first model the data, i.e. the *measurements*. Since, the *time series*

$$Z^k = \{z(j)\}_{j=1,\dots,k}$$

is *inherently random*, it can be described by a *joint pdf*, which is *parametrised* by the unknown parameter x , i.e.

$$p(z(1), \dots, z(k); x).$$

For a single measure

$$p(z(1); x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z(1)-x)^2}{2\sigma^2}}.$$

The choice of the modelling pdf turns to be *very critical* for the estimator design. Let us make an example: three possible values of $x \in \{-10, 10, 20\}$, with $\sigma = 2$, as reported in Figure 12.1. What is the *most probable value* if $z(1) < 0$? Of course, $x = -10$. Notice that here we use a prior (i.e., $x \in \{-10, 10, 20\}$) and we compute *how likely* each of the three prior hypotheses matches the measurements.

But there is also an alternative representation of this fact. The pdf $\Pr[z(1) < z < z(1) + \delta_z] \approx p(z(1); x)dz(1)$ gives the *probability of observing* $z(1)$ in a small area for a given x . As an example, we can plot the pdfs $p(z(1); x)$ for a *given* $z(1) = \bar{z} < 0$ considering x *variable*, i.e. $x \in \mathbb{R}$ (see Figure 12.2). In practice, the value of $p(z(1) = \bar{z}; x)dz(1)$ for each possible x tell us the *probability of observing* $z(1)$ *in a region* \mathbb{R} *centred around* \bar{z}

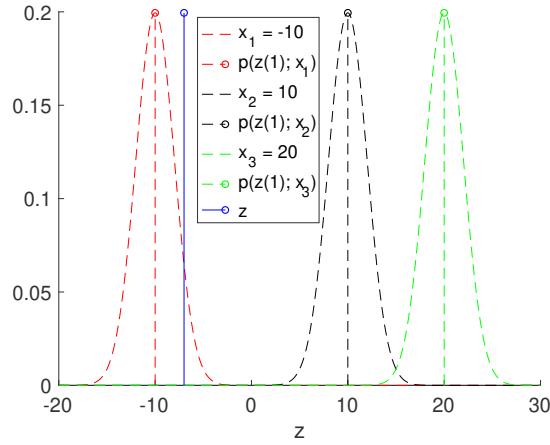


Figure 12.1: Three different pdfs for three different possible values of x , i.e. $x \in \{-10, 10, 20\}$, with $\sigma = 2$ and Gaussian pdf.

with are $dz(1)$ assuming the given value of x . This is the *likelihood function*. Observing $z(1) = \bar{z}$ it is quite unlikely that $x = 10$ or $x = 20$. Indeed, in those cases *the probability of observing that measurement would be very low!* On the other hand, it is more *likely* that we have observed $x = -10$, i.e. *assuming that $x = -10$, we have a higher probability that we actually observe $z(1) = \bar{z}$.* As a consequence, our estimate would be $\hat{x} = -10$!

This quite immediate idea carries a lot of information with it. Indeed, we may actually try to estimate x by *maximising the likelihood function without any prior*, i.e. by choosing the estimate \hat{x} that give the *highest probability of having a measurement $z(1) = \bar{z}$* . This estimator is the *Maximum Likelihood* estimator that we will see in a while in a more general framework.

Definition 66. An estimator is *unbiased* if, on average, it converges to the true value x , i.e. if $E\{\hat{x}\} = x$, for all the possible values of x in its domain.

Alternatively, we can say that since $\hat{x} = \hat{x}[k, Z^k]$, we have

$$E\{\hat{x}\} = \int \hat{x}[k, Z^k] p(Z^k; x) dZ^k = x.$$

Remark 10. A *biased estimator* is affected by a *systematic error*, i.e. a bias $E\{\hat{x}\} - x$.

A natural way to synthesise an estimator is to use an *optimal criterium*. For example, one may try to minimise the *Mean Square Error* (MSE) that we have already introduced:

$$\text{mse}(\hat{x}) = E\{(\hat{x} - x)^2\}.$$

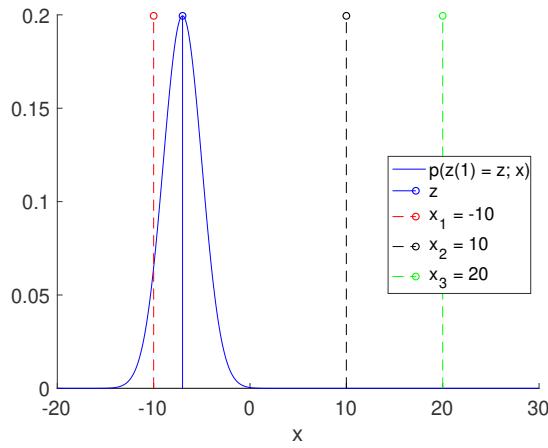


Figure 12.2: A likelihood function for the problem at hand, with $x \in \{-10, 10, 20\}$, $\sigma = 2$ and a Gaussian pdf centred around $z(1) = \bar{z} = -7$.

However, for a *biased estimator* with bias $b(x) = E\{\hat{x}\} - x$, we have

$$\text{mse}(\hat{x}) = E\{((\hat{x} - E\{\hat{x}\}) + (E\{\hat{x}\} - x))^2\} = V\{\hat{x}\} + b(x)^2.$$

Hence, to minimise the MSE, it is necessary to use the *bias* $b(x)$, which is usually *unknown* since it depends on the *actual value* of x , which is *not available*. Therefore, synthesise an estimator that *minimises the MSE* is in general *not possible!*

If the estimator is instead *unbiased*, as we noticed for the arithmetic mean, the MSE turns to be the *variance*. Therefore, minimising the MSE corresponds to minimise the *variance* and the estimator is then called *Minimum Variance Unbiased Estimator* (MVUE). In general the MVUE *does not* always exists. Indeed, it needs to be based on an *unbiased* estimator with *minimum variance* but *for all the possible values* of x .

Remark 11. Notice that for multidimensional parameters $x \in \mathbb{R}^n$, the MVUE has the property that $V\{x_i\}$ is minimum.

In general, the *Minimum Mean Square Error* (MMSE) leads to *unrealisable estimators*, while the *Minimum variance unbiased estimators* (MVUE) do not exist [24]. However, when an MVUE exists, it can be found using the *Cramer-Rao lower bound* (CRLB) or the *Rao-Blackwell-Lehmann-Scheffe* theorem. Instead, when an MVUE does not exist, an approximation of an estimator that is *linear in the measurements* can be derived.

Theorem 3 ((MVUE for the Linear Models)). *Assuming a linear model $Z = Hx + w$ for the observations, where $Z \in \mathbb{R}^k$, $x \in \mathbb{R}^n$, $H \in \mathbb{R}^{k \times n}$ and*

$w \in \mathbb{R}^k$ with $w \sim \mathcal{N}(0, R)$, then the MVUE is

$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} Z,$$

with covariance matrix

$$C\{\hat{x}\} = I^{-1}(x) = (H^T R^{-1} H)^{-1}.$$

For the linear model, the MVUE is efficient and attains the CRLB.

A measure of the *information* an *estimator* of a quantity x uses is called *Fisher information* $I(x)$. The *Fisher information* has the properties of the *information measure*: a) it is non negative; b) it is additive for independent measures.

Remark 12. When the CRLB is attained, the Fisher information is the reciprocal of the variance: more information, less variance.

For the previous linear MVUE, the *covariance matrix*

$$C\{\hat{x}\} = I^{-1}(x) = (H^T R^{-1} H)^{-1}.$$

12.1 Best Linear Unbiased Estimator

In general, even if the pdf is known and the CRLB available, it is not given for granted that an MVUE could be designed. In those cases, it is needed to use a *suboptimal estimator*. Usually, we can restrict to estimators that are *linear in the measurements* and hence find the *Best Linear Unbiased Estimator* (BLUE). The BLUE can be determined knowing only the *the first two moments* of the pdf.

Theorem 4 (Gauss-Markov theorem). If the measurements are linear, i.e. $Z^k = Hx + w$ where w is zero-mean and covariance R , the BLUE is given by

$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} Z^k,$$

with minimum variance

$$C\{\hat{x}\} = (H^T R^{-1} H)^{-1}.$$

A few remarks are now in order:

- The *arithmetic mean* is the BLUE for *any* pdf;

- If the noise sequence w is Gaussian, than the BLUE is also the MVUE;
- If the noise sequence w is not Gaussian, this is no longer true and the BLUE is clearly a *sub-optimal* solution

Another important aspect is related to the estimator *consistency*.

Definition 67 (Consistency). *The estimator $\hat{x}[k, Z^k]$ of the quantity x is consistent if:*

$$\lim_{k \rightarrow +\infty} \Pr[|\hat{x}[k, Z^k] - x| \geq \varepsilon] = 0, \forall \varepsilon > 0.$$

For example, the arithmetic mean is consistent since the MSE is $\frac{\sigma^2}{k}$.

12.2 Bayesian approaches

Suppose we have a *time series* representing the time evolution of the price of the oil on the market, which shows a *linear trend on average*. We can assume that a valid model is

$$z(j) = a + bj + w(j), \quad j = 1, \dots, k.$$

To have a model tractable we assume that $w(j) \sim \mathcal{N}(0, \sigma^2)$ and that $E\{w(j)w(i)\} = \sigma^2 \delta_{ji}$. Defining $x = [a, b]^T$, we finally have

$$p(Z^k, x) = \frac{1}{\sqrt{(2\pi\sigma^2)^k}} e^{-\frac{\sum_{j=1}^k (z(j) - a - bj)^2}{2\sigma^2}}.$$

Of course, the choice of the pdf is *critical* for the estimator design, which relies exactly on the chosen model. In general, we hope that the estimator is *robust*, i.e. changes in the pdf *do not affect* that much the estimates. Otherwise, *robust statistical procedures* should be adopted. In the *classical estimation* theory the parameter(s) x is assumed *unknown but constant*.

If instead, we know for the previous example that $a \in [\underline{a}, \bar{a}]$, we can incorporate this knowledge with a *stochastic prior* $p(x)$, thus having a *joint pdf*

$$p(Z^k, x) = p(Z^k|x)p(x),$$

and a *Bayesian estimator*. Usually, when the parameter(s) x is assumed *unknown but constant*, the approaches of the *classical estimation* are referred to as *non Bayesian*.

The prior pdf assumed in the problem is the *subjective assessment of the phenomenon*. If the prior is *uniform*, we assume that the Nature is *indifferent*.

In game theory, the Nature is assumed to *play against our purposes* (which is the basis of the H_∞ filter design). None of the previous assumptions are strictly correct, however there is always the *principle of perversity of inanimate objects*. According to Richard Bellman, this fact is proved by a set of experiments.

Example 28. *If you drop a piece of buttered toast on a rug, you have 79.3% possibilities that the toast fell buttered face down. There is also a mathematical proof...*

The *prior knowledge* can be given by the pdf $p_x(x)$. For example if x is in a certain interval, we can define

$$p_x(x) \sim \mathcal{U}(x_{\min}, x_{\max}).$$

If x has some typical values

$$p_x(x) \sim \mathcal{N}(\mu_x, \sigma_x^2).$$

Example 29. *To understand the nature of Bayesian Theorem, let us study the coin toss example.*

Suppose we have a coin where $\Pr[x_i = T] = \pi$ is the probability of having a tail T at the i -th toss (x_i is the **rv** modelling the coin i -th toss). Of course, the probability of having a head is $\Pr[x_i = H] = 1 - \pi$. Knowing that at the first toss we had $x_1 = T$, can we make a **prediction** about $x_2 = T$ at the second toss? We are hence interested in $\Pr[x_2 = T | x_1 = T]$.

Since $\Pr[x_i = T] = \pi$ and since $\Pr[x_1 = T | x_2 = T] = \Pr[x_1 = T] = \pi$ (the events of coin tosses are **independent**), we have

$$\Pr[x_2 = T \cap x_1 = T] = \Pr[x_1 = T | x_2 = T] \Pr[x_2 = T] = \pi^2,$$

which yields to

$$\Pr[x_2 = T | x_1 = T] = \frac{\Pr[x_2 = T \cap x_1 = T]}{\Pr[x_1 = T]} = \pi = \Pr[x_2 = T],$$

i.e. the events of coin tosses are **independent**. Therefore, the knowledge of $x_1 = T$ does not improve our confidence about $x_2 = T$. The **Bayes inference does not add anything to our prediction capabilities in this case**.

Let us now suppose that the coin is randomly selected from a pool of 2 coins, one having $\Pr[x_i = T] = 0$ and one $\Pr[x_i = T] = 1$, $\forall i$, i.e. $\pi = \{\pi_1, \pi_2\} = \{0, 1\}$. The probability of selecting the coin with $\Pr[x_i = T] = 0$ is q_1 , while the probability for $\Pr[x_i = T] = 1$ is $q_2 = 1 - q_1$. Knowing

that at the first toss we had $x_1 = T$, what is the probability $x_2 = T$ at the second toss? Again, a higher prediction confidence would be given if $\Pr[x_2 = T|x_1 = T] > \Pr[x_2 = T]$. Arguably, the additional prior information about the unfairness of the coins is of relevance.

Defining with “ $c = j$ ” the event “the j -th coin has been selected”, we will have for the two coins in the pool

$$\Pr[c = 1] = q_1 \text{ and } \Pr[c = 2] = q_2.$$

Since

$$c = \begin{cases} 1, & \Pr[x_i = T] = \pi_1 = 0, \\ 2, & \Pr[x_i = T] = \pi_2 = 1, \end{cases}$$

we can define this situation with a joint pmf with four values $p(x, c) = \{\pi_{T,1}, \pi_{T,2}, \pi_{H,1}, \pi_{H,2}\}$. The four values can be computed noticing that the pmf elements are just probabilities and that

$$\begin{aligned} \Pr[x = T \cap c = j] &= \Pr[x = T|c = j] \Pr[c = j] = \pi_j q_j, \\ \Pr[x = H \cap c = j] &= \Pr[x = H|c = j] \Pr[c = j] = (1 - \pi_j) q_j. \end{aligned}$$

As a consequence

$$\begin{aligned} \pi_{T,1} &= \Pr[x = T \cap c = 1] = \pi_1 q_1 = 0 \\ \pi_{T,2} &= \Pr[x = T \cap c = 2] = \pi_2 q_2 = q_2 \\ \pi_{H,1} &= \Pr[x = H \cap c = 1] = (1 - \pi_1) q_1 = q_1 \\ \pi_{H,2} &= \Pr[x = H \cap c = 2] = (1 - \pi_2) q_2 = 0 \end{aligned}$$

Notice that this is a well posed joint pmf, since $p(x, c) \geq 0$ and since

$$\sum_{k \in \{T,H\}} \sum_{j \in \{1,2\}} \Pr[x = k \cap c = j] = 1.$$

Moreover, given the joint probabilities we have immediately form the Total Probability Law

$$\begin{aligned} \Pr[x = T] &= \sum_{j \in \{1,2\}} \Pr[x = T \cap c = j], \\ \Pr[c = j] &= \sum_{k \in \{T,H\}} \Pr[x = k \cap c = j]. \end{aligned}$$

In terms of the joint pmf, those are the marginal pmfs:

$$\begin{aligned} p_m(x) &= \sum_{j \in \{1,2\}} p(x, c = j), \\ p_m(c) &= \sum_{k \in \{T,H\}} p(x = k, c). \end{aligned}$$

Now that we have framed the problem correctly and recalling that $\pi_1 = 0$ and $\pi_2 = 1$, we can solve the problem by noticing that

$$\begin{aligned} \Pr[x_i = T] &= \sum_{j \in \{1,2\}} \Pr[x_i = T \cap c = j] = \\ &= \sum_{j \in \{1,2\}} \Pr[x_i = T | c = j] \Pr[c = j] = \\ &= \sum_{j \in \{1,2\}} \pi_j q_j = q_2. \end{aligned}$$

For the joint probability

$$\begin{aligned} \Pr[x_1 = T \cap x_2 = T] &= \sum_{j \in \{1,2\}} \Pr[x_1 = T \cap x_2 = T | c = j] \Pr[c = j] = \\ &= \sum_{j \in \{1,2\}} \Pr[x_1 = T \cap x_2 = T | c = j] q_j = \\ &= \sum_{j \in \{1,2\}} \pi_j^2 q_j = q_2, \end{aligned}$$

since the two consecutive coin tosses are independent. Hence

$$\Pr[x_2 = T | x_1 = T] = \frac{\Pr[x_2 = T \cap x_1 = T]}{\Pr[x_1 = T]} = 1 \geq \Pr[x_2 = T],$$

which is independent from q_2 ! The Bayesian inference has radically improved our confidence about the predictions!

Let us now suppose that the coin is randomly selected from a pool of n coins, having $\Pr[x_i = T] = \pi_j$, $\forall i$, i.e. $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$. The probability of selecting the coin with $\Pr[x_i = T] = \pi_j$ is q_j ($j = 1, 2, \dots, n$). What is the probability $\Pr[x_2 = T | x_1 = T]$? As in the previous case, we can define this situation with a joint pmf with $2n$ values $p(x, c) = \{\pi_{T,1}, \dots, \pi_{T,n}, \pi_{H,1}, \dots, \pi_{H,n}\}$.

Since to each coin is associated a certain probability of tail, the joint pmf $p(x, c)$ that relies on the event “ $c = j$ ”, i.e. the event “the j -th coin has been selected”, could be equivalently expressed with $p(x, \Pi)$ that relies on the event “ $\Pi = \pi_j$ ”, i.e. the event “the coin with a probability of tail equals to π_j has been selected”. As a consequence, we have as before

$$\Pr[x_i = T] = \sum_{j=1}^n q_j \pi_j,$$

and

$$\Pr[x_2 = T \cap x_1 = T] = \sum_{j=1}^n q_j \pi_j^2,$$

hence

$$Pr[x_2 = T | x_1 = T] = \frac{Pr[x_2 = T \cap x_1 = T]}{Pr[x_1 = T]} = \frac{\sum_{j=1}^n q_j \pi_j^2}{\sum_{j=1}^n q_j \pi_j}.$$

Let us make an example: $\pi = \{0, 1/2, 1\}$ with $q = \{1/2, 1/4, 1/4\}$, i.e. the number of fair coins or coins with only tails is equal to the number of coins with only heads. What is the probability $Pr[x_2 = T | x_1 = T]$? Since

$$Pr[x_i = T] = \sum_{j=1}^n q_j \pi_j = \frac{1}{2}0 + \frac{1}{4}\frac{1}{2} + \frac{1}{4}1 = \frac{3}{8},$$

we have

$$Pr[x_2 = T \cap x_1 = T] = \sum_{j=1}^n q_j \pi_j^2 = \frac{1}{2}0^2 + \frac{1}{4}\frac{1}{2}^2 + \frac{1}{4}1^2 = \frac{5}{16},$$

which is now less than $Pr[x_i = T]$. Finally

$$Pr[x_2 = T | x_1 = T] = \frac{Pr[x_2 = T \cap x_1 = T]}{Pr[x_1 = T]} = \frac{5}{6} \geq Pr[x_i = T].$$

Again, in the general case, the Bayesian inference improves our confidence on the predictions!

We expect that the things will go even better if we keep observing the phenomenon of interest. Indeed, let us suppose we observe $x_1 = T$ and $x_2 = T$. Now, what is $Pr[x_3 = T | x_2 = T \cap x_1 = T]$? Will it be greater or lower than $Pr[x_2 = T | x_1 = T]$? Since

$$Pr[x_2 = T \cap x_1 = T] = \sum_{j=1}^n q_j \pi_j^2 = \frac{1}{2}0^2 + \frac{1}{4}\frac{1}{2}^2 + \frac{1}{4}1^2 = \frac{5}{16},$$

we have

$$Pr[x_3 = T \cap x_2 = T \cap x_1 = T] = \sum_{j=1}^n q_j \pi_j^3 = \frac{1}{2}0^3 + \frac{1}{4}\frac{1}{2}^3 + \frac{1}{4}1^3 = \frac{9}{32}.$$

Hence

$$\begin{aligned} Pr[x_3 = T | x_2 = T \cap x_1 = T] &= \frac{Pr[x_3 = T \cap x_2 = T \cap x_1 = T]}{Pr[x_2 = T \cap x_1 = T]} \\ &= \frac{9}{10} \geq Pr[x_2 = T | x_1 = T]. \end{aligned}$$

Indeed, more information we have, more precise will be our **predictions**! Increasing the observations, since we observed w consecutive tails in the first w coin tosses

$$\Pr \left[\bigcap_{i=1}^w x_i = T \right] = \sum_{j=1}^n q_j \pi_j^w,$$

We have

$$\begin{aligned} \Pr \left[x_{w+1} = T \mid \bigcap_{i=1}^w x_i = T \right] &= \frac{\Pr \left[\bigcap_{i=1}^{w+1} x_i = T \right]}{\Pr \left[\bigcap_{i=1}^w x_i = T \right]} \\ &= \frac{\sum_{j=1}^n q_j \pi_j^{w+1}}{\sum_{j=1}^n q_j \pi_j^w}. \end{aligned}$$

Let us now suppose that the coin is randomly selected from a **continuous pool of infinite coins**, having a certain **pdf** $p(c)$ (recall that the **rv** c models the **coin selection**). What is now the probability $\Pr[x_2 = T \mid x_1 = T]$? First recall that the **joint pdf** between the coin selection c and the outcome of the coin toss x is $p(x, c)$, which is an **hybrid pdf**, i.e. **continuous and discrete**. The probability of selecting the coin between \underline{c} and \bar{c} is

$$\Pr[\underline{c} < c < \bar{c}] = \int_{\underline{c}}^{\bar{c}} p_m(c) dc,$$

where $p_m(c)$ is the marginal computed along the possible outcomes

$$p_m(c) = \sum_{k \in \{T, H\}} p(x = k, c).$$

Notice that to each coin is associated a **certain probability** of having a tail π . In other words, there exists a mapping M (possibly not bijective) such that $\pi = M(c)$. Therefore, the marginal $p(c)$ can be represented equivalently as a function of π . In other words, the probability of selecting the coin with a probability of tail between $\pi - \varepsilon$ and π is

$$\Pr[\pi - \varepsilon < \Pi < \pi] = \int_{\pi-\varepsilon}^{\pi} p_m(\Pi) d\Pi,$$

where $p(\Pi)$ is the marginal computed along the possible outcomes

$$p_m(\Pi) = \sum_{k \in \{T, H\}} p(x = k, \Pi).$$

Then, recall that for a **pmf** for c we had the marginals along c

$$\Pr[x_i = T] = \sum_{j=1}^n q_j \pi_j,$$

and

$$Pr[x_2 = T \cap x_1 = T] = \sum_{j=1}^n q_j \pi_j^2.$$

If a pdf for c is considered, it means that both the sets π and q are continuous! As a consequence, considering that $\pi \in (a, b)$, with $0 \leq a < b \leq 1$ since they are probabilities, we have

$$\begin{aligned} Pr[x_i = T] &= \int_a^b p(x_i = T, \pi) d\pi = \int_a^b p_c(x_i = T | \pi) p_m(\pi) d\pi = \\ &= \int_a^b \pi p_m(\pi) d\pi, \end{aligned}$$

Similarly

$$Pr[x_2 = T \cap x_1 = T] = \int_a^b p_m(\pi) \pi^2 d\pi.$$

To make a numerical example, let us assume that $\pi \sim \mathcal{U}(0, 1)$. Hence

$$Pr[x_i = T] = \int_a^b p_m(\pi) \pi d\pi = \int_0^1 \pi d\pi = \frac{1}{2}.$$

Similarly

$$Pr[x_2 = T \cap x_1 = T] = \int_a^b p_m(\pi) \pi^2 d\pi = \int_0^1 \pi^2 d\pi = \frac{1}{3},$$

Therefore:

$$Pr[x_2 = T | x_1 = T] = \frac{Pr[x_2 = T \cap x_1 = T]}{Pr[x_1 = T]} = \frac{2}{3}.$$

Again, the knowledge of $x_1 = T$ improves our prediction capabilities using Bayesian inference.

What happens if the bucket is filled with unfair coins in favour of the tails? To model this fact $c \sim \mathcal{U}(1/2, 1)$. Hence

$$Pr[x_i = T] = \int_{1/2}^1 2\pi d\pi = \frac{3}{4}.$$

Similarly

$$Pr[x_2 = T \cap x_1 = T] = \int_{1/2}^1 2\pi^2 d\pi = \frac{7}{12},$$

Therefore:

$$Pr[x_2 = T | x_1 = T] = \frac{Pr[x_2 = T \cap x_1 = T]}{Pr[x_1 = T]} = \frac{7}{9} \geq \frac{2}{3}.$$

Let us again suppose that the coin is randomly selected from a pool of n coins, having $Pr[x_i = T] = \pi_j$, $\forall i$, i.e. $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$. The probability of selecting the coin with $Pr[x_i = T] = \pi_j$ is q_j ($j = 1, 2, \dots, n$). We saw that the Bayesian inference allows us to improve our prediction capabilities knowing that $x_1 = T$ and computing $Pr[x_2 = T | x_1 = T]$.

Let us now set-up an estimation problem. We want to estimate the coin we have selected (i.e. the quantity of interest is c) using as measurements the outcome of the tosses. As in the previous case, we can define this situation with a joint pmf with $2n$ values $p(x, c) = \{\pi_{T,1}, \dots, \pi_{T,n}, \pi_{H,1}, \dots, \pi_{H,n}\}$. Let us suppose we had $x_1 = T$. For this estimation problem we are now interested in computing

$$Pr[c = j | x_1 = T] = \frac{Pr[x_1 = T | c = j] Pr[c = j]}{Pr[x_1 = T]} = \frac{\pi_j q_j}{\sum_{j=1}^n q_j \pi_j}.$$

Considering the numerator, it is now extremely clear that:

- $Pr[c = j] = q_j$ is the prior: the probability of selecting the j -th coin a priori;
- $Pr[x_1 = T | c = j] = \pi_j$ is the likelihood: how likely the first outcome would be a tail given we have selected the j -th coin.

Substituting the numbers of the previous example, we will notice how the posterior probability $Pr[c = j | x_1 = T]$ changes (left as exercise).

Let us now suppose we increase the number of measurements, i.e. $x_1 = T$ and $x_2 = T$. In this case

$$\begin{aligned} Pr[c = j | x_1 = T \cap x_2 = T] &= \frac{Pr[x_1 = T \cap x_2 = T | c = j] Pr[c = j]}{Pr[x_1 = T \cap x_2 = T]} = \\ &= \frac{\pi_j^2 q_j}{\sum_{j=1}^n q_j \pi_j^2}. \end{aligned}$$

Let us now suppose we have collected w measurements, i.e. $x_i = T$, $\forall i = 1, \dots, w$. In this case

$$\begin{aligned} Pr\left[c = j | \bigcap_{i=1}^w x_i = T\right] &= \frac{Pr[\bigcap_{i=1}^w x_i = T | c = j] Pr[c = j]}{Pr[\bigcap_{i=1}^w x_i = T]} \\ &= \frac{q_j \pi_j^w}{\sum_{j=1}^n q_j \pi_j^w}. \end{aligned}$$

In practice, as $w \rightarrow +\infty$ and we keep observing tails, we end up with

- If $\pi_j \geq \pi_i, \forall i \neq j$ and $\exists l$ indices $\{i_1, i_2, \dots, i_l\}$ such that $\pi_j = \pi_{i_k}$, we have $\Pr[c = j | \bigcap_{i=1}^w x_i = T] \rightarrow \frac{q_j}{q_j + \sum_{k=1}^l q_{i_k}}$;
- If $\pi_j \leq \pi_i, \forall i \neq j$ and $\exists \pi_i > \pi_j$, we have $\Pr[c = j | \bigcap_{i=1}^w x_i = T] \rightarrow 0$.

In the case of a continuous pdf for the coin choice

$$p_c(c | \bigcap_{i=1}^w x_i = T) = \frac{p_c(\bigcap_{i=1}^w x_i = T | c)p_p(c)}{\Pr[\bigcap_{i=1}^w x_i = T]} = \frac{\pi^w p_p(c)}{\int_a^b \pi^w p_m(\pi) d\pi}.$$

For example if $p_p(c) \sim \mathcal{U}(0, 1)$, we have immediately for $w = 1$

$$p_c(c | x_i = T) = \frac{\pi p_p(c)}{\int_a^b \pi p_m(\pi) d\pi} = 2\pi, \text{ where } \pi \in [0, 1],$$

which is more informative than the uniform pdf and shifted towards the higher values of π due to the presence of $x_1 = T$.

In general, for w measurements

$$p_c(c | \bigcap_{i=1}^w x_i = T) = \frac{\pi^w p_p(c)}{\int_a^b \pi^w p_m(\pi) d\pi} = (w+1)\pi^w, \text{ where } \pi \in [0, 1],$$

where it is increasingly shifted towards the values with $\pi \rightarrow 1$. Figure 12.3 represents the pdf $p_c(c | \bigcap_{i=1}^w x_i = T)$ for a varying number of measurements w . The pdf tends towards a Dirac delta in $\pi = 1$.

As a final comment, let us consider the case in which $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and also $c \sim \mathcal{N}(\mu_c, \sigma_c^2)$. The same steps given previously apply starting from the prior $p_p(c)$ (that is Gaussian by definition) and considering the jointly Gaussian pdf $p(x, c)$. As seen previously

$$p_c(c|x) = \frac{p_c(x|c)p_p(c)}{p_m(x)},$$

where of course $p_m(x)$ is again Gaussian by definition. The nice property is that the posterior $p_c(c|x)$ is Gaussian as well. Therefore, the marginal mean and the marginal variance (that, we recall, can be computed directly for the posterior) give the new values for μ'_c and σ'^2_c . At this point, $c \sim \mathcal{N}(\mu'_c, \sigma'^2_c)$ can be used as a prior for the next step, thus implementing an iterative algorithm when new measurements arrive. This is the basis of Bayesian filtering (e.g. Kalman filtering).

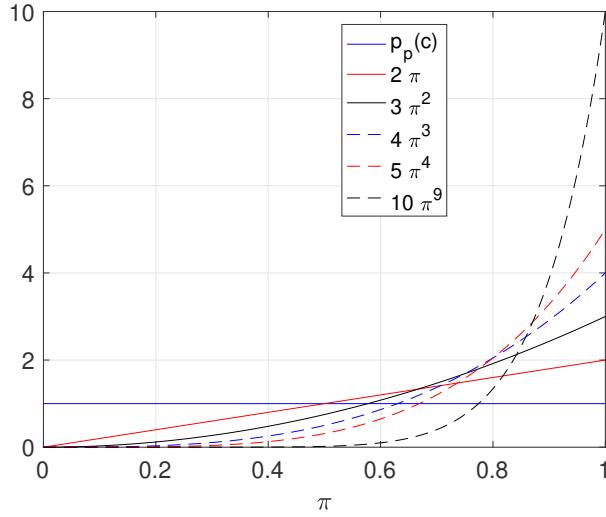


Figure 12.3: Increasing number of measurements $x_i = T$ for the continuous pdf example.

12.3 Most Popular Estimators

12.3.1 Maximum Likelihood

If no prior $p(x)$ is available, no *posterior* pdf can be defined. However, we can define the *pdf of the measurements conditioned on the parameter* x , which is called the *likelihood function* (LF) of x and it is given by

$$\Lambda_k(x) = p(Z^k|x).$$

The *likelihood function*, that measures how “likely” a parameter value is given the observations, measure the *evidence from the data*. The *Maximum Likelihood* (ML) estimator maximises the LF, i.e.,

$$\hat{x}^{ML}[k, Z^k] = \arg \max_{\hat{x}} \Lambda_k(\hat{x}) = \arg \max_{\hat{x}} p(Z^k|\hat{x}).$$

Notice that since $\hat{x}^{ML}[k, Z^k]$ is a function of the random variables Z^k , it is a **rv** even if the parameter x is not.

The solution of the ML is given by

$$\frac{d\Lambda_k(\hat{x})}{d\hat{x}} = \frac{dp(Z^k|\hat{x})}{d\hat{x}} = 0.$$

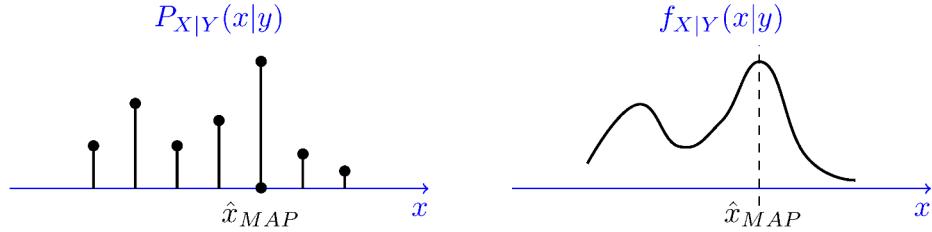


Figure 12.4: The work of the MAP estimator: the estimate is the *mode* of the posterior $p(x|z)$.

12.3.2 Maximum A Posteriori

As pointed out in the previous slides, the *Bayesian* estimator computes the *posterior* $p(\hat{x}|Z^k)$ given the *prior* $p(\hat{x})$ and the *likelihood function* $p(Z^k|\hat{x})$, i.e.,

$$p(\hat{x}|Z^k) = \frac{p(Z^k|\hat{x})p(\hat{x})}{p(Z^k)} = cp(Z^k|\hat{x})p(\hat{x}),$$

where c is the normalisation constant computed in the denominator, which is not a function of \hat{x} . In practice, the *Bayesian* estimators apply the *Bayes' Theorem*. Once the *posterior* pdf is known, several different algorithms can be applied.

An example is the *Maximum A Posteriori* (MAP) that computes the maximum of the *posterior* pdf

$$\hat{x}^{MAP}[k, Z^k] = \arg \max_{\hat{x}} p(\hat{x}|Z^k) = \arg \max_{\hat{x}} p(Z^k|\hat{x})p(\hat{x}),$$

where the normalisation constant has been removed since it does not depend on \hat{x} . Obviously, the MAP estimator $\hat{x}^{MAP}[k, Z^k]$ returns a value that is the realisation of a random variable.

Remark 13. *The MAP estimates are point estimates, whereas Bayesian methods are characterised by the use of distributions to summarise data and draw inferences, hence the MAP is marginally representative of the potentiality of the Bayesian inference, as represented in Figure 12.4.*

Let us make an example: consider a single measurement:

$$z = x + w,$$

and suppose that x is an unknown parameter and that $w \sim \mathcal{N}(0, \sigma^2)$. Therefore, we know that z is a Gaussian **rv**, characterised by knowing only the first two moments, i.e.

$$\mathbb{E}\{z\} = \mathbb{E}\{x + w\} = x + \mathbb{E}\{w\} = x,$$

and

$$\text{V}\{z\} = \text{E}\{(z - \text{E}\{z\})^2\} = \text{E}\{(x + w - x)^2\} = \text{E}\{w^2\} = \sigma^2.$$

As a consequence $z \sim \mathcal{N}(x, \sigma^2)$.

First assume that no prior is given. Hence, for the ML estimator, the *likelihood function* will be given by

$$\Lambda(\hat{x}) = p(z|\hat{x}) = \mathcal{N}(\hat{x}, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\hat{x})^2}{2\sigma^2}}.$$

Therefore,

$$\hat{x}^{ML} = \arg \max_{\hat{x}} \Lambda(\hat{x}) = \arg \min_x (z - \hat{x})^2 = z.$$

This idea can be generalised to a set of *i.i.d.*, e.g. *independent and identically distributed*, measures Z^k , each normally distributed with the likelihood function $\Lambda(\hat{x}) = \mathcal{N}(\hat{x}, \sigma^2)$. In this case, we have

$$\begin{aligned} p(Z^k|\hat{x}) &= p(z(1), \dots, z(k)|\hat{x}) = \\ &= \prod_{j=1}^k \mathcal{N}(\hat{x}, \sigma^2) = c'' e^{-\frac{1}{2\sigma^2} \sum_{i=1}^k (z(i)-\hat{x})^2} \end{aligned}$$

Therefore,

$$\hat{x}^{ML} = \arg \max_{\hat{x}} \Lambda(\hat{x}) = \arg \min_x \sum_{i=1}^k (z(i) - \hat{x})^2 = \frac{1}{k} \sum_{i=1}^k z(i).$$

Moreover, if they are not *i.i.d.* but still *uncorrelated* (i.e. *independent*, since we are considering Gaussian rvs), measures Z^k , each normally distributed with likelihood function $\Lambda_i(\hat{x}) = \mathcal{N}(\hat{x}, \sigma_i^2)$. In this case, we have

$$\begin{aligned} p(Z^k|\hat{x}) &= p(z(1), \dots, z(k)|\hat{x}) = \\ &= \prod_{j=1}^k \mathcal{N}(\hat{x}, \sigma_j^2) = c'' e^{-\frac{1}{2} \sum_{i=1}^k \frac{(z(i)-\hat{x})^2}{\sigma_i^2}} \end{aligned}$$

Therefore,

$$\hat{x}^{ML} = \arg \max_{\hat{x}} \Lambda(\hat{x}) = \arg \min_x \sum_{i=1}^k \frac{(z(i) - \hat{x})^2}{\sigma_i^2} = \frac{\sum_{i=1}^k \frac{z(i)}{\sigma_i^2}}{\sum_{i=1}^k \frac{1}{\sigma_i^2}},$$

i.e. a *weighted arithmetic mean*.

Suppose now that an a priori information is given, i.e., $\hat{x} \sim \mathcal{N}(\mu_x, \sigma_x^2)$, where \hat{x} is *independent* from w . Hence the posterior is given by

$$p(\hat{x}|z) = c' p(z|\hat{x}) p(\hat{x}) = c e^{-\frac{(z-\hat{x})^2}{2\sigma^2} - \frac{(\hat{x}-\mu_x)^2}{2\sigma_x^2}}.$$

Notice that

$$\hat{x}^{MAP}(z) = \arg \max_{\hat{x}} p(\hat{x}|z) = \arg \min_{\hat{x}} -\frac{(z-\hat{x})^2}{2\sigma^2} - \frac{(\hat{x}-\mu_x)^2}{2\sigma_x^2}.$$

It immediately follows that

$$\hat{x}^{MAP}(z) = \frac{\sigma^2}{\sigma^2 + \sigma_x^2} \mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} z.$$

Since the conditional pdf of a Gaussian distribution *is* a Gaussian distribution, it follows immediately that

$$p(\hat{x}|z) = \mathcal{N}(\eta_{x|z}, \sigma_{x|z}^2) = \frac{1}{\sqrt{2\pi}\sigma_{x|z}} e^{-\frac{(\hat{x}-\eta_{x|z})^2}{2\sigma_{x|z}^2}}.$$

In this case, we have obviously that

$$\hat{x}^{MAP}(z) = \arg \max_{\hat{x}} p(\hat{x}|z) \Rightarrow \hat{x}^{MAP}(z) = \eta_{x|z}.$$

In other words, we have immediately that the posterior Gaussian is expressed in terms of the *conditional mean* and *conditional variance*!

From the previous analysis, $\eta_{x|z}$ is given by

$$\eta_{x|z} = \frac{\sigma^2}{\sigma^2 + \sigma_x^2} \mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} z \text{ and } \sigma_{x|z}^2 = \frac{\sigma^2 \sigma_x^2}{\sigma^2 + \sigma_x^2},$$

which is the formula of parallel resistors. Notice how the component with the highest uncertainty, weights less in the estimation. Moreover,

$$\eta_{x|z} = \mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} (z - \mu_x).$$

This idea can be generalised to a set of *i.i.d.*, e.g. *independent and identically distributed*, measures Z^k , each normally distributed $\mathcal{N}(x, \sigma^2)$. In this case, we have

$$\begin{aligned} p(Z^k|x) &= p(z(1), \dots, z(k)|x) = \\ &= \prod_{j=1}^k \mathcal{N}(x, \sigma^2) = c'' e^{-\frac{1}{2\sigma^2} \sum_{i=1}^k (z(i)-x)^2} \end{aligned}$$

Hence, the posterior is given by

$$p(\hat{x}|Z^k) = c' p(Z^k|\hat{x}) p(\hat{x}) = ce^{-\left[\frac{1}{2\sigma^2} \sum_{i=1}^k (z(i)-\hat{x})^2\right] - \frac{(\hat{x}-\mu_x)^2}{2\sigma_x^2}}.$$

For the MAP we have

$$\begin{aligned}\hat{x}^{MAP}[k, Z^k] &= \arg \max_{\hat{x}} p(\hat{x}|Z^k) = \\ &= \arg \min_{\hat{x}} \left[-\frac{1}{2\sigma^2} \sum_{i=1}^k (z(i) - \hat{x})^2 \right] - \frac{(\hat{x} - \mu_x)^2}{2\sigma_x^2}.\end{aligned}$$

It immediately follows that

$$\hat{x}^{MAP}(z) = \frac{\frac{\sigma^2}{k}}{\frac{\sigma^2}{k} + \sigma_x^2} \mu_x + \frac{\sigma_x^2}{\frac{\sigma^2}{k} + \sigma_x^2} \left(\frac{1}{k} \sum_{i=1}^k z(i) \right) = \eta_{x|z}.$$

Some remarks are now in order:

- The MAP estimator is a weighted sum between: the ML estimator and μ_x , which is the peak of the prior.
- Depending on the number of the available measurements Z^k , the MAP chooses either the prior (when k is small), the measurements (when k is relatively high) or a combination thereof.

12.3.3 Least Squares

Another non-Bayesian estimator for nonrandom parameters is the *Least Squares* (LS) method. Let us consider a set of measurements about a *nonrandom* variable x affected by random noise w as

$$z(j) = x + w(j), \quad j = 1, \dots, k,$$

the *Least Squares Estimator* is given by

$$\hat{x}^{LS}[k, Z^k] = \arg \min_{\hat{x}} \sum_{j=1}^k [z(j) - \hat{x}]^2.$$

This is a *linear LS problem*, the simplest form of *linear regression*. This solution *makes no assumption* about the noise, however, if this is *i.i.d.* (independent and identically distributed) and zero-mean Gaussian $w(j) \sim \mathcal{N}(0, \sigma^2)$, then it turns out that

$$z(j) \sim \mathcal{N}(x, \sigma^2), \quad j = 1, \dots, k.$$

Since the *likelihood* function of x is

$$\begin{aligned}\Lambda_k(\hat{x}) &= p(Z^k|\hat{x}) = p(z(1), \dots, z(k)|\hat{x}) = \\ &= \prod_{j=1}^k \mathcal{N}(\hat{x}, \sigma^2) = ce^{-\frac{1}{2\sigma^2} \sum_{i=1}^k (z(j)-\hat{x})^2},\end{aligned}$$

it follows that the maximisation of the ML is *equivalent* to the minimisation of the LS, that is the LS method is a *disguised* ML approach for Gaussian noises.

It has to be noted that the *Least Squares* (LS) method works also for *nonlinear* and possibly *time varying* output functions. Given a nonlinear function of the parameter x affected by random noise w as

$$z(j) = h(j, x) + w(j), \quad j = 1, \dots, k,$$

the *Least Squares Estimator* is given by

$$\hat{x}^{LS}(k) = \arg \min_{\hat{x}} \sum_{j=1}^k [z(j) - h(j, \hat{x})]^2.$$

This is a *nonlinear LS problem*. If, by chance, this noise is *i.i.d.* (independent and identically distributed) and zero-mean Gaussian $w(j) \sim \mathcal{N}(0, \sigma^2)$, then it turns out that

$$z(j) \sim \mathcal{N}(h(j, x), \sigma^2), \quad j = 1, \dots, k.$$

The *likelihood* function of x is again given by

$$\begin{aligned}\Lambda_k(\hat{x}) &= p(Z^k|\hat{x}) = p(z(1), \dots, z(k)|\hat{x}) = \\ &= \prod_{j=1}^k \mathcal{N}(h(j, \hat{x}), \sigma^2) = ce^{-\frac{1}{2\sigma^2} \sum_{i=1}^k (z(j)-h(j,\hat{x}))^2}\end{aligned}$$

which again reveals that the maximisation of the ML is equivalent to the minimisation of the LS for Gaussian noises.

12.3.4 Minimum Mean Square Error

For random parameters, the “counterpart” of the LS is the *Minimum Mean Square Error* (MMSE) estimator. In practice, we consider the *square error* of the estimator

$$(\hat{x} - x)^2.$$

Since we are dealing with a **rv**, we want to consider its *mean* assuming the knowledge of the measurements Z^k , i.e.

$$\mathrm{E} \{ (\hat{x} - x)^2 | Z^k \}.$$

Finally, the MMSE is given by

$$\hat{x}^{MMSE}[k, Z^k] = \arg \min_{\hat{x}} \mathrm{E} \{ (\hat{x} - x)^2 | Z^k \}.$$

To find the solution to the MMSE estimator, i.e.,

$$\hat{x}^{MMSE}[k, Z^k] = \arg \min_{\hat{x}} \mathrm{E} \{ (\hat{x} - x)^2 | Z^k \},$$

we set the derivative with respect to \hat{x} to zero

$$\frac{d\mathrm{E} \{ (\hat{x} - x)^2 | Z^k \}}{d\hat{x}} = \mathrm{E} \{ 2(\hat{x} - x) | Z^k \} = 2(\hat{x} - \mathrm{E} \{ x | Z^k \}) = 0.$$

It then follows that

$$\hat{x}^{MMSE}[k, Z^k] = \mathrm{E} \{ \hat{x} | Z^k \} \triangleq \int_{-\infty}^{+\infty} \hat{x} p(\hat{x} | Z^k) d\hat{x},$$

i.e. the *conditional mean*!.

12.3.5 Comments on LS, MAP and MMSE

A few remarks:

- Both $\hat{x}^{LS}[k, Z^k]$ (with noise) and $\hat{x}^{MMSE}[k, Z^k]$ returns random variables.
- The MMSE is a particular case of *Bayesian* estimator where the *expected value* of a *quadratic* function has to be minimised.
- The difference between the MAP and the MMSE is that the MAP finds the most probable x , i.e. the *mode* of the posterior $p(\hat{x} | Z^k)$, while the MMSE finds the *mean* value of the posterior $p(\hat{x} | Z^k)$.

The MAP and the MMSE estimators for the Gaussian can be rewritten in terms of the *conditional mean* and the *conditional variance*, i.e.

$$\eta_{x|z} = \frac{\sigma^2}{\sigma^2 + \sigma_x^2} \mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} z \quad \text{and} \quad \sigma_{x|z}^2 = \frac{\sigma^2 \sigma_x^2}{\sigma^2 + \sigma_x^2}.$$

Notice that in the case of the Gaussian, the mean and the mode actually coincides, hence the equivalence between the two methods.

Remark 14. The MAP and the MMSE estimators for the Gaussian can be rewritten as

$$\begin{aligned}\hat{x}^{MMSE} &= \sigma_x^{-2}(\sigma^{-2} + \sigma_x^{-2})^{-1}\mu_x + \sigma^{-2}(\sigma^{-2} + \sigma_x^{-2})^{-1}z = \\ &= (\sigma^{-2} + \sigma_x^{-2})^{-1} \left(\frac{\mu_x}{\sigma_x^2} + \frac{z}{\sigma^2} \right),\end{aligned}$$

which again expresses that the weightings of the prior and the measurements are inversely proportional to their variances.

Remark 15. Since the inverse of the variances is called the information and since $\sigma_{x|z}^{-2} = \sigma^{-2} + \sigma_x^{-2}$, the inverse of the variances are additive. This additivity property holds in general when the pdfs are independent.

Let us make the same example. Consider a single measurement:

$$z = x + w,$$

and suppose that x is an unknown nonrandom parameter. The *Least Squares* criterion leads to

$$\hat{x}^{LS} = \arg \max_{\hat{x}} (z - \hat{x})^2 = z.$$

Recall that if the noise is Gaussian, i.e. $w \sim \mathcal{N}(0, \sigma^2)$, $\hat{x}^{LS} = \hat{x}^{ML}$.

Let us consider a single measurement:

$$z = x + w,$$

and suppose that x is an unknown random parameter. Assuming both \hat{x} and w normally distributed, the *conditional* pdf is given by

$$p(\hat{x}|z) = \mathcal{N}(\eta_{x|z}, \sigma_{x|z}^2) = \frac{1}{\sqrt{2\pi}\sigma_{x|z}} e^{-\frac{(\hat{x}-\eta_{x|z})^2}{2\sigma_{x|z}^2}}.$$

Notice that for the Gaussian, the peak (the *mode*) coincides with the *mean* of the conditional pdf, which is the value returned by the MMSE, hence

$$\hat{x}^{MMSE} = \mathbb{E}\{\hat{x}|z\} = \eta_{x|z} = \hat{x}^{MAP}.$$

12.3.6 Unbiased estimators

For a nonrandom parameter x_0 , we say that an estimator is *unbiased* if

$$\mathbb{E}\{\hat{x}[k, Z^k]\} = x_0,$$

where x_0 is the true value of x . For a random parameter x with prior $p(x)$, we say that an estimator is *unbiased* if

$$\mathrm{E}\{\hat{x}[k, Z^k]\} = \mathrm{E}\{x\},$$

where the left term is computed on the *joint* pdf $p(x, Z^k)$, while the term on the right is computed on the $p(x)$. In the general case, an estimator is *unbiased* if

$$\mathrm{E}\{\tilde{x}(k)\} = 0,$$

where $\tilde{x}(k) = x - \hat{x}(k)$ is the *estimation error* at the k -th iteration of the estimator.

An estimator is *asymptotically unbiased* if the previous holds only for $k \rightarrow +\infty$.

For example, let us consider a single measurement:

$$z = x + w,$$

and suppose that x is an unknown parameter.

For the ML we have

$$\mathrm{E}\{\hat{x}^{ML}\} = \mathrm{E}\{z\} = \mathrm{E}\{x + w\} = x.$$

Equivalently:

$$\mathrm{E}\{\tilde{x}\} = \mathrm{E}\{x - \hat{x}^{ML}\} = x - \mathrm{E}\{\hat{x}^{ML}\} = 0.$$

For the MAP, we have

$$\begin{aligned} \mathrm{E}\{\hat{x}^{MAP}\} &= \mathrm{E}\left\{\frac{\sigma^2}{\sigma^2 + \sigma_x^2}\mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2}z\right\} = \\ &= \frac{\sigma^2}{\sigma^2 + \sigma_x^2}\mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2}\mathrm{E}\{z\} = \\ &= \frac{\sigma^2}{\sigma^2 + \sigma_x^2}\mu_x + \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2}(\mu_x + \mathrm{E}\{w\}) = \mu_x. \end{aligned}$$

We always mentioned that the choice of the prior is crucial. Indeed, if an incorrect choice is made, we have:

$$\begin{aligned} \mathrm{E}\{\hat{x}^{MAP}\} &= \mathrm{E}\left\{\frac{\sigma^2}{\sigma^2 + \bar{\sigma}_x^2}\bar{\mu}_x + \frac{\bar{\sigma}_x^2}{\sigma^2 + \bar{\sigma}_x^2}z\right\} = \\ &= \frac{\sigma^2}{\sigma^2 + \bar{\sigma}_x^2}\bar{\mu}_x + \frac{\bar{\sigma}_x^2}{\sigma^2 + \bar{\sigma}_x^2}\mu_x \neq 0. \end{aligned}$$

However, when a sufficiently large number of measurements are available, we have

$$E\{\hat{x}^{MAP}\} = \frac{\frac{\sigma^2}{k}}{\frac{\sigma^2}{k} + \bar{\sigma}_x^2} \bar{\mu}_x + \frac{\bar{\sigma}_x^2}{\frac{\sigma^2}{k} + \bar{\sigma}_x^2} \hat{\mu}_x,$$

where $\hat{\mu}_x$ is the arithmetic mean. Notice that when $k \rightarrow +\infty$, we have that $\hat{\mu}_x \rightarrow \mu_x$ and the MAP rewards the measurements, hence

$$\lim_{k \rightarrow +\infty} E\{\hat{x}^{MAP}\} = \mu_x.$$

With the definition of the estimation error $\tilde{x} = x - \hat{x}$, we can say that:

$$E\{\tilde{x}^2\} = V\{\hat{x}\},$$

if \hat{x} is *unbiased* and x is *nonrandom*. On the contrary, if \hat{x} is *biased* and/or x is *random*, we have

$$E\{\tilde{x}^2\} = MSE(\hat{x}),$$

where MSE stands for the *mean squared error*.

To summarise the basic estimator performances, there are basically two main approaches to estimation given the peculiarity of the time-invariant parameter x to be estimated.

Definition 68. *If the parameter is nonrandom, then x has a true value x_0 that has to be estimated. To this end, non-Bayesian or Fisher approaches will be used.*

Definition 69. *If the parameter is random and if a prior on x in terms of a pdf $p(x)$ is known, a particular realisation is supposed to be available and, hence, the Bayesian approaches will be used.*

12.3.7 Multidimensional MMSE for Gaussian uncertainties

Let us generalise the MMSE analysis thus introduced and let us consider two vectors of rvs $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ that are jointly Gaussian, we have:

$$y = \begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}(\mu_y, P_{yy}).$$

In other words:

$$p(y) = p(x, z) = \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}(y-\mu_y)^T P_{yy}^{-1}(y-\mu_y)},$$

where

$$\mu_y = \begin{bmatrix} \mu_x \\ \mu_z \end{bmatrix} \quad \text{and} \quad P_{yy} = \begin{bmatrix} P_{xx} & P_{xz} \\ P_{zx} & P_{zz} \end{bmatrix},$$

where $P_{xz} = P_{zx}^T$ is the *cross covariance matrix*. Let us first try derive the marginals, i.e. $p_x(x)$ and $p_z(z)$, by defining $a = x - \mu_x$ and $b = z - \mu_z$, i.e.

$$p(x, z) = \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}[a^T \ b^T] P_{yy}^{-1} \begin{bmatrix} a \\ b \end{bmatrix}}.$$

Let us consider

$$P_{yy}^{-1} = \begin{bmatrix} P_{xx} & P_{xz} \\ P_{zx} & P_{zz} \end{bmatrix}^{-1} = \begin{bmatrix} Q_{xx} & Q_{xz} \\ Q_{zx} & Q_{zz} \end{bmatrix},$$

where by definition if the covariance matrix $P_{xx} = P_{xx}^T$, $P_{zz} = P_{zz}^T$ and $P_{xz} = P_{zx}^T$. By computing the matrix product at the exponent, we have the scalar value

$$g = a^T Q_{xx} a + a^T Q_{xz} b + b^T Q_{zx} a + b^T Q_{zz} b.$$

By the matrix inversion lemma we have immediately that:

$$\begin{aligned} Q_{xx} &= (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} = P_{xx}^{-1} + P_{xx}^{-1} P_{xz} (P_{zz} - P_{zx} P_{xx}^{-1} P_{xz})^{-1} P_{zx} P_{xx}^{-1}, \\ Q_{zz} &= (P_{zz} - P_{zx} P_{xx}^{-1} P_{xz})^{-1} = P_{zz}^{-1} + P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} P_{xz} P_{zz}^{-1}, \\ Q_{xz} &= -P_{xx}^{-1} P_{xz} (P_{zz} - P_{zx} P_{xx}^{-1} P_{xz})^{-1} = Q_{zx}^T, \\ Q_{zx} &= -P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} = Q_{xz}^T, \end{aligned}$$

which ensures the fact that the inverse is symmetric as well, i.e. $Q_{xx} = Q_{xx}^T$, $Q_{zz} = Q_{zz}^T$ and $Q_{xz} = Q_{zx}^T$.

Substituting the previous equations in g , we have

$$\begin{aligned} g &= a^T Q_{xx} a + a^T Q_{xz} b + b^T Q_{zx} a + b^T Q_{zz} b = a^T Q_{xx} a + 2b^T Q_{zx} a + b^T Q_{zz} b = \\ &= a^T (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} a - 2b^T P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} a + \\ &\quad + b^T [P_{zz}^{-1} + P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} P_{xz} P_{zz}^{-1}] b = \\ &= a^T (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} a - 2b^T P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} a + \\ &\quad + b^T P_{zz}^{-1} b + b^T P_{zz}^{-1} P_{zx} (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} P_{xz} P_{zz}^{-1} b = \\ &= [a - P_{xz} P_{zz}^{-1} b]^T (P_{xx} - P_{xz} P_{zz}^{-1} P_{zx})^{-1} [a - P_{xz} P_{zz}^{-1} b] + b^T P_{zz}^{-1} b \end{aligned}$$

Recalling that $a = x - \mu_x$ and $b = z - \mu_z$, we can define

$$\gamma = \mu_x + P_{xz} P_{zz}^{-1} (z - \mu_z) \quad \text{and} \quad \Gamma = P_{xx} - P_{xz} P_{zz}^{-1} P_{zx},$$

so hence to have

$$G_1(z) = (z - \mu_z)^T P_{zz}^{-1}(z - \mu_z) \text{ and } G_2(x) = (x - \gamma)^T \Gamma^{-1}(x - \gamma),$$

and finally

$$g = G_1(z) + G_2(x).$$

Therefore, we have

$$\begin{aligned} p(x, z) &= \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}[a^T \ b^T] P_{yy}^{-1} \begin{bmatrix} a \\ b \end{bmatrix}} = \\ &= \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}(G_1(z) + G_2(x))}. \end{aligned}$$

First notice that the term $|2\pi P_{yy}| = (2\pi)^{n+m}|P_{yy}|$ and then we notice that

$$|P_{yy}| = |P_{zz}||P_{xx} - P_{xz}P_{zz}^{-1}P_{zx}| = |P_{zz}||\Gamma|.$$

Indeed, for any block partitioned matrix

$$P_{yy} = \begin{bmatrix} P_{xx} & P_{xz} \\ P_{zx} & P_{zz} \end{bmatrix},$$

it can be expressed as the product of two *triangular* matrices, i.e.

$$P_{yy} = \begin{bmatrix} P_{xx} & P_{xz} \\ P_{zx} & P_{zz} \end{bmatrix} = \begin{bmatrix} I & P_{xz} \\ 0 & P_{zz} \end{bmatrix} \begin{bmatrix} P_{xx} - P_{xz}P_{zz}^{-1}P_{zx} & 0 \\ P_{zz}^{-1}P_{zx} & I \end{bmatrix}.$$

Recalling that $|AB| = |A||B|$, the proof follows.

As a consequence

$$\begin{aligned} p(x, z) &= \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}(G_1(z) + G_2(x))} = \\ &= \frac{1}{\sqrt{(2\pi)^m |P_{zz}|}} e^{-\frac{1}{2}G_1(z)} \frac{1}{\sqrt{(2\pi)^n |\Gamma|}} e^{-\frac{1}{2}G_2(x)}. \end{aligned}$$

It then follows that the marginal

$$\begin{aligned} p_z(z) &= \int_{-\infty}^{+\infty} p(x, z) dx = \frac{1}{\sqrt{(2\pi)^m |P_{zz}|}} e^{-\frac{1}{2}G_1(z)} = \\ &= \frac{1}{\sqrt{(2\pi)^m |P_{zz}|}} e^{-\frac{1}{2}(z - \mu_z)^T P_{zz}^{-1}(z - \mu_z)}. \end{aligned}$$

hence we have shown that the marginal $p_z(z) \sim \mathcal{N}(\mu_z, P_{zz})$!

In practice, given the joint Gaussian pdf defined as

$$p(y) = p(x, z) = \frac{1}{\sqrt{|2\pi P_{yy}|}} e^{-\frac{1}{2}(y-\mu_y)^T P_{yy}^{-1}(y-\mu_y)},$$

where

$$\mu_y = \begin{bmatrix} \mu_x \\ \mu_z \end{bmatrix} \text{ and } P_{yy} = \begin{bmatrix} P_{xx} & P_{xz} \\ P_{zx} & P_{zz} \end{bmatrix},$$

the marginal $p_z(z) \sim \mathcal{N}(\mu_z, P_{zz})$ is a Gaussian whose mean value and covariance matrix are *the correct portion of the mean values vector and the correct portion of the covariance matrix!* Of course, by exchanging the role of z and x , we can compute similarly the marginal $p_x(x) \sim \mathcal{N}(\mu_x, P_{xx})$. The interesting thing is that if we compute the conditional $p_c(x|z)$, we have immediately by definition

$$\begin{aligned} p_c(x|z) &= \frac{p(x, z)}{p_z(z)} = \frac{\frac{1}{\sqrt{(2\pi)^m |P_{zz}|}} e^{-\frac{1}{2}G_1(z)}}{\frac{1}{\sqrt{(2\pi)^n |\Gamma|}} e^{-\frac{1}{2}G_2(x)}} = \\ &= \frac{1}{\sqrt{(2\pi)^n |\Gamma|}} e^{-\frac{1}{2}G_2(x)}, \end{aligned}$$

which states that $p_c(x|z) \sim \mathcal{N}(\gamma, \Gamma)$, i.e. a Gaussian with mean value and covariance matrix

$$\gamma = \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z) \text{ and } \Gamma = P_{xx} - P_{xz}P_{zz}^{-1}P_{zx}.$$

We know that if x is the unknown vector and z are the measurements, we have to compute $p(x|z)$ to have an *estimate*. Moreover, we know that the *minimum mean square error* (MMSE) is a *Bayesian* estimator that is computed with the *conditional mean*, i.e.

$$\hat{x}^{MMSE} = \mathbb{E}\{x|z\}.$$

Hence, it follows immediately that

$$\hat{x}^{MMSE} = \mathbb{E}\{x|z\} = \gamma = \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z).$$

Remark 16. *The expressions of the conditional mean and conditional covariance are then*

$$E\{x|z\} = \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z) \text{ and } C\{x|z\} = \Gamma = P_{xx} - P_{xz}P_{zz}^{-1}P_{zx},$$

that are referred to as the fundamental equations of linear estimation.

Indeed, the *conditional mean* depends *linearly* by the observations z , while the *conditional covariance* is *independent* from the observations. Notice that, again, the results for the mean and the covariance are a weighted composition between the prior and the measures. Of course it is possible to compute the *marginal mean* and the *marginal covariance* using the *Total Expectations* and *Total Variance* Laws.

Remark 17. The linear MMSE is also called in the literature least mean squares (LMS), minimum variance (MV) or least squares (LS).

Personally I prefer *minimum variance* since it immediately explains what is the job of the estimator.

Facts on the MMSE:

- The *linear MMSE* estimator is identical to the expression of the *conditional mean* of Gaussian random vectors.
- The *linear MMSE* estimator is the *best estimator* if the random variables are *Gaussian*.
- The quite interesting thing is that the *linear MMSE* estimator is the *best linear estimator* if the random variables are *not Gaussian* and generically distributed.

Chapter 13

Estimators

13.1 Linear Estimators

13.1.1 The Least Squares solution

Let us consider two sensors measuring the same nonrandom quantity x with a different linear characteristic

$$z_1 = \alpha_1 x \text{ and } z_2 = \alpha_2 x.$$

One possible choice is just to select one of the two. However we can decide (again) to minimise the *sum of the squares of the residuals* (LS approach):

$$\min_x J(x) = \min_x (z_1 - \alpha_1 x)^2 + (z_2 - \alpha_2 x)^2.$$

The solution would then be given

$$\frac{dJ(x)}{dx} = 0 \Rightarrow x = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1^2 + \alpha_2^2}.$$

In general, for n distributed sensors we have:

$$\hat{x}^{LS} = \frac{\sum_{i=1}^n \alpha_i z_i}{\sum_{i=1}^n \alpha_i^2}.$$

If the sensor is the same measured multiple times or if we have multiple equal sensors, we have $\alpha_i = \alpha, \forall i$. Hence

$$\hat{x}^{LS} = \frac{1}{\alpha} \frac{\sum_{i=1}^n z_i}{n},$$

which is the *arithmetic mean*.

Let us consider the previous problem, differently weighting the sensors

$$\min_x J(x) = \min_x k_1(z_1 - \alpha_1 x)^2 + k_2(z_2 - \alpha_2 x)^2,$$

if $k_1 > k_2$ then the solution will be more “attracted” by z_1 . In general, for n distributed sensors we have:

$$\hat{x}^{LS} = \frac{\sum_{i=1}^n k_i \alpha_i z_i}{\sum_{i=1}^n k_i \alpha_i^2}.$$

If the sensor is the same measured multiple times or if we have multiple equal sensors, we have $\alpha_i = \alpha, \forall i$. Hence

$$\hat{x}^{LS} = \frac{1}{\alpha} \frac{\sum_{i=1}^n k_i z_i}{\sum_{i=1}^n k_i},$$

which is a weighted sum with coefficients

$$\frac{k_i}{\sum_{i=1}^n k_i},$$

i.e. *a convex linear combination*.

Let us now consider two measurements about sensors measuring the same nonrandom quantity x with a different linear characteristic

$$z_1 = \alpha_1 x + \varepsilon_1 \text{ and } z_2 = \alpha_2 x + \varepsilon_2,$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. If, in general, $\sigma_i^2 = \sigma^2, \forall i$, there is not a preferred choice. Hence the solution is the usual *arithmetic mean*. Notice that, this is also the solution *if no stochastic description is available upfront about ε_i* .

Let us now assume that $\sigma_i^2 \neq \sigma_j^2, \forall i \neq j$. As we saw from the MAP, considering all the sensors as *equally precise* may lead to *increased estimation uncertainty*, while using *less* data as available leads to *under-exploitation* of the available information. Is it possible to prove this case?

Since we already know the solution when all the variances are equal to σ^2 , let us reduce to that case by *scaling the i -th measurement equation* with a factor k_i

$$z_i = \alpha_i x + \varepsilon_i \Rightarrow k_i z_i = k_i \alpha_i x + k_i \varepsilon_i \Rightarrow z_i^* = \alpha_i^* x + \varepsilon_i^*$$

In particular we want that the variance of ε_i^* be equal to $\sigma^2 = 1$ (which the simplest possible choice for the free parameter σ^2), that is

$$V\{\varepsilon_i^*\} = V\{k_i \varepsilon_i\} = k_i^2 V\{\varepsilon_i\} = k_i^2 \sigma_i^2 = \sigma^2 = 1 \Rightarrow k_i = \frac{1}{\sigma_i}.$$

It then follows, that the LS index to minimise is

$$J(x) = \sum_{i=1}^n (z_i^* - \alpha_i^* x)^2 = \sum_{i=1}^n k_i^2 (z_i - \alpha_i x)^2$$

Therefore the solution is given by the weighted LS as

$$\hat{x}^{LS} = \arg \min_x J(x) = \frac{\sum_{i=1}^n k_i^2 \alpha_i z_i}{\sum_{i=1}^n k_i^2 \alpha_i^2} = \frac{\sum_{i=1}^n \frac{\alpha_i}{\sigma_i^2} z_i}{\sum_{i=1}^n \frac{\alpha_i^2}{\sigma_i^2}}$$

that is the contribution of each sensor is weighted by the amount of *information* it adds to the estimation problem. Moreover, the solution depends on the combination of the σ_i^2 and on the output function, i.e., $z = h(x)$, exactly as what happened for *Maximum likelihood* in the Gaussian case.

Generalisation to the multidimensional case

The interesting point is that the previous solutions can be generalised to a generic set of *linear* measure. Indeed, let us consider $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, $H \in \mathbb{R}^{m \times n}$, with $m \geq n$ and H full rank. We have that

$$\begin{aligned} z = Hx \Rightarrow J(x) &= (z - Hx)^T (z - Hx) \Rightarrow \\ \Rightarrow \hat{x}^{LS} &= \arg \min_x J(x) = (H^T H)^{-1} H^T z, \end{aligned}$$

that is the solution of the sensor calibration we have seen previously. Similarly, if we add a *weighting matrix* W (which has to be symmetric and p.d. in order to have the Hessian p.d.), we have

$$\begin{aligned} z = Hx \Rightarrow J(x) &= (z - Hx)^T W (z - Hx) \Rightarrow \\ \Rightarrow \hat{x}^{LS} &= \arg \min_x J(x) = (H^T W H)^{-1} H^T W z. \end{aligned}$$

In other words, the *Weighted Least Squares* (WLS) is actually a BLUE for a generic pdf, while it is the MVUE for the Gaussian case.

We are now in a position to present the *LS batch* algorithm with noisy measures. Consider the following problem:

Problem 5. *We have to determine the value of a nonrandom constant parameter. We collect the sensor readings from multiple sensors deployed in an environment by means of a communication system. The data are collected in different time instants $k = 0, \dots$ and, to ensure a correct sensor fusion, the data are timestamped at the source location and all the sensors are supposed*

to be synchronised at the Coordinated Universal Time (UTC), for example using GPS signals. The set of sensor readings is not necessarily the same and, hence, may change in time, while each sensor provides the measures with a different (possibly time varying) uncertainty, i.e., the stochastic process affecting the sensors is non-stationary. The data are collected by a single entity fusing all the data, i.e., centralised approach. \square

Believe it or not, you know perfectly how to solve this problem:

- First, the synchronisation of the nodes allow us to assume that all the sensor readings are not affected by the *network induced problems*, e.g., jitter, latency, etc.;
- Second, since the quantity to estimate is *nonrandom* and *constant*, a *non Bayesian* approach should be used;
- Since no information is given about the pdf of the noise, it is impractical to use the ML, and hence we go for the LS approach, performed in a centralised way.

To properly model this problem, let us define

$$z(i) = H(i)x + \varepsilon(i), \quad i = 1, \dots, k \quad (13.1)$$

the time varying set of measurements collected at time i . Then, let us define the following aggregating vectors

$$Z^k = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(k) \end{bmatrix}, \quad H^k = \begin{bmatrix} H(1) \\ H(2) \\ \vdots \\ H(k) \end{bmatrix}, \quad \varepsilon^k = \begin{bmatrix} \varepsilon(1) \\ \varepsilon(2) \\ \vdots \\ \varepsilon(k) \end{bmatrix}. \quad (13.2)$$

For the covariance matrix of the noises

$$C^k = \begin{bmatrix} C\{\varepsilon(1)\} & 0 & \dots & 0 \\ 0 & C\{\varepsilon(2)\} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C\{\varepsilon(k)\} \end{bmatrix} \quad (13.3)$$

Notice how the noise is assumed uncorrelated in different time instants, while instead it can be correlated for each time instant ($C\{\varepsilon(i)\}$ is not necessarily diagonal).

Following the same idea presented previously, we have

$$\begin{aligned}
J(x, k) &= \sum_{i=1}^k (z(i) - H(i)x)^T C \{\varepsilon(i)\}^{-1} (z(i) - H(i)x) = \\
&= (Z^k - H^k x)^T C^{k-1} (Z^k - H^k x) \Rightarrow \\
\Rightarrow \hat{x}^{LS} &= \arg \min_x J(x, k) = (H^{k^T} C^{k-1} H^k)^{-1} H^{k^T} C^{k-1} Z^k.
\end{aligned} \tag{13.4}$$

Notice how the solution has the same properties of the scalar case.

If the sequence of the $\varepsilon(i)$ comprise noises that are zero-mean, normally distributed, the uncorrelatedness becomes *independence*. Therefore

$$p(\varepsilon(i)) = \frac{1}{\sqrt{|2\pi C \{\varepsilon(i)\}|}} e^{-\frac{1}{2}\varepsilon(i)^T C \{\varepsilon(i)\}^{-1} \varepsilon(i)}.$$

As a consequence, the *likelihood function*

$$p(Z^k|x) = \prod_{i=1}^k p(z(i)|x) = \prod_{i=1}^k p(H(i)x + \varepsilon(i)|x) = \prod_{i=1}^k \mathcal{N}(H(i)x, C \{\varepsilon(i)\})$$

Therefore,

$$\begin{aligned}
p(H(i)x + \varepsilon(i)|x) &= ce^{-\frac{1}{2}(H(i)x + \varepsilon(i) - H(i)x)^T C \{\varepsilon(i)\}^{-1} (H(i)x + \varepsilon(i) - H(i)x)} = \\
&= cp(\varepsilon(i)|x)
\end{aligned}$$

Hence, the *likelihood function* becomes

$$\begin{aligned}
p(Z^k|x) &= \prod_{i=1}^k p(z(i)|x) = \prod_{i=1}^k p(H(i)x + \varepsilon(i)|x) = \prod_{i=1}^k p(\varepsilon(i)|x) = \\
&= \prod_{i=1}^k \frac{1}{\sqrt{|2\pi C \{\varepsilon(i)\}|}} \prod_{i=1}^k e^{-\frac{1}{2}\varepsilon(i)^T C \{\varepsilon(i)\}^{-1} \varepsilon(i)} = \\
&= ce^{-\frac{1}{2}\sum_{i=1}^k \varepsilon(i)^T C \{\varepsilon(i)\}^{-1} \varepsilon(i)}
\end{aligned}$$

Since

$$z(i) = H(i)x + \varepsilon(i) \Rightarrow \varepsilon(i) = z(i) - H(i)x,$$

we have

$$\begin{aligned}
p(Z^k|x) &= ce^{-\frac{1}{2}\sum_{i=1}^k \varepsilon(i)^T C \{\varepsilon(i)\}^{-1} \varepsilon(i)} = \\
&= ce^{-\frac{1}{2}\sum_{i=1}^k (z(i) - H(i)x)^T C \{\varepsilon(i)\}^{-1} (z(i) - H(i)x)} = \\
&= ce^{-\frac{1}{2}(Z^k - H^k x)^T C \{\varepsilon(i)^k\}^{-1} (Z^k - H^k x)},
\end{aligned}$$

which has its maximum wherever the exponent is minimum. Again, if the measurement noise is normally distributed, zero-mean and white, the LS is a “disguised” version of the ML.

If the noises $\varepsilon(i)$ are uncorrelated and zero-mean (without *any other assumption on the noise stochastic process description*), the LS is *unbiased*. Indeed:

$$\begin{aligned}\mathrm{E}\{\hat{x}(k)^{LS}\} &= (H^{kT} C^{k-1} H^k)^{-1} H^{kT} C^{k-1} \mathrm{E}\{Z^k\} = L^k \mathrm{E}\{Z^k\} = \\ L^k \mathrm{E}\{H^k x + \varepsilon^k\} &= L^k H^k x = x\end{aligned}$$

The *estimation error* is instead given by:

$$\tilde{x} = x - \hat{x}^{LS} = x - L^k Z^k = x - L^k (H^k x + \varepsilon^k) = -L^k \varepsilon^k.$$

The *uncertainty* of the estimation is instead given by the *covariance matrix* $P(k)$ of the estimates:

$$\begin{aligned}P(k) &= \mathrm{E}\{(\hat{x}(k) - \mathrm{E}\{\hat{x}(k)\})(\hat{x}(k) - \mathrm{E}\{\hat{x}(k)\})^T\} = \\ &= \mathrm{E}\{(\hat{x}(k) - x)(\hat{x}(k) - x)^T\} = \\ &= \mathrm{E}\{\tilde{x}(k)\tilde{x}(k)^T\} = \mathrm{E}\{-L^k \varepsilon^k (-L^k \varepsilon^k)^T\} = \\ &= L^k C^k L^{kT} = (H^{kT} C^{k-1} H^k)^{-1}.\end{aligned}$$

As a consequence:

$$\hat{x}^{LS} = (H^{kT} C^{k-1} H^k)^{-1} H^{kT} C^{k-1} Z^k = P(k) H^{kT} C^{k-1} Z^k.$$

Recursive solution

The main problem of the LS solution presented till now is that it is *batch*: all the data collected up to k are used simultaneously in the same instant. As time goes by, this solution becomes easily unpractical. However, is it possible to derive a *recursive implementation* of the LS. In practice, each time that a new set of measurements are available at time $k+1$, we make use of the results of the estimates up to time k . This way, we have a remarkable saving in the computation time.

To this end, let us define

$$Z^{k+1} = \begin{bmatrix} Z^k \\ z(k+1) \end{bmatrix}, \quad H^{k+1} = \begin{bmatrix} H^k \\ H(k+1) \end{bmatrix}, \quad \varepsilon^{k+1} = \begin{bmatrix} \varepsilon^k \\ \varepsilon(k+1) \end{bmatrix},$$

and

$$C^{k+1} = \begin{bmatrix} C^k & 0 \\ 0 & C\{\varepsilon(k+1)\} \end{bmatrix}.$$

Let us start with the definition of the estimator *covariance matrix* given previously. In particular, since it involves an inverse, we are interested in the *inverse of the covariance matrix*:

$$\begin{aligned} P(k+1)^{-1} &= H^{k+1T} C^{k+1-1} H^{k+1} = \\ &= H^k T C^{k-1} H^k + H(k+1)^T C \{\varepsilon(k+1)\}^{-1} H(k+1) = \\ &= P(k)^{-1} + H(k+1)^T C \{\varepsilon(k+1)\}^{-1} H(k+1) \end{aligned}$$

This equation is of *paramount importance*. Indeed, since $P(k)^{-1}$ represents the *Fisher Information Matrix* (FIM), i.e., the amount of information collected from the sensor, it states that *whatever is the uncertainty of the new measurements, it will increase the information about x* .

In other words, *all* the sensor readings are useful for the estimates: the information grows *linearly*. To recover the *covariance matrix* $P(k+1)$ we have to invert the previous matrix equation, which is not trivial. However, we can make use of the *matrix inversion lemma*:

$$(A + BCB^T)^{-1} = A^{-1} - A^{-1}B(B^TA^{-1}B + C^{-1})^{-1}B^TA^{-1}.$$

Therefore:

$$\begin{aligned} P(k+1) &= P(k) - P(k)H(k+1)^T \\ &\quad (H(k+1)P(k)H(k+1)^T + C\{\varepsilon(k+1)\})^{-1}H(k+1)P(k). \end{aligned}$$

The term

$$S(k+1) = H(k+1)P(k)H(k+1)^T + C\{\varepsilon(k+1)\},$$

is called the *covariance of the residuals*. It corresponds to the uncertainty accumulated in the estimates of x up to the previous step plus the uncertainty for the new measures.

The matrix

$$\begin{aligned} W(k+1) &= P(k)H(k+1)^T \\ &\quad (H(k+1)P(k)H(k+1)^T + C\{\varepsilon(k+1)\})^{-1} = \\ &= P(k)H(k+1)^T S(k+1)^{-1}, \end{aligned}$$

is called the *update gain*, and it weights the decrease of the uncertainty as a function of the new measures. Finally we have:

$$\begin{aligned} P(k+1) &= (I - W(k+1)H(k+1))P(k) = \\ &= P(k) - W(k+1)S(k+1)W(k+1)^T. \end{aligned}$$

For the estimates we have:

$$\begin{aligned}\hat{x}(k+1) &= P(k+1)H^{k+1T}C^{k+1-1}Z^{k+1} = \\ &= P(k+1)H^kT C^{k-1} Z^k + \\ &\quad + P(k+1)H(k+1)^T C \{\varepsilon(k+1)\}^{-1} z(k+1)\end{aligned}$$

First, notice that (*prove it*)

$$P(k+1)H(k+1)^T C \{\varepsilon(k+1)\}^{-1} = W(k+1).$$

Hence, by substituting the update equation for $P(k+1)$, we have

$$\begin{aligned}\hat{x}(k+1) &= \\ &= (I - W(k+1)H(k+1)) P(k)H^kT C^{k-1} Z^k + W(k+1)z(k+1).\end{aligned}$$

We then recognise that

$$\hat{x}(k) = P(k)H^kT C^{k-1} Z^k.$$

Finally we get:

$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k)).$$

Remark 18. *The update rule of the estimates for the recursive LS estimator*

$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k)),$$

is given by the previous estimates plus a correction term. This term is the product between a gain and a residual.

Remark 19. *The residual is the difference between the new measures $z(k+1)$ and the predicted value of the measures based on the sensor model $H(k+1)$ and the available estimates $\hat{x}(k)$.*

Remark 20. *It is easy now to see that the covariance of the residuals is:*

$$\begin{aligned}S(k+1) &= \\ &= E\{(z(k+1) - H(k+1)\hat{x}(k))(z(k+1) - H(k+1)\hat{x}(k))^T\}.\end{aligned}$$

To recap, the *non Bayesian* WLS recursive solution is given by the following equations, to be computed iteratively once a new set of measures become available:

$$\begin{aligned}S(k+1) &= H(k+1)P(k)H(k+1)^T + C \{\varepsilon(k+1)\} \\ W(k+1) &= P(k)H(k+1)^T S(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k)) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k)\end{aligned}$$

Remark 21. Since the presented scheme is recursive, it needs an **initialisation**. This is done by using a batch technique on a small amount of data or using “*a priori*” information on the value of x and of its covariance matrix P .

Example 30. Consider noisy scalar observations of a **nonrandom** quantity

$$z(i) = x + \varepsilon(i), \quad i = 1, \dots, k$$

Assuming $\varepsilon(i)$ zero-mean **i.i.d.** with variance σ^2 , we have:

$$H^k = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad \text{and} \quad C^k = \sigma^2 I.$$

For the **batch** solution we have

$$\hat{x}(k) = \left(H^{kT} C^{k-1} H^k \right)^{-1} C^{k-1} H^k Z^k = \frac{1}{k} \sum_{i=1}^k z(i),$$

and

$$P(k) = \left(H^{kT} C^{k-1} H^k \right)^{-1} = \frac{\sigma^2}{k},$$

the same results obtained previously. For the **recursive** solution, we obviously start from $\hat{x}(1) = z(1)$ and $P(1) = \sigma^2$. Hence, after k steps we have:

$$\hat{x}(k) = \left(H^{kT} C^{k-1} H^k \right)^{-1} C^{k-1} H^k Z^k = \frac{1}{k} \sum_{i=1}^k z(i),$$

and

$$P(k) = \left(H^{kT} C^{k-1} H^k \right)^{-1} = \frac{\sigma^2}{k},$$

as in the **batch** case.

For the $k+1$ step, involving the $k+1$ -th measure, we have then:

$$\begin{aligned} S(k+1) &= H(k+1)P(k)H(k+1)^T + C\{\varepsilon(k+1)\} = \\ &= \frac{\sigma^2}{k} + \sigma^2 = \frac{k+1}{k}\sigma^2 \\ W(k+1) &= P(k)H(k+1)^T S(k+1)^{-1} = \\ &= \frac{\sigma^2}{k} \left(\frac{k+1}{k}\sigma^2 \right)^{-1} = \frac{1}{k+1} \end{aligned}$$

Hence:

$$\begin{aligned}
\hat{x}(k+1) &= \hat{x}(k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k)) = \\
&= \hat{x}(k) + \frac{1}{k+1}(z(k+1) - H(k+1)\hat{x}(k)) = \\
&= \frac{1}{k+1} \sum_{i=1}^{k+1} z(i), \\
P(k+1) &= (I - W(k+1)H(k+1))P(k) = \\
&= (1 - \frac{1}{k+1})\frac{\sigma^2}{k} = \frac{\sigma^2}{k+1}
\end{aligned}$$

as expected. \square

13.1.2 The Kalman filter

We first present the Kalman filter derivation as an extension of the Luenberger observer. Let us consider the problem of *estimating a certain state variable* $x \in \mathbb{R}^n$, with *known linear time-varying dynamic* governed by the *known inputs* $u \in \mathbb{R}^m$. The information on the state x are provided by a set of sensors returning the measurement values $z \in \mathbb{R}^p$. Therefore

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) \\ z(k) &= H(k)x(k) \end{cases}$$

In this case it is possible to nullify the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ using the following observer

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that, providing the system is *observable* (or at least *detectable*), ensures $\tilde{x}(k) \rightarrow 0$ by properly choosing $L(k)$, i.e.

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k).$$

Let us now consider the problem of *estimating a certain state variable* $x \in \mathbb{R}^n$, with *known linear dynamic* governed by the *known inputs* $u \in \mathbb{R}^m$ and *model noise* $\nu \in \mathbb{R}^q$. The information on the variable x are provided by a set of sensors returning the measures $z \in \mathbb{R}^p$, which are affected by the *measurement noise*, i.e. *random effects*, $\varepsilon \in \mathbb{R}^p$. The *time varying, discrete-time linear* model we are dealing with is then

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) \\ z(k) &= H(k)x(k) + \varepsilon(k) \end{cases}$$

Notice that $\nu(k)$ models either a) the *imperfect* application of the actuation, b) the *noise* in the actuation commands sensors, c) *no or partial knowledge* of the system inputs.

Hence, starting from

$$\begin{cases} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) \\ z(k) &= H(k)x(k) + \varepsilon(k) \end{cases}$$

we can apply the same idea of the *Luenberger observer*, as

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that yields the modified version

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k).$$

It is evident that the terms $G(k)\nu(k) + L(k)\varepsilon(k)$ modify the picture a little bit. Indeed, the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ is now a **rv**, hence the convergence concept $\tilde{x}(k) \rightarrow 0$ *cannot be applied* as is. In these cases, a *convergence under the expected operator* (asymptotic unbiasedness) is instead needed, i.e. $E\{\tilde{x}(k)\} \rightarrow 0$. By considering

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k),$$

it then follows that convergence is attained if *necessarily* $E\{\nu(k)\} = E\{\varepsilon(k)\} = 0$, which yields to

$$E\{\tilde{x}(k+1)\} = (A(k) - L(k)H(k))E\{\tilde{x}(k)\},$$

as in the *Luenberger observer* case. Hence, we can easily recognised that the *estimated* value is the *mean value* of the **rv** $x(k)$, i.e. $\hat{x}(k) = E\{x(k)\}$. This is perfectly in agreement with the *Type A evaluation* in *metrology* and reported in the GUM [23]. We finally noticed that

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

can be rewritten in *two steps* (using $z(k+1)$ instead of $z(k)$):

$$\begin{aligned} \hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-), \end{aligned}$$

where the superscript $^-$ stands for *model-based* estimate, i.e. *prior estimate*, while the second equation is the *measurement-based* estimation, i.e. *posterior estimate*.

We define, as usual, $x^k = \{x(i)\}_{i=0,\dots,k}$ and $Z^k = \{z(i)\}_{i=1,\dots,k}$ and we are considering the prior $p(x(0))$, which is a given. Notice that both $x(k)$ and $z(k)$ are *discrete time continuous stochastic processes*. Hence, the *Bayes theorem* application reads like this:

$$p(x(k)|Z^k) = \frac{p(Z^k|x(k))p(x(k))}{p(Z^k)},$$

in which the *posterior*, the *likelihood* and the *prior* are clearly visible. We can easily recognise here that the *likelihood* can be rewritten as

$$p(Z^k|x(k)) = p(z(k)|x(k), Z^{k-1})p(Z^{k-1}),$$

and that

$$p(Z^k) = p(z(k)|Z^{k-1})p(Z^{k-1}).$$

As a consequence:

$$p(x(k)|Z^k) = \frac{p(z(k)|x(k), Z^{k-1})p(x(k))}{p(z(k)|Z^{k-1})},$$

We then notice that the *prior* is given by all the measurements up to time $k - 1$ and by the marginalisation on the previous positions, i.e.

$$p(x(k)) = p(x(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} p(x(k)|x^{k-1}, Z^{k-1})dx^{k-1}.$$

Assuming that the noise acting on the system dynamics is *white*, the system is *Markovian* and hence we can make use of the *Markovian property* and hence

$$p(x(k)|x^{k-1}, Z^{k-1}) = p(x(k)|x(k-1), Z^{k-1}),$$

thus yielding the prior

$$p(x(k)) = p(x(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} p(x(k)|x(k-1))p(x(k-1)|Z^{k-1})dx(k-1).$$

For the denominator, we consider the *Total Probability Law*, thus having

$$p(z(k)|Z^{k-1}) = \int_{-\infty}^{+\infty} p(z(k)|x(k))p(x(k)|Z^{k-1})dx(k).$$

Finally, for the likelihood function we readily recognise that if $z(k)$ is affected by a *white* stochastic process we have

$$p(z(k)|x(k), Z^{k-1}) = p(z(k)|x(k)).$$

Hence, we finally have

$$p(x(k)|Z^k) = \frac{p(z(k)|x(k))p(x(k)|Z^{k-1})}{p(z(k)|Z^{k-1})},$$

where all the components are described previously and that can be applied recursively (remember the spirit of the Bayes experiment).

We will now revisit the previous concepts using the linear model previously introduced, i.e.

$$\begin{aligned}\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-),\end{aligned}$$

where the superscript \cdot^- stands for *model-based* estimate, i.e. *prior estimate*, while the second equation is the *measurement-based* estimation, i.e. *posterior estimate*. The dynamic of the *estimation error* $\tilde{x}(k) = x(k) - \hat{x}(k)$ using only the *prior knowledge on the model*, i.e. the *prior estimation error* $\tilde{x}(k)^-$, is

$$\begin{aligned}\tilde{x}(k+1)^- &= x(k+1) - \hat{x}(k+1)^- = \\ &= A(k)x(k) + B(k)u(k) + G(k)\nu(k) - (A(k)\hat{x}(k) + B(k)u(k)) = \\ &= A(k)(x(k) - \hat{x}(k)) + G(k)\nu(k) = A(k)\tilde{x}(k) + G(k)\nu(k).\end{aligned}$$

By the previous considerations, we have

$$E\{\tilde{x}(k+1)^-\} = E\{x(k+1)\} - \hat{x}(k+1)^- = 0.$$

For the *prior covariance matrix* of the *prior estimation error* $\tilde{x}(k+1)^-$ we have then

$$\begin{aligned}P(k+1)^- &= E\{\tilde{x}(k+1)\tilde{x}(k+1)^T\} = \\ &= E\{(A(k)\tilde{x}(k) + G(k)\nu(k))(A(k)\tilde{x}(k) + G(k)\nu(k))^T\} \\ &= E\{A(k)\tilde{x}(k)\tilde{x}(k)^T A(k)^T\} + E\{G(k)\nu(k)\nu(k)^T G(k)^T\} + \\ &\quad E\{A(k)\tilde{x}(k)\nu(k)^T G(k)^T\} + E\{G(k)\nu(k)\tilde{x}(k)^T A(k)^T\} = \\ &= A(k)E\{\tilde{x}(k)\tilde{x}(k)^T\} A(k)^T + G(k)E\{\nu(k)\nu(k)^T\} G(k)^T,\end{aligned}$$

where the mixed terms

$$A(k)E\{\tilde{x}(k)\nu(k)^T\} G(k)^T \text{ and } G(k)E\{\nu(k)\tilde{x}(k)^T\} A(k)^T$$

are both zero if and only if $\nu(k)$ is generated by a *white* process (*prove it!*). Hence, for the *prior covariance matrix* of the *prior estimation error* $\tilde{x}(k+1)^-$ we finally have

$$\begin{aligned}P(k+1)^- &= A(k)E\{\tilde{x}(k)\tilde{x}(k)^T\} A(k)^T + G(k)E\{\nu(k)\nu(k)^T\} G(k)^T = \\ &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T,\end{aligned}$$

where $Q(k)$ is the *covariance matrix* of the model uncertainties.

For the measurements, we have the *innovations* given by

$$\begin{aligned} z(k+1) - \hat{z}(k+1) &= H(k+1)x(k+1) + \varepsilon(k+1) - H(k+1)\hat{x}(k+1)^- = \\ &= H(k+1)(x(k+1) - \hat{x}(k+1)^-) + \varepsilon(k+1) = \\ &= H(k+1)\tilde{x}(k+1)^- + \varepsilon(k+1) = e_z(k+1)^-. \end{aligned}$$

In the spirit of the *Luenberger observer* we have the *measurement updated* estimates as

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - \hat{z}(k+1)) = \\ &= \hat{x}(k+1) + W(k+1)e_z(k+1)^- = \\ &= \hat{x}(k+1) + W(k+1)(H(k+1)\tilde{x}(k+1)^- + \varepsilon(k+1)). \end{aligned}$$

As in the observer case, the matrix $W(k+1)$ is a *design parameter* to be determined, so it can be *optimally* determined. For instance, we can determine the optimal gain $W(k+1)$ so that the *covariance of the error* between the posterior state estimates $\hat{x}(k+1)$ and the actual state $x(k+1)$, i.e. the *posterior covariance matrix*, is minimum! To this end, the *posterior estimation error* $\tilde{x}(k+1)$, i.e. *after* the measurement updates, is by definition:

$$\begin{aligned} \tilde{x}(k+1) &= x(k+1) - \hat{x}(k+1) = x(k+1) - \hat{x}(k+1)^- + \\ &\quad - W(k+1)H(k+1)\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1) = \\ &= \tilde{x}(k+1)^- - W(k+1)H(k+1)\tilde{x}(k+1)^- + \\ &\quad - W(k+1)\varepsilon(k+1) = \\ &= (I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1). \end{aligned}$$

The *posterior covariance matrix* $P(k+1)$ of the *posterior estimation error* $\tilde{x}(k+1)$ is then given by

$$\begin{aligned} P(k+1) &= E \left\{ \tilde{x}(k+1)\tilde{x}(k+1)^T \right\} \\ &= E \left\{ ((I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1)) \cdot \right. \\ &\quad \left. ((I - W(k+1)H(k+1))\tilde{x}(k+1)^- - W(k+1)\varepsilon(k+1))^T \right\}. \end{aligned}$$

As for the *prior covariance matrix* $P(k+1)^-$, if $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ are *uncorrelated*, the previous equation simplifies a lot. To understand what is the condition that should be satisfied in order to have $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ *uncorrelated*, we noticed that:

- $\tilde{x}(k+1)^- = A(k)\tilde{x}(k) + G(k)\nu(k);$
- $\tilde{x}(k)^- = A(k-1)\tilde{x}(k-1) + G(k)\nu(k-1);$

- $\tilde{x}(k) = (I - W(k)H(k))\tilde{x}(k)^- - W(k)\varepsilon(k)$;
- The *model uncertainties* $\nu(k)$ are supposed to be generated by a *white* stochastic process.

Therefore,

$$\begin{aligned}\tilde{x}(k+1)^- &= A(k)\tilde{x}(k) + G(k)\nu(k) = \\ &= A(k)[(I - W(k)H(k))\tilde{x}(k)^- - W(k)\varepsilon(k)] + G(k)\nu(k) = \\ &= A(k)[(I - W(k)H(k))\{A(k-1)\tilde{x}(k-1) + G(k)\nu(k-1)\} + \\ &\quad - W(k)\varepsilon(k)] + G(k)\nu(k),\end{aligned}$$

i.e., it is *correlated* with $\nu(k)$, $\nu(k-1)$ and $\varepsilon(k)$. Therefore, to have $\tilde{x}(k+1)^-$ and $\varepsilon(k+1)$ *uncorrelated*, we need to have:

- $\varepsilon(k)$ generated by a *white* stochastic process;
- $E\{\varepsilon(k)\nu(k)\} = 0$.

For ease of notation, let us drop the reference to the $(k+1)$ -th time instant, i.e. $P(k+1)^- = P^-$. With the previous conditions on *uncorrelatedness*, the *posterior covariance matrix* $P(k+1)$ simplifies

$$\begin{aligned}P &= E\{((I - WH)\tilde{x}^- - W\varepsilon) ((I - WH)\tilde{x}^- - W\varepsilon)^T \} = \\ &= (I - WH)P^- (I - WH)^T + WRW^T.\end{aligned}$$

where R is the *covariance matrix* of the *measurement uncertainties* $\varepsilon(k)$. As a consequence, the *posterior covariance matrix* $P(k+1)$ is finally given by

$$\begin{aligned}P &= (I - WH)P^- (I - WH)^T + WRW^T = \\ &= P^- - WHP^- - P^- H^T W^T + WHP^- H^T W^T + WRW^T = \\ &= P^- - WHP^- - P^- H^T W^T + W(HP^- H^T + R)W^T = \\ &= P^- - WHP^- - P^- H^T W^T + WSW^T.\end{aligned}$$

Let us focus on the term $S = HP^- H^T + R$. It is related to the *measurements* $z = Hx + \varepsilon$, since R is the covariance matrix of ε . Let us consider the *innovations* $e_z(k+1)^-$: the *mean value* is

$$E\{e_z^-\} = E\{(Hx + \varepsilon - H\hat{x}^-)\} = HE\{\tilde{x}^-\} + E\{\varepsilon\} = E\{\varepsilon\} = 0.$$

Therefore, the *covariance matrix* of the innovations is

$$E\{e_z^-(e_z^-)^T\} = E\{(H\tilde{x}^- + \varepsilon)(H\tilde{x}^- + \varepsilon)^T\}.$$

Again, there is the problem of the *cross products*. However, being ε generated by a *white* stochastic process as stated previously, we have

$$\mathbb{E} \{e_z^- (e_z^-)^T\} = HP^- H^T + R,$$

which is exactly S !

Let us switch back to the *posterior covariance matrix* $P(k+1)$

$$P = P^- - WHP^- - P^-H^TW^T + WSW^T.$$

One way to minimise the *posterior covariance matrix* is to *minimise its trace*, i.e. the sum of the variances on the matrix $P(k+1)$ diagonal, i.e.

$$\begin{aligned}\frac{\partial \text{Trace}(P)}{\partial W} &= -2(HP^-)^T + 2WS = 0 \\ WS &= P^-H^T \\ W &= P^-H^TS^{-1}\end{aligned}$$

which is the *optimal Kalman gain* to minimise the *posterior covariance matrix*, i.e. *maximising the estimator precision*. Therefore, by plugging the *Kalman gain* W in the following

$$P = P^- - WHP^- - P^-H^TW^T + WSW^T,$$

we finally have

$$\begin{aligned}P &= P^- - WHP^- - P^-H^TW^T + P^-H^TS^{-1}SW^T = \\ &= P^- - WHP^- = (I - WH)P^-.\end{aligned}$$

Summary We assume a *wide sense whiteness* property on the noises, i.e.,

$$\begin{aligned}\mathbb{E} \{\nu(k)\} &= \mu_\nu \\ \mathbb{C} \{\nu(k)\} &= \mathbb{E} \{(\nu(k) - \mu_\nu)(\nu(j) - \mu_\nu)^T\} = Q(k)\delta_{kj} \\ \mathbb{E} \{\varepsilon(k)\} &= \mu_\varepsilon \\ \mathbb{C} \{\varepsilon(k)\} &= \mathbb{E} \{(\varepsilon(k) - \mu_\varepsilon)(\varepsilon(j) - \mu_\varepsilon)^T\} = R(k)\delta_{kj}\end{aligned}$$

with $\mu_\nu = \mu_\varepsilon = 0$ and noises *uncorrelatedness*

$$\mathbb{E} \{(\nu(k) - \mu_\nu)(\varepsilon(j) - \mu_\varepsilon)^T\} = 0, \quad \forall k, j.$$

Notice that *no assumption has been made on the pdf*.

The Kalman filter is split into two phases:

- *Prediction step*: it computes the evolution of the estimates based on the knowledge of the stochastic description of $\tilde{x}(k)$, i.e., $P(k)$, and the stochastic noise process $\nu(k)$. In practice this is the *prior* we want to use;
- *Update step*: it computes the updated values of the estimates based on the *likelihood function* of the *measurements*.

Hence, the *Kalman filter* is a linear *Bayesian estimator*, which combines the prior with the likelihood function. Hence, the filter is the *best linear filter* for noises that are generically distributed. The two equations of the *Kalman Filter* are:

- *Prediction step* (based on the model, i.e., the *prior*):

$$\begin{aligned}\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\ P(k+1)^- &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T\end{aligned}$$

- *Update step* (based on the measurements, i.e., the *posterior*):

$$\begin{aligned}S(k+1) &= H(k+1)P(k+1)^-H(k+1)^T + R(k+1) \\ W(k+1) &= P(k+1)^-H(k+1)^TS(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^-\end{aligned}$$

Notice that the *Update step* comprises the WLS solution for the *iterative solution*.

Remark 22. The relation $P = (I - WH)P^- (I - WH)^T + WRW^T$ retrieved previously in the derivation of W is called *Joseph form of the covariance update* and can be used in place of $P = (I - WH)P^-$ because more numerically stable.

What happens when the noises are Gaussian?

- If the *process noise* $\nu(k) \sim \mathcal{N}(0, Q(k))$, the state $x(k) \sim \mathcal{N}(\hat{x}(k), C\{x(k)\})$;
- If the *measurement noise* $\varepsilon(k) \sim \mathcal{N}(0, R(k))$, then $z(k) = H(k)x(k) + \varepsilon(k)$ is $z(k) \sim \mathcal{N}(\mu_z, C\{z(k)\})$.

Recall that $C\{x(k)\} = E\{(x(k) - \hat{x}(k))(x(k) - \hat{x}(k))^T\}$ by definition. Hence, $C\{x(k)\} = E\{\tilde{x}(k)\tilde{x}(k)^T\} = P(k)$. Recall that if $\mu_\varepsilon \neq 0$, this bias can be removed by sensor calibration. Let us see how μ_x , $C\{x(k)\}$, μ_z and $C\{z(k)\}$ are given by the *Kalman filter*.

- $\mu_x = \hat{x}(k)$;
- $C\{x(k)\} = A(k-1)P(k-1)A(k-1)^T + G(k-1)Q(k-1)G(k-1)^T$;
- $\mu_z = E\{H(k)x(k) + \varepsilon(k)\} = H(k)\hat{x}(k)$;
- $C\{z(k)\} = E\{(z - \mu_z)(z - \mu_z)^T\}$, hence:

$$C\{z(k)\} = E\{(H(k)\tilde{x}(k) + \varepsilon(k))(H(k)\tilde{x}(k) + \varepsilon(k))^T\},$$

Notice that $\tilde{x}(k)$ depends on the noise $\nu(k-1)$. If $\nu(k)$ and $\varepsilon(j)$ are *uncorrelated* $\forall k, j$, it follows

$$C\{z(k)\} = H(k)P(k)H(k)^T + R(k),$$

that is the *covariance of the residuals* $S(k)$ found previously;

- $C\{x, z\} = E\{(x - \mu_x)(z - \mu_z)^T\}$, that is:

$$C\{x, z\} = E\{\tilde{x}(k)(H(k)\tilde{x}(k) + \varepsilon(k))^T\}$$

If $\nu(k)$ and $\varepsilon(j)$ are *uncorrelated* $\forall k, j$, it follows

$$C\{x, z\} = E\{\tilde{x}(k)\tilde{x}(k)\} H(k)^T = P(k)H(k)^T.$$

It then follows that x and z are *jointly Gaussian*! Let us see what are the implications of being *jointly Gaussian*.

If x and z are jointly Gaussian, the optimal estimator is the MMSE, whose solution equation are given by:

$$E\{x|z\} = \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z) \text{ and } C\{x|z\} = P_{xx} - P_{xz}P_{zz}^{-1}P_{zx}.$$

Hence, if the noises are Gaussian, zero-mean and white, i.e. $x(k) \sim \mathcal{N}(\hat{x}(k), C\{x(k)\})$ and $z(k) \sim \mathcal{N}(\mu_z, C\{z(k)\})$, is the Kalman filter the *optimal linear MMSE estimator*?

To answer this question, let us see how those quantities are *computed and combined* in the Kalman filter. At time k , we have:

- $\mu_x = \hat{x}(k)$;
- $P_{xx} = A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T = P(k)$;
- $\mu_z = E\{H(k)x(k) + \varepsilon(k)\} = H(k)\hat{x}(k)$;
- $P_{zz} = E\{(z - \mu_z)(z - \mu_z)^T\} = H(k)P(k)H(k)^T + R(k) = S(k)$;

- $P_{xz} = P_{zx}^T = \mathbb{E} \{(x - \mu_x)(z - \mu_z)^T\} = \mathbb{E} \{\tilde{x}(k)\tilde{x}(k)\} H(k)^T = P(k)H(k)^T$.

By plugging the previous equations into the MMSE solution we finally have:

$$\begin{aligned}\mathbb{E} \{x|z\} &= \mu_x + P_{xz}P_{zz}^{-1}(z - \mu_z) = \\ &= \hat{x}(k) + P(k)H(k)^T S(k)^{-1}(z - H(k)\hat{x}(k)) = \\ &= \hat{x}(k) + W(k)(z - H(k)\hat{x}(k)), \\ \mathbb{C} \{x|z\} &= P_{xx} - P_{xz}P_{zz}^{-1}P_{zx} = \\ &= P(k) - P(k)H(k)^T S(k)^{-1}H(k)P(k) = \\ &= P(k) - W(k)H(k)P(k) = (I - W(k)H(k))P(k),\end{aligned}$$

i.e. *exactly the equation of the Kalman filter*. Hence, the *Kalman filter is an MMSE estimator* and, hence, *the Kalman filter is the optimal filter for Gaussian noises*, while it is the *best linear filter in all the other cases*. This fact is not surprising at all since the Kalman filter is split into two phases:

- *Prediction step*: it computes the evolution of the estimates based on the knowledge of the stochastic description of $\hat{x}(k)$, i.e., $P(k)$, and the stochastic noise process $\nu(k)$. In practice this is the *prior* we want to use;
- *Update step*: it computes the updated values of the estimates based on the *likelihood function*, which is the same function maximised in the LS for Gaussian noises.

Hence, the *Kalman filter* is a linear MMSE *Bayesian estimator*, which combines the prior with the likelihood function.

Let us switch back to the Bayesian formulation, i.e.

$$p(x(k+1)|Z^{k+1}) = \frac{p(z(k+1)|x(k+1))p(x(k+1)|Z^k)}{p(z(k+1)|Z^k)}.$$

In case of a linear system and of Gaussian, zero-mean white noises, we have immediately that

$$\begin{aligned}p(x(k+1)|x(k)) &= \mathcal{N}(A(k)x(k) + B(k)u(k), G(k)Q(k)G(k)^T), \\ p(z(k+1)|x(k+1)) &= \mathcal{N}(H(k+1)x(k+1), R(k+1)), \\ p(x(k+1)|Z^{k+1}) &= \mathcal{N}(\hat{x}(k+1), P(k+1)).\end{aligned}$$

Remark 23. *The Kalman filter presented in these slides is just the discrete-discrete Kalman filter, in which both the model and the measurement process are assumed to be discrete-time.*

This is not the only choice available. Indeed there exists also other versions, such as the continuous-continuous or the continuous-discrete Kalman filters, that can be adopted depending on the particular problem.

Remark 24. We restrict ourselves to the discrete-discrete Kalman filters because are the most relevant for applications, especially when the information on the models and on the measures are known only at discrete time intervals. This is always the case when we have to deals with distributed systems.

Alternative derivation using the WLS

Recall the *time varying, discrete-time linear* model we are dealing with is

$$\begin{cases} x(k+1) = A(k)x(k) + B(k)u(k) + G(k)\nu(k) \\ z(k) = C(k)x(k) + \varepsilon(k) \end{cases}$$

We first notice that the discrete time system dynamic

$$x(k+1) = A(k)x(k) + B(k)u(k) + G(k)\nu(k)$$

driven by white noise is a *Markovian stochastic process*. We also notice that if $\nu(k)$ is distributed with a Gaussian, this is a *Gauss Markov* process. Since we have to deal with a dynamic system, a *dynamic estimator* is needed.

A possible estimate of x can be given by the *recursive LS solution*, which gives us the opportunity to efficiently combine a sequence of observations in order to minimise the squares of the residuals, whose equations are recalled next:

$$\begin{aligned} S(k+1) &= H(k+1)P(k)H(k+1)^T + C\{\varepsilon(k+1)\} \\ W(k+1) &= P(k)H(k+1)^TS(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k)) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k) \end{aligned}$$

However, this solution does not take into account the knowledge of the system. Moreover, the system is a *Markovian stochastic process*, so the variable x is random and, hence, a *Bayesian* approach is needed. In other words, we missed the prior of the *Bayesian estimator* and we are only using the *likelihood*. For a linear estimator, the $x(k)$ prior is defined by the *mean* and the *covariance matrix*. Hence, the $x(k)$ prior is given by the discrete time dynamic propagation of those quantities.

To this end, let us first compute the time evolution of the *mean* of x , i.e., the mean of the *state* of the system

$$\begin{aligned} E\{x(k+1)\} &= E\{A(k)x(k) + B(k)u(k) + G(k)\nu(k)\} = \\ &= A(k)E\{x(k)\} + B(k)u(k) + G(k)E\{\nu(k)\} = \\ &= A(k)E\{x(k)\} + B(k)u(k) + G(k)\mu_\nu \end{aligned}$$

We know that to be *unbiased*, the estimator must have $\hat{x}(k) \triangleq \mathbb{E}\{x(k)\}$. Notice that this equation represents the *discrete-time evolution of the estimates that can be computed by using $\hat{x}(k)$ and the knowledge of the input $u(k)$ and of the disturbance mean ν* . We further notice that since $u(k)$ and μ_ν are known (we can assume that μ_ν is an *observed* constant value that sums to the inputs), the estimates, i.e. the *mean values*, propagates as

$$\hat{x}(k+1) = \mathbb{E}\{x(k+1)\} = A(k)\hat{x}(k) + B(k)u(k)$$

In other words, the knowledge of the mean of the noise μ_ν allows us to consider the noise as *zero mean*.

Since

$$x(k+1) = A(k)x(k) + B(k)u(k) + G(k)\nu(k)$$

and

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k),$$

the *estimation error* $\tilde{x} = x(k+1) - \hat{x}(k+1)$ has zero mean and *covariance matrix*:

$$\begin{aligned} P(k+1) &= \mathbb{C}\{\tilde{x}(k+1)\} = \mathbb{E}\{\tilde{x}(k+1)\tilde{x}(k+1)^T\} = \\ &= \mathbb{E}\{(x(k+1) - \hat{x}(k+1))(x(k+1) - \hat{x}(k+1))^T\} = \\ &= \mathbb{E}\{(A(k)\tilde{x}(k) + G(k)\nu(k))(A(k)\tilde{x}(k) + G(k)\nu(k))^T\} \end{aligned}$$

We have to notice that

$$\tilde{x}(k) = A(k-1)\tilde{x}(k-1) + G(k-1)\nu(k-1),$$

due to the Markovian property. Due to the *whiteness property*, $\nu(k)$ is uncorrelated with $\nu(k-1)$. Hence, $\tilde{x}(k)$, which is a linear function of $\nu(k-1)$, is *uncorrelated* with $\nu(k)$. As a consequence

$$\mathbb{E}\{A(k)\tilde{x}(k)\nu(k)^T G(k)^T\} = \mathbb{E}\{G(k)\nu(k)\tilde{x}(k)^T A(k)^T\} = 0$$

It then follows that:

$$\begin{aligned} P(k+1) &= \\ &= \mathbb{E}\{(A(k)\tilde{x}(k) + G(k)\nu(k))(A(k)\tilde{x}(k) + G(k)\nu(k))^T\} = \\ &= \mathbb{E}\{(A(k)\tilde{x}(k)\tilde{x}(k)^T A(k)^T\} + \mathbb{E}\{G(k)\nu(k)\nu(k)^T G(k)^T\} = \\ &= A(k)\mathbb{E}\{\tilde{x}(k)\tilde{x}(k)^T\} A(k)^T + G(k)\mathbb{E}\{\nu(k)\nu(k)^T\} G(k)^T = \\ &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T \end{aligned}$$

To summarise, the *Kalman filter* propagates the *mean* of x and the *covariance* of the estimation error P . As a new measure is available at time $k + 1$, we will have:

$$\begin{aligned}\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\ P(k+1)^- &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T \\ S(k+1) &= H(k+1)P(k+1)^-H(k+1)^T + R(k+1) \\ W(k+1) &= P(k+1)^-H(k+1)^TS(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^-\end{aligned}$$

where the superscript $(\cdot)^-$ stands for the *estimates given the model*, to disambiguate with $\hat{x}(k+1)$ and $P(k+1)$, which are the *estimates*.

Alternative derivation using optimal estimation approach

We start from the idea of the *Luenberger observer* as

$$\hat{x}(k+1) = A(k)\hat{x}(k) + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)),$$

that yields the modified version of the *estimation error*

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k).$$

We still assume a *wide sense whiteness* property on the noises, i.e.,

$$\begin{aligned}\mathbb{E}\{\nu(k)\} &= \mu_\nu \\ \mathbb{C}\{\nu(k)\} &= \mathbb{E}\{(\nu(k) - \mu_\nu)(\nu(j) - \mu_\nu)^T\} = Q(k)\delta_{kj} \\ \mathbb{E}\{\varepsilon(k)\} &= \mu_\varepsilon \\ \mathbb{C}\{\varepsilon(k)\} &= \mathbb{E}\{(\varepsilon(k) - \mu_\varepsilon)(\varepsilon(j) - \mu_\varepsilon)^T\} = R(k)\delta_{kj}\end{aligned}$$

with $\mu_\nu = \mu_\varepsilon = 0$ and noises *uncorrelatedness*

$$\mathbb{E}\{(\nu(k) - \mu_\nu)(\varepsilon(j) - \mu_\varepsilon)^T\} = 0, \quad \forall k, j.$$

Notice that *no assumption has been made on the pdf*. To find the *optimal gain* of $L(k)$, we solve the following optimal problem

$$L(k) = \arg \min_{L(k)} a^T P(k+1) a, \quad \forall a \in \mathbb{R}^n.$$

Given the *estimation error* description derived from the *observer* description

$$\tilde{x}(k+1) = (A(k) - L(k)H(k))\tilde{x}(k) + G(k)\nu(k) + L(k)\varepsilon(k),$$

we have immediately

$$\mathbb{E}\{\tilde{x}(k+1)\} = (A(k) - L(k)H(k))\mathbb{E}\{\tilde{x}(k)\}.$$

Hence:

$$\begin{aligned} P(k+1) &= \mathbb{E}\{\tilde{x}(k+1)\tilde{x}(k+1)^T\} = \\ &= (A(k) - L(k)H(k))P(k)(A(k) - L(k)H(k))^T + \\ &\quad G(k)Q(k)G(k)^T + L(k)R(k)L(k)^T. \end{aligned}$$

Therefore, by computing the derivative with respect to $L(k)$ and imposing it to be equal to zero, we have:

$$a^T(-2A(k)P(k)H(k)^T + 2L(k)(H(k)P(k)H(k)^T + R(k)))a = 0.$$

Since the previous equation should be verified $\forall a$, we have

$$L(k) = A(k)P(k)H(k)^T(H(k)P(k)H(k)^T + R(k))^{-1}$$

The *estimation error covariance* relation becomes

$$\begin{aligned} P(k+1) &= \mathbb{E}\{\tilde{x}(k+1)\tilde{x}(k+1)^T\} = \\ &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T + \\ &\quad - A(k)P(k)H(k)^T(H(k)P(k)H(k)^T + R(k))^{-1}H(k)P(k)A(k)^T, \end{aligned}$$

which is the *Riccati difference equation*. If the system is *time invariant* (i.e. $A(k) = A$, etc.), the system is *controllable* and *observable*, and the noise stochastic processes are *wide sense stationary*, we have that $P(k) \rightarrow P$, thus yielding the *Discrete Algebraic Riccati Equation* (DARE)

$$\begin{aligned} P &= APA^T + GQG^T + \\ &\quad - AP(HPH^T + R)^{-1}HPA^T, \end{aligned}$$

and $L = AP(HPH^T + R)^{-1}$ is the *steady state solution*.

Remark 25. The *optimal gain* and the *Kalman gain* are related by $L(k) = A(k)W(k)$.

The two formulations are identical. Indeed, with $L(k)$ we can write the *Kalman filter* in its *a priori* form:

$$\begin{aligned} L(k) &= A(k)P(k)H(k)^T(H(k)P(k)H(k)^T + R(k))^{-1} \\ \hat{x}(k+1)^- &= A(k)\hat{x}(k)^- + B(k)u(k) + L(k)(z(k) - H(k)\hat{x}(k)^-) \\ P(k+1)^- &= A(k)P(k)^-(A(k) - L(k)H(k))^T + G(k)Q(k)G(k)^T \end{aligned}$$

which is equivalent to the *standard form* of the *Kalman Filter*.

13.2 Nonlinear estimators

13.2.1 The Non-linear Least Squares solution

Usually the measurement process is *nonlinear*, i.e.

$$z(k) = h(x) + \varepsilon(k),$$

where $h(x)$ is a generic nonlinear function of the states. The WLS problem thus becomes

$$\hat{x}^{WLS}(k) = \arg \min_x (Z^k - h(x))^T W (Z^k - h(x)) = \arg \min_x J(x, k)$$

where $W = W^T$ and W positive definite (p.d.). In such a case it becomes very hard to find the solution to the previous problem, since we have to find

$$\frac{dJ(x, k)}{dx} = 0$$

where $J(x, k)$ is a *generic nonlinear vectorial function*, i.e. the *gradient*. We recall the derivative of nonlinear functions of x , i.e., given $s(x)$ and $r(x)$ be two *generic nonlinear vectorial functions* of the vector x and A a matrix of suitable dimension, we have

$$\frac{ds(x)^T Ar(x)}{dx} = \left(\frac{ds(x)}{dx} \right)^T Ar(x) + \left(\frac{dr(x)}{dx} \right)^T A^T s(x);$$

We then compute the explicit form of the *gradient* $J(x, k)$ by expanding the products as

$$J(x, k) = Z^k^T W Z^k - h(x)^T W Z^k - Z^k^T W h(x) + h(x)^T W h(x)$$

It then follows that we are searching for the solution to

$$\frac{dJ(x, k)}{dx} = -H(x)^T W (Z^k - h(x)) = 0,$$

where $H(x) = \frac{dh(x)}{dx}$ is the *Jacobian* of the output forward map $h(x)$. To this end, let us first recall that given a generic nonlinear scalar function $g(x)$, without *local minima* and *differentiable*, the problem of finding a solution \bar{x} such that

$$g(\bar{x}) = 0,$$

can be approached with the *Newton-Raphson method*.

1. Select a generic point x_0 ;
2. Compute the first order *Taylor expansion* around x_0

$$0 = g(\bar{x}) \approx g(x_0) + \frac{dg(x)}{dx} \Big|_{x=x_0} (\bar{x} - x_0) = g(x_0) + g'(x_0)(\bar{x} - x_0);$$

3. Hence we have

$$\bar{x} = x_0 - \frac{g(x_0)}{g'(x_0)}.$$

Of course, if the function $g(x)$ is linear then the first order *Taylor expansion* holds with an equality, hence \bar{x} is the desired solution. If the function $g(x)$ is nonlinear, we have to resort to an *iterative solution*. Hence:

1. Select a generic point x_q , with $q = 0$ (as before);
2. Compute the first order *Taylor expansion* around x_q (as before);
3. Compute the solution \bar{x} , which now is renamed x_{q+1} , i.e.,

$$x_{q+1} = x_q - \frac{g(x_q)}{g'(x_q)} = x_q + \Delta x_q;$$

4. If $|x_{q+1} - x_q| = |\Delta x_q|$ is *below a certain threshold*, the algorithm stops and returns x_{q+1} as the point in which $g(x_{q+1}) \approx 0$. Otherwise, switch back to Step 2.

This is an *estimator*, since $x_{q+1} = \hat{x}$, i.e. the point in which $g(\bar{x}) = 0$, but it is a *numerical algorithm*. Using this idea, we can apply the following algorithm for the nonlinear WLS. Let's start with the *gradient* function

$$g(x) = \frac{dJ(x, k)}{dx} = -H(x)^T W(Z^k - h(x)) = 0,$$

and let us apply the *Newton-Raphson method*, as follows:

1. Select a generic state x_q ;
2. Compute the first order *Taylor expansion* around x_q :

$$g(x) \approx g(x_q) + \frac{dg(x)}{dx} \Big|_{x=x_q} (x_{q+1} - x_q) = g(x_q) + G(x_q)\Delta x_q = 0,$$

where $G(x_q)$ is the *Hessian matrix* of $J(x, k)$ evaluated in x_q ;

3. Solve with respect to $\Delta x_q = x_{q+1} - x_q$, i.e.,

$$\Delta x_q = -G(x_q)^{-1}g(x_q) \Rightarrow x_{q+1} = x_q - G(x_q)^{-1}g(x_q).$$

4. If $|x_{q+1} - x_q| = |\Delta x_q|$ is *below a certain threshold*, the algorithm stops and returns x_{q+1} as the point in which $g(x_{q+1}) \approx 0$. Otherwise, switch back to Step 2.

In general, the *Newton-Raphson method* can be described as:

$$x_{q+1} = x_q - \mathcal{H}(J(x, k))^{-1}\nabla J(x, k),$$

where $\mathcal{H}(J(x, k))$ and $\nabla J(x, k)$ are the *Hessian* and the *gradient* of $J(x, k)$. Since we are dealing with the *Nonlinear WLS*, recall that the *gradient* is

$$g(x) = \frac{dJ(x, k)}{dx} = -H(x)^T W(Z^k - h(x)),$$

Hence the *Hessian* evaluated in x_q is

$$G(x_q) = \left. \frac{dg(x)}{dx} \right|_{x=x_q} = H(x_q)^T W H(x_q)$$

and the *gradient*

$$g(x_q) = -H(x_q)^T W(Z^k - h(x_q))$$

We can substitute the *explicit* values of the *gradient* and the *Hessian* at step q of the *Newton-Raphson method*, yielding

$$\begin{aligned} x_{q+1} &= x_q + \Delta x_q = x_q - G(x_q)^{-1}g(x_q) = \\ &= x_q + (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W (Z^k - h(x_q)) \end{aligned}$$

where $\Delta y_q = Z^k - h(x_q)$ are the *linearised residuals* (more on this in what follows).

Again, if $|x_{q+1} - x_q| = |\Delta x_q|$ is below a certain threshold, the algorithm stops and returns x_{q+1} as x . Otherwise, switch back to Step 2. This solution is called the *Gauss-Newton method*.

Notice that since the *Hessian* is a function of the *Jacobian* of the nonlinear forward map $h(x)$ for the *Nonlinear WLS*, it turns out that *the Hessian is not computed de facto*. Notice that for the *Gauss-Newton method*

$$\Delta x_q = x_{q+1} - x_q = (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W (Z^k - h(x_q)),$$

is the WLS solution for a *linearised problem*. Therefore, the *Gauss-Newton method* can also be interpreted like this: given x_q , compute the first order *Taylor expansion* around x_q of the output function:

$$h(x_{q+1}) \approx h(x_q) + \frac{dh(x)}{dx} \Big|_{x=x_q} (x_{q+1} - x_q) = h(x_q) + H(x_q)\Delta x_q,$$

where $H(x_q)$ is the *Jacobian matrix* of the output function $h(x)$ evaluated in x_q , as reported previously.

Define the following *linear* WLS

$$\begin{aligned} \Delta x_q^{WLS} &= \arg \min_{\Delta x_q} (Z^k - h(x_{q+1}))W(Z^k - h(x_{q+1}))^T \approx \\ &\approx \arg \min_{\Delta x_q} (Z^k - h(x_q) - H(x_q)\Delta x_q)W(Z^k - h(x_q) - H(x_q)\Delta x_q)^T = \\ &= \arg \min_x (\Delta y_q - H(x_q)\Delta x_q)W(\Delta y_q - H(x_q)\Delta x_q)^T, \end{aligned}$$

where we have substituted the *first order Taylor linearised* function $h(x_{q+1})$ and $\Delta y_q = Z^k - h(x_q)$ are, again and more clearly, the *linearised residuals*. The solution is then given as customary

$$\Delta x_q = (H(x_q)^T W H(x_q))^{-1} H(x_q)^T W \Delta y_q,$$

which finally generates $x_{q+1} = x_q + \Delta x_q$.

Remark 26. In many practical problems, the *Hessian matrix* $G(x_q) = H(x_q)^T W H(x_q)$ is quite *sparse* (i.e., it has a lot of zeros) and it is huge. However, its inverse $G(x_q)^{-1}$ will not be sparse.

Remark 27. In general, the *Hessian matrix* $G(x_q)$ is *not inverted*. Instead, the following *normal equation* is considered:

$$G(x_q)\Delta x_q = H(x_q)^T W \Delta y_q.$$

To solve the previous equations, the matrix $G(x_q)$ is factorised in its triangular components (*Cholesky decomposition*) and then the forward/backward substitution algorithm is used.

As a final comment, another approach that actually involves just the computation of the *gradient* of the function and follows exactly *the same iterative procedure* is the *steepest descent method*. In general, the *steepest descent method* can be described as:

$$x_{q+1} = x_q - \alpha \nabla J(x, k),$$

where $\nabla J(x, k)$ is again the *gradient* of $J(x, k)$. The parameter $\alpha > 0$ is a constant which controls the *convergence rate* and *stability*: the larger is α , the faster is the convergence at the price of a reduced stability. The main idea underlying this simple approach is to *move* the solution following the *gradient* direction.

13.2.2 The Extended Kalman filter

Let us consider the following *time varying nonlinear discrete time* dynamic system:

$$\begin{cases} x(k+1) = f_k(x(k), u(k), \nu(k)) \\ z(k) = h_k(x(k), \varepsilon(k)) \end{cases}$$

We have assumed a *wide sense whiteness* property on the noises, i.e.,

$$\begin{aligned} E\{\nu(k)\} &= \mu_\nu \\ C\{\nu(k)\} &= E\{(\nu(k) - \mu_\nu)(\nu(j) - \mu_\nu)^T\} = Q(k)\delta_{kj} \\ E\{\varepsilon(k)\} &= \mu_\varepsilon \\ C\{\varepsilon(k)\} &= E\{(\varepsilon(k) - \mu_\varepsilon)(\varepsilon(j) - \mu_\varepsilon)^T\} = R(k)\delta_{kj} \end{aligned}$$

Moreover, it is usually assumed that

$$E\{(\nu(k) - \mu_\nu)(\varepsilon(j) - \mu_\varepsilon)^T\} = 0, \quad \forall k, j,$$

i.e., *uncorrelated noises*. Since the system is nonlinear, the *Kalman filter* (KF) cannot be applied as is. To *extend* the application of the KF to this class of systems, we first notice that the process prediction and the measurement equations *can be used as they are* in the formulation of the KF. What changes is the way in which the *covariances* are propagated and the way in which the *Kalman gain* $W(k)$ is computed, which indeed subsumes a *linear process*.

We recall here that in order to compute an approximated propagation of covariances for a *nonlinear combination* of rvs, we can make use of the *law of propagation of errors*, aka *law of propagation of the uncertainties*. In that case, a *Taylor linearised* model can be used. Therefore, the *Extended Kalman Filter* (EKF) is just a filter that makes use of linearised models for covariances propagation and gain computation. The linearisation involves the knowledge of the actual estimates $\hat{x}(k)$ at time k .

For the *Prediction step* (based on the model, i.e., in *open loop*), we have:

$$\begin{aligned} \hat{x}(k+1)^- &= f_k(\hat{x}(k), u(k)) \\ P(k+1)^- &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T \end{aligned}$$

where the *linearised model* for the covariances is given by

$$A(k) = \left. \frac{f_k(x, u, \nu)}{dx} \right|_{\begin{array}{l} x = \hat{x}(k) \\ u = u(k) \\ \nu = 0 \end{array}}$$

$$G(k) = \left. \frac{f_k(x, u, \nu)}{d\nu} \right|_{\begin{array}{l} x = \hat{x}(k) \\ u = u(k) \\ \nu = 0 \end{array}}$$

Similarly, for the *Update step* (based on the measurements, i.e., in *closed loop*), we have:

$$\begin{aligned} S(k+1) &= H(k+1)P(k+1)^{-}H(k+1)^T + R(k+1) \\ W(k+1) &= P(k+1)^{-}H(k+1)^TS(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k+1)^{-} + W(k+1)(z(k+1) - h_{k+1}(\hat{x}(k+1)^{-})) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^{-} \end{aligned}$$

where the *linearised model* for the covariances and the gain is given by

$$H(k+1) = \left. \frac{h_{k+1}(x, \varepsilon)}{dx} \right|_{\begin{array}{l} x = \hat{x}(k+1)^{-} \\ \varepsilon = 0 \end{array}}$$

Remark 28. There are other solutions provided in the literature for the KF, which can be efficiently used in particular cases. For example, the **Unscented Kalman Filter**, that makes use of the **unscented transformation**, is a possible choice in the nonlinear case. The **unscented transformation** is explicitly conceived for the nonlinear propagation of the uncertainties.

Remark 29. Other approaches, instead, consider the model as a **blended combination of elementary models**, among which the process may switch. These are called **Interactive Multiple Models**.

13.2.3 Unscented Kalman Filter

The EKF relies on linearisation to propagate the mean and covariance of the state, which may lead to *overoptimistic estimates*, or even *inconsistency*, when the nonlinearities are severe. Another approach in this cases is the **Unscented Kalman Filter** (UKF), an extension of the Kalman filter that reduces the linearisation errors of the EKF. The use of the UKF can provide significant improvement over the EKF.

Consider a LIDAR acquiring data in *polar coordinates* $p = [r, \theta]^T$, i.e. r being the range and θ the angle. Assume that both the measurement quantities are affected by the noises η_r and η_θ that are generated by two *wide-sense stationary* and *independent processes* with symmetric pdfs and such that $\mu_{\eta_r} = \mu_{\eta_\theta} = 0$ and variances $\sigma_{\eta_r}^2$ and $\sigma_{\eta_\theta}^2$, i.e. $p_m = [r_m, \theta_m]^T = [r + \eta_r, \theta + \eta_\theta]^T$. We want to express those quantities in term of the vector of Cartesian coordinates $z = [x, y]^T$, therefore

$$z = \begin{bmatrix} r_m \cos(\theta_m) \\ r_m \sin(\theta_m) \end{bmatrix} = h(p_m)$$

Let us compute a first order approximation:

$$\mu_z = E\{h(p_m)\} \approx E\left\{h(E\{p_m\}) + \frac{\partial h(p_m)}{\partial p_m}\Big|_{E\{p_m\}} (p_m - E\{p_m\})\right\}$$

Since $E\{p_m\} = p$, we have

$$\mu_z \approx h(p) + \frac{\partial h(p_m)}{\partial p_m}\Big|_p E\{p_m - p\} = h(p)$$

Suppose $p = [1, \pi/2]^T$, we have $\mu_z = [0, 1]$. However, if we consider

$$E\{\mu_x\} = E\{r_m \cos \theta_m\} = E\{(r + \eta_r)(\cos \theta \cos \eta_\theta - \sin \theta \sin \eta_\theta)\}$$

that assuming independence of the noises and symmetry of the pdfs, we have

$$E\{\mu_x\} = r \cos \theta E\{\cos \eta_\theta\} = 0$$

If we consider

$$E\{\mu_y\} = E\{r_m \sin \theta_m\} = E\{(r + \eta_r)(\sin \theta \cos \eta_\theta - \cos \theta \sin \eta_\theta)\}$$

that assuming independence of the noises and symmetry of the pdfs, we have

$$E\{\mu_y\} = r \sin \theta E\{\cos \eta_\theta\} = E\{\cos \eta_\theta\}$$

Let us assume that η_θ is *uniformly* distributed (if no assumption is made, *it is not possible go any further*), with range in $\pm \eta_\theta^*$. Therefore

$$E\{\mu_y\} = E\{\cos \eta_\theta\} = \int_{-\eta_\theta^*}^{\eta_\theta^*} \cos \eta_\theta \frac{1}{2\eta_\theta^*} d\eta_\theta = \frac{\sin \eta_\theta^*}{\eta_\theta^*}$$

Notice that, instead of 1 we have a number that is equal to 1 *only if* $\eta_\theta^* = 0$, while a value that is < 1 , $\forall \eta_\theta^* > 0$. Hence, the *first order approximation* $\mu_z = [0, 1]$ is *incorrect!*

Indeed, by expanding up to the *second order* we have:

$$\begin{aligned}\mu_z = E\{h(p_m)\} &\approx E\left\{h(E\{p_m\}) + \frac{\partial h(p_m)}{\partial p_m}\Big|_{E\{p_m\}} (p_m - E\{p_m\})\right\} + \\ &+ \frac{1}{2}E\left\{\frac{\partial^2 h(p_m)}{\partial r_m^2}\Big|_{E\{p_m\}} (r_m - E\{r_m\})^2\right\} + \\ &+ \frac{1}{2}E\left\{\frac{\partial^2 h(p_m)}{\partial \theta_m^2}\Big|_{E\{p_m\}} (\theta_m - E\{\theta_m\})^2\right\} + \\ &+ E\left\{\frac{\partial^2 h(p_m)}{\partial r_m \partial \theta_m}\Big|_{E\{p_m\}} (r_m - E\{r_m\})(\theta_m - E\{\theta_m\})\right\}\end{aligned}$$

We notice that:

$$\frac{1}{2}E\left\{\frac{\partial^2 h(p_m)}{\partial r_m^2}\Big|_{E\{p_m\}} (r_m - E\{r_m\})^2\right\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Then:

$$E\left\{\frac{\partial^2 h(p_m)}{\partial r_m \partial \theta_m}\Big|_{E\{p_m\}} (r_m - E\{r_m\})(\theta_m - E\{\theta_m\})\right\} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

since the two **rvs** are *uncorrelated*. Finally:

$$\frac{1}{2}E\left\{\frac{\partial^2 h(p_m)}{\partial \theta_m^2}\Big|_{E\{p_m\}} (\theta_m - E\{\theta_m\})^2\right\} = \frac{1}{2}\sigma_{\eta_\theta}^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

From this result, it follows that

$$\mu_z = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \frac{1}{2}\sigma_{\eta_\theta}^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Hence, while $E\{\mu_x\} = 0$, we have

$$E\{\mu_y\} = E\{\cos \eta_\theta\} \approx 1 - \frac{\sigma_{\eta_\theta}^2}{2}$$

where the approximation is *up to the second order*. This is clearly visible in the example of Figure 13.1.

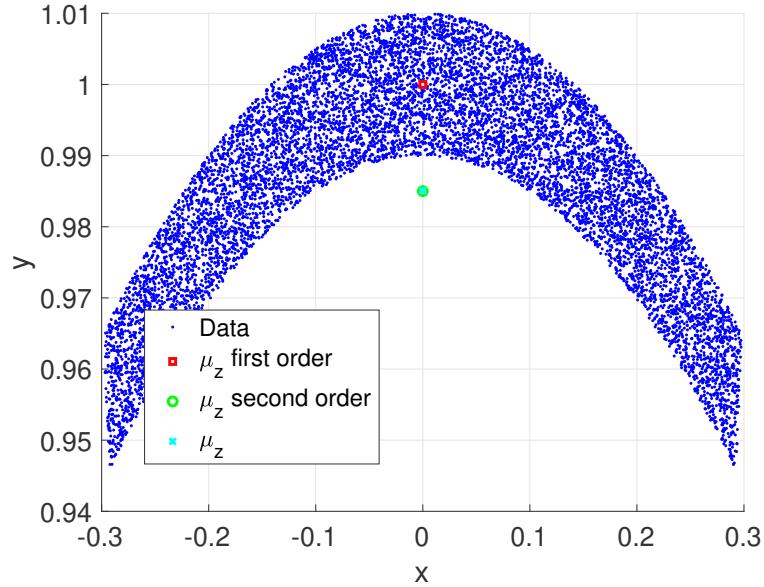


Figure 13.1: Approximation of the means for the previous example (with $\eta_r \sim \mathcal{U}(-0.01, 0.01)$ metres and $\eta_\theta \sim \mathcal{U}(-0.4, 0.4)$ radians) for the *first order* Taylor, *second order* Taylor and the *analytic* solution using a Monte Carlo approach (i.e. *Type B evaluation*).

Let us try to compute the covariance of z , i.e.

$$\mathbf{C}\{z\} = \mathbf{E}\{(z - \mu_z)(z - \mu_z)^T\}$$

If we resort to the *first order approximation*, we have:

$$\mathbf{C}\{z\} \approx H \mathbf{C}\{p_m\} H^T$$

where

$$H = \left. \frac{\partial h(p_m)}{\partial p_m} \right|_{\mathbf{E}\{p_m\}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Moreover

$$\begin{aligned} \mathbf{C}\{p_m\} &= \mathbf{E}\{(p_m - \mathbf{E}\{p_m\})(p_m - \mathbf{E}\{p_m\})^T\} = \\ &= \mathbf{E}\left\{\begin{bmatrix} \eta_r \\ \eta_\theta \end{bmatrix} \begin{bmatrix} \eta_r & \eta_\theta \end{bmatrix}\right\} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \end{aligned}$$

Therefore

$$\mathbf{C}\{z\} \approx H \mathbf{C}\{p_m\} H^T = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

which is, of course, approximated.

Instead, if we make use of the approximation *up to the second order*, we have

$$\begin{aligned} C\{z\} &= E\{(z - \mu_z)(z - \mu_z)^T\} = \\ &= E\left\{\begin{bmatrix} r_m \cos \theta_m \\ r_m \sin \theta_m - \left(1 - \frac{\sigma_\theta^2}{2}\right) \end{bmatrix} \begin{bmatrix} r_m \cos \theta_m & r_m \sin \theta_m - \left(1 - \frac{\sigma_\theta^2}{2}\right) \end{bmatrix}\right\} = \\ &= E\left\{\begin{bmatrix} r_m^2 \cos^2 \theta_m & r_m^2 \cos \theta_m \sin \theta_m - r_m \cos \theta_m \left(1 - \frac{\sigma_\theta^2}{2}\right) \\ r_m^2 \cos \theta_m \sin \theta_m - r_m \cos \theta_m \left(1 - \frac{\sigma_\theta^2}{2}\right) & r_m^2 \sin^2 \theta_m + \left(1 - \frac{\sigma_\theta^2}{2}\right)^2 - 2r_m \sin \theta_m \left(1 - \frac{\sigma_\theta^2}{2}\right) \end{bmatrix}\right\} \end{aligned}$$

We now notice that if $\eta_r \sim \mathcal{U}(-r^*, r^*)$ with variance σ_r^2 and $\eta_\theta \sim \mathcal{U}(-\eta_\theta^*, \eta_\theta^*)$, we have

$$\begin{aligned} E\{r_m^2 \cos^2 \theta_m\} &= E\{r_m^2\} E\{\cos^2 \theta_m\} \\ E\{r_m^2\} &= E\{r^2\} + E\{\eta_r^2\} + 2rE\{\eta_r\} = 1 + \sigma_r^2 \\ E\{\cos^2 \theta_m\} &= \frac{1}{2}(1 + E\{\cos 2\theta_m\}) \\ E\{\sin^2 \theta_m\} &= \frac{1}{2}(1 - E\{\cos 2\theta_m\}) \\ E\{\cos 2\theta_m\} &= -E\{\cos 2\eta_\theta\} \\ E\{\cos 2\eta_\theta\} &\approx 1 - 2\sigma_\theta^2 \\ E\{\cos \theta_m\} &= 0 \\ E\{\sin \theta_m\} &= E\{\cos \eta_\theta\} \approx 1 - \frac{\sigma_\theta^2}{2} \\ E\{\cos \theta_m \sin \theta_m\} &= -\sin_\theta^2 E\{\cos \eta_\theta \sin \eta_\theta\} = 0 \end{aligned}$$

Substituting

$$\begin{aligned} C\{z\} &= E\{(z - \mu_z)(z - \mu_z)^T\} = \\ &= \begin{bmatrix} (1 + \sigma_r^2)\sigma_\theta^2 & 0 \\ 0 & (1 + \sigma_r^2)(1 + \sigma_\theta^2) - \left(1 - \frac{\sigma_\theta^2}{2}\right)^2 \end{bmatrix} \end{aligned}$$

Instead, if we make use of the *analytic solution* with no approximation (which *is only valid* for uniform pdfs), we have

$$\begin{aligned} C\{z\} &= E\{(z - \mu_z)(z - \mu_z)^T\} = \\ &= E\left\{\begin{bmatrix} r_m \cos \theta_m \\ r_m \sin \theta_m - \frac{\sin \eta_\theta^*}{\eta_\theta^*} \end{bmatrix} \begin{bmatrix} r_m \cos \theta_m & r_m \sin \theta_m - \frac{\sin \eta_\theta^*}{\eta_\theta^*} \end{bmatrix}\right\} = \\ &= E\left\{\begin{bmatrix} r_m^2 \cos^2 \theta_m & r_m^2 \cos \theta_m \sin \theta_m - r_m \cos \theta_m \frac{\sin \eta_\theta^*}{\eta_\theta^*} \\ r_m^2 \cos \theta_m \sin \theta_m - r_m \cos \theta_m \frac{\sin \eta_\theta^*}{\eta_\theta^*} & r_m^2 \sin^2 \theta_m + \frac{\sin \eta_\theta^{*2}}{\eta_\theta^{*2}} - 2r_m \sin \theta_m \frac{\sin \eta_\theta^*}{\eta_\theta^*} \end{bmatrix}\right\} \end{aligned}$$

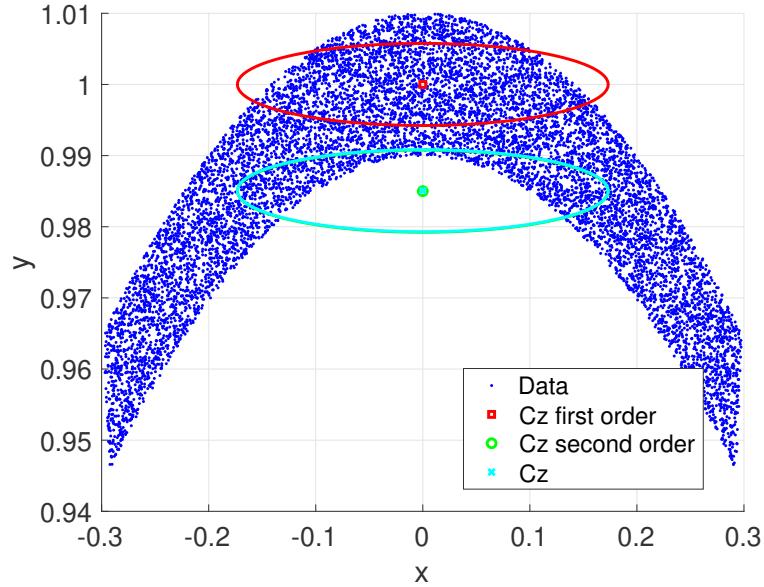


Figure 13.2: Approximation of the covariances for the previous example (with $\eta_r \sim \mathcal{U}(-0.01, 0.01)$ metres and $\eta_\theta \sim \mathcal{U}(-0.4, 0.4)$ radians) for the *first order* Taylor, *second order* Taylor and the *analytic* solution using a Monte Carlo approach (i.e. *Type B evaluation*).

With respect to the previous case we only have the following differences

$$E \{ \cos 2\eta_\theta \} = \frac{\sin 2\eta_\theta^*}{2\eta_\theta^*}$$

$$E \{ \sin \theta_m \} = E \{ \cos \eta_\theta \} = \frac{\sin \eta_\theta^*}{\eta_\theta^*}$$

Hence, substituting we have

$$\begin{aligned} C \{ z \} &= E \{ (z - \mu_z)(z - \mu_z)^T \} = \\ &= \begin{bmatrix} \frac{1}{2}(1 + \sigma_r^2)(1 - \frac{\sin 2\eta_\theta^*}{2\eta_\theta^*}) & 0 \\ 0 & \frac{1}{2}(1 + \sigma_r^2)(1 + \frac{\sin 2\eta_\theta^*}{2\eta_\theta^*}) - \frac{\sin \eta_\theta^{*2}}{\eta_\theta^{*2}} \end{bmatrix} \end{aligned}$$

For a pictorial description of such a behaviour, please refer to Figure 13.2.

The main message is that a second order approximation *fits better* when the nonlinearities becomes relevant. The *mean* (Figure 13.1) and the *variance* (Figure 13.2) are not sufficient to fully describe the pdf in many cases, as the approximation in Figure 13.3 details. Since the EKF uses the first order approximation, sometimes it may be useful to use something more appropriate: the *Unscented Kalman Filter*.

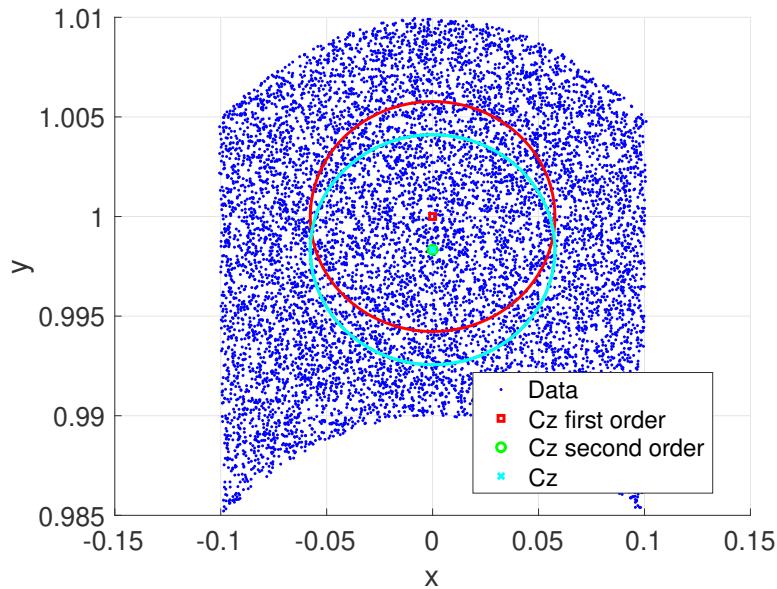


Figure 13.3: Approximation of the covariances for the previous example when $\eta_\theta \sim \mathcal{U}(-0.1, 0.1)$ radians.

13.3 Application of the Least Squares in Distributed Systems

13.3.1 Clock synchronisation example

Dictionary definition(s) of *synchronisation*:

1. An adjustment that causes something to occur or recur in unison;
2. Coordinating by causing to indicate the same time.

Intrinsic ambiguity: event-based synchronisation is *not* the same as time-based synchronisation. There is a *lot of confusion about the meaning of synchronisation*. A clock is an electronic device, which counts the number of periods of a given oscillator running at a given nominal frequency. *Clock synchronisation* in digital circuits as well as *symbol synchronisation in communication systems* are related to a common *event* triggered by a specific signal, hence *no explicit notion of time is needed*. The difference between internal and external synchronisation is depicted in Figure 13.4.

Definition 70 (IEEE 1588 def.). *Two clocks are synchronised to a specified uncertainty if they have the same epoch (i.e. the same time origin) and*

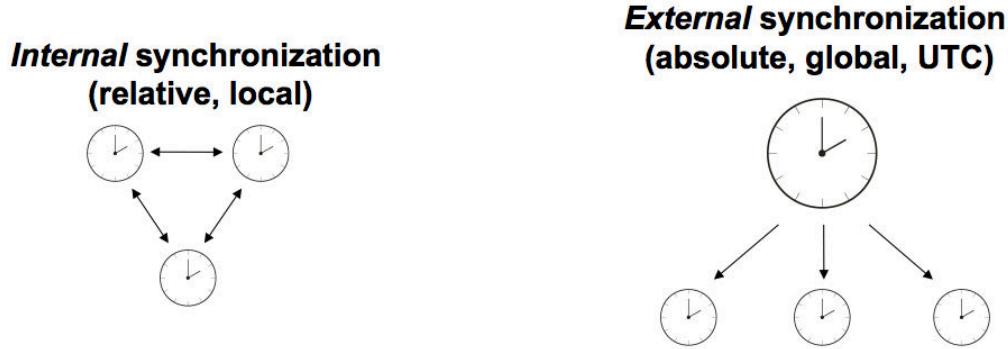


Figure 13.4: Difference between internal and external synchronisation.

their measurement of time of a single event differ by no more than that uncertainty.

In a world where system connectivity steadily grows, a *common timescale* is essential:

- To coordinate and to schedule distributed tasks (e.g. power management);
- To aggregate data/events to support decisions (e.g. data fusion);
- To schedule communication activities (e.g. slotted CSMA/CA, TDMA).

Imagine three microphones on the side of a road and measuring the noise generated by a vehicle passing by (Figure 13.5). By knowing the intensity of the sound, the geometry of the sensor deployment and the *common time reference*, it is actually possible to localise the vehicle on the road.

The *clock* on an embedded node is generated by an *oscillator* generating a signal

$$s(t) = A \sin(\phi(t))$$

The *instantaneous frequency* is affected by systematic and random *uncertainty fluctuations*

$$f_c(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt} = f_0(1 + \nu + \rho t + \frac{1}{2\pi f_0} \frac{d\xi(t)}{dt})$$

where

- f_0 is the *nominal frequency*;

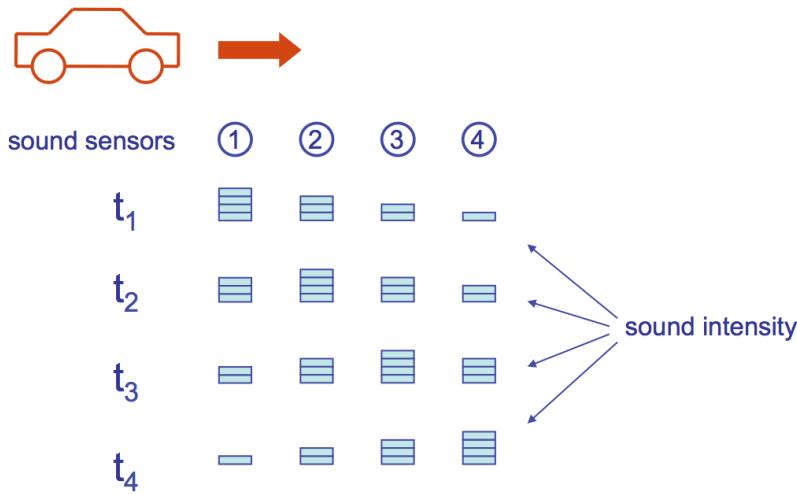


Figure 13.5: A *data fusion* example: speed and direction estimation.

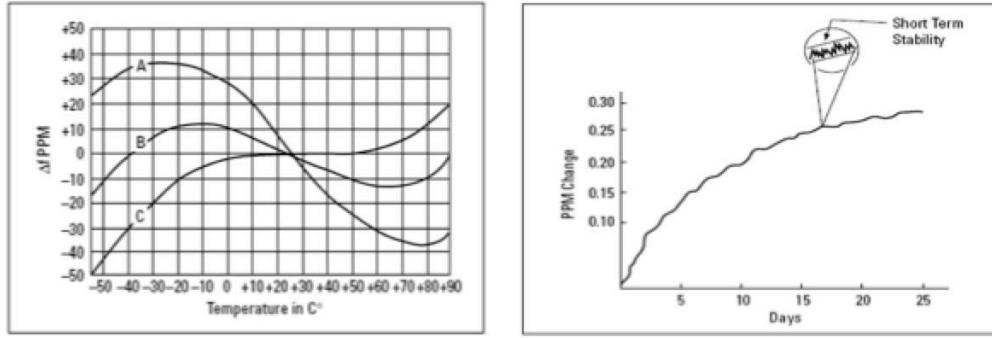
- ν is the *fractional frequency offset*, which depends on tolerance and it is sensitive to *temperature*;
- ρ is the *ageing coefficient*;
- $x(t) = \frac{1}{2\pi f_0} \xi(t)$ is responsible for the *random frequency deviation*.

In general, all clocks tend to drift away due to accumulation of 5 types of noise

$$x(t) = \frac{1}{2\pi f_0} \xi(t) \Rightarrow S_x(f) = \begin{cases} \frac{1}{(2\pi)^2} \sum_{\alpha=-4}^0 h_\alpha f^\alpha, & f \leq f_h \\ 0 & f > f_h \end{cases}$$

where f_h is the *upper cutoff frequency* due to low-pass filtering in the oscillator. The contributions of the disturbances are:

- $\alpha = -4$ is the *random walk* intrinsic noise sources within the quartz and electrode;
- $\alpha = -3$ is the *flicker frequency intermodulation* of white frequency (carrier noise) particularly within the oscillator loop;
- $\alpha = -2$ is the *white frequency* carrier noise, predominately the Crystal noise;
- $\alpha = -1$ is the *flicker phase* pink noise (equal power per decade frequency), predominately buffer amplifier flicker noise;



*Temperature vs. frequency plots
of XO's with different crystal cuts
(by courtesy of Agilent)*

*Aging short-term fractional
frequency variations of a plain XO
(by courtesy of Agilent)*

Figure 13.6: Oscillator noise measured in *part per million* (PPM), that is the fraction of error accumulated for a million of oscillations.

- $\alpha = 0$ is the *white phase* thermal noise (Johnson noise), given by the buffer amplifier noise, the resistor noise and the shot noise.

Two typical graphs describing the uncertainty contributions for clocks are reported for reference in Figure 13.6.

The model of the clock is in general given by a linear model comprising two different effects: the *time offset* is $O_{ij} = T_i(0) - T_j(0)$; the *frequency offset skew* is $\Delta\nu_{ij}(t) = \nu_i(t) - \nu_j(t)$. The effect of those quantities on the clock model is pictorially described in Figure 13.7. Hence, with the term *clock synchronisation* we refer to the *periodic offsets estimation and compensation within a given target uncertainty*. *Time synchronisation* can be regarded as a special kind of *calibration* (although UTC traceability is not always required nor possible). At each synchronisation *both time offsets and the frequency offsets* should be compensated. Synchronisation have to be repeated *periodically* as the process is *not stationary*. Besides the *native* limited oscillator stability, time synchronisation accuracy is limited by 2 further groups of uncertainty contributions:

- *Timestamping contributions:*
 - Clock resolution;
 - Transmission/reception (TX/RX) clock reading latency.
- *Communication latency contributions:*

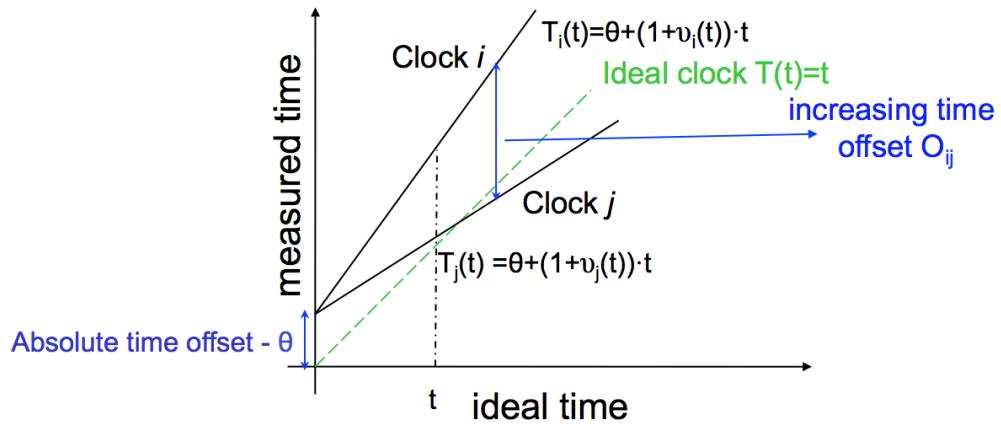


Figure 13.7: Graphical representation of the *clock synchronisation problem*.

- Medium access time (related to the MAC);
- Point-to-point communication delay;
- Intermediate node delays (especially for *wireless sensor networks* - WSN);
- Traffic and channels asymmetries (especially for WSN).

Accuracy *improves* with PHY-layer (*Physical Layer*) timestamping and drift removal. Accuracy *degrades* as the distance between nodes increases (especially for WSN).

Synchronisation in WSNs

We recall that the overall *communication latency contributions* can be expressed as

$$\delta = \delta_s + \delta_a + \delta_T + \delta_R$$

where, for the case of a WSN (but similar data can be derived for a wired network):

- δ_s - *send time*: packet construction time, including the data transmission time between the micro-controller and the radio-module. Mostly deterministic with some random fluctuations;
- δ_a - *access time*: time spent to access the channel. Mostly random and possibly very large;

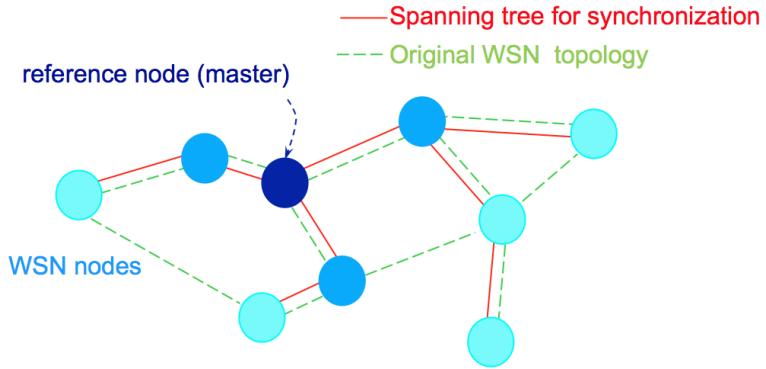


Figure 13.8: *Master election* in a WSN.

- δ_T - *transmission and propagation time*: time for a packet to reach the receiver after leaving the sender. Mostly deterministic;
- δ_R - *reception and receive time*: time to process the incoming packet. Similar to δ_s .

Usually, WSN synchronisation algorithm search for a *hierarchical structure*: a tree topology is built from the existing mesh topology (see Figure 13.8) after the *master is elected* (temporarily or permanently). Absence of *loops* makes convergence *faster*. Some solutions readily available for clock synchronisation are:

- Timing-Sync protocol for sensor network (TPSN);
- Tiny-Sync/Mini-Sync;
- Light-weight time syntonisation (LTS);
- Precision time protocol (PTP) – IEEE 1588.

One of the simplest clock synchronisation algorithm is the *two way message approach*, reported in Figure 13.9. For $k = 1, \dots, N$ we have:

$$T_{2,k} = \frac{1}{\nu_{ij}} (T_{1,k} + \delta_{ij} + \varepsilon_k) + \theta_j - \frac{\theta_i}{\nu_{ij}}$$

$$T_{3,k} = \frac{1}{\nu_{ij}} (T_{4,k} - \delta_{ij} - \eta_k) + \theta_j - \frac{\theta_i}{\nu_{ij}}$$

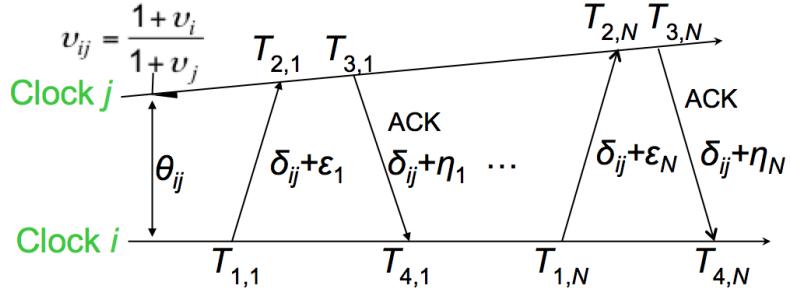


Figure 13.9: The most popular synchronisation algorithms use the *two way message approach*.

where δ_{ij} is the *systematic latency*, while ε_k and η_k are its zero-mean random fluctuations. Denoting with $\theta_{ij} = \theta_j - \frac{\theta_i}{\nu_{ij}}$

$$T_{2,k} = \frac{1}{\nu_{ij}} (T_{1,k} + \delta_{ij} + \varepsilon_k) + \theta_{ij} \quad \text{and} \quad T_{3,k} = \frac{1}{\nu_{ij}} (T_{4,k} - \delta_{ij} - \eta_k) + \theta_{ij},$$

for $k = 1, \dots, N$, we have

$$\begin{aligned} T_{1,1} &= \nu_{ij} T_{2,1} - \nu_{ij} \theta_{ij} - \delta_{ij} - \varepsilon_1 \\ &\vdots \\ T_{1,N} &= \nu_{ij} T_{2,N} - \nu_{ij} \theta_{ij} - \delta_{ij} - \varepsilon_N \\ -T_{4,1} &= -\nu_{ij} T_{3,1} + \nu_{ij} \theta_{ij} - \delta_{ij} - \eta_1 \\ &\vdots \\ -T_{4,N} &= -\nu_{ij} T_{3,N} + \nu_{ij} \theta_{ij} - \delta_{ij} - \eta_N \end{aligned}.$$

By solving with respect to $T_{1,k}$ and $T_{4,k}$ and stacking the previous set of equations, we have:

$$\begin{bmatrix} T_{1,1} \\ \vdots \\ T_{1,N} \\ -T_{4,1} \\ \vdots \\ -T_{4,N} \end{bmatrix} = \begin{bmatrix} T_{2,1} & -1 & -1 \\ \vdots & \vdots & \vdots \\ T_{2,N} & -1 & -1 \\ -T_{3,1} & 1 & -1 \\ \vdots & \vdots & \vdots \\ -T_{3,N} & 1 & -1 \end{bmatrix} \begin{bmatrix} \nu_{ij} \\ \nu_{ij} \theta_{ij} \\ \delta_{ij} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \\ \eta_1 \\ \vdots \\ \eta_N \end{bmatrix} \Rightarrow Z = Hx + E.$$

Therefore, yielding the WLS batch solution:

$$x = (H^T H)^{-1} H^T Z.$$

For the recursive solution, we need first to initialise the *Recursive WLS*. As consequence, the first *two* sequence of messages (otherwise the number of measurements is *less* than the number of unknowns), are used to initialise the filter using the *batch LS solution*, i.e.

$$P(2) = (H_2^T W_2 H_2)^{-1}, \\ \hat{x}(2) = P(2) H_2^T W_2 z(2).$$

where $z(2) = [T_{1,1}, T_{1,2}, -T_{4,1}, -T_{4,2}]^T$ and

$$H_2 = \begin{bmatrix} T_{2,1} & -1 & -1 \\ T_{2,2} & -1 & -1 \\ -T_{3,1} & 1 & -1 \\ -T_{3,2} & 1 & -1 \end{bmatrix}, \text{ and } W_2 = \begin{bmatrix} \frac{1}{\sigma_{\varepsilon_1}^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_{\varepsilon_2}^2} & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_{\eta_1}^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_{\eta_2}^2} \end{bmatrix}$$

Then, for any new send/receive message $k = 2, \dots$, we can compute:

$$\begin{aligned} S(k+1) &= H(k+1)P(k)H(k+1)^T + W_{k+1}^{-1} \\ G(k+1) &= P(k)H(k+1)^T S(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k) + G(k+1)(z(k+1) - H(k+1)\hat{x}(k)) \\ P(k+1) &= (I - G(k+1)H(k+1))P(k) \end{aligned}$$

where $z(k) = [T_{1,k}, -T_{4,k}]^T$ and

$$H(k) = \begin{bmatrix} T_{2,k} & -1 & -1 \\ -T_{3,k} & 1 & -1 \end{bmatrix}, \text{ and } W_k = \begin{bmatrix} \frac{1}{\sigma_{\varepsilon_k}^2} & 0 \\ 0 & \frac{1}{\sigma_{\eta_k}^2} \end{bmatrix}$$

Once \hat{x} is available, we can retrieve:

$$\begin{aligned} \hat{\nu}_{ij} &= \hat{x}_1, \\ \hat{\theta}_{ij} &= \frac{\hat{x}_2}{\hat{x}_1}, \\ \hat{\delta}_{ij} &= \hat{x}_3, \end{aligned}$$

therefore the clocks dynamic become

$$\begin{aligned} T_i(t) &= \theta_i + (1 + \nu_i) \frac{t}{\nu_{ij}} - \hat{\theta}_{ij}, \\ T_j(t) &= \theta_j + (1 + \nu_j)t. \end{aligned}$$

As alternative of the previous algorithm is the *one way message approach*, reported in Figure 13.10. Sensible when δ_{ij} is comparable or smaller than

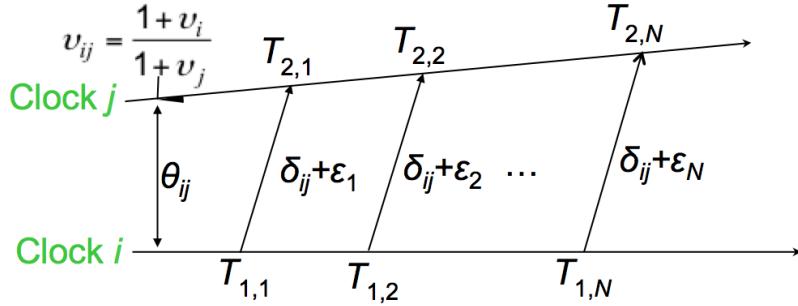


Figure 13.10: Some synchronisation algorithms use the *one way message approach*.

timestamping uncertainty (e.g. when timestamping at MAC layer). Message/beacon broadcasting can be used (no ACK is needed). Used in Flooding Time Synchronisation Protocol (FTSP). In this case, try to infer the solution (which is still a WLS).

Other solutions are available. For example, a well known algorithm is the *Reference Broadcast Protocol* (RBS):

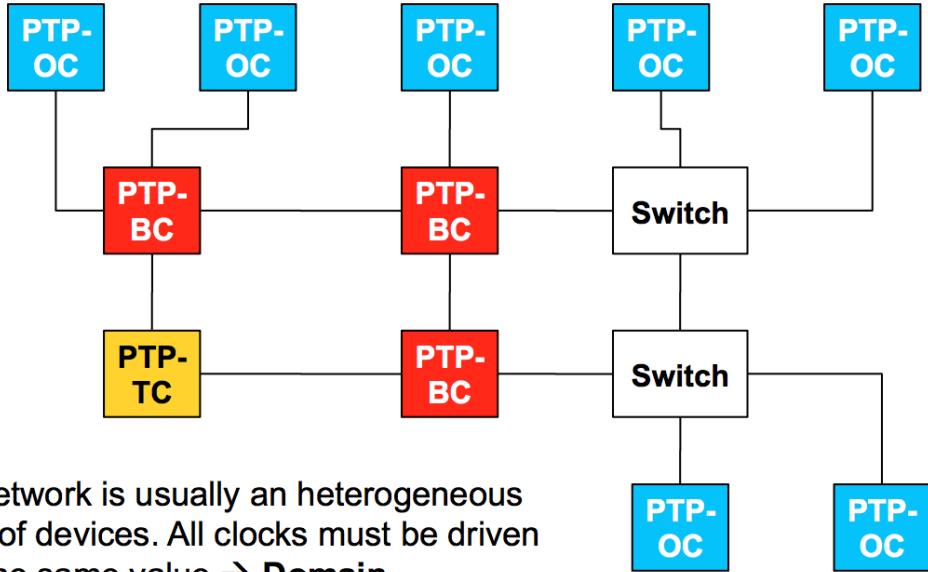
1. One chosen node broadcasts N reference packets to a set of visible receivers. The local timestamp does not matter;
2. Each node records its clock value as soon as the k -th reference packet is received;
3. All receivers exchange their own timestamps;
4. Each receiving node estimate its offset and skew to any other receiver.

The result is once again the WLS and it is quite close to the Global Positioning System (GPS) pseudo-ranging method.

Wired Networks Synchronisation: the IEEE 1588

The *Precision Time Protocol* (PTP - IEEE 1588) is an application layer protocol provides global synchronisation over a local network. Accuracy depends on the *chosen timestamping mechanism*. No need for GPS receivers in each node, only one in the *master*. Two versions: 2002 and 2008. Conceived for *wired LAN synchronisation* (and partially extended to WiFi), it has been recently applied to WSNs thanks to 6LowPAN (TCP/IP over IEEE 802.15.4).

In PTP there are three different type of devices:



A network is usually an heterogeneous set of devices. All clocks must be driven to the same value → Domain

Figure 13.11: The network is potentially full meshed. The OCs can just be only on the boundaries of the PTP network. Unlike OCs and BCs, TC do *not* correct their own clock, they work as switch but they *estimate the bridge delay and the line delay*.

- *Ordinary clock* (OC): a clock that has a *single* PTP port in a time domain and maintains the timescale used in the same domain. It may be used as a master clock or as a slave clock;
- *Boundary clock* (BC): a clock that has *multiple* (e.g. M) PTP ports in a domain and maintains the timescale used in the domain. It may be used as a master clock or as a slave clock. In case it is a slave, the other M-1 ports may become the master of the corresponding subnets;
- *Transparent clock* (TC): A device that *measures the time taken for a PTP event message to transit the device* and provides this information to clocks receiving this PTP event message.

An example of interconnected components is reported in Figure 13.11. A network may potentially consist of multiple domains. PTP devices may potentially participate in multiple domains; however, unless otherwise specified in the standard, the operation of the protocol and the timescale in different domains is independent. The *master slave hierarchy* of the PTP is reported in Figure 13.12. The first task of PTP protocol is to *logically configure* net-

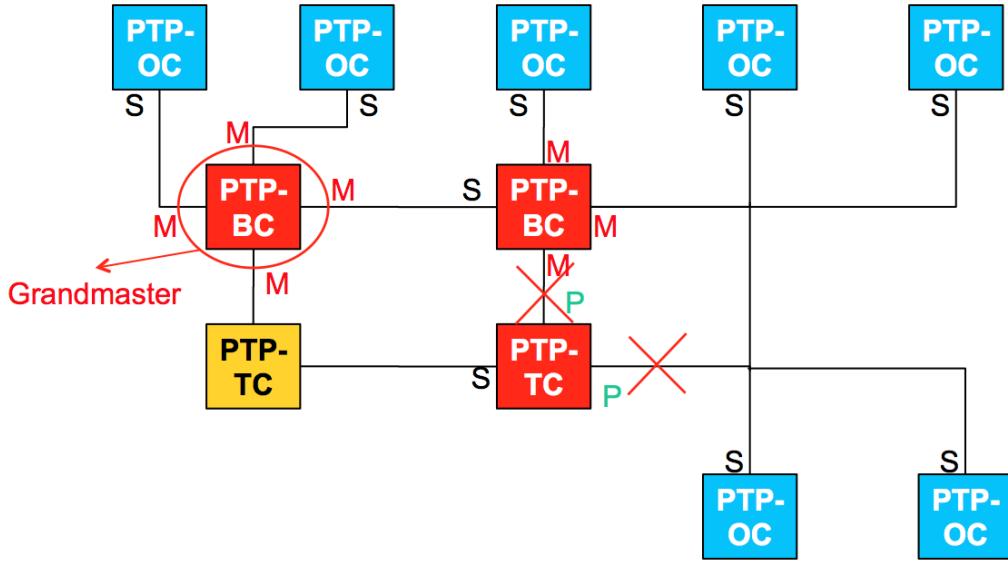


Figure 13.12: Non-PTP devices are invisible to PTP. All clocks in IEEE 1588 systems are organised into a *master slave hierarchy*.

work communications used by IEEE 1588 into a tree structure. The root of the tree is referred to as *grandmaster*. Such a grandmaster can be traceable to UTC or not. The election of the master is performed through a so-called *Best Master Clock* (BMC) algorithm. *BMC works by changing the states of the ports:* M , S and P . Usually 1 clock only for each clock. PTP implements the Two-way message exchange (see Figure 13.13), where *Sync Messages*:

- Issued by clocks in the *Master state*;
- Contain an estimate of the sending time, i.e., \tilde{t}_1 ;
- When received by a slave clock the receipt time t_2 is noted;
- Can be distinguished from other legal messages on the network;
- For best accuracy these messages can be easily identified and detected at or near the physical layer and the precise sending (or receipt) time recorded.

Instead, the *Follow-Up Messages*:

- Issued by clocks in the *Master state*;
- Always associated with the preceding *Sync message*;

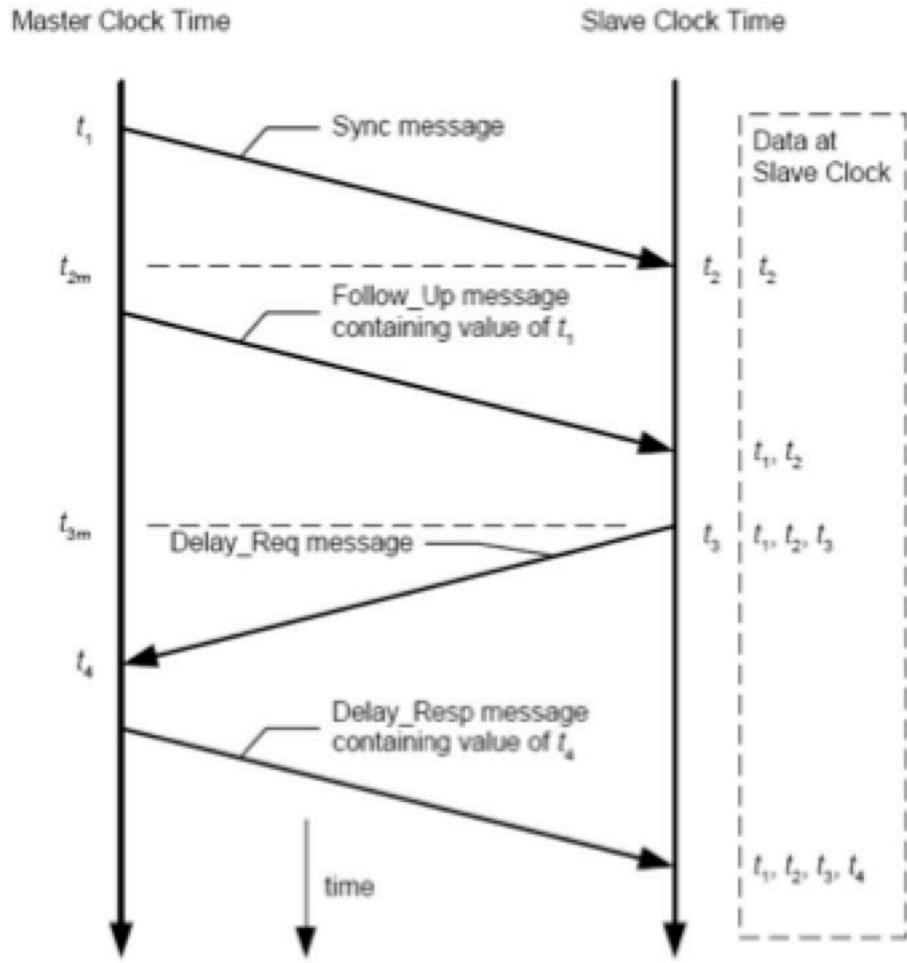


Figure 13.13: Two-way message exchange of PTP.

- Contain the *precise sending time* t_1 as measured as close as possible to the *Physical layer* of the network;
- When received by a slave clock the *precise sending time* is used in computations rather than the estimated sending time contained in the *Sync message*.

Next, the *Delay_Req Messages*:

- Issued by clocks in the *Slave state*;
- The slave *measures and records* the sending time t_3 ;
- When received by the master clock the receipt time t_4 is noted;

- Can be distinguished from other legal messages on the network;
- For best accuracy these messages can be easily identified and detected at or near the physical layer and the precise sending (or receipt) time recorded.

Finally, the *Delay-Resp Messages*:

- Issued by clocks in the *Master state*;
- Always associated with a preceding *Delay-Req message* from a specific slave clock;
- Contain the receipt time of the associated *Delay-Req message* t_4 ;
- When received by a slave clock the receipt time is noted and *used in conjunction with the sending time of the associated Delay-Req message t_3* as part of the latency calculation.

It then follows that:

- The *transmission delay* or *latency* is given by

$$\delta = \frac{(t_4 - t_3) - (t_2 - t_1)}{2}$$

- The *master-slave* offset is given by

$$O_{sm} = t_2 - t_1 + \delta$$

As reported in [25], the time stamps of the clocks are hardware assisted to limit the latency and the jitter induced by the software (Figure 13.14).

Let us now consider an example for large linear topologies [26], which is depicted in Figure 13.16. If we just consider the effect of *white frequency and random walk frequency* noises, the evolution of the free-running clock of the i -th node can be described by:

$$\begin{cases} \dot{\nu}_i(t) = 0 \\ \dot{\rho}_i(t) = \nu_i + \delta_{\rho_i}(t) \\ \dot{\tau}_i(t) = \rho_i(t) + \delta_{\tau_i}(t) \end{cases},$$

where

- $\nu_i(t)$ is the *frequency drift* rate mostly due to ageing. In a first approximation this term can be assumed to be constant;

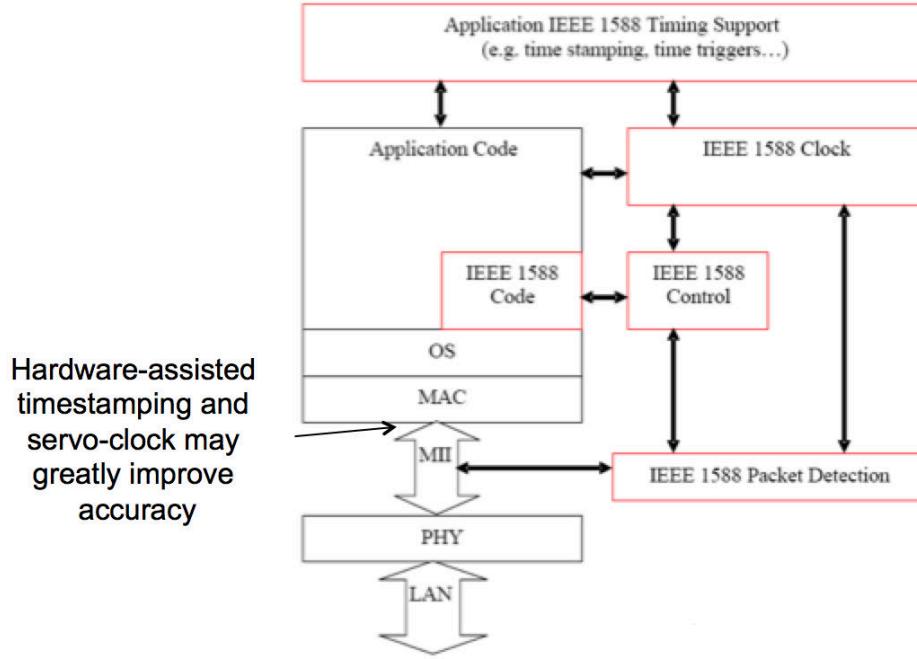


Figure 13.14: Hardware assisted generation of time stamps is potentially the most accurate [25].

- $\rho_i(t)$ is the process modelling the *normalised frequency* of the i -th clock (i.e. $1 \pm$ some fractional systematic skew);
- $\tau_i(t)$ is the process modelling the *time* of the i -th node at time t on an ideal time scale;
- $\delta_{\rho_i}(t)$ is a *zero-mean white noise* causing the random walk time fluctuations;
- $\delta_{\tau_i}(t)$ is the *zero-mean white frequency noise*.

Additionally, the time measured by each clock is also affected by the finite resolution of the clock itself and it results from:

$$c_i(t) = \frac{\lfloor f_i \tau_i(t) \rfloor}{f_i} = \tau_i(t) + \tilde{q}_i,$$

where f_i is the *nominal frequency* of the i -th and $\tilde{q}_i \in \mathcal{U}(0, \frac{1}{f_i})$ is the *corresponding quantisation error*.

Let $e_{\nu_i}(t) = \nu_0 - \nu_i$, $e_{\rho_i}(t) = \rho_0(t) - \rho_i(t)$ and $e_{\tau_i}(t) = \tau_0(t) - \tau_i(t)$ be the drift rate difference, frequency skew and time offset of the i -th TC with

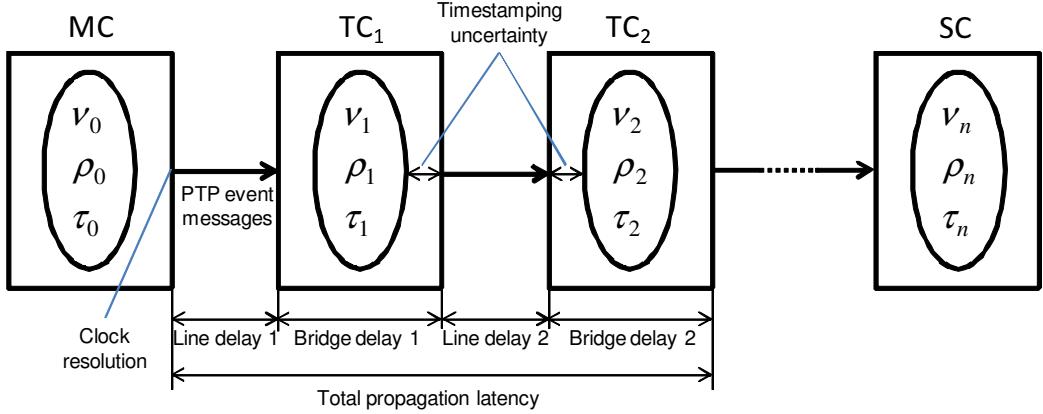


Figure 13.15: Qualitative scheme of a PTP-based network with linear topology. Between the master clock (MC) and the slave (ordinary or boundary) clock (SC), $n - 1$ peer-to-peer transparent clocks (TCs) are used [26].

respect to the MC at time t . The time-continuous evolution and the output $y_e(t)$ of the system result respectively from:

$$\begin{cases} \dot{e}_{\nu_i}(t) = 0 \\ \dot{e}_{\rho_i}(t) = e_{\nu_i}(t) + \delta_{\rho_0}(t) - \delta_{\rho_i}(t) = e_{\nu_i}(t) + \Delta_{\rho_i}(t) \\ \dot{e}_{\tau_i}(t) = e_{\rho_i}(t) + \delta_{\tau_0}(t) - \delta_{\tau_i}(t) = e_{\rho_i}(t) + \Delta_{\tau_i}(t) \\ y_e(t) = e_{\tau_i}(t) \end{cases}$$

Since the ideal time scale is unknowable, in the following the absolute reference time scale will be the MC time τ_0 and the dependence on t will be omitted. Accordingly, we can be rewrite:

$$\begin{cases} \dot{\mathbf{e}}_i(\tau_0) = A\mathbf{e}_i(\tau_0) + B\tilde{\mathbf{v}}_i(\tau_0) \\ y_e(\tau_0) = C\mathbf{e}_i(\tau_0) \end{cases},$$

where

- $\dot{\mathbf{e}}_i(\tau_0)$ is the derivative of $\mathbf{e}_i = [e_{\nu_i}(t), e_{\rho_i}(t), e_{\tau_i}(t)]$ with respect to τ_0 ;
- $\tilde{\mathbf{v}}_i(\tau) = [\Delta_{\rho_i}(\tau), \Delta_{\tau_i}(\tau)]^T$.

Notice that this system is *observable*, i.e. its state can be estimated by measuring $y_e(\tau_0) = e_{\tau_i}(\tau_0)$.

For the matrices in

$$\begin{cases} \dot{\mathbf{e}}_i(\tau_0) = A\mathbf{e}_i(\tau_0) + B\tilde{\mathbf{v}}_i(\tau_0) \\ y_e(\tau_0) = C\mathbf{e}_i(\tau_0) \end{cases},$$

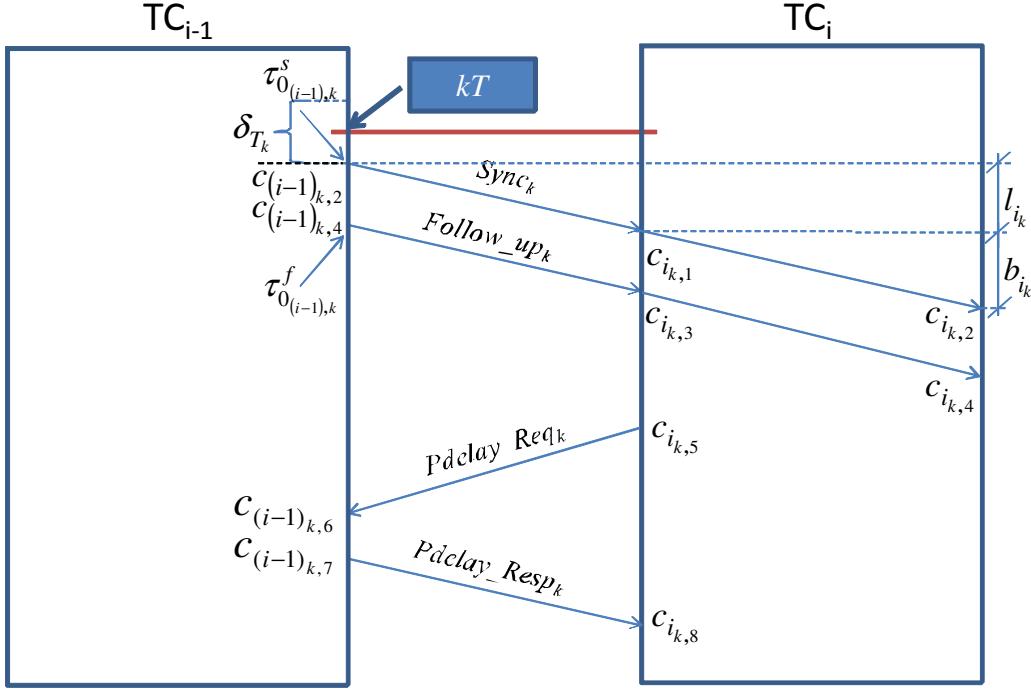


Figure 13.16: Message sequence diagram between TCs $i - 1$ and i [26].

we have

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad C = [0 \ 0 \ 1]$$

In particular, for the k -th message sent by the MC

$$y_e(\tau_{0k}) = C\mathbf{e}_i(\tau_{0k}) = e_{\tau_i}(\tau_{0k}) + \tilde{w}(\tau_{0k}),$$

where $\tilde{w}(\tau_{0k})$ comprises quantisation, timestamping errors as well as bridge and line delay estimation errors. The message sequence is reported in Figure 13.16. It can be easily proved that the *discrete-time* error model associated with is

$$\begin{cases} \mathbf{e}_i(k+1) &= F(k)\mathbf{e}_i(k) + \mathbf{v}_i(k) \\ y_e(k) &= C\mathbf{e}_i(k) + \mathbf{w}_i(k) \end{cases},$$

Moreover, it can be proved that

$$\begin{aligned} \mathbb{E}\{\mathbf{v}_i(k)\} &\approx 0 \\ \mathbb{E}\{\mathbf{v}_i(k)\mathbf{v}_i(j)^T\} &\approx Q(k)\delta_{kj} \end{aligned}$$

and

$$\begin{aligned} \mathbb{E}\{\mathbf{w}_i(k)\} &\approx 0 \\ \mathbb{E}\{\mathbf{w}_i(k)\mathbf{w}_i(j)^T\} &\approx R(k)\delta_{kj} \end{aligned}$$

From the previous assumptions, it follows that a *Kalman filter* in each TC can return an optimal estimate of the model state \mathbf{e}_i :

- *Prediction step*:

$$\begin{aligned} \mathbf{e}_i^+(k+1) &= F(k)\hat{\mathbf{e}}_i(k) \\ P^+(k+1) &= F(k)P(k)F^T(k) + Q(k) \end{aligned}$$

- *Update step*:

$$\begin{aligned} \bar{K}(k+1) &= P^+(k+1)C^T[C P^+(k+1) C^T + R(k+1)]^{-1} \\ \hat{\mathbf{e}}_i(k+1) &= \mathbf{e}_i^+(k+1) + \bar{K}(k)[y_e(k+1) - C\mathbf{e}_i^+(k+1)] \\ P(k+1) &= [I - \bar{K}(k+1)C]\bar{P}^+(k+1) \end{aligned}$$

Applying the Kalman filer described to this problem (see [26]), the absolute values of the drift rate difference, of the frequency skew difference and of the time difference between some peer-to-peer transparent clocks of a long daisy chain and the master clock is reported in Figure 13.17, while the line delay estimation uncertainty in Figure 13.18. Finally, the average time to reduce the synchronisation standard uncertainty below 1 ms is depicted in Figure 13.19.

13.3.2 Smart grid state estimation example

Power systems are composed of *transmission, sub-transmission, distribution and generation* systems. The *transmission systems* may contain large numbers of substations which are interconnected by *transmission lines, transformers, and other devices* for system control and protection. *Power* may be injected into the system by the generators or absorbed from the system by the loads at these substations. The output voltages of *generators* (e.g., *hydropower plants*) typically do not exceed 30 kV. *Transformers* are used to increase the voltage levels to levels ranging from 69 kV all the way up to 765 kV at the generator terminals for efficient power transmission. High voltage is preferred at the transmission system for different reasons, one of which is to *minimise the copper losses* that are proportional to the *ampere flows* along lines.

At the *receiving end*, the transmission systems are connected to the *sub-transmission or distribution systems* which are operated at lower voltage

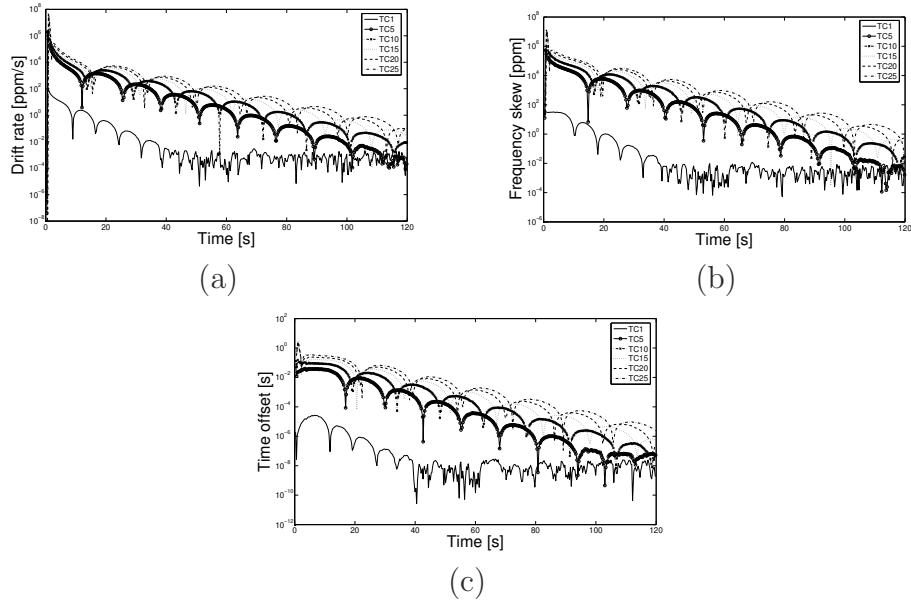


Figure 13.17: Absolute values of the drift rate difference (a), frequency skew difference (b) and of the time difference (c) as reported in [26].

levels ranging from 115 KV to 4.16 KV. *Distribution systems* are typically configured to operate in a radial configuration, where *feeders* stretch from distribution substations and form a *tree structure* with their roots at the substation and *branches spreading over the distribution area* (recall Figure 9.1-a).

Power systems are operated by system operators from the *area control centres*. To maintain the grid in *secure* state, a continuous monitoring of the system conditions is needed, which goes under the name of *grid state estimation* (see Figure 6.5). IED are the *Intelligent Electronic Devices*, which complements the RTU.

State estimators facilitate accurate and efficient monitoring of operational constraints on quantities such as the *transmission line loadings or bus voltage magnitudes*. In practice, *state estimators* provides:

- *Observability analysis*: Determines if a state estimation solution for the entire system can be obtained using the available set of measurements. Identifies the unobservable branches, and the observable islands in the system if any exist;
 - *State estimation solution*: Determines the optimal estimate for the system state, which is composed of complex bus voltages in the entire

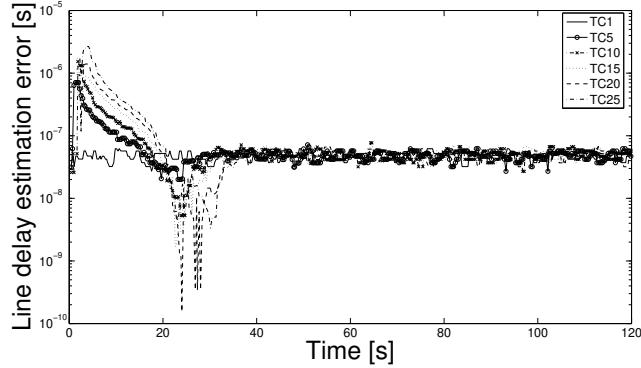


Figure 13.18: Line delay estimation uncertainty of different TCs as reported in [26].

power system, based on the network model and the gathered measurements from the system. Also provides the best estimates for all the line flows, loads, transformer taps, and generator outputs.

The equivalent circuit of a transmission line is represented in Figure 13.21-a. A transmission line with a positive sequence series impedance of $R + jX$ and total line charging susceptance of $j2B$, will be modelled by this equivalent circuit. By writing a set of nodal equations which are derived by applying *Kirchhoff's current law* at each bus, a model of the entire power system is given by

$$\begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1n} \\ Y_{21} & Y_{22} & \dots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \dots & Y_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \Rightarrow I = YV$$

where

- i_l is the net *current injection phasor* at bus l ;
- v_l is the *voltage phasor* at bus l ;
- Y is the *admittance matrix*, whose element $Y_{lr} = G_{lr} + jB_{lr}$ is the (l, r) -th element of Y .

Usually, the *measures* available are given by the two-port model [27] (see Figure 13.21-b):

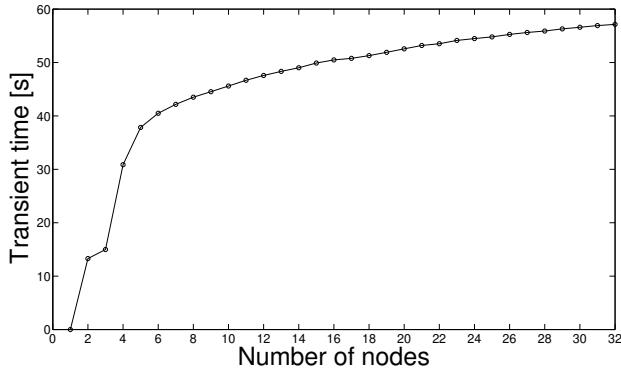


Figure 13.19: Average time to reduce the synchronisation standard uncertainty below 1 ms, as reported in [26].

- *Real and reactive power injection at bus l :*

$$P_l = V_l \sum_{r \in \mathcal{N}_l} V_r (G_{lr} \cos(\theta_{lr}) + B_{lr} \sin(\theta_{lr}))$$

$$Q_l = V_l \sum_{r \in \mathcal{N}_l} V_r (G_{lr} \sin(\theta_{lr}) - B_{lr} \cos(\theta_{lr}))$$

where V_l is the *voltage magnitude* at bus l , θ_l is the *phase angle* at bus l , $G_{lr} + jB_{lr}$ is the (l, r) -th element of the *admittance matrix* Y and $\theta_{lr} = \theta_l - \theta_r$. \mathcal{N}_l are the set of bus numbers that are directly connected to bus l ;

- *Real and reactive power flow from bus l to bus r :*

$$\begin{aligned} P_{lr} &= V_l^2 (g_{sl} + g_{lr}) - V_l V_r (g_{lr} \cos(\theta_{lr}) + b_{lr} \sin(\theta_{lr})) \\ Q_{lr} &= -V_l^2 (b_{sl} + b_{lr}) - V_l V_r (g_{lr} \sin(\theta_{lr}) - b_{lr} \cos(\theta_{lr})) \end{aligned}$$

where $g_{lr} + jb_{lr}$ is the *admittance* of the series branch connecting bus l with bus r and $g_{sl} + jb_{sl}$ is the *admittance* of the series *shunt* branch connected at bus l as in figure;

- *Line current flow magnitude from bus l to bus r :*

$$I_{lr} = \frac{P_{lr}^2 + Q_{lr}^2}{V_l},$$

or, by ignoring the shunt admittance $g_{sl} + jb_{sl}$, is given by:

$$I_{lr} = \sqrt{(g_{lr}^2 + b_{lr}^2)(V_l^2 V_r^2 - 2V_l V_r \cos(\theta_{lr}))}.$$

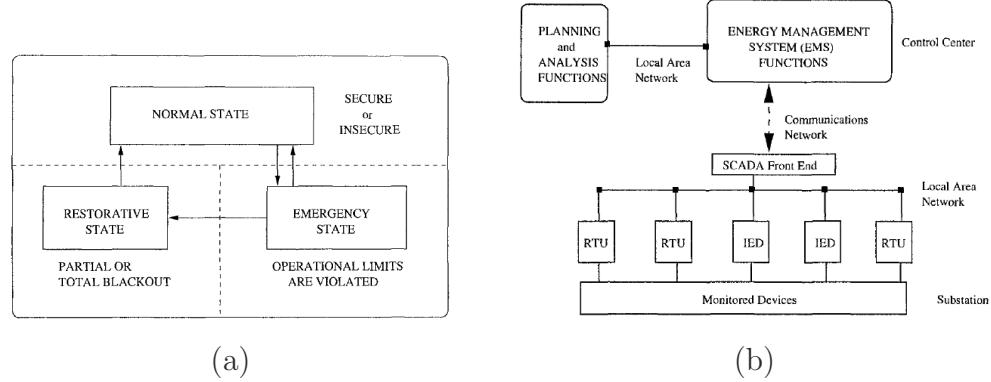


Figure 13.20: (a) Power grid monitor and security analysis and (b) EMS/SCAOA system configuration [27].

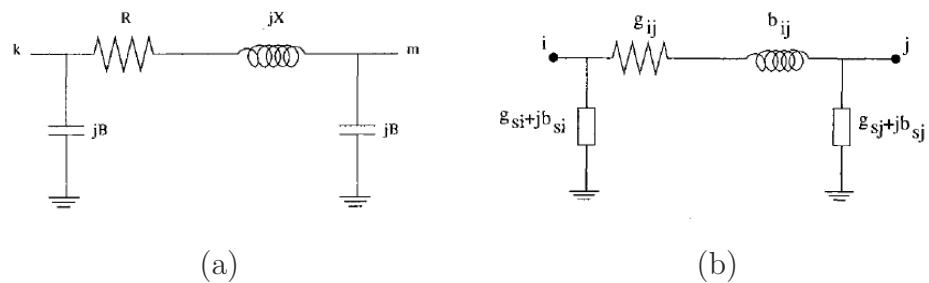


Figure 13.21: (a) Equivalent circuit for a transmission line and (b) two port π -model of a network branch [27].

The *state* of the power grid is instead given by the vector

$$x = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \\ V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

where θ_1 is considered as the *reference bus phase*, and set equal to 0. Whatever is the measured quantity (power, current, etc.) and the amount of measures available, we have that

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} h_1(\theta_2, \theta_3, \dots, \theta_n, V_1, V_2, \dots, V_n) \\ h_2(\theta_2, \theta_3, \dots, \theta_n, V_1, V_2, \dots, V_n) \\ \vdots \\ h_m(\theta_2, \theta_3, \dots, \theta_n, V_1, V_2, \dots, V_n) \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix} \Rightarrow z = h(x) + \varepsilon$$

Under the assumption that

$$E\{\varepsilon_i\} = 0, \quad C\{\varepsilon_i, \varepsilon_j\} = 0, \quad \text{and, hence } C\{\varepsilon\} = \text{diag}(\sigma_1^2, \dots, \sigma_m^2),$$

we can safely apply the *nonlinear WLS* to obtain the optimal solution in the LS sense.

Chapter 14

Advanced Estimators

14.1 Maximum Likelihood Estimation

The Maximum Likelihood Estimation method belongs to the so-called “Parameter Estimation Procedures” [28]. The results of this notes are inspired by [28, 29, 30]. Once data have been collected and the fitting model has been fixed, the parameters are getting changed to get the best possible fit.

MLE is mostly frequentist, but can be motivated from a Bayesian perspective.

Bayes’ theorem in the case of a discrete set of mutually exclusive H'_n s is:

$$P(H_n|B, I) = \frac{P(B|H_n, I)P(H_n, I)}{\sum_k P(B|H_k, I)P(H_k, I)},$$

where I is some *a priori* information. Now let the H_n be a competing, mutually exclusive hypothesis and let $B=D$ be a set of data, then the theorem can be rewritten as

$$P(H_n|D, I) = \frac{P(D|H_n, I)}{\sum_k P(D|H_k, I)P(H_k, I)}P(H_n, I),$$

which shows that the probability of the hypothesis H_n given the data D is the old probability times a factor that includes the data. In this expression $P(H_n, I)$ represents the *a priori* probability of the hypothesis H_n , $L = P(D|H_k, I)$ is the *likelihood*, $\sum_k P(D|H_k, I)P(H_k, I)$ which is the normalisation factor represents the *evidence* and $P(H_n|D, I)$ represents the *a posteriori* probability of the hypothesis H_n .

If we have a model that depends on a set of parameters θ , then different sets correspond to different hypotheses. If, as often happens, parameters can

change continuously, then hypotheses are identified by parameters, probabilities become pdf's, and the sum becomes an integral

$$p(\theta|D, I) = \frac{L(D, \theta)}{\int_{\Theta} L(D, \theta)p(\theta|I)d\theta} p(\theta|I),$$

and since the normalising integral (the evidence) does not influence the functional shape of the posterior probability, we see that maximising the posterior probability is equivalent to maximising the likelihood. This is the probabilistic basis for the principle of maximum likelihood.

We can almost invariably assume that single measurements d_k are independent and identically distributed (i.i.d.) random variables with pdf $p(d, \theta)$, so that

$$L(D, \theta) = \prod_k p(d_k, \theta).$$

To maximise this equation, the logarithmic function can help us. In fact, being the logarithm a monotonic function, maximise the logarithm of a function means maximise the function itself. So the quantity that can easily be maximised is

$$\log L(D, \theta) = \sum_k \log p(d_k, \theta),$$

where "log" is the natural logarithm.

14.1.1 Finding the MLE

The first thing to do when estimating the maximum likelihood is to collect the data and find the likelihood. After that, the probability distribution which underlies the data has to be found. Finding the distribution is a crucial step to find the most accurate parameters, since to different probability distributions correspond different parameters. The vector parameter resulting from the estimation is called "MLE estimate" and is denoted by $w_{MLE} = (w_{1,MLE}, \dots, w_{k,MLE})$.

To summarise, maximum likelihood estimation is a method to seek the probability distribution that makes the observed data most likely.

Once the likelihood has been found, there are three methods to maximise it:

- *Analytic*: The likelihood function is differentiated with respect to the parameter vector and the resulting gradient vector is set to zero. The system of equations is solved to find extrema. The second derivative is taken to make sure to have a maximum rather than a minimum. This method only works if there is an analytical solution.

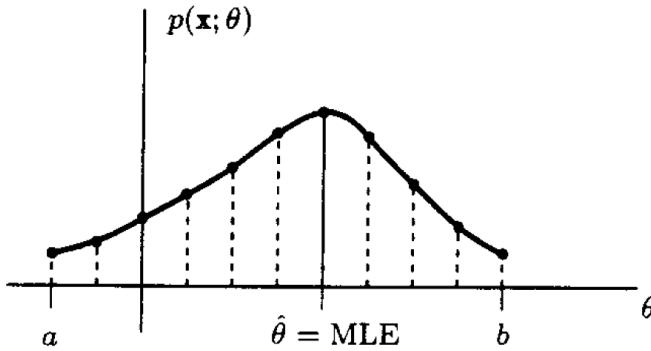


Figure 14.1: Grid search for MLE

- *Grid Search:* If you know $\hat{\theta}$ lies in a subspace of \mathbb{R} , you can do an exhaustive search over that region in the interval $[a, b]$ for the $\hat{\theta}$ that produces the largest likelihood. In other words, you can try each possible value of $\hat{\theta}$ and see which produces the largest likelihood as shown in Figure 14.1. The grid search method is a good way of showing that you can find the maximum of the likelihood function by repeated approximation and iteration. However, it is not practical in most cases and becomes much more difficult when the number of parameters rises beyond one or two.
- *Numerical:* This is the most common. Essentially, you can give the computer a set of starting values, θ_0 , and let a hill climbing algorithm (Newton-Raphson, BHHH, DFP, etc.) find the maximum.

Example 31. Let's show an example of MLE for the Poisson distribution, which is often used as the underlying distribution for a count model. This can be written as:

$$y_i \sim f(\theta, y_i) = \frac{e^{-\theta} \theta^{y_i}}{y_i!},$$

For the Poisson distribution, the likelihood function would be

$$L = \frac{e^{-\theta} \theta^{y_1}}{y_1!} \times \frac{e^{-\theta} \theta^{y_2}}{y_2!} \times \dots \times \frac{e^{-\theta} \theta^{y_N}}{y_N!} = \prod_{i=1}^N \frac{e^{-\theta} \theta^{y_i}}{y_i!} = \frac{e^{-N\theta} \theta^{\sum_{i=1}^N y_i}}{\prod_{i=1}^N y_i!},$$

and the log-likelihood function would be

$$\begin{aligned}\log L &= \sum_{i=1}^N \log \left(\frac{e^{-\theta} \theta^{y_i}}{y_i!} \right) = \sum_{i=1}^N \log \left(\frac{e^{-N\theta} \theta^{\sum_{i=1}^N y_i}}{\prod_{i=1}^N y_i!} \right) \\ &= -N\theta + \sum_{i=1}^N y_i \log(\theta) - \sum_{i=1}^N \log(y_i!).\end{aligned}\quad (14.1)$$

Now an analytical solution can be found. To do this the derivative with respect to θ of 14.1 has to be considered

$$\frac{\partial \log L}{\partial \theta} = -N + \frac{\sum_{i=1}^N y_i}{\theta}$$

and set to zero to solve for θ

$$-N + \frac{\sum_{i=1}^N y_i}{\theta} = 0 \Rightarrow \hat{\theta} = \frac{\sum_{i=1}^N y_i}{N}.$$

□

14.1.2 Properties of the MLE

The Maximum Likelihood Estimator (MLE) satisfies some important properties of the estimators.

Sufficiency

It provides complete information about the parameters.

Consistency

As the number of data increases, the parameter value converges asymptotically to its true value. This means that the MLE is asymptotically unbiased; The consistency property can be proved as follows. Let's define the mean log-likelihood of i.i.d. data:

$$l(\mathbf{x}; \theta) = \frac{1}{n} L(\mathbf{x}; \theta) = \frac{1}{n} \sum_{k=1}^n \log p(x_k, \theta).$$

This is a sort of sample mean and it is itself a random variable.

We can notice that maximising the likelihood we also maximise $l(x; \theta)$ and that the *the mean log-likelihood approaches a limit that does not depend on*

the data set: $l(\mathbf{x}; \sigma) \rightarrow l(\sigma)$ for large n (because of the law of large numbers¹) and in particular:

$$l(\mathbf{x}; \theta) \rightarrow l(\theta) = \mathbf{E}[\log p(x, \theta)] = \int_{\{\mathbf{x}\}} [\log p(x, \theta)] p(x, \theta_0) dx,$$

because the value of $l(\mathbf{x}; \theta)$ follow a pdf $p(x, \theta_0)$ that depends on the actual distribution of the data values, and therefore on the true value θ_0 .

We can see that:

$$\begin{aligned} l(\theta) - l(\theta_0) &= \mathbf{E}[l(\mathbf{x}; \theta) - l(\mathbf{x}; \theta_0)] = \int_{\{\mathbf{x}\}} [\log p(x, \theta) - \log p(x, \theta_0)] p(x, \theta_0) dx \\ &= \int_{\{\mathbf{x}\}} \log \frac{p(x, \theta)}{p(x, \theta_0)} p(x, \theta_0) dx \leq \int_{\{\mathbf{x}\}} \left[\frac{p(x, \theta)}{p(x, \theta_0)} - 1 \right] p(x, \theta_0) dx \\ &= \int_{\{\mathbf{x}\}} [p(x, \theta) - p(x, \theta_0)] dx = 1 - 1 = 0, \end{aligned}$$

where the inequality $\log x \leq x - 1$ has been used. Therefore, for large n, $\log l(\mathbf{x}; \theta_0)$, and equivalently $\log L(\mathbf{x}; \theta_0)$ correspond to a maximum, i.e., θ_0 is equal to the value $\hat{\theta}$ that maximises the likelihood, and we conclude that $\hat{\theta}$ is a consistent estimator. However, the several assumptions used to prove this property of the MLE are not always true and some MLEs are not consistent.

Asymptotic Normality

The estimator not only converges to the unknown parameter, but it converges fast enough, at a rate $1/\sqrt{N}$

$$\sqrt{N}(\hat{\theta} - \theta_0) \rightarrow \mathcal{N}(0, \sigma_{\theta_{MLE}}^2) \text{ for some } \sigma_{\theta_{MLE}}^2.$$

This asymptotic variance in some sense measures the quality of MLE.

Theorem 6 (Asymptotic normality of MLE).

$$\sqrt{N}(\hat{\theta} - \theta_0) \rightarrow \mathcal{N}\left(0, \frac{1}{I(\theta_0)}\right).$$

where $I(\theta_0)$ is the Fisher Information.

¹The law of large numbers (LLN) is a theorem which states that the average of the results obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed.

As we can see, the asymptotic variance/dispersion of the estimate around true parameter will be smaller when Fisher information is larger.

Example 32. *The family of Bernoulli distributions $B(\theta)$ has probability function*

$$f(x|\theta) = \theta^x(1-\theta)^{1-x}$$

and taking the logarithm

$$\log f(x|\theta) = x \log \theta + (1-x) \log(1-\theta).$$

The first and second derivative with respect to parameter θ are

$$\begin{aligned}\frac{\partial}{\partial \theta} \log f(x|\theta) &= \frac{x}{\theta} - \frac{1-x}{1-\theta} \\ \frac{\partial^2}{\partial \theta^2} \log f(x|\theta) &= -\frac{x}{\theta^2} - \frac{1-x}{(1-\theta)^2}\end{aligned}$$

Then the Fisher information can be computed as

$$I(\theta) = -E\left\{ \frac{\partial^2}{\partial \theta^2} \log f(X|\theta) \right\} = \frac{E\{X\}}{\theta^2} + \frac{1-E\{X\}}{(1-\theta)^2} = \frac{\theta}{\theta^2} + \frac{1-\theta}{(1-\theta)^2} = \frac{1}{\theta(1-\theta)}$$

The MLE of θ is $\hat{\theta} = \bar{X}$ and the asymptotic normality result states that

$$\sqrt{N}(\hat{\theta} - \theta_0) \rightarrow \mathcal{N}(0, \theta_0(1-\theta_0))$$

which of course, also follows directly from the Central Limit Theorem². \square

Example 33. *The family of exponential distributions $E(\alpha)$ has pdf*

$$f(x|\alpha) = \begin{cases} \alpha e^{-\alpha x}, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

and, therefore,

$$\log f(x|\alpha) = \log \alpha - \alpha x \Rightarrow \frac{\partial^2}{\partial \alpha^2} \log f(x|\alpha) = -\frac{1}{\alpha^2}.$$

This does not depend on X and we get

$$I(\alpha) = -E\left\{ \frac{\partial^2}{\partial \alpha^2} \log f(X|\alpha) \right\} = \frac{1}{\alpha^2}.$$

Therefore, the MLE $\hat{\alpha} = 1/\bar{X}$ is asymptotically normal and

$$\sqrt{N}(\hat{\alpha} - \alpha_0) \rightarrow \mathcal{N}(0, \alpha^2)$$

\square

²In probability theory, the central limit theorem (CLT) establishes that, in some situations, when independent random variables are added, their properly normalised sum tends toward a normal distribution (informally a "bell curve") even if the original variables themselves are not normally distributed.

Efficiency

The parameter value has the minimum variance. As an example, consider the (unbiased) estimator of the mean time of the exponential distribution:

$$\hat{\tau} = \frac{1}{n} \sum_{k=1}^n t_k.$$

We have $b_n = 0$ due to the unbiased estimator. If we consider n exponentially distributed data, with the pdf

$$p(t) = \frac{1}{\tau} e^{-t/\tau},$$

so there is just a parameter: τ . The Log-likelihood is:

$$\log L(\mathbf{x}; \tau) = \sum_{k=1}^n \left(-\log \tau - \frac{t_k}{\tau} \right).$$

Taking the derivative of $\log L(\mathbf{x}; \tau)$ we find

$$\frac{\partial \log L}{\partial \tau} = \sum_{k=1}^n \left(-\frac{1}{\tau} + \frac{t_k}{\tau^2} \right).$$

If we take the second derivative, we get

$$\frac{\partial^2 \log L}{\partial \tau^2} = \frac{1}{\tau^3} \left(n\tau - 2 \sum_{k=1}^n t_k \right),$$

and

$$-\mathbb{E} \left\{ \frac{\partial^2 \log L}{\partial \tau^2} \right\} = -\frac{1}{\tau^3} \left(n\tau - 2\mathbb{E} \left\{ \sum_{k=1}^n t_k \right\} \right) = -\frac{1}{\tau^3} (n\tau - 2n\tau) = \frac{n}{\tau^2}.$$

Using the CRLB we find:

$$V \{ \hat{\tau} \} \geq \frac{\tau^2}{n}.$$

Since we already know that the variance of the estimator $\hat{\tau}$ is $\frac{\tau^2}{n}$ we can conclude that the MLE is the estimator of τ with maximal efficiency.

Invariance

If you can know the MLE for a parameter, you can know the MLE for functions of the initial parameter.

Theorem 7. *If $\hat{\theta}$ is a MLE of θ and if $g(\cdot)$ is a function, then $g(\hat{\theta})$ is a MLE of $g(\theta)$.*

Proof. If $g(\cdot)$ is a one-to-one³ function, then

$$L(\theta) = L(g^{-1}(g(\theta))),$$

are both maximised by $\hat{\theta}$, so

$$\hat{\theta} = g^{-1}(g(\hat{\theta})),$$

which means that

$$g(\hat{\theta}) = \hat{g}(\theta).$$

If $g(\cdot)$ is a many-to-one function⁴, then $\hat{\theta}$ which maximises $L(\theta)$ still corresponds to $g(\hat{\theta})$, so $g(\hat{\theta})$ still corresponds to the maximum of $L(\theta)$. \square \square

Example 34. Suppose $x = \{X_1, X_2, \dots, X_n\}$ is a random sample from a Bernoulli distribution $B(1, \theta)$. Consider the MLE of the mean, θ , and variance, $\theta(1 - \theta)$. Note that $\theta(1 - \theta)$ is not a one-to-one function of θ . The log-likelihood is

$$l(\theta) = \sum_i x_i \log \theta + (n - \sum_i x_i) \log(1 - \theta),$$

$$\frac{dl(\theta)}{d\theta} = \sum_i \frac{x_i}{\theta} - \frac{(n - \sum_i x_i)}{(1 - \theta)}.$$

Putting $\frac{dl(\theta)}{d\theta} = 0$ it can be shown that the MLE of θ is $\hat{\theta} = \bar{X}$. If we set $\nu = \theta(1 - \theta)$, then

$$\frac{dl(\nu)}{d\nu} = \frac{dl(\nu(\theta))}{d\theta} \frac{d\theta}{d\nu}.$$

Since $\frac{d\theta}{d\nu}$ is not, in general, equal to zero, then

$$\hat{\nu} = \nu(\hat{\theta}) = \bar{X}(1 - \bar{X}).$$

\square

³A one-to-one function is a function where every element of the range of the function corresponds to ONLY one element of the domain.

⁴A many-to-one function is a function where an element of range corresponds to more than one element in the domain.

Example 35. We observe n samples of WGN with variance σ^2 whose power in dB is to be estimated. To do so we first find the MLE of σ^2 . Then, we use the invariance principle to find the power P in dB, which is defined as

$$P = 10 \log_{10} \sigma^2.$$

The PDF is given by

$$p(\mathbf{x}; \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2} \sum_{i=0}^{n-1} x^2[i]}.$$

Differentiating the log-likelihood function produces

$$\begin{aligned} \frac{\partial \log(\mathbf{x}; \sigma^2)}{\partial \sigma^2} &= \frac{\partial}{\partial \sigma^2} \left[-\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=0}^{n-1} x^2[i] \right] \\ &= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=0}^{n-1} x^2[i], \end{aligned}$$

and setting it equal to zero yields the MLE

$$\sigma^2 = \frac{1}{n} \sum_{i=0}^{n-1} x^2[i].$$

The MLE of the power in dB follows as

$$\hat{P} = 10 \log_{10} \hat{\sigma}^2 = 10 \log \frac{1}{n} \sum_{i=0}^{n-1} x^2[i].$$

□

14.1.3 Numerical Determination of the MLE

Up to now only the so-called "analytical method" of finding a MLE $\hat{\theta}$ has been explained in detail. The concept of numerical determination of the MLE has been already introduced in the Section— 14.1.1 when we dealt with the problem of a detector which can only measure times smaller than t_{min} or larger than t_{max} . In many cases, there is no analytical solution and grid searches are impractical, so the MLE can be found only using numerical methods. In particular, this approach should only be used when an analytic expression can not be found because:

- Finding the numerical solution is computationally intensive;

- Numerical search may provide only an approximate solution;
- The search algorithm may fail to converge to the global maximum.

The goal of MLE is to find values of parameters θ that maximise the likelihood function. To do this, we could start with a guess of $\hat{\theta}$ that could be called $\hat{\theta}_0$. We could then adjust this guess based on the value of the likelihood that it gives, i.e.

$$\hat{\theta}_1 = \hat{\theta}_0 + A_0,$$

or, in general

$$\hat{\theta}_k = \hat{\theta}_{k-1} + A_{k-1}.$$

Our goal is to move $\hat{\theta}$ to the point at which the likelihood is the highest. To do this we could move the adjustment parameter A which can be considered as the derivative of the likelihood with respect to the parameter θ_k i.e. considering it as the gradient matrix

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \frac{\partial \log L}{\partial \theta_k}.$$

If the gradient matrix is positive, then the likelihood is increasing in $\hat{\theta}$ and so we need to increase our guess of $\hat{\theta}$ some more. And if the gradient matrix is negative, then the likelihood is decreasing in $\hat{\theta}$ and so we need to decrease our guess of $\hat{\theta}$. By reportedly doing this, we gradually climb to the top of the likelihood. This general method is called *method of steepest ascent*. But this approach should be modified due to the fact that it does not consider how fast the slope is changing. This approach needs to be generalised so that the estimators could change not only in terms of their direction but also by a factor, the "step size", determining how far we should change them

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \lambda_{k-1} \Delta_{k-1},$$

where:

- Δ tells us the direction we want to take a step in;
- λ tells us the amount by which we want to change our estimates.

To determine how fast the slope of the likelihood is changing the second derivative can be used. If the functions is very steep (the second derivative is big), then the parameters do not need to be changed very much from an iteration to another. If the function is flat (the second derivative is small), then the parameters need to be adjusted more. To do this there are several methods, each with a maximisation algorithm associated:

- **Newton-Raphson**

It uses the inverse of Hessian

$$H = \frac{\partial^2 \log L}{\partial \theta \partial \theta'},$$

to get

$$\hat{\theta}_k = \hat{\theta}_{k-1} - \left[\left(\frac{\partial^2 \log L}{\partial \hat{\theta}_{k-1} \partial \hat{\theta}'_{k-1}} \right)^{-1} \frac{\partial \log L}{\partial \hat{\theta}_{k-1}} \right].$$

The new parameter estimates are equal to the old ones. However, they are adjusted in the direction of the first derivative (steepest ascent) and the amount of change is inversely related to the size of the second derivative. Newton-Raphson works pretty well and quickly for simply functions with global maxima;

- **Scoring**

It uses the inverse of the information matrix

$$I = -\mathbf{E} \left[\frac{\partial^2 \log L}{\partial \theta \partial \theta'} \right],$$

to get

$$\hat{\theta}_k = \hat{\theta}_{k-1} - \left[\mathbf{E} \left(\frac{\partial^2 \log L}{\partial \hat{\theta}_{k-1} \partial \hat{\theta}'_{k-1}} \right)^{-1} \right] \frac{\partial \log L}{\partial \hat{\theta}_{k-1}}.$$

Sometimes it can be easier to compute the information matrix than the Hessian matrix;

- **Berndt, Hall, Hall and Hausman (BHHH)**

It uses the inverse of the outer product approximation to the information matrix

$$\sum_{i=0}^N \frac{\partial \log L_i}{\partial \theta} \frac{\partial \log L'_i}{\partial \theta},$$

to get

$$\hat{\theta}_k = \hat{\theta}_{k-1} - \left(\sum_{i=0}^N \frac{\partial \log L_i}{\partial \theta} \frac{\partial \log L'_i}{\partial \theta} \right)^{-1} \frac{\partial \log L}{\partial \hat{\theta}_{k-1}}.$$

This method has the advantage that the second derivatives do not need to be considered. The BHHH is usually used in case of complicated likelihood function or when the data are poor.

The difficulty with the use of these iterative methods is that in general we do not know beforehand if they will converge and, even if convergence is attained, whether the value produced is the MLE. The function to be maximised is not known *a priori*. The likelihood changes for each data set, requiring the maximisation of a *random function*. Nevertheless, these methods can produce good results.

Example 36. Let's consider n exponentially distributed data, with the pdf

$$p(t) = \frac{1}{\tau} e^{-t/\tau},$$

so that there is just one parameter $\theta = \tau$, and

$$\log L(D, \tau) = \sum_{k=1}^n \left(-\log \tau - \frac{t_k}{\tau} \right).$$

Taking the derivative of $\log L(D, \tau)$ we find

$$\frac{\partial \log L}{\partial \tau} = \sum_{k=1}^n \left(-\frac{1}{\tau} + \frac{t_k}{\tau^2} \right) = -\frac{1}{\tau^2} \sum_{k=1}^n (\tau - t_k) = -\frac{1}{\tau^2} \left(n\tau - \sum_{k=1}^n t_k \right) = 0,$$

therefore the MLE for τ is

$$\hat{\tau} = \frac{1}{n} \sum_{k=1}^n t_k.$$

In this case the sample mean coincides with the MLE. The method can easily be adapted to more complex situations. Consider the case where the detector cannot measure times smaller than t_{min} or larger than t_{max} , then

$$p(t) = C e^{-t/\tau},$$

must be normalised, so that

$$1 = \int_{t_{min}}^{t_{max}} p(t) dt = C \int_{t_{min}}^{t_{max}} e^{-t/\tau} dt = C\tau [e^{-t_{min}/\tau} - e^{-t_{max}/\tau}],$$

thus

$$p(t) = \frac{1}{\tau} \frac{e^{-t/\tau}}{e^{-t_{min}/\tau} - e^{-t_{max}/\tau}}.$$

Therefore we find

$$\log L(D, \tau) = \sum_{k=1}^n \left\{ -\log \tau - \frac{t_k}{\tau} - \log \left[e^{-t_{min}/\tau} - e^{-t_{max}/\tau} \right] \right\}, \quad (14.2)$$

and

$$\begin{aligned}\frac{\partial \log L}{\partial \tau} &= \sum_{k=1}^n \left[-\frac{1}{\tau} - \frac{t_k}{\tau^2} + \frac{(t_{min}/\tau^2)e^{-t_{min}/\tau} - (t_{max}/\tau^2)e^{-t_{max}/\tau}}{e^{-t_{min}/\tau} - e^{-t_{max}/\tau}} \right] \\ &= -\frac{n}{\tau} + \frac{n}{\tau^2} \frac{t_{min}e^{-t_{min}/\tau} - t_{max}e^{-t_{max}/\tau}}{e^{-t_{min}/\tau} - e^{-t_{max}/\tau}} - \frac{1}{\tau^2} \sum_{k=1}^n t_k = 0.\end{aligned}$$

The solution of this equation cannot be found with analytical methods, and we must use a numerical method: the most common is the Newton-Raphson method. Now note that measurements are limited by our ability to measure very short times, rather than by very long waits, so that we can confidently set $t_{max} = \infty$ in (14.2), and obtain

$$\log L(D, \tau) = \sum_{k=1}^n \left(-\log \tau - \frac{t_k}{\tau} + \frac{t_{min}}{\tau} \right) = -n \log \tau - \frac{1}{\tau} \sum_{k=1}^n t_k + n \frac{t_{min}}{\tau}.$$

This log-likelihood yields the following MLE

$$\hat{\tau} = \frac{1}{n} \sum_{k=1}^n t_k - t_{min},$$

which does not coincide with the true value τ even in the limit of large n . \square

14.1.4 Extension to a Vector Parameter

The MLE for a vector parameter $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_p]^T$ is defined to be the value that maximises the likelihood function $p(\mathbf{x}; \boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{ML} = \arg \max_{\boldsymbol{\theta}} p(\mathbf{x}; \boldsymbol{\theta}),$$

over the allowable domain of $\boldsymbol{\theta}$. Assuming a differentiable likelihood function, the MLE is found from

$$\frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0.$$

If multiple solution exist, then the one that maximises the likelihood function is the MLE.

Example 37. We can take as an example the ML estimation of sinusoidal parameters. If we consider the model

$$x[n] = A + w[n], \quad n = 0, 1, \dots, N-1,$$

where $w[n]$ is WGN with variance σ^2 and the vector parameter $\boldsymbol{\theta} = [A, \sigma^2]^T$ has to be estimated. The pdf is

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2}.$$

The ML estimator is found by calculating the differential of the log likelihood function

$$\frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0.$$

This means that the PDF is differentiated over each of the components individually, and they are all set to zero

$$\begin{aligned} \frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial A} &= \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (x[n] - A), \\ \frac{\partial \log p(\mathbf{x}; \boldsymbol{\theta})}{\partial \sigma^2} &= -\frac{N}{2\sigma^2} + \frac{1}{\sigma^4} \sum_{n=0}^{N-1} (x[n] - A), \end{aligned}$$

which gives

$$\hat{A} = \bar{x} \text{ and } \hat{\sigma}^2 = \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2.$$

The MLE is

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} \bar{x} \\ \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2 \end{bmatrix}.$$

□

14.1.5 Properties of MLE for Vector Parameters

Asymptotic normality

The asymptotic properties of the MLE for vector parameters are summarised in the following theorem.

Theorem 8. If the PDF $p(\mathbf{x}; \boldsymbol{\theta})$ of the data \mathbf{x} satisfies some "regularity" conditions, then the MLE of the unknown parameter $\boldsymbol{\theta}$ is asymptotically distributed according to

$$\hat{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\theta}, \mathbf{I}^{-1}(\boldsymbol{\theta})),$$

where $\mathbf{I}(\boldsymbol{\theta})$ is the Fisher information matrix evaluated at the true value of the unknown parameter.

This theorem tells us that the vector MLE approaches an efficient estimator as $N \rightarrow \infty$. Moreover, the distribution of the estimates approaches a Gaussian distribution.

Invariance

The invariance properties of MLE for vector parameters are summarised in the following theorem.

Theorem 9. *The MLE of the parameter $\alpha = g(\theta)$, where the pdf $p(x; \theta)$ is parametrised by θ , is given by*

$$\hat{\alpha} = g(\hat{\theta}),$$

where $\hat{\theta}$ is the MLE of θ . If g is not a one-to-one function $\bar{p}(x; \theta)$, defined as

$$\bar{p}(x; \theta) = \max_{\theta: \alpha = g(\theta)} p(x; \theta).$$

This theorem tells us that if you need to estimate a function of a parameter (for example A^2 instead of A), the ML property carries over any transformation $g(\hat{\theta}) = g(\hat{\theta})$. For MVUE, this is not always the case as the estimator may become unbiased.

Optimality

The optimality property of the MLE is summarised in the following theorem.

Theorem 10. *If the observed data x are described by the linear model*

$$x = H\theta + w,$$

then the MLE of θ is

$$\hat{\theta} = (H^T C^{-1} H)^{-1} H^T C^{-1} x,$$

where H is a known $N \times p$ observation matrix, with $N > p$ and rank p , θ is a $p \times 1$ parameter vector to be estimated and w is a noise vector with PDF $\mathcal{N}(0, C)$.

Because the MLE has exactly the same form as the Minimum Variance Unbiased Estimator (MVUE) for the general linear model, the MLE is always optimal for the linear model.

14.1.6 Signal Processing Example: Sinusoidal Parameter Estimation

Let us now consider the model taken from [28]

$$x[n] = A \cos(2\pi f_0 n + \phi) + w[n],$$

with $w[n] \sim \mathcal{N}(0, \sigma^2)$. It is possible to find the MLE for all three parameters: $\boldsymbol{\theta} = [A, f_0, \phi]^T$. The pdf is given as

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} [x[n] - A \cos(2\pi f_0 n + \phi)]^2}.$$

Instead of proceeding through the log-likelihood function, it can be noted that the above function is maximised when

$$J(A, f_0, \phi) = \sum_{n=0}^{N-1} [x[n] - A \cos(2\pi f_0 n + \phi)]^2$$

is minimised. Usually, it is possible to differentiate $J(A, f_0, \phi)$ with respect to all the three components and solve the roots. However, this becomes difficult for the variable f_0 . Therefore, we will first transform the problem into a different form and find an approximate solution. Applying

$$\cos(\alpha + \beta) = \cos \alpha \cos \beta - \sin \alpha \sin \beta,$$

we get

$$J(A, f_0, \phi) = \sum_{n=0}^{N-1} [x[n] - A \cos(2\pi f_0 n) \cos \phi + A \sin(2\pi f_0 n) \sin \phi]^2$$

Now, the transformation to simplify the problem introduces the new variables α_1 and α_2 , which resembles the transformation from polar to cartesian coordinates:

$$\alpha_1 = A \cos \phi \text{ and } \alpha_2 = -A \sin \phi,$$

whose inverse is

$$A = \sqrt{\alpha_1^2 + \alpha_2^2} \text{ and } \phi = \arctan \left(-\frac{\alpha_2}{\alpha_1} \right).$$

Now the function J becomes a new function J_2 to be minimised:

$$J_2(\alpha_1, \alpha_2, f_0) = \sum_{n=0}^{N-1} [x[n] - \alpha_1 \cos(2\pi f_0 n) - \alpha_2 \sin(2\pi f_0 n)]^2$$

Let us define the following cosine and sine vectors

$$\mathbf{c} = \begin{bmatrix} 1 \\ \cos(2\pi f_0) \\ \cos(4\pi f_0) \\ \dots \cos(2\pi f_0)(N-1) \end{bmatrix} \text{ and } \mathbf{s} = \begin{bmatrix} 0 \\ \sin(2\pi f_0) \\ \sin(4\pi f_0) \\ \dots \sin(2\pi f_0)(N-1) \end{bmatrix},$$

so as to have

$$J_2(\alpha_1, \alpha_2, f_0) = (\mathbf{x} - \alpha_1 \mathbf{c} - \alpha_2 \mathbf{s})^T (\mathbf{x} - \alpha_1 \mathbf{c} - \alpha_2 \mathbf{s}).$$

Moreover, by defining

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \cos(4\pi f_0) & \sin(4\pi f_0) \\ \vdots & \ddots & \ddots & \cos(2\pi f_0)(N-1) & \sin(2\pi f_0)(N-1) \end{bmatrix} = [\mathbf{c}, \mathbf{s}],$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix},$$

we get the simplest form

$$J_2(\alpha_1, \alpha_2, f_0) = (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}).$$

We know that the minimising $\boldsymbol{\alpha}$ will be:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$$

Since $\boldsymbol{\alpha}$ depends on f_0 via \mathbf{H} , the estimation procedure estimates first the frequency f_0 and the rest of the parameters after that. The exact form of J_2 when $\boldsymbol{\alpha}$ is optimal becomes

$$\begin{aligned} J_2(\hat{\alpha}_1, \hat{\alpha}_2, f_0) &= (\mathbf{x} - \mathbf{H}\hat{\boldsymbol{\alpha}})^T (\mathbf{x} - \mathbf{H}\hat{\boldsymbol{\alpha}}) \\ &= (\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x})^T (\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}) \\ &= (\mathbf{I}\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x})^T (\mathbf{I}\mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}) \\ &= \mathbf{x}^T (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)^T (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T) \mathbf{x}. \end{aligned}$$

It can be shown that the matrix $(\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)$ is idempotent⁵. Thus,

$$J_2(\hat{\alpha}_1, \hat{\alpha}_2, f_0) = \mathbf{x}^T (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T) \mathbf{x}.$$

In order to minimise J_2 with respect to f_0 , we can equivalently maximise

$$J_3(f_0) = \mathbf{x}^T \mathbf{H} (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$$

The maximisation can be done, for example, using grid search for $f_0 = 0, \dots, 0.5$. We can simplify J_3 in order to have an analytical approximate formula for the estimator \hat{f}_0 . Using the definition of \mathbf{H} ,

$$J_3(f_0) = \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix}^T \begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix}$$

⁵Matrix A is idempotent if $A^2 = A$.

A numerical approximation with $f_0 = 0.2$, $N = 184$, the centre matrix becomes

$$\begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix} = \begin{pmatrix} 92.4045 & 0.2939 \\ 0.2939 & 91.5955 \end{pmatrix},$$

with $f_0 = 0.136$, $N = 121$

$$\begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix} = \begin{pmatrix} 60.4341 & 0.1177 \\ 0.1177 & 60.5659 \end{pmatrix},$$

with $f_0 = 0.423$, $N = 84$

$$\begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix} = \begin{pmatrix} 41.7965 & -0.1868 \\ -0.1868 & 42.2035 \end{pmatrix}.$$

Hence, we can approximate the matrix as

$$\begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix} \approx \begin{pmatrix} \frac{N}{2} & 0 \\ 0 & \frac{N}{2} \end{pmatrix}.$$

The hypothesis holds because

$$\frac{1}{N} \mathbf{c}^T \mathbf{s} = \frac{1}{N} \sum_{n=0}^{N-1} \cos(2\pi f_0 n) \sin(2\pi f_0 n) = \frac{1}{2N} \sum_{n=0}^{N-1} \sin(4\pi f_0 n) \approx 0,$$

and

$$\begin{aligned} \frac{1}{N} \mathbf{c}^T \mathbf{c} &= \frac{1}{N} \sum_{n=0}^{N-1} \cos^2(2\pi f_0 n) = \frac{1}{2N} \sum_{n=0}^{N-1} (1 + \cos(4\pi f_0 n)) \\ &= \underbrace{\frac{1}{2N} \sum_{n=0}^{N-1} 1}_{=\frac{N}{2}} + \underbrace{\frac{1}{2N} \sum_{n=0}^{N-1} \cos(4\pi f_0 n)}_{\approx 0} \approx \frac{N}{2}, \end{aligned}$$

and

$$\begin{aligned} \frac{1}{N} \mathbf{s}^T \mathbf{s} &= \frac{1}{N} \sum_{n=0}^{N-1} \sin^2(2\pi f_0 n) = \frac{1}{2N} \sum_{n=0}^{N-1} (1 - \cos(4\pi f_0 n)) \\ &= \underbrace{\frac{1}{2N} \sum_{n=0}^{N-1} 1}_{=\frac{N}{2}} - \underbrace{\frac{1}{2N} \sum_{n=0}^{N-1} \cos(4\pi f_0 n)}_{\approx 0} \approx \frac{N}{2}. \end{aligned}$$

Thus, J_3 can be approximated by

$$\begin{aligned} J_3(f_0) &= \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix}^T \begin{pmatrix} \frac{2}{N} & 0 \\ 0 & \frac{N}{2} \end{pmatrix} \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix} \\ &= \frac{2}{N} [(\mathbf{c}^T \mathbf{x})^2 + (\mathbf{s}^T \mathbf{x})^2] \\ &= \frac{2}{N} \left[\left(\sum_{n=0}^{N-1} x[n] \cos(2\pi f_0 n) \right)^2 + \left(\sum_{n=0}^{N-1} x[n] \sin(2\pi f_0 n) \right)^2 \right]. \end{aligned}$$

The formula can be still simplified using the observation that the cosine and the sine terms together form a complex exponential, $e^{jx} = \cos(x) + j \sin(x)$, i.e.

$$J_3(f_0) = \frac{2}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi f_0 n} \right|^2.$$

To conclude, the MLE of frequency is obtained by maximising the periodogram over f_0

$$\hat{f}_0 = \underset{f}{\operatorname{argmax}} \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi f n} \right|$$

Once \hat{f}_0 is available, we can proceed by calculating the other parameters:

$$\begin{aligned} \hat{\alpha} &= (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} = \begin{pmatrix} \mathbf{c}^T \mathbf{c} & \mathbf{c}^T \mathbf{s} \\ \mathbf{s}^T \mathbf{c} & \mathbf{s}^T \mathbf{s} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix} \approx \frac{2}{N} \begin{pmatrix} \mathbf{c}^T \mathbf{x} \\ \mathbf{s}^T \mathbf{x} \end{pmatrix} \\ &= \left(\frac{2}{N} \sum_{n=0}^{N-1} x[n] \cos(2\pi \hat{f}_0 n) \right), \end{aligned}$$

from which we get (using Euler's formula in a similar way as before)

$$\begin{aligned} \hat{A} &= \sqrt{\hat{\alpha}^2 + \hat{\phi}^2} = \frac{2}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi \hat{f}_0 n} \right| \\ \hat{\phi} &= \arctan \left(-\frac{\hat{\alpha}_2}{\hat{\alpha}_1} \right) = \arctan \left(-\frac{\sum_{n=0}^{N-1} x[n] \sin(2\pi \hat{f}_0 n)}{\sum_{n=0}^{N-1} x[n] \cos(2\pi \hat{f}_0 n)} \right). \end{aligned}$$

Figure 14.2 shows the algorithm running on four examples.

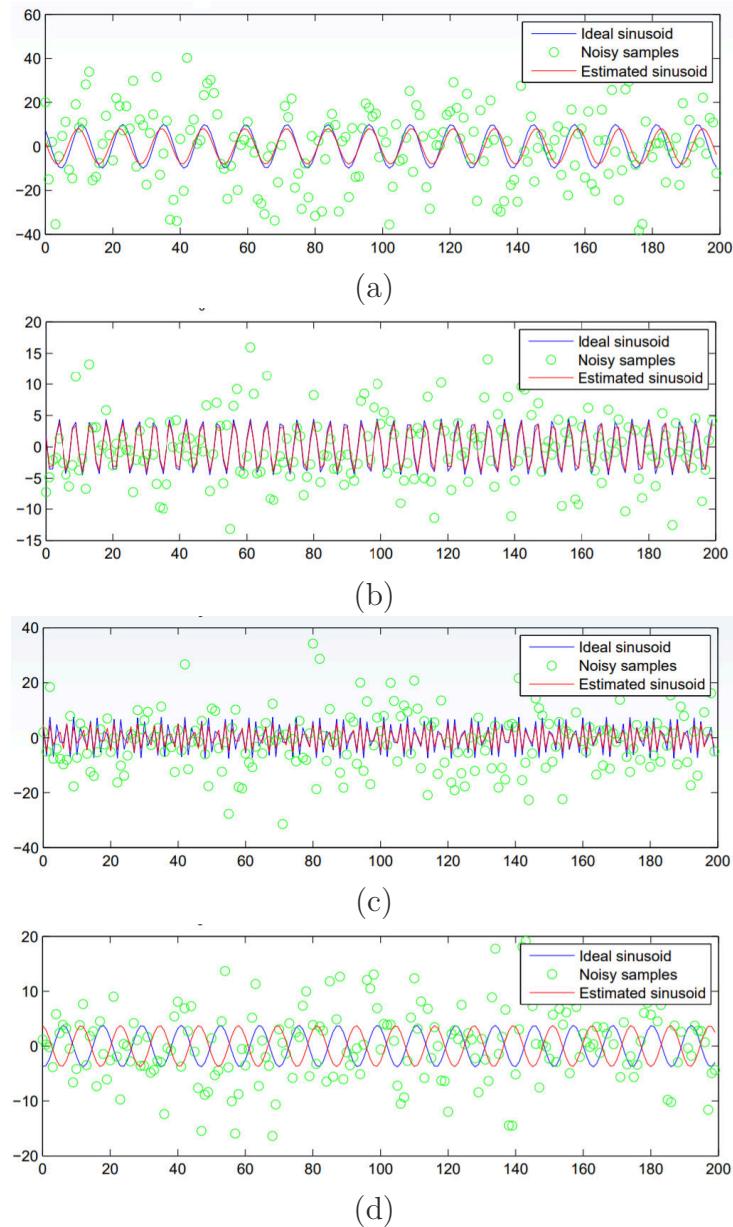


Figure 14.2: (a) Estimation with $f_0 = 0.082$, $A = 9.943$, $\phi = 0.742$. Estimated values are: [0.081, 8.095, 1.316], (b) Estimation with $f_0 = 0.210$, $A = 4.433$, $\phi = 1.172$. Estimated values are: [0.210, 3.913, 1.221], (c) Estimation with $f_0 = 0.424$, $A = 7.460$, $\phi = 1.022$. Estimated values are: [0.425, 5.164, 0.387], (d) Estimation with $f_0 = 0.086$, $A = 3.784$, $\phi = 2.933$. Estimated values are: [0.086, 3.739, 0.111].

Chapter 15

An example of distributed estimation

Consider the following toy example:

- There are two points x_1 and x_2 moving on a line, starting from unknown initial conditions $x_1(0)$ and $x_2(0)$ and subject to a first order discrete dynamic affected by noise:

$$x_i(k+1) = x_i(k) + a_i d_i(k) + b_i \varepsilon_i(k), \quad k = 0, 1, 2, \dots,$$

where $d_i(k)$ is the linear (known) displacement and $\varepsilon_i(k) \sim \mathcal{N}(0, \sigma_i^2)$ is the uncertainty affecting the displacement, supposed to be white.

- Suppose that every $n_i \geq 1$ time steps, each point receives its *absolute* position measurement from an *exogenous* sensor, e.g. GPS, landmark detection (e.g. visual feature on a map or RFID position), anchor-based position, etc., with a certain uncertainty

$$z_i(n_i k) = x_i(n_i k) + \eta_i(n_i k), \quad k = 0, 1, 2, \dots,$$

where $\eta_i(k) \sim \mathcal{N}(0, \xi_i^2)$ is a white Gaussian process.

To properly estimate the point positions, it is possible to carry out an (*intermittent*) *Kalman filter* (provided that the two noise sequences are *uncorrelated*). *Intermittency* takes place whenever $n_i > 1$, i.e. an *update step* every n_i *prediction steps*. In other words, the estimator acts *alternatively* as a *predictor* (when there is no *update step*) or as a *filter*. Recall that the Kalman filter comprises two steps:

- *Prediction step:*

$$\begin{aligned}\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\ P(k+1)^- &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T\end{aligned}$$

- *Update step:*

$$\begin{aligned}S(k+1) &= H(k+1)P(k+1)^-H(k+1)^T + R(k+1) \\ W(k+1) &= P(k+1)^-H(k+1)^TS(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1)^-) \\ P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^-\end{aligned}$$

Therefore:

- *Prediction step:*

$$\begin{aligned}\hat{x}_i(k+1)^- &= \hat{x}_i(k) + a_id_i(k) \\ P_i(k+1)^- &= P_i(k) + b_i^2\sigma_i^2\end{aligned}$$

- *Update step* (every n_i steps):

$$\begin{aligned}S_i(k+1) &= P_i(k+1)^- + \xi_i^2 \\ W_i(k+1) &= \frac{P_i(k+1)^-}{S_i(k+1)} \\ \hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + W_i(k+1)(z_i(k+1) - \hat{x}_i(k+1)^-) \\ P_i(k+1) &= (1 - W_i(k+1))P_i(k+1)^-\end{aligned}$$

Simplifying the terms:

- *Prediction step:*

$$\begin{aligned}\hat{x}_i(k+1)^- &= \hat{x}_i(k) + a_id_i(k) \\ P_i(k+1)^- &= P_i(k) + b_i^2\sigma_i^2\end{aligned}$$

- *Update step* (every n_i steps):

$$\begin{aligned}\hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + \frac{P_i(k+1)^-}{P_i(k+1)^- + \xi_i^2}(z_i(k+1) - \hat{x}_i(k+1)^-) \\ P_i(k+1) &= \frac{\xi_i^2}{P_i(k+1)^- + \xi_i^2}P_i(k+1)^-\end{aligned}$$

From the update equations,

$$\begin{aligned}\hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + \frac{P_i(k+1)^-}{P_i(k+1)^- + \xi_i^2} (z_i(k+1) - \hat{x}_i(k+1)^-) \\ P_i(k+1) &= \frac{\xi_i^2}{P_i(k+1)^- + \xi_i^2} P_i(k+1)^-\end{aligned}$$

it is evident that whenever the sensor readings are very accurate compared to the available *a-priori knowledge*, i.e. $\xi_i^2 \ll P_i(k+1)^-$, we have:

$$\begin{aligned}\hat{x}_i(k+1) &\approx z_i(k+1) \\ P_i(k+1) &\approx \xi_i^2\end{aligned}$$

Moreover, whenever the *a-priori knowledge* is very accurate compared to the sensor readings, i.e. $\xi_i^2 \gg P_i(k+1)^-$, we have:

$$\begin{aligned}\hat{x}_i(k+1) &\approx \hat{x}_i(k+1)^- \\ P_i(k+1) &\approx P_i(k+1)^-\end{aligned}$$

It is worthwhile to note that this condition is temporary, since sooner or later $P_i(k+1)^-$ becomes comparable to ξ_i^2 , i.e. uncertainty increases over time due to *dead reckoning*.

Finally notice that the steady state *estimation uncertainty* $P_i(k)$ can be computed in closed form by computing the fixed point of the following equation:

$$\begin{aligned}P_i(k) &= \frac{\xi_i^2}{P_i(k)^- + \xi_i^2} P_i(k)^- = \\ &= \frac{\xi_i^2}{P_i(k) + b_i^2 \sigma_i^2 + \xi_i^2} (P_i(k) + b_i^2 \sigma_i^2)\end{aligned}$$

Notice that this results hold for a multidimensional system as well, computing the solution of the *Riccati difference equation*.

15.1 A simplified example with relative measurements

Consider now the previous toy example, but with *relative measurements*. Suppose that every $m_i \geq 1$ time steps, the i -th point measures the *relative position* w.r.t. $x_j(k)$ from an *exogenous* sensor, e.g. a LIDAR, a camera, an RGB-D sensor, etc., with a certain uncertainty

$$\Delta_{ij}(m_i k) = (x_j(m_i k) - x_i(m_i k)) + \eta_{ij}(m_i k), \quad k = 0, 1, 2, \dots,$$

where $\eta_{ij}(m_i k) \sim \mathcal{N}(0, \xi_{ij}^2)$ is a white Gaussian process. Therefore, the output function is

$$z_{ij}(m_i k) = \Delta_{ij}(m_i k) - x_j(m_j k) = -x_i(m_i k) + \eta_{ij}(m_i k), \quad k = 0, 1, \dots$$

The overall uncertainty affecting this measurement is then given by $\eta_{ij}(m_i k) + P_j(m_i k)^-$. Notice that we are now implicitly assuming that the agent j is able to send $x_j(m_j k)$ and $P_j(m_i k)$ to the i -th agent. This is the essence of *collaborative localisation*.

Therefore, for the i -th or j -th agent:

- *Prediction step*:

$$\begin{aligned}\hat{x}_i(k+1)^- &= \hat{x}_i(k) + a_i d_i(k) \\ P_i(k+1)^- &= P_i(k) + b_i^2 \sigma_i^2\end{aligned}$$

- *Update step only absolute* (every n_i steps, with $n_i k \neq m_i k$):

$$\begin{aligned}S_i(k+1) &= P_i(k+1)^- + \xi_i^2 \\ W_i(k+1) &= \frac{P_i(k+1)^-}{S_i(k+1)} \\ \hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + W_i(k+1) (z_i(k+1) - \hat{x}_i(k+1)^-) \\ P_i(k+1) &= (1 - W_i(k+1)) P_i(k+1)^-\end{aligned}$$

- For only the i -th agent, *update step only relative* (every m_i steps, with $n_i k \neq m_i k$):

$$\begin{aligned}S_i(k+1) &= P_i(k+1)^- + \xi_{ij}^2 + P_j(k+1)^- \\ W_i(k+1) &= \frac{P_i(k+1)^-}{S_i(k+1)} \\ \hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + W_i(k+1) (z_{ij}(k+1) - (-\hat{x}_i(k+1)^-)) \\ P_i(k+1) &= (1 - W_i(k+1)) P_i(k+1)^-\end{aligned}$$

- For only the i -th agent, *update step with absolute and relative* (when $n_i k = m_i k$):

$$\begin{aligned}S(k+1) &= H_i(k+1) P_i(k+1)^- H_i(k+1)^T + R_i(k+1) \\ W(k+1) &= P_i(k+1)^- H_i(k+1)^T S(k+1)^{-1} \\ \hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + W(k+1) (\bar{z}(k+1) - H_i(k+1) \hat{x}_i(k+1)^-) \\ P_i(k+1) &= (1 - W(k+1) H_i(k+1)) P_i(k+1)^-\end{aligned}$$

where $\bar{z}(k+1) = [z_i(k+1), z_{ij}(k+1)]^T$, $H_i(k+1) = [1, -1]^T$ and $R_i(k+1) = \text{diag}([\xi_i^2, \xi_{ij}^2 + P_j(m_i k)^-])$.

- It is possible to show that, in the spirit of the *Fisher matrix*, using also the relative measure *the uncertainty decreases*.

It is then reasonable that this idea remains valid if *also* the j -th agent measure the i -th agent every m_j time steps. Unfortunately this is not always the case.

The problem can be easily understood by rewriting the complete equations as it would be a centralised Kalman filter:

- Initial conditions: $\hat{x}(0) = [\hat{x}_i(0), \hat{x}_j(0)]^T$ and $P(0) = \text{diag}(P_i(0), P_j(0))$.
- *Prediction step*:

$$\begin{aligned}\hat{x}(k+1)^- &= \begin{bmatrix} \hat{x}_i(k) \\ \hat{x}_j(k) \end{bmatrix} + \begin{bmatrix} a_i & 0 \\ 0 & a_j \end{bmatrix} \begin{bmatrix} d_i(k) \\ d_j(k) \end{bmatrix} \\ P(k+1)^- &= \begin{bmatrix} P_i(k) & 0 \\ 0 & P_j(k) \end{bmatrix} + \begin{bmatrix} b_i^2 \sigma_i^2 & 0 \\ 0 & b_j^2 \sigma_j^2 \end{bmatrix}\end{aligned}$$

Notice that after the prediction step nothing different happens.

- *Update step*: We model that j -th agent *does not* updates its estimate, but the i -th *does* with only the relative measure from the j -th:

$$\begin{aligned}S(k+1) &= H_{ij}(k+1)P(k+1)^-H_{ij}(k+1)^T + R_{ij}(k+1) = \\ &= [-1 \quad 1] P(k+1)^- \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \xi_{ij}^2 = \\ &= P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2 \\ W(k+1) &= P(k+1)^- H_{ij}(k+1)^T S(k+1)^{-1} = \\ &= \frac{1}{P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2} \begin{bmatrix} -P_i(k+1)^- \\ 0 \end{bmatrix}\end{aligned}$$

So we have to force a zero in the second element of the gain matrix of W .

- Moreover:

$$\begin{aligned}\hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1) (z_{ij}(k+1) - H_{ij}(k+1)\hat{x}(k+1)^-) \\ P(k+1) &= (I - W(k+1)H_{ij}(k+1))P(k+1)^- = \\ &= \frac{1}{P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2} \cdot \\ &\quad \cdot \begin{bmatrix} P_j(k+1)^- + \xi_{ij}^2 & P_i(k+1)^- \\ 0 & P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2 \end{bmatrix} \cdot \\ &\quad \cdot \begin{bmatrix} P_i(k+1)^- & 0 \\ 0 & P_j(k+1)^- \end{bmatrix}\end{aligned}$$

So the random variable x_i is *correlated* to the random variable x_j !

As a consequence, the first time that x_j updates its position with a relative measure from x_i , it will consider the measurement H_{ji} as independent but *this is not the case!* This leads to the Kalman filter *inconsistency* in covariances, i.e. *the covariance decreases even if it should not!* Indeed, both are updated with the relative measure of the i -th agent w.r.t. the j -th agent:

$$\begin{aligned} S(k+1) &= H_{ij}(k+1)P(k+1)^{-1}H_{ij}(k+1)^T + R_{ij}(k+1) = \\ &= \begin{bmatrix} -1 & 1 \end{bmatrix} P(k+1)^{-1} \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \xi_{ij}^2 = \\ &= P_i(k+1)^{-1} + P_j(k+1)^{-1} + \xi_{ij}^2 \\ W(k+1) &= P(k+1)^{-1}H_{ij}(k+1)^T S(k+1)^{-1} = \\ &= \frac{1}{P_i(k+1)^{-1} + P_j(k+1)^{-1} + \xi_{ij}^2} \begin{bmatrix} -P_i(k+1)^{-1} \\ P_j(k+1)^{-1} \end{bmatrix} \end{aligned}$$

Moreover:

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}(k+1)^{-1} + W(k+1) (z_{ij}(k+1) - H_{ij}(k+1)\hat{x}(k+1)^{-1}) \\ P(k+1) &= (I - W(k+1)H_{ij}(k+1))P(k+1)^{-1} = \\ &= \frac{1}{P_i(k+1)^{-1} + P_j(k+1)^{-1} + \xi_{ij}^2} \cdot \\ &\quad \cdot \begin{bmatrix} P_j(k+1)^{-1} + \xi_{ij}^2 & P_i(k+1)^{-1} \\ P_j(k+1)^{-1} & P_i(k+1)^{-1} + \xi_{ij}^2 \end{bmatrix} \cdot \\ &\quad \cdot \begin{bmatrix} P_i(k+1)^{-1} & 0 \\ 0 & P_j(k+1)^{-1} \end{bmatrix} \end{aligned}$$

There is a sub-space that is not *observable*, i.e. $x_i + x_j$, hence the associated eigenvalue *grows unbounded* (i.e. *dead reckoning*).

To overcome this problem, a correct *covariance update* should be considered among the agents.

This solution for a team of mobile robots can be found in [31].

Chapter 16

Digital Systems

In modern control systems a *digital discrete-time* system is interfaced to a *continuous-time* physical plant. In a *distributed control system*, the interface takes place for different reasons: a) to share information in digital networks; b) to control plants using digitally implemented controllers; c) to control networked systems. To show the problems related to digitalisation of systems, we will make use of the classic *feedback control* and then we will extend the results on the general case.

Usually *three* actors take place in a discretisation process:

- *Hold*, which is used to convert a discrete-time sequence $\{u_k\}$ into a continuous-time sequence suitable for the physical plant: $u(t) = u_k$, for $k\Delta_t \leq t \leq (k + 1)\Delta_t$, where Δ_t is the sampling time;
- *Physical system*, usually described by a set of linear or nonlinear differential equations;
- *Sampler*, that is the device that creates the discrete-time sequence.

The basic elements and their interconnections, with the continuous time equivalent, are reported in Figure 16.1. Sometimes an *Anti-aliasing filter*, which prepares the continuous-time output signal, is adopted. This is certainly true for *high-sampling rates*, as shown in Figure 16.2 taken from [32].

16.1 Discretisation of SISO Linear Continuous Time Systems

Linear continuous time controllers need to be *discretised* in order to be implemented in a digital system, i.e., by an algorithm. *Discretisation* is the

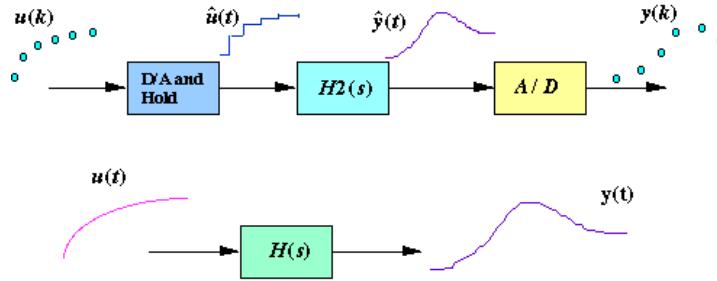


Figure 16.1: Components of a digital system as approximation of a continuous time behaviour.

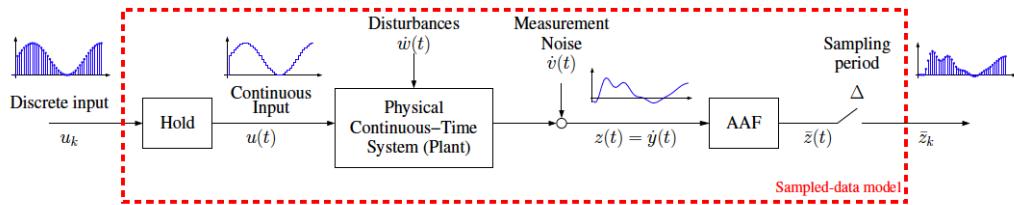


Figure 16.2: Components of a digital system with anti-aliasing filter and noise [32].

process of transferring continuous models into *discrete* models. In the case of a dynamic system, the models are transferred from continuous time to *discrete time*, hence obtaining a *discrete time controller*. Therefore, the system is sampled using a well defined *sampling time* Δ_t . In digital systems, the sampling time is usually lower bounded by feasibility reasons.

There exist different solutions to discretise a system:

1. A trivial solution may be to use a sampling time that is the smaller possible, while keeping the controller continuous and using numeric tools for differential equations.
2. Alternatively, there are different solutions that tries to minimise the *discretisation approximations*, which can be applied to transfer functions $G(s)$ or to *state space* models.

In the latter case, the approximation introduced are strictly related to the discrete time approximation of the integral of a continuous function. Consider the following transfer function and its time representation

$$\frac{Y(s)}{U(s)} = G(s) = \frac{\alpha}{s + \alpha} \Rightarrow \dot{y}(t) + \alpha y(t) = \alpha u(t).$$

Therefore, considering the continuous-time integral, one gets

$$y(t) = \int_0^t -\alpha y(\tau) + \alpha u(\tau) d\tau,$$

that, assuming a sampling time Δ_t for the discrete-time approximation, turns to

$$y(k\Delta_t) = \int_0^{k\Delta_t - \Delta_t} -\alpha y(\tau) + \alpha u(\tau) d\tau + \int_{k\Delta_t - \Delta_t}^{k\Delta_t} -\alpha y(\tau) + \alpha u(\tau) d\tau = y(k\Delta_t - \Delta_t) + A,$$

where

$$A = \int_{k\Delta_t - \Delta_t}^{k\Delta_t} -\alpha y(\tau) + \alpha u(\tau) d\tau$$

is the area under the function $-\alpha y(\tau) + \alpha u(\tau)$ between $k\Delta_t - \Delta_t$ and $k\Delta_t$.

A suitable description in terms of the *shift operator*. As previously depicted, a linear *differential equation* expressed in continuous time is expressed in discrete time by means of a linear *difference equation*, where the difference is supposed with respect to time. An analogous of the *differential-operator* of continuous time equations can be defined for linear difference equations with constant coefficients. The *forward-shift operator* is denoted by q , i.e.,

$$qf(k) = f(k+1),$$

where the sampling period is assumed to be the *time unit*, i.e., if $f(t)$ is sampled every Δ_t seconds and $f(k) \leftrightarrow f(k\Delta_t)$, then

$$qf(k) = f(k+1) \leftrightarrow f(k\Delta_t + \Delta_t).$$

Instead, the *backward-shift operator*, or *delay operator*, is denoted by q^{-1} , i.e.,

$$q^{-1}f(k) = f(k-1).$$

With this definitions, we can consider the different ways in which a continuous differential equation can be discretised, which stems from different integration rules. The first approximation of the integral is given by the *Euler's method*

$$\frac{dx(t)}{dt} \approx \frac{x(t + \Delta_t) - x(t)}{\Delta_t},$$

also known as *forward difference* or *forward rectangular rule*, which is depicted in Figure 16.3-a. An alternative interpretation of the forward difference method comes with the shift operator:

$$\frac{dx(t)}{dt} \approx \frac{x(t + \Delta_t) - x(t)}{\Delta_t} = \frac{q - 1}{\Delta_t} x(t),$$

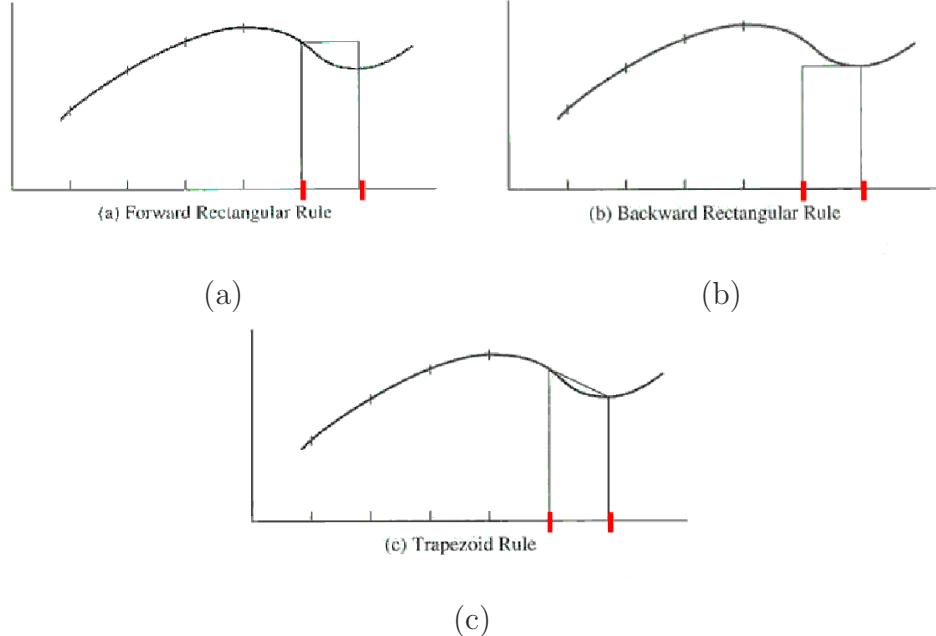


Figure 16.3: (a) Forward rectangular rule, (b) backward rectangular rule and (c) trapezoidal rule.

or, by using the discrete index k ,

$$\frac{dx(k)}{dt} \approx \frac{q-1}{\Delta_t} x(k).$$

Therefore, by recalling the fact that s corresponds to the differential operator, one gets

$$\frac{dx(k)}{dt} \approx \frac{q-1}{\Delta_t} x(k) \Rightarrow s \leftarrow \frac{q-1}{\Delta_t},$$

that is, *it is sufficient to substitute s in $G(s)$ with the proper shift operator equation.*

Remark 30. Notice that the δ -operator, i.e., $\delta = \frac{q-1}{\Delta_t}$, is not given only by the Euler integration rule. The δ -operator simply re-parametrises discrete models through a transformation of general validity.

The second method, the inverse of the previous, is given by the *backward difference* or *backward rectangular rule* (Figure 16.3-b)

$$\frac{dx(t)}{dt} \approx \frac{x(t) - x(t - \Delta_t)}{\Delta_t} \quad \text{or} \quad \frac{dx(t + \Delta_t)}{dt} \approx \frac{x(t + \Delta_t) - x(t)}{\Delta_t}.$$

Using the shift operator

$$\frac{dx(t + \Delta_t)}{dt} \approx \frac{x(t + \Delta_t) - x(t)}{\Delta_t} = \frac{q - 1}{\Delta_t}x(t),$$

or, by using the discrete index k ,

$$\frac{dx(k + 1)}{dt} = \frac{dqx(k)}{dt} \approx \frac{q - 1}{\Delta_t}x(k).$$

Therefore, by recalling the fact that s corresponds to the differential operator, one gets

$$\frac{dqx(k)}{dt} \approx \frac{q - 1}{\Delta_t}x(t) \Rightarrow sq \leftarrow \frac{q - 1}{\Delta_t},$$

and, more easily,

$$s \leftarrow \frac{q - 1}{q\Delta_t},$$

that is again sufficient to substitute s in $G(s)$ with the proper shift operator equation.

Finally, the third and more accurate approximation of the integral is given by the *trapezoidal rule* (Figure 16.3-c)

$$\int_{t_1}^{t_2} x(t)dt = (t_2 - t_1) \frac{x(t_2) - x(t_1)}{2}.$$

The trapezoidal rule can be interpreted as the integral of the *time derivatives* using the shift operator

$$\frac{1}{2} \left(\frac{dx(t + \Delta_t)}{dt} + \frac{dx(t)}{dt} \right) \approx \frac{q - 1}{\Delta_t}x(t),$$

or, using the discrete index k ,

$$\frac{1}{2} \left(\frac{dqx(k)}{dt} + \frac{dx(k)}{dt} \right) \approx \frac{q - 1}{\Delta_t}x(k).$$

Therefore, by recalling the fact that s corresponds to the differential operator, one gets

$$\frac{1}{2} \left(\frac{dqx(k)}{dt} + \frac{dx(k)}{dt} \right) \approx \frac{q - 1}{\Delta_t}x(k) \Rightarrow \frac{qs + s}{2} \leftarrow \frac{q - 1}{\Delta_t}$$

and, more easily,

$$s \leftarrow \frac{2}{\Delta_t} \frac{q - 1}{q + 1}.$$

Example 38. Consider the following transfer function and its time representation

$$\frac{Y(s)}{U(s)} = G(s) = \frac{\alpha}{s + \alpha} \Rightarrow \dot{y}(t) + \alpha y(t) = \alpha u(t).$$

Using the forward rectangular rule, one gets

$$\frac{q - 1}{\Delta_t} y(k) + \alpha y(k) = \alpha u(k) \Rightarrow y(k + 1) = (1 - \alpha \Delta_t) y(k) + \alpha \Delta_t u(k),$$

that is the difference equation that expresses the discrete-time approximation of $Y(s) = G(s)U(s)$. Notice that the initial condition for $y(k)$ is needed. The same result can be obtained by simply write the equation in s and than substituting $Y(s)$ with $y(k)$, $U(s)$ with $u(k)$ and s with $\frac{q-1}{\Delta_t}$. This is very useful as soon as the transfer function has terms of the type s^n .

Instead, using the backward rectangular rule, one gets

$$\frac{q - 1}{q \Delta_t} y(k) + \alpha y(k) = \alpha u(k) \Rightarrow y(k + 1) = \frac{y(k)}{1 + \alpha \Delta_t} + \frac{\alpha \Delta_t u(k + 1)}{1 + \alpha \Delta_t}.$$

Notice that the initial condition for $y(k)$ and for $u(k)$ is needed. The same result can be obtained by simply write the equation in s and than substituting $Y(s)$ with $y(k)$, $U(s)$ with $u(k)$ and s with $\frac{q-1}{q \Delta_t}$. This is very useful as soon as the transfer function has terms of the type s^n .

Finally, using the trapezoidal rule, one gets

$$\frac{2}{\Delta_t} \frac{q - 1}{q + 1} y(k) + \alpha y(k) = \alpha u(k) \Rightarrow y(k + 1) = \frac{2 - \alpha \Delta_t}{2 + \alpha \Delta_t} y(k) + \frac{\alpha \Delta_t}{2 + \alpha \Delta_t} u(k) + \frac{\alpha \Delta_t}{2 + \alpha \Delta_t} u(k + 1).$$

Notice that the initial condition for $y(k)$ and for $u(k)$ is needed. The same result can be obtained by simply write the equation in s and than substituting $Y(s)$ with $y(k)$, $U(s)$ with $u(k)$ and s with $\frac{2}{\Delta_t} \frac{q-1}{q+1}$. This is very useful as soon as the transfer function has terms of the type s^n . \square

A quite simple way to automatically discretise the system is to:

1. Substitute to the variable s in $G(s)$ the function of the variable q ;
2. Simplify the expressions of the numerator and the denominator;
3. The denominator will be multiplied by $y(k)$, the numerator by $u(k)$;
4. The coefficients of the two polynomials in q are the coefficients of $y(k)$, i.e., $(a_n q^n + a_{n-1} q^{n-1} + \dots + a_1 q + a_0) y(k)$ turns to $a_n y(k+n) + a_{n-1} y(k+n-1) + \dots + a_1 y(k+1) + a_0 y(k)$. The same for $u(k)$.

Usually, the shift operator is a convenient way to express the complex variable z , the variable of the *\mathcal{Z} -transform*, that is the discrete time counterpart of the Laplace transform. As soon as the \mathcal{Z} -transform is considered, the approximation of the trapezoidal rule is also called *Tustin's approximation* or *bilinear transformation*. The Tustin's approximation is derived by the approximation of the continuous time delay of the sampling time Δ_t , i.e.,

$$z = e^{s\Delta_t} \approx \frac{1 + s\Delta_t/2}{1 - s\Delta_t/2},$$

which is the *Padé approximant* of the first order.

The most accurate discretisation algorithm among the previous is the one given by the trapezoidal rule. Indeed, it gives the best approximation of the integral operator.

More precisely:

- The forward difference may generate an unstable discrete-time system starting from a stable continuous-time system;
- The backward difference may generate stable discrete-time systems starting from unstable continuous-time systems;
- The trapezoidal rule maps continuous-time stable systems into discrete-time stable systems and unstable into unstable systems.

16.2 Forced and Unforced Response of a Linear System

Consider a time-invariant continuous-time linear system:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ z(t) &= Cx(t)\end{aligned}$$

The model can be represented in incremental form by

$$\begin{aligned}dx(t) &= Ax(t)dt + Bu(t)dt \\ z(t)dt &= Cx(t)dt\end{aligned}$$

The idea is to express the evolution of the system in the *inter-sampling*, i.e., during the time Δ_t . To this end, due to the linearity of system, we can express the evolution of the system using the *unforced response*, i.e., when the input is zero and the system evolves from a generic initial state $x(0)$, and

the *forced response*, i.e., when the initial condition is $x(0) = 0$ and the input is generic. To derive the associated solutions, we will examine the results of the scalar difference equation

$$\dot{x}(t) = ax(t) + bu(t)$$

and then extend such results to the multidimensional case.

The *homogeneous* solution, i.e., the *unforced response*, of the differential scalar equation is simply given by

$$x_u(t) = e^{at}x(0),$$

where using the Taylor expansion around $t_0 = 0$, can be represented as

$$x_u(t) = \sum_{i=0}^{+\infty} \frac{a^i t^i}{i!} x(0).$$

Similarly, we can assume that the *unforced response* for the multidimensional case is given by

$$x_u(t) = \left(I + At + \frac{A^2 t^2}{2} + \dots \right) x(0) = \sum_{i=0}^{+\infty} \frac{A^i t^i}{i!} x(0).$$

Now, we have to show that such a solution is effectively a solution of the differential equation $\dot{x}_u(t) = Ax_u(t)$. It is easy to see that

$$\begin{aligned} \dot{x}_u(t) &= \left(0 + A + At + \frac{A^2 t^2}{2} + \dots \right) x(0) = \\ &= A \left(I + At + \frac{A^2 t^2}{2} + \dots \right) x(0) = Ax_u(t), \end{aligned}$$

as desired. Due to the similarity here defined, we can state that

$$e^{At} = \sum_{i=0}^{+\infty} \frac{A^i t^i}{i!},$$

and then

$$x_u(t) = e^{At}x(0) = \Phi(t)x(0),$$

where $\Phi(t)$ is in general called the *state transition matrix*.

Example 39. As an exercise, compute the unforced response of

$$\begin{aligned} \dot{x}_1 &= x_1 \\ \dot{x}_2 &= -2x_1 - x_2 + u \end{aligned}$$

when the initial conditions are $x_1(0) = 1$ and $x_2(0) = 2$. \square

It is now necessary to compute the *particular* solution of the differential equation, i.e., *forced response*, to derive the complete output of the system, which will be given by the *superposition principle*, i.e., the sum of forced and unforced responses. There are several ways to compute this. One possibility is to compute *directly* the sum of the forced and unforced responses. Consider again the scalar system

$$\dot{x}(t) = ax(t) + bu(t) \Rightarrow \dot{x}(t) - ax(t) = bu(t).$$

Since

$$\frac{de^{-at}x(t)}{dt} = e^{-at}(\dot{x}(t) - ax(t)),$$

it follows that

$$\frac{de^{-at}x(t)}{dt} = e^{-at}bu(t).$$

Integrating both sides

$$\int_0^t \frac{de^{-a\tau}x(\tau)}{d\tau} d\tau = e^{-at}x(t) - x(0) = \int_0^t e^{-a\tau}bu(\tau)d\tau,$$

that finally yields to

$$x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau.$$

As a consequence, we have that the *forced response* is given by

$$x_f(t) = \int_0^t e^{a(t-\tau)}bu(\tau)d\tau.$$

Following the same steps for the multidimensional case, for which we notice that $e^{0t} = I$ and $(e^{-At})^{-1} = e^{At}$, we get

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau.$$

Notice how the integral of the forced response is the *convolution integral*.

Example 40. As an exercise, compute the response of the system

$$\begin{aligned}\dot{x}_1 &= x_1 \\ \dot{x}_2 &= -2x_1 - x_2 + u\end{aligned}$$

when the initial conditions are $x_1(0) = 1$ and $x_2(0) = 2$ and the input $u(t) = 10$, $\forall t \geq 0$. \square

16.3 Discretisation of Linear Continuous Time Systems

Suppose that the system is sampled through a *sample and hold*, aka *zero order hold* (ZOH), i.e.,

$$u(t) = u_k \text{ for } k\Delta_t \leq t \leq (k+1)\Delta_t,$$

where Δ_t is the sampling time. Hence, the discrete-time system dynamic is given by

$$\begin{aligned} x_{k+1} &= A_s x_k + B_s u_k \\ z_k &= C x_k \end{aligned}$$

where $x_k = x(k\Delta_t)$ and

$$A_s = e^{A\Delta_t} \text{ and } B_s = \int_0^{\Delta_t} e^{A\tau} B d\tau.$$

If the Euler integration rule is adopted, the dynamic becomes:

$$\begin{aligned} x_{k+1} &= A_e x_k + B_e u_k \\ z_k &= C x_k \end{aligned}$$

where $x_k = x(k\Delta_t)$ and

$$A_e = I + A\Delta_t \text{ and } B_e = B\Delta_t.$$

Therefore, an error of the order of Δ_t^2 is then unavoidable, since $e^{A\Delta_t} \approx I + A\Delta_t$. This approach is sometimes called *simple derivative replacement* (SDR).

16.4 An Example for Nonlinear Discretisation

As an example of discretisation for nonlinear systems, let us consider the case of the unicycle vehicle, depicted in Figure 16.4. Using the discretised model, it is possible to make a prediction about the position of the vehicle at each time instant Δ_t . The input of the discrete model is given by the incremental encoders on the left and right wheels. Let us recall the unicycle like vehicle kinematic model

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}.$$

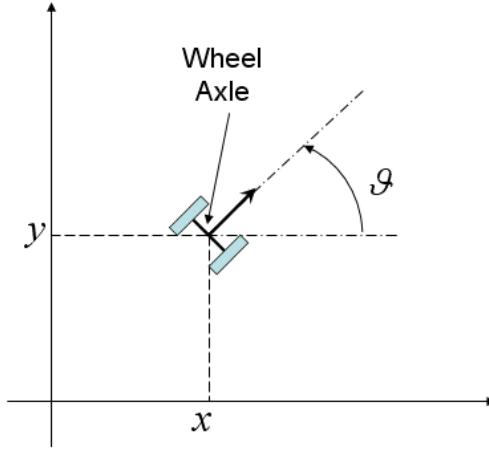


Figure 16.4: Unicycle like vehicle generalised coordinates.

From a practical view-point, the vehicle is usually controlled using the rotational velocities (ω_l, ω_r) of the left and right wheel respectively. To obtain the velocities of the kinematic model, the following relations are used

$$\begin{cases} v = \frac{R}{2}(\omega_r + \omega_l) \\ \omega = \frac{R}{D}(\omega_r - \omega_l) \end{cases}$$

where R is the wheels radius and D is the length of the wheel axle. Since the encoder values represent the angular position η of each wheel and assuming realistically that we sample the encoders with a period of Δt , we can apply the Euler integration to approximate the time derivatives, i.e.,

$$\begin{cases} \omega_r \approx \frac{\eta_r(t+\Delta t) - \eta_r(t)}{\Delta t} \\ \omega_l \approx \frac{\eta_l(t+\Delta t) - \eta_l(t)}{\Delta t} \end{cases} \Rightarrow \begin{cases} \omega_r \Delta t \approx \eta_r(t + \Delta t) - \eta_r(t) \\ \omega_l \Delta t \approx \eta_l(t + \Delta t) - \eta_l(t) \end{cases}$$

Hence, we have

$$\begin{cases} v \Delta t \approx \frac{R}{2}(\eta_r(t + \Delta t) - \eta_r(t) + \eta_l(t + \Delta t) - \eta_l(t)) \\ \omega \Delta t \approx \frac{R}{D}(\eta_r(t + \Delta t) - \eta_r(t) - \eta_l(t + \Delta t) + \eta_l(t)) \end{cases}$$

Similarly and finally

$$\dot{\mathbf{q}} \Delta t = \begin{bmatrix} \dot{x} \Delta t \\ \dot{y} \Delta t \\ \dot{\theta} \Delta t \end{bmatrix} = \begin{bmatrix} x(t + \Delta t) - x(t) \\ y(t + \Delta t) - y(t) \\ \theta(t + \Delta t) - \theta(t) \end{bmatrix} = \begin{bmatrix} \cos \theta(t) & 0 \\ \sin \theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \Delta t \\ \omega \Delta t \end{bmatrix}$$

from which we have the sampled, reconstructed dynamic given by

$$\begin{bmatrix} x(t + \Delta t) \\ y(t + \Delta t) \\ \theta(t + \Delta t) \end{bmatrix} = \begin{bmatrix} x(t) + \cos(\theta(t))v\Delta t \\ y(t) + \sin(\theta(t))v\Delta t \\ \theta(t) + \omega\Delta t \end{bmatrix}$$

16.4.1 The Application of the Extended Kalman Filter

The estimator here proposed reconstruct the actual position of the robot given the *history* of the encoder values. It is trivial that such an algorithm can be successfully implemented whenever the initial condition for the robot, i.e., $[x(0), y(0), \theta(0)]$, and for the encoders, i.e., $[\eta_r(0), \eta_l(0)]$, are known or chosen by the designer. Notice that the wheel encoders are the only sources of information, hence the high inaccuracy of the *dead-reckoning* for robot position reconstruction. Adding more sensors, all the additional data can be coherently fused together using *Extended Kalman Filters* to get a finer estimate of the state \mathbf{q} .

16.5 Some issues for discretisation

Sometimes it is useful to have the dynamic of the system in the inter-sample, i.e., with a time that is more fine grained than the sampling time Δ_t adopted in the S&H. This is very convenient when, for example, the time of arrival of messages, e.g., *control signals* $u(k)$, are ruled by an external random process. For example, this is what happens when the control signals are computed by a platform with unpredictable time of execution or when there is a network in between. Hence, to have a picture of the inter-sampling behaviour, it is needed just to sample the system with a *finer* sampling time δ_t , i.e., $\Delta_t = n\delta_t$, with $n > 1$. To recover the original dynamic, it is sufficient to compute the forced and unforced response of the system for the *discrete time*.

It is possible to sample the system using a different approach for sampling, i.e., modifying the strategy from the classic S&H *time based* to an *event based* strategy. We already know that this is very advantageous for *networked* control systems. This is also of relevance for *soft real-time* systems. The basic idea can be restated by simply saying that the sampling changes from the *Riemann sampling* (time-based) to the *Lebesgue sampling* (event-based), both reported in Figure 16.5.

16.5.1 Noise discretisation

Unfortunately, the system we are dealing with are *stochastic* linear systems, hence the dynamic will be given by

$$\dot{x}(t) = Ax(t) + Bu(t) + G\tilde{\nu}(t),$$

where $\tilde{\nu}(t)$ is supposed to be white with zero mean and possibly time varying variance $V(t)$, i.e.,

$$E\{\tilde{\nu}(t)\} = 0 \text{ and } E\{\tilde{\nu}(t)\tilde{\nu}(\tau)^T\} = V(t)\delta(t - \tau),$$

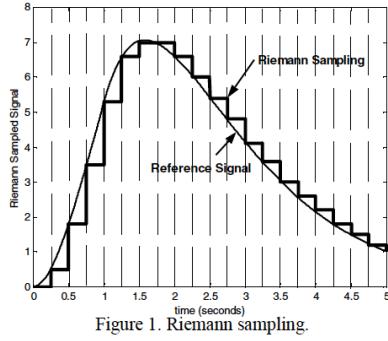


Figure 1. Riemann sampling.

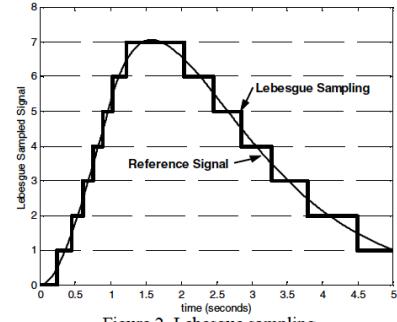


Figure 2. Lebesgue sampling.

Figure 16.5: Riemann (left) and Lebesgue (right) samplers [33].

where $\delta(\cdot)$ is the standard *Dirac delta function*. Due to the discretisation, the system is rewritten with the difference equation

$$x(k+1) = A_D x(k) + B_D u(K) + \nu(k),$$

being $\nu(k)$ is the discretised version of the continuous noise $\tilde{\nu}(t)$ weighted by the matrix G . Following the same arguments of the previous analysis, we may rewrite

$$x(t_1) = e^{A(t_1-t_0)} x(t_0) + \int_{t_0}^{t_1} e^{A(t_1-\tau)} [B(\tau)u(\tau) + G(\tau)\tilde{\nu}(\tau)] d\tau,$$

from which the *Markov property* is further highlighted, i.e.,

$$f(x(t_1)|x_{(-\infty, t_0)}, u_{(t_0, t_1)}, \tilde{\nu}_{(t_0, t_1)}) = f(x(t_1)|x(t_0), u_{(t_0, t_1)}, \tilde{\nu}_{(t_0, t_1)}).$$

From the previous analysis and the linearity of the integral operator, it follows immediately that

$$\nu(k) = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)} G(\tau) \tilde{\nu}(\tau) d\tau.$$

Let us derive the *stochastic description* of this new discrete time random variable. For the mean value, it is immediate (due to the linearity of the integral operator) to show that

$$\mathbb{E}\{\nu(k)\} = 0.$$

For the variance, the analysis is a little bit more complicated

$$\begin{aligned}
& \mathbb{E} \{ \nu(k) \nu(j)^T \} = \\
&= \mathbb{E} \left\{ \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau_k)} G(\tau_k) \tilde{\nu}(\tau_k) d\tau_k \int_{t_j}^{t_{j+1}} \tilde{\nu}(\tau_j)^T G(\tau_j)^T e^{A(t_{j+1}-\tau_j)^T} d\tau_j \right\} = \\
&= \int_{t_k}^{t_{k+1}} \int_{t_j}^{t_{j+1}} e^{A(t_{k+1}-\tau_k)} G(\tau_k) \mathbb{E} \{ \tilde{\nu}(\tau_k) \tilde{\nu}(\tau_j)^T \} G(\tau_j)^T e^{A(t_{j+1}-\tau_j)^T} d\tau_k d\tau_j = \\
&= \int_{t_k}^{t_{k+1}} \int_{t_j}^{t_{j+1}} e^{A(t_{k+1}-\tau_k)} G(\tau_k) V(\tau_k) \delta(\tau_k - \tau_j) G(\tau_j)^T e^{A(t_{j+1}-\tau_j)^T} d\tau_k d\tau_j.
\end{aligned}$$

Hence

$$\begin{aligned}
& \mathbb{E} \{ \nu(k) \nu(j)^T \} = \\
&= \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau_k)} G(\tau_k) V(\tau_k) G(\tau_k)^T e^{A(t_{k+1}-\tau_k)^T} d\tau_k \delta_{k,j}
\end{aligned}$$

where $\delta_{k,j}$ is the *Kronecker delta* function. In practice, the larger is the sampling period, the larger would be the variance of the *discrete white noise* injected in the discrete time system.

Chapter 17

Introduction to Linear Consensus

What we have seen so far are *Distributed Control Systems* in which, essentially, local controllers are networked to ensure a correct course of actions. Usually, a *centralised supervisory controller* modifies the executions of the local controllers modifying their behaviours, e.g., changing their reference or tuning parameters, or aggregating the measures, e.g., Kalman filter. Nevertheless, there is the possibility to avoid the presence of a *master* ruling the executions and, hence, having a fair distribution of the decisions among the local controllers. This is the paradigm of the *networked control systems*. Examples of networked control systems are:

- Wireless sensor networks;
- Swarm robotics;
- Communication networks;
- Next generation smart grids;
- Water distribution;
- Ground or air traffic control.

Furthermore, examples taken from various scientific contexts:

- *Statistical mechanics*: The local interactions of millions of particles may yield simple thermodynamics laws describing the global behaviour.

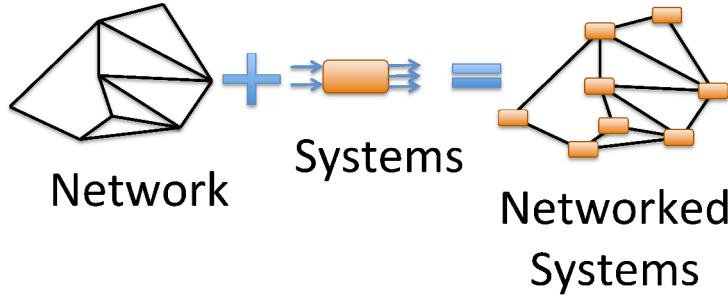


Figure 17.1: The graph representation of the networked control system: a network whose nodes are dynamic systems.

- *Cooperation*: Simple global behaviours are obtained from local interactions. One example is *flocking*: collective animal behaviour is given by the motion of a large number of coordinated individuals.
- *Social networks*: Individual social interactions produce global social phenomena.
- *Economic networks*: Economic entities take part to the global market producing global behaviours, e.g., world wide economic crisis.

The objective is the study of the behaviour of complex systems constituted by the *interconnection of many units* which are themselves dynamical systems (see Figure 17.1). The behaviour of these systems will depend on the *dynamics of the units* and on the *interconnection topology*. In general, the main purpose is to study how the topology and the dynamic systems produce the *global dynamics*. In this course, we will limit the analysis to *distributed estimation* and to *distributed control* for very specific (and simple) systems.

In general, two main problems can be tackled for this class of systems:

- *Distributed estimation*: Using the “opinion”, e.g., local measurement, of each system, construct a global estimate of the quantity of interest through *messages exchange*.
- *Distributed control*: Using the “local understanding” upon the relative configuration of each system, e.g., local measurement, solve the problem of coordinated motions, e.g., rendezvous, deployment, etc., through *messages exchange*.

One of the most promising tools are the *linear consensus algorithms*, which are simple distributed algorithms to compute averages of local quantities. These algorithms stem from the analysis of *Markov chains* and have

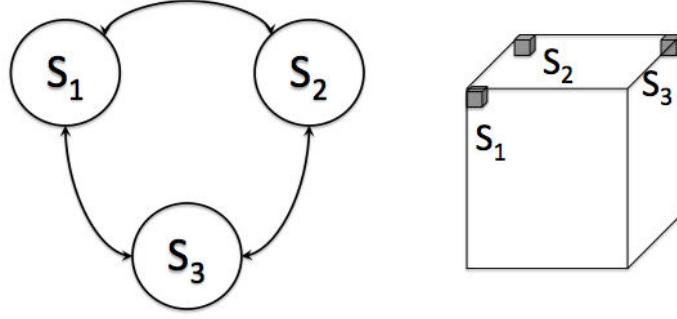


Figure 17.2: Sensors communication links and sensors deployment.

been applied in the 80s to the computer science community for *load balancing* [34]. In the recent years, they have been adapted and applied to cooperative coordination of multi-agent systems. Alternative naming conventions are agreement problems (social networks, economy); synchronisation (statistical mechanics); society of robots, rendezvous, coordinated motion (robotics).

17.1 A Practical Example

Let us consider two sensors measuring the constant quantity x affected by the same zero mean noise, i.e.

$$z_1 = x + \varepsilon \text{ and } z_2 = x + \varepsilon$$

The Least Squares solution would be the arithmetic mean

$$\hat{x}^{LS} = \frac{\sum_{i=1}^n z_i}{n} = x + \frac{\sum_{i=1}^n \varepsilon}{n}.$$

Let us now consider the problem to compute the arithmetic mean in a *distributed way*. For example, consider a network comprising $n = 3$ sensors measuring the temperature of a given environment (Figure 17.2). The communication is bidirectional between the nodes and each node has visibility of all the other nodes of the network. Once S_1 receives the temperature message from S_2 and S_3 (with a proper *timestamp*), it can compute the mean. Similarly, S_2 and S_3 .

A convenient way to represent this distributed operation is to collect the value measured by each sensor, say $x_i(k)$, in a column vector, i.e. $x(k) =$

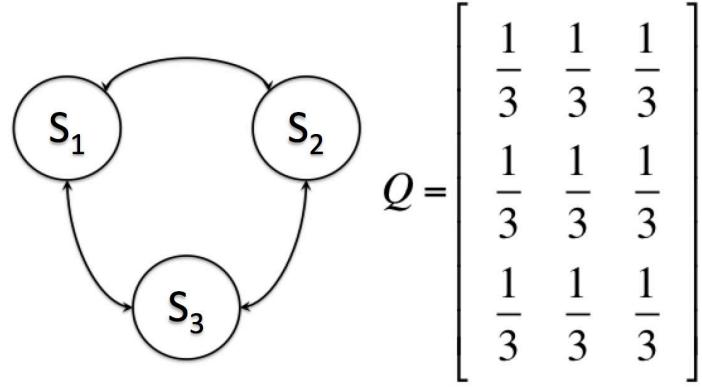


Figure 17.3: Sensors communication links and algebraic representation.

$[x_1(k), x_2(k), x_3(k)]^T$, and then write the temperature update with

$$x(k+1) = Qx(k) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} = \begin{bmatrix} \frac{\sum_{j=1}^3 x_j(k)}{3} \\ \frac{\sum_{j=1}^3 x_j(k)}{3} \\ \frac{\sum_{j=1}^3 x_j(k)}{3} \end{bmatrix},$$

where Q is called the *transition matrix* (see Figure 17.3). It is easy to see that being $x(0)$ the value of the temperature at the beginning, after one *protocol* iteration, i.e. after three broadcasts, one for each node S_i , the *agreement* is reached, i.e. *each node has the same mean value stored*. Moreover, the idea can be extended to an arbitrary number of nodes n . Notice that this algorithm is *distributed* since each node reaches the mean by simply knowing its own value and the value of its *neighbours*, i.e. *nodes that can share the information with node S_i* . To highlight the distributed nature of the protocol, the update equation can be rewritten as

$$\begin{aligned} x_i(k+1) &= \frac{\sum_{j=1}^n x_j(k)}{n} = \frac{1}{n} x_i(k) + \sum_{j=1, j \neq i}^n \frac{1}{n} x_j(k) = \\ &= \left(1 - \frac{n-1}{n}\right) x_i(k) + \sum_{j=1, j \neq i}^n \frac{1}{n} x_j(k) = \\ &= x_i(k) + \sum_{j=1}^n \frac{1}{n} (x_j(k) - x_i(k)) = \\ &= x_i(k) + \sum_{j=1}^n q_{ij} (x_j(k) - x_i(k)) \end{aligned}$$

Let us make a closer look to this distributed estimation protocol. Let $x(0)$ be the value of the temperature measured by three sensors. After one iteration of the protocol, one has:

$$x(1) = Qx(0) = \begin{bmatrix} \frac{\sum_{j=1}^3 x_j(0)}{3} \\ \frac{\sum_{j=1}^3 x_j(0)}{3} \\ \frac{\sum_{j=1}^3 x_j(0)}{3} \end{bmatrix} = \frac{\sum_{j=1}^3 x_j(0)}{3} \mathbf{1} = \beta \mathbf{1},$$

where $\mathbf{1}$ is a vector of ones. What happens when another round of messages is broadcasted? For $x(2)$ we have:

$$x(2) = Qx(1) = \beta Q \mathbf{1} = \beta \mathbf{1}.$$

It then follows that $\beta \mathbf{1}$ remains constant, no matter what is the number of messages exchanged. In fact:

$$\begin{aligned} x(1) &= Qx(0) = \beta \mathbf{1}, \\ x(2) &= Qx(1) = \beta \mathbf{1}, \\ x(3) &= Qx(2) = \beta \mathbf{1}, \\ &\vdots \\ x(k+1) &= Qx(k) = \beta \mathbf{1}, \\ &\vdots \end{aligned}$$

Therefore, $\beta \mathbf{1}$ is an *equilibrium point* of the distributed protocol. And this is true *for all* the possible values of the scalar β . In other words, if all the sensors measure the same value at the beginning, i.e. $x_i(0) = \beta \forall i$, that is already the arithmetic mean. Technically speaking, this property holds since $\mathbf{1}$ is the right *eigenvector* of Q associated to the *eigenvalue* 1, i.e.

$$Q\mathbf{1} = \mathbf{1}.$$

This is a fundamental property of the *stochastic matrices*.

Definition 71 (Stochastic matrix). A *stochastic matrix* is a matrix $Q \in \mathbb{R}^{n \times n}$ if and only if $q_{ij} \geq 0$ and $\sum_{j=0}^n q_{ij} = 1$, $\forall i$, i.e. the *row sum* is equal to 1.

This is the necessary property for the existence of a stable agreement (e.g. the arithmetic mean) in *linear consensus theory*. Hence, an agreement is reached if the protocol matrix Q is a *stochastic matrix* (plus other technical requirements on matrix aperiodicity and irreducibility).

Since the value stored in each node evolves according to the previous value stored, i.e. *discrete dynamic*, we may notice that

$$\begin{aligned} x(1) &= Qx(0) \\ x(2) &= Qx(1) = Q^2x(0), \\ x(3) &= Qx(2) = Q^2x(1) = Q^3x(0), \\ &\vdots \\ x(k) &= Qx(k-1) = Q^kx(0), \\ &\vdots \end{aligned}$$

Hence, if Q^k is a matrix with *all equal rows* from some k , then after k rounds of the protocol, the system reaches an agreement, aka a *consensus*. Q^k is the *k -step transition matrix*, i.e. the transition matrix representing the aggregates, one shot transition from $x(0)$ to $x(k)$. This is trivial to show, since in that case the entries of $x(k)$, i.e. the values of the nodes after k rounds of the protocol, have all the same values.

The fact that Q^k is a matrix with *all equal rows*, for some k , holds for a *stochastic matrix*. Moreover, the product of two stochastic matrices *is still a stochastic matrix*, hence if Q is a *stochastic matrix*, then Q^k is a *stochastic matrix*. In general nothing can be said about the reached equilibrium, i.e. what is the value of β . In other words, further properties should be verified to ensure that $\beta = \frac{\sum_{j=1}^n x_j(0)}{n}$, i.e. the arithmetic mean.

Let us consider now the case in which there is not complete visibility among nodes, e.g. sensor S_2 does not receive the information from sensor S_3 and vice-versa (see Figure 17.4). In this case the matrix Q changes accordingly. One idea can still be to compute the mean among the neighbouring nodes, as represented by the Q matrix in Figure 17.4. Using the matrix product rule, for a sufficiently large k , one has for the selected Q that

$$\lim_{k \rightarrow +\infty} Q^k = \begin{bmatrix} a_1 & a_2 & a_2 \\ a_1 & a_2 & a_2 \\ a_1 & a_2 & a_2 \end{bmatrix}$$

with $a_1 > a_2$. Hence an agreement is reached, i.e. all the nodes will have the same quantities, but that is not the arithmetic mean, since

$$x(k) = Q^k x(0) \Rightarrow x_i(k) = a_1 x_1(0) + a_2 x_2(0) + a_2 x_3(0).$$

To reach an *average consensus*, that is convergence towards the mean, the matrix Q should be *doubly stochastic*.

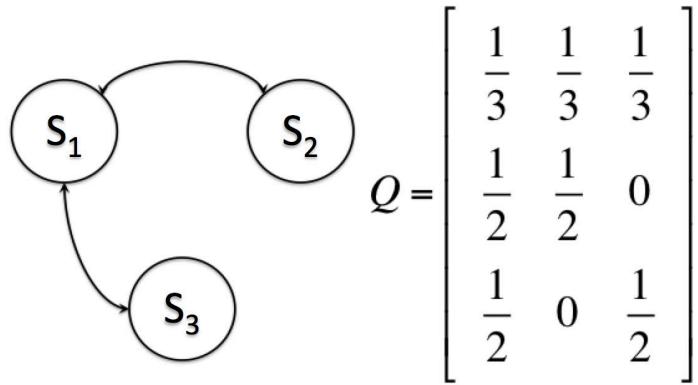


Figure 17.4: Sensors communication links and algebraic representation without complete visibility.

Definition 72 (Doubly-stochastic matrix). A stochastic matrix Q is **doubly-stochastic** iff both $\sum_{j=0}^n q_{ij} = 1$, $\forall i$, and $\sum_{i=0}^n q_{ij} = 1$, $\forall j$.

Clearly, if Q is a stochastic **symmetric** matrix, i.e., $Q = Q^T$, then it is also doubly-stochastic. So, how we can derive a **doubly stochastic matrix** if S_2 does not see S_3 ?

Let us find the answer in a general way. Let us define a generic matrix Q with seven unspecified parameters (two entries are 0, see Figure 17.4), i.e.

$$Q = \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{2,1} & q_{2,2} & 0 \\ q_{3,1} & 0 & q_{3,3} \end{bmatrix},$$

and let us identify the value of the unknown parameters $q_{ij} \geq 0$ with the constraints that the row and the column sum should be equal to 1, and the limiting condition that

$$\lim_{k \rightarrow +\infty} Q^k = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}. \quad (17.1)$$

We then end up with the following form

$$Q = \begin{bmatrix} x+y-1 & 1-x & 1-y \\ 1-x & x & 0 \\ 1-y & 0 & y \end{bmatrix}$$

for $0 < x < 1$ and $0 < y < 1$, with $x+y \geq 1$. In other words, for any value of x and y , we have a **doubly stochastic matrix**, that, for a sufficiently large k , converging to the averaging matrix (17.1).

However, for different values x and y what changes is the number of messages *to reach the consensus*, i.e. the number of k to reach a matrix with all rows equal. One possible standard choice (independent from the number of nodes involved) is

$$q_{ij} = \begin{cases} \varepsilon & \text{if } j \text{ can communicate with } i, \\ 1 - \varepsilon d(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (17.2)$$

where q_{ij} is the element in position i and j of the matrix Q . For bidirectional connections, the *number of neighbours that can send and can receive* the information to and from node i is called the *node degree* and denoted with $d(i)$. A common usual choice is to use $\varepsilon = \frac{1}{1 + \max_j d(j)}$, aka *maximum degree weights*. Notice that to compute the value of ε at least a bound on the degree of each node is needed, hence the algorithm is *partially local*.

A different strategy for the same problem is instead given by

$$q_{ij} = \begin{cases} \frac{1}{\max(d(i), d(j)) + 1} & \text{if } j \text{ can communicate with } i, \\ 1 - \sum_{j=1, i \neq j}^n q_{ij} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (17.3)$$

Notice that $\forall i$:

$$q_{ii} = 1 - \sum_{j=1, i \neq j}^n q_{ij} \geq 1 - \sum_{j=1, i \neq j}^n \frac{1}{d(i) + 1} = 1 - \frac{d(i)}{d(i) + 1} > 0,$$

which ensures that the matrix Q is again doubly-stochastic. This solution adopts the *Metropolis-Hastings* weights, which is a *local solution* and ensures *faster* convergence with respect to the previous solution.

In both cases, the *Metropolis-Hastings* weights and the *maximum degree weight* give the following solution:

$$Q = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} \end{bmatrix}.$$

Notice how this matrix is *doubly stochastic*.

17.1.1 Variable topology

Let us now consider a different situation in which the communication topology *changes in time* as per Figure 17.5. This situation usually arises when two nodes cannot broadcast their messages *simultaneously*. In such a case,

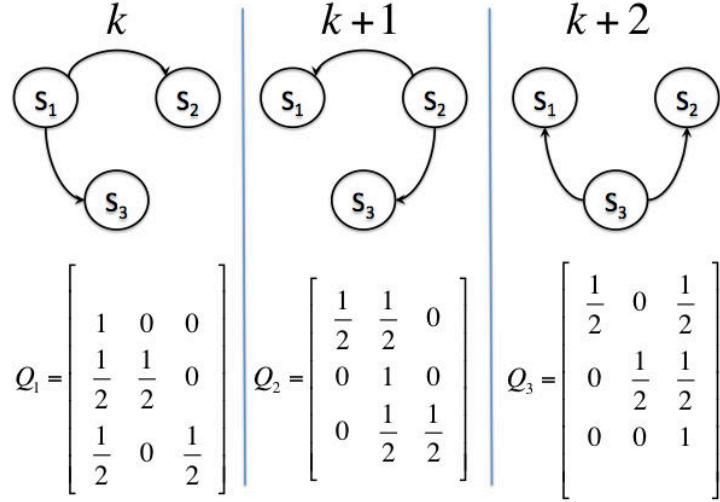


Figure 17.5: Sensors communication links and algebraic representation with complete visibility but sequential broadcasts.

different protocol matrices should be considered. The simple choice reported in Figure 17.5, i.e.

$$Q(k) = Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}, \quad Q(k+1) = Q_2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

$$Q(k+2) = Q_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix},$$

computes the mean pairwise. Notice that Q_i , $i = 1, \dots, 3$, is *stochastic* but not *doubly stochastic*. At time k node S_1 broadcasts, at time $k+1$ node S_2 , at time $k+2$ node S_3 and then the sequence is repeated, i.e. *round robin* scheduling. Hence, at time k , nodes S_2 and S_3 receive the S_1 node value and then compute:

$$x_1(k+1) = x_1(k), \quad x_i(k+1) = \frac{x_1(k) + x_i(k)}{2} \text{ for } i = 2, 3.$$

The stochastic matrix after a cycle of the round robin scheduling would be

$$Q = Q_3 Q_2 Q_1 = \begin{bmatrix} \frac{5}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{2} & \frac{3}{8} & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

Since this Q is now *stochastic* but not *doubly stochastic*, a consensus would be reached, but not an *averaged consensus*. Since our objective is to compute the *arithmetic mean*, the solution would be to impose $Q = Q_3Q_2Q_1$ *doubly stochastic*, and then derive the entries of the matrices Q_i . A possible solution in this case (computed as in the fixed topology example) is

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{x}{2x+1} & \frac{x+1}{2x+1} & 0 \\ \frac{x}{2x+1} & 0 & \frac{x+1}{2x+1} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} \frac{1}{1+x} & \frac{x}{1+x} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{x}{1+x} & \frac{1}{x+1} \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} 1-x & 0 & x \\ 0 & 1-x & x \\ 0 & 0 & 1 \end{bmatrix},$$

with $0 < x < 1$. For example, by selecting $x = \frac{1}{2}$ one has

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \quad Q = Q_3Q_2Q_1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}.$$

Notice that Q is *doubly stochastic* as desired, while the Q_i are *stochastic*. After some k round robin executions, $Q^k = (Q_3Q_2Q_1)^k$ converges to a matrix whose entries are all equal to $\frac{1}{3}$, as desired.

The fastest convergence, i.e., the minimum number of messages k to reach the average consensus, is instead obtained for any $x \neq 0$ and

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{6x} & \frac{1}{3x} & 0 \\ \frac{1}{6x} & 0 & \frac{1}{3x} \end{bmatrix}, \quad Q_2 = \begin{bmatrix} x & x & 0 \\ 0 & 1 & 0 \\ 0 & x & x \end{bmatrix},$$

$$Q_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

Indeed, $\forall x \neq 0$ we get that matrix Q_3 has the role to substitute the value of $x_1(k)$ and $x_2(k)$ with $x_3(k)$, i.e.

$$x(k+1) = Q_3x(k) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

while

$$Q_2 Q_1 = \begin{bmatrix} x + \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{6x} & \frac{1}{3x} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

that is, after the broadcasts of S_1 and S_2 , S_3 has the correct value of the arithmetic mean (last row of $Q_2 Q_1$), which has to be broadcasted to the other two nodes with the last step of the round robin scheduling (i.e. matrix Q_3). As a consequence, only one round robin cycle is needed to reach the *average consensus*. This is obvious by noting that

$$Q = Q_3 Q_2 Q_1 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

just for $k = 1$: *fastest convergence*.

17.2 Examples

Problems that can be solved in a distributed sense with linear consensus algorithms:

- Node counting;
- Minimum variance estimates;
- Vehicle rendezvous;
- Least squares problems;
- Sensor calibration;
- Distributed estimation.

17.2.1 Node counting

Let us consider a network with n nodes, but at start-up this number is *unknown*. If the network satisfies the condition to reach an *average consensus*, i.e., the graph contains all the self loops and Q is doubly stochastic, then

$$\lim_{t \rightarrow +\infty} x_i(t) = \frac{1}{n} \sum_{j=1}^n x_j(0), \quad \forall i = 1, \dots, n.$$

Is it possible to exploit this property?

If we impose that only one node, say $x_1(0)$, is equal to one and all the others equal to 0, one has

$$\lim_{t \rightarrow +\infty} x_i(t) = \frac{1}{n} \sum_{j=1}^n x_j(0) = \frac{1}{n}, \quad \forall i = 1, \dots, n.$$

Therefore, the number of nodes is simply given by

$$\frac{1}{\lim_{t \rightarrow +\infty} x_i(t)} = n, \quad \forall i = 1, \dots, n.$$

17.2.2 Minimum variance estimates

Consider a network of n sensors all measuring the same scalar quantity

$$z_i = x + \nu_i,$$

where $\nu_i \sim \mathcal{N}(0, \sigma_i^2)$, $\forall i = 1, \dots, n$. The *minimum variance* solution if all the sensors have the same *precision* σ_i is just the *arithmetic mean*, hence an average consensus is sufficient. However, in the general case of different sensors, we know that the solution is given by the *least squares* that is computed as

$$\hat{x}^{LS} = \sum_{i=1}^n \frac{\frac{z_i}{\sigma_i^2}}{\sum_{j=1}^n \frac{1}{\sigma_j^2}}.$$

Notice that this relation can be equivalently computed as

$$\hat{x}^{LS} = \frac{\frac{1}{n} \sum_{i=1}^n \frac{z_i}{\sigma_i^2}}{\frac{1}{n} \sum_{j=1}^n \frac{1}{\sigma_j^2}}.$$

It is easy to see that this is the ratio of two average consensus algorithms. Therefore, by defining the variables $a_i(0) = \frac{z_i}{\sigma_i^2}$ and $b_i(0) = \frac{1}{\sigma_i^2}$, two parallel average consensus algorithms can be executed, i.e.

$$a(t+1) = Qa(t) \text{ and } b(t+1) = Qb(t).$$

Therefore, the local estimate of the LS estimator is $\hat{x}_i^{LS}(t) = \frac{a_i(t)}{b_i(t)}$, that asymptotically converge to

$$\lim_{t \rightarrow +\infty} \hat{x}_i^{LS}(t) = \lim_{t \rightarrow +\infty} \frac{a_i(t)}{b_i(t)} = \hat{x}^{LS}, \quad \forall i = 1, \dots, n.$$

Notice how the solution is completely distributed.

17.2.3 Vehicle rendezvous

With the simple linear average consensus it is also possible to solve the problem of *rendezvous*, i.e. let the vehicles meet in a common point. This problem can be solved assuming that vehicles *only* use *relative distance information*. A very simple (mono-dimensional) vehicle kinematic can be described by

$$x_i(t+1) = x_i(t) + u_i(t).$$

It is then easy to see that a control law such as

$$u_i(t) = \sum_{j=1}^n q_{ij}(x_j(t) - x_i(t)),$$

with properly chosen weights to guarantee average consensus leads to vehicle rendezvous.

17.3 A Theoretical Approach to Linear Consensus

Many results in the field of distributed systems can be reformulated with graph properties.

Definition 73 (Graph). A *graph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ comprising a set \mathcal{N} of *nodes* or *vertices* together with a set \mathcal{E} of *edges*, which are 2-element subsets of \mathcal{N} .

Usually the graph is used to describe the *topology* of the systems connection in a network. Each node represents a system while an edge the communication between the two connected systems (see Figure 17.7).

Definition 74 (Edge). Let $\mathcal{N} = \{1, 2, \dots, n\}$ be the set of nodes. Hence, the pair $(j, i) \in \mathcal{E}$ implies that node i can receive information from node j .

Definition 75 (Digraph). A *directed graph* or *digraph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which the edges, known also as *arcs* in this case, connects an ordered pairs of nodes.

Definition 76 (Undirected Graph). An *undirected graph* is an ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which if $(j, i) \in \mathcal{E}$ hence $(i, j) \in \mathcal{E}$.

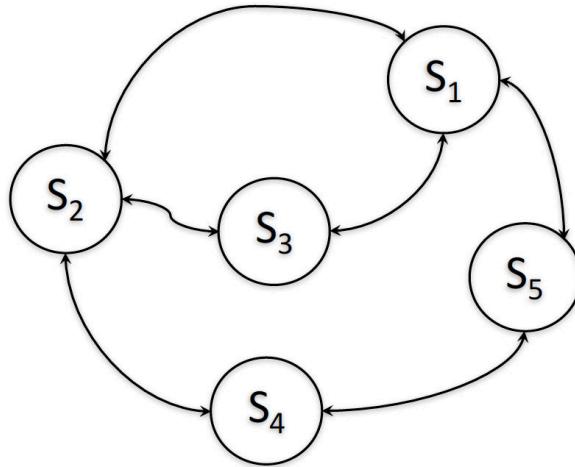


Figure 17.6: The graph representation of the distributed systems for a generic network topology.

An example of an undirected and a directed graph can be seen in Figure 17.7. In practice, the edge represents a *communication link* between two nodes. In a digraph, the communication obeys a *simplex* modality. In an undirected graph, the communication obeys to the *half-duplex* or *full-duplex* modality. The difference between the two is that in the latter the communication is bidirectional and in the same time instant.

Definition 77 (Self loops). A graph includes all the *self-loops* if and only if $(i, i) \in \mathcal{E}$, $\forall i \in \mathcal{N}$.

Definition 78 (Set of sending neighbours and in-degree). The set of neighbours that can send the information to node i is defined by $\mathcal{V}_{in}(i) = \{j | (j, i) \in \mathcal{E}, i \neq j\}$. Hence, the *in-degree* of node i is defined as $d_{in}(i) = |\mathcal{V}_{in}(i)|$, where $|\cdot|$ represents the cardinality of a set.

Definition 79 (Set of receiving neighbours and out-degree). The set of neighbours that can receive the information from node i is defined by $\mathcal{V}_{out}(i) = \{j | (i, j) \in \mathcal{E}, i \neq j\}$. Hence, the *out-degree* of node i is defined as $d_{out}(i) = |\mathcal{V}_{out}(i)|$.

for the undirected graph of Figure 17.7-a, we have $\mathcal{V}(1) = \{2, 3, 5\}$, $\mathcal{V}(2) = \{1, 3, 4\}$. Hence, $d(1) = 3$, $d(2) = 3$. Instead, for the directed graph of Figure 17.7-b, we have $\mathcal{V}_{in}(1) = \{2\}$, $\mathcal{V}_{in}(2) = \{1, 4\}$, $\mathcal{V}_{out}(1) = \{2, 3, 5\}$, $\mathcal{V}_{out}(2) = \{1, 3, 4\}$. Hence, $d_{in}(1) = 1$, $d_{in}(2) = 2$, $d_{out}(1) = 3$, $d_{out}(2) = 3$. In practice, *self-loops are always considered* since it is commonplace that each

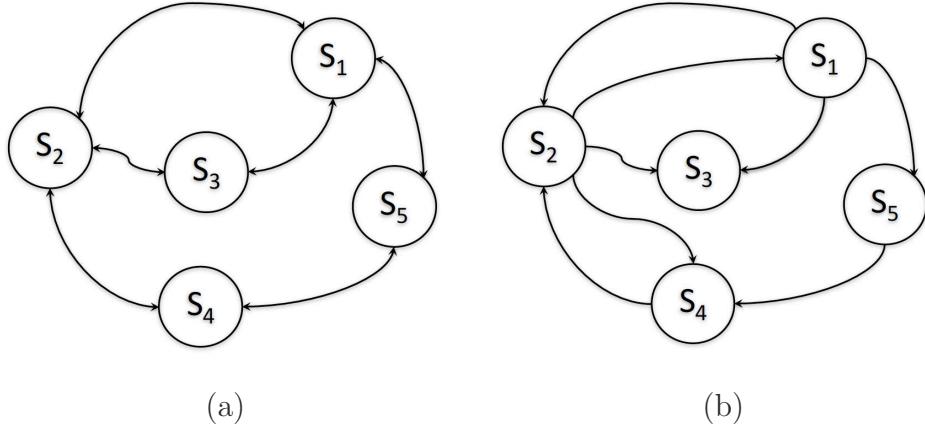


Figure 17.7: Undirected (a) and directed (b) graphs.

node can have information from its sensors/actuators. For an undirected graph, in-neighbours and out-neighbours of a node i coincide and they are simply denoted by the set $\mathcal{V}(i)$, whose degree is $d(i) = |\mathcal{V}(i)|$.

Definition 80 (Rooted). A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is **rooted** if there exists a node $k \in \mathcal{N}$ such that for any other node $j \in \mathcal{N}$ there is a unique path from k to j .

Definition 81 (Strongly connected). A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is **strongly connected** if there exists a path from any node to any other node.

Definition 82 (Complete). A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is **complete** if $(i, j) \in \mathcal{E}$, $\forall i, j \in \mathcal{N}$.

The undirected graph of Figure 17.7-a is *rooted* and *strongly connected*. It is not *complete*, but it becomes complete if $\mathcal{E}^{new} = \mathcal{E} \cup \{(1, 4), (2, 5), (3, 4), (3, 5)\}$. Instead, the directed graph of Figure 17.7-b is *rooted* in all the nodes apart from S_3 . It is *not strongly connected*. It is *not complete*.

Definition 83 (Diameter). The diameter of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as the length of the longest among all shortest paths connecting any two nodes in a strongly connected graph.

The diameter of the undirected graph of Figure 17.7-a is 2.

Definition 84 (Adjacency matrix). The adjacency matrix $A \in \mathbb{R}^{n \times n}$ of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a matrix having $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and $i \neq j$, and $a_{ij} = 0$.

Definition 85 (Degree matrix). The **degree matrix** D of an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as $D = \text{diag}(d(1), d(2), \dots, d(n))$.

Definition 86 (Laplacian matrix). The **Laplacian matrix** L of an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as $L = D - A$.

The **adjacency matrix** of the undirected graph of Figure 17.7-a is

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix},$$

while its **degree matrix** is

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix},$$

and its **Laplacian matrix** is

$$L = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}.$$

For the directed graph of Figure 17.7-b the **adjacency matrix** is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The Laplacian matrix is **positive semidefinite**¹ and verifies $L\mathbf{1} = 0$.

Let us consider the previous example in Figure 17.2. Each measured value is a variable, $x_i(t)$ and the distributed system **state** can then be represented by

¹A matrix $M \in \mathbb{R}^{n \times n}$ is **positive definite** if and only if $x^T M x > 0$, $\forall x \in \mathbb{R}^n$ and $x \neq 0$. Accordingly, is **negative definite**, **positive semidefinite** and **negative semidefinite** if respectively $x^T M x < 0$, $x^T M x \geq 0$ and $x^T M x \leq 0$.

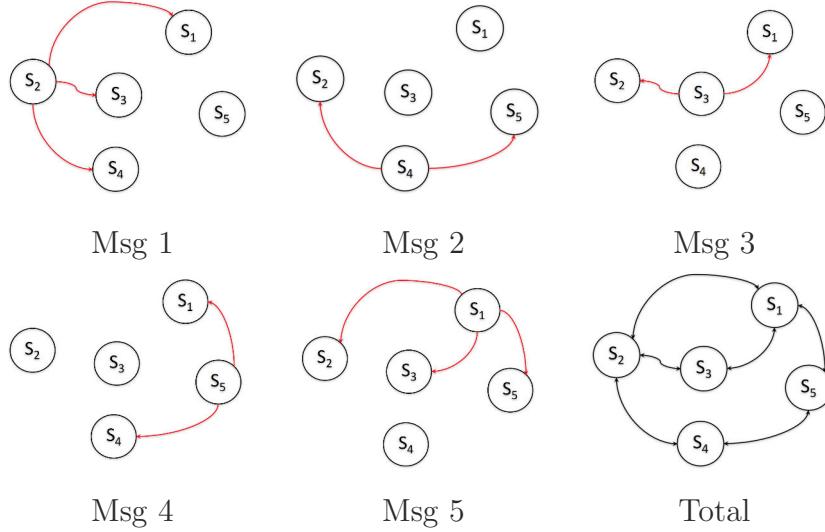


Figure 17.8: Time varying combination of graphs.

$x(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T \in \mathbb{R}^n$. Suppose we want to construct a *global* agreement on the measured phenomenon starting from each *local* information of each sensor. In practice, we want the network to reach an *agreement* on the measured quantities. One possible solution is to let each sensor to carry out a *local estimation process* based on the information coming from the other sensors. The agreement is then reached if *all the estimation processes give the same result*. A straightforward solution may be to implement *in each* node a simple estimator: the mean. However, the information sharing depends on the *topology* of the network. This problem can be circumvented making use of the adjacency matrix. Indeed, assuming a *complete graph*, we can have for example the following *update equation*:

$$x(t+1) = \frac{1}{n}(I_n + A)x(t) = Qx(t).$$

One important property of the matrix Q is that its rows and its columns, in this case, sums up to one, i.e. a *double stochastic* matrix. We will see in the following that this is an essential characteristic of such algorithms even if the graph is *not* complete and even if the matrix Q changes in time, i.e., the communication topology is time varying as in Figure 17.8.

17.3.1 Stochastic matrices

Definition 87 (Stochastic matrix). A *stochastic matrix* is a matrix $Q \in \mathbb{R}^{n \times n}$ if and only if $q_{ij} \geq 0$ and $\sum_{j=0}^n q_{ij} = 1$, $\forall i$.

Using the previously introduced definitions, we say that the graph $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ is associated to the stochastic matrix $Q \in \mathbb{R}^{n \times n}$ if $\mathcal{N} = \{1, 2, \dots, n\}$ and $\mathcal{E}_q = \{(j, i) | q_{ij} > 0\}$.

Definition 88 (Doubly-stochastic matrix). A stochastic matrix Q is **doubly-stochastic** if also $\sum_{i=0}^n q_{ij} = 1, \forall j$.

Clearly, if Q is a stochastic **symmetric** matrix, i.e., $Q = Q^T$, then it is also doubly-stochastic.

Definition 89 (Circulant matrix). A **circulant matrix** is a matrix having the sequence of the rows with elements shifted by one position.

An example of a circulant matrix is the following

$$C = \begin{bmatrix} c_1 & c_2 & c_3 & \dots & c_n \\ c_n & c_1 & c_2 & \dots & c_{n-1} \\ \vdots & \vdots & \ddots & & \vdots \\ c_2 & c_3 & c_4 & \dots & c_1 \end{bmatrix}$$

Definition 90 (Spectral radius). The **spectral radius** of a matrix $M \in \mathbb{R}^{n \times n}$ is defined as $\rho(M) \triangleq \max_i |\lambda_i|$, $i = 1, \dots, n$ and λ_i is the i -th eigenvalue of M .

Definition 91 (Essential spectral radius). The **essential spectral radius** $\rho_2(M)$ of a matrix $M \in \mathbb{R}^{n \times n}$ is defined as the second largest eigenvalue of M .

If Q is a stochastic matrix, then the following facts hold:

- If λ_i is an eigenvalue of Q , then $|\lambda_i| \leq 1, \forall i$.
- Moreover, $Q\mathbf{1} = \mathbf{1}$. Notice that this condition ensures that any vector with constant entries, i.e. $x(t) = \alpha\mathbf{1}$, is a **fixed point** for the dynamic $x(t+1) = Qx(t)$.
- If the eigenvalues are ordered, it follows that $\rho(Q) = |\lambda_1| = 1$, while $\rho_2(Q) = |\lambda_2| \leq 1$.

17.4 Linear Consensus

Let us consider the following general **update equation**:

$$x(t+1) = Q(t)x(t), \text{ with } Q(t) \in \mathbb{R}^{n \times n} \text{ and } x(t) \in \mathbb{R}^n.$$

We will assume that $Q(t)$ is a *stochastic matrix*. It is easy to see that the *update equation* for node i is given by:

$$x_i(t+1) = \sum_{j=1}^n q_{ij}(t)x_j(t) = x_i(t) + \sum_{j=1}^n q_{ij}(t)(x_j(t) - x_i(t)), \quad (17.4)$$

i.e., it is associated to a graph with all the self-loops. Notice that the formulation (17.4) expresses that this updating rule is *distributed*: each node uses its own information plus the information it can receive. Nonetheless, we will give now some properties that ensures the *global* convergence.

Definition 92 (Linear consensus problem). *With respect to the previous update equation, the matrix $Q(t)$ solves the consensus problem if*

$$\lim_{t \rightarrow +\infty} x_i(t) = \alpha, \quad \forall i,$$

or, in matrix form, assuming $x(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T$,

$$\lim_{t \rightarrow +\infty} x(t) = \alpha \mathbf{1}.$$

Notice that since

$$\begin{aligned} x(t+1) &= Q(t)x(t) = Q(t)Q(t-1)x(t-1) = \\ &= Q(t)Q(t-1)\dots Q(0)x(0) = \Phi(t)x(0), \end{aligned}$$

where $\Phi(t)$ is the *t-step transition matrix*, we have that

$$\lim_{t \rightarrow +\infty} x(t) = \lim_{t \rightarrow +\infty} \Phi(t)x(0) = \alpha \mathbf{1}.$$

Definition 93 (Linear average consensus problem). *With respect to the previous update equation, the matrix $Q(t)$ solves the average consensus problem if*

$$\lim_{t \rightarrow +\infty} x_i(t) = \frac{1}{n} \sum_{i=1}^n x_i(0), \quad \forall i,$$

or, in matrix form

$$\lim_{t \rightarrow +\infty} x(t) = \left(\frac{1}{n} \mathbf{1}^T x(0) \right) \mathbf{1} = \frac{1}{n} \mathbf{1} \mathbf{1}^T x(0).$$

The next theorems describe some *sufficient conditions* which guarantee deterministic consensus, i.e., when $Q = Q(t), \forall t$.

Theorem 11 (Deterministic convergence). *If the graph $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ contains all the self-loops and it is also rooted, then*

$$\lim_{t \rightarrow +\infty} Q^t = \mathbf{1}\beta^T,$$

where $\beta \in \mathbb{R}^n$ is the left eigenvector of Q for $\lambda_1 = 1$. Moreover, we have

$$\beta_j \geq 0 \text{ and } \mathbf{1}^T \beta = 1.$$

Notice that being the left eigenvector of Q for $\lambda_1 = 1$, means $\beta^T Q = \beta^T$, which implies $\beta^T x(t+1) = \beta^T x(t)$ at each step.

Theorem 12 (Average consensus). *If $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ contains all the self-loops and it is also strongly connected, then $\beta_j > 0, \forall j$.*

If Q is doubly-stochastic, then $\mathcal{G}_q = (\mathcal{N}, \mathcal{E}_q)$ is strongly connected and

$$\beta = \frac{1}{n} \mathbf{1},$$

i.e., it solves the average consensus problem.

In other words, since

$$\lim_{t \rightarrow +\infty} x(t) = \lim_{t \rightarrow +\infty} \Phi(t)x(0),$$

the *t-step transition matrix* converges to

$$\lim_{t \rightarrow +\infty} \Phi(t) = \frac{1}{n} \mathbf{1} \mathbf{1}^T = J_a.$$

An alternative formulation of the convergence is given by the following Theorem (see [35]).

Theorem 13 (Average consensus bis). *The previous condition hold if and only if: a) $\mathbf{1}^T Q = \mathbf{1}^T$; b) $Q\mathbf{1} = \mathbf{1}$; c) $\rho(Q - J_a) < 1$.*

Notice that if the elements of Q are all non negative, these are the conditions of the previous Theorem formulation, i.e., Q doubly stochastic and the graph containing all the self-loops.

Existence of self-loops is *not necessary* to reach consensus. Indeed, taking Q with only one column equal to $\mathbf{1}$ reaches a consensus. However, the fact of being rooted *without self-loops* does not guarantee consensus. For example, taking Q with the anti-diagonal equal to $\mathbf{1}$ defines a periodic dynamic.

Theorem 14 (Rate of convergence). *The rate of convergence of all the cases in the previous theorem is exponential and its rate is given by $\rho_2(Q)$.*

17.4.1 Continuous time systems

Before going into the details of this case, let us consider a special class of matrices.

Definition 94. A *Metzler matrix* is a matrix whose off-diagonal elements are nonnegative and the row-sum is null.

It then follows that if M is Metzler, $M\mathbf{1} = 0$ [36]. Notice that the *negative* graph Laplacian $-L$ is a *Metzler matrix*.

Consider a continuous time system

$$\dot{x}(t) = M(t)x(t).$$

If $M(t)$ is a *Metzler matrix* than the network achieves a consensus under general connectivity properties of the associated graph. Indeed, $\forall x(t) = c\mathbf{1}$, we have $\dot{x} = 0$, i.e., an *equilibrium point*. To prove stability, let's compute the *agreement error*:

$$e_i(t) = x_{i+1}(t) - x_1(t), \forall i = 1, \dots, n-1,$$

and $e = [e_1, e_2, \dots, e_{n-1}]^T$. Therefore, assuming for simplicity $M(t) = -L$, one gets:

$$\dot{e} = -\tilde{L}e,$$

where

$$\tilde{L} = \begin{bmatrix} l_{22} - l_{12} & \cdots & l_{2n} - l_{1n} \\ \vdots & \ddots & \vdots \\ l_{n2} - l_{12} & \cdots & l_{nn} - l_{1n} \end{bmatrix}.$$

Since the eigenvalues of L are $\lambda_1, \lambda_2, \dots, \lambda_n$, with $\lambda_1 = 0$, it is possible to show that the eigenvalues of \tilde{L} are $\lambda_2, \dots, \lambda_n$.

Theorem 15. For any eigenvalue λ_i , $i = 1, \dots, n$, of the Laplacian matrix L associated to $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ either $\lambda_i = 0$ or $\text{Re}(\lambda_i) > 0$.

Indeed, L is a *positive semidefinite* matrix.

Theorem 16. The (di)graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is rooted if and only if $\text{Rank}(L) = n - 1$, with L being the Laplacian of \mathcal{G} .

It then follows that the continuous system reaches a consensus solution *asymptotically if and only if* the (di)graph \mathcal{G} is rooted [36, 37].

There is a lot of literature of the last ten years that extends this basic idea to more complicated and more realistic scenarios:

- *Multidimensionality*: The consensus for continuous time systems is indeed used for multidimensional systems, even though specific and difficult technical solutions are needed.
- *Delays*: The consensus protocols are ideal, since they don't explicitly consider the presence of the communication delays.
- *Noise*: Optimal consensus protocols have been also designed, which are able to minimise the presence of the noise in the sent data.
- *Nonlinear*: The linear consensus have been also extended to nonlinear systems, like robotic vehicles.

17.4.2 Design of the consensus algorithm

In the previous sections we have considered the *analysis* of consensus algorithms. From an engineering point of view, it is also important to understand how to *design* such algorithms, aka *consensus protocols*. The design can be synthesised in what follows: *Given the communication graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ of a network with n nodes, find a matrix $Q(t)$ compatible with \mathcal{G} that achieve (average) consensus.*

In practice this problem amounts to find the values of the elements of the matrix $q_{ij} > 0$ corresponding to an edge (j, i) , i.e., j sends to i .

There are basically two approaches to the design of the consensus algorithms:

- *Global optimal design*: This approach tends to find an optimal solution to some *global performance index*. In this case, a *centralised* solution is quite often necessary, which is feasible for networks with a *limited number* of nodes and *fixed topology*.
- *Local design*: This approach designs the consensus algorithm using only *local information* and independently from other nodes. Of course, optimality is *not guaranteed*. This is the approach we adopt in this course.

Continuous time case

For the *local* design of the consensus protocol in the continuous time case, it is easy to see that in the case of simple integrator dynamics

$$\dot{x}_i(t) = \alpha_i u_i(t), \alpha \in \mathbb{R},$$

where $u_i(t)$ is the input, a simple *consensus protocol* like the following

$$u_i(t) = \frac{\beta}{\alpha_i} \sum_{j=1}^n l_{ij}(x_j(t) - x_i(t)), \beta \in \mathbb{R}$$

where l_{ij} are the elements of the Laplacian matrix L , reaches asymptotically a consensus.

Discrete time case

A possible strategy for the *local* design of a consensus protocol for a discrete time system:

$$x_i(t+1) = x_i(t) + \alpha_i u_i(t), \alpha \in \mathbb{R},$$

is given by the input

$$u_i(t) = \frac{1}{\alpha_i} \sum_{j=1}^n \frac{1}{1 + d_{in}(i)} (x_j(t) - x_i(t)), \forall (j, i) \in \mathcal{E}.$$

Substituting $u_i(t)$, one has

$$\begin{aligned} x_i(t+1) &= x_i(t) + \alpha_i u_i(t) = x_i(t) + \sum_{j=1}^n \frac{1}{1 + d_{in}(i)} (x_j(t) - x_i(t)) \\ &= x_i(t) + \sum_{j=1}^n q_{ij} (x_j(t) - x_i(t)). \end{aligned}$$

In matrix form, defining $Q = (q_{ij})$, i.e., $x(t+1) = Qx(t)$, it is easy to verify how Q is a stochastic matrix. Notice how this solution reaches consensus, **but not** average consensus for general graphs.

If we want to design a discrete time consensus protocol with *convergence rate requirements*, let us define $\bar{x} = \frac{1}{n} \mathbf{1}^T x(0) = \frac{1}{n} \sum_{i=1}^n x_i(0)$ and the *asymptotic convergence factor* as

$$r_a = \sup_{x(0) \neq \bar{x}} \lim_{t \rightarrow +\infty} \left(\frac{\|x(t) - \bar{x}\|_2}{\|x(0) - \bar{x}\|_2} \right)^{\frac{1}{t}},$$

which expresses how *fast* the average consensus is reached. Similarly, let us define the *per-step convergence factor* as

$$r_s = \sup_{x(t) \neq \bar{x}} \frac{\|x(t+1) - \bar{x}\|_2}{\|x(t) - \bar{x}\|_2},$$

which expresses how *fast* is the contraction *per step* towards the average consensus. Recall that the *t-step transition matrix* converges to

$$\lim_{t \rightarrow +\infty} \Phi(t) = \frac{1}{n} \mathbf{1} \mathbf{1}^T = J_a,$$

if and only if. a) $\mathbf{1}^T Q = \mathbf{1}^T$; b) $Q\mathbf{1} = \mathbf{1}$; c) $\rho(Q - J_a) < 1$, i.e. Theorem 13. In such a case, we have that the *asymptotic convergence factor* r_a is given by

$$r_a = \rho(Q - J_a).$$

Similarly, the *per-step convergence factor* r_s is given by

$$r_s = \|Q - J_a\|_2.$$

Let us understand why the *per-step convergence factor* is relevant. Since $\mathbf{1}^T Q = \mathbf{1}^T$, we have $\mathbf{1}^T x(t+1) = \mathbf{1}^T x(t)$, $\forall t$, we have

$$x(t+1) - J_a x(0) = Q(x(t) - J_a x(0)) = (Q - J_a)(x(t) - J_a x(0)).$$

Using then the Euclidean norms, we have

$$\begin{aligned} \|x(t+1) - J_a x(0)\|_2 &= \|(Q - J_a)(x(t) - J_a x(0))\|_2 \\ &\leq \|Q - J_a\|_2 \|x(t) - J_a x(0)\|_2. \end{aligned}$$

Therefore, if $r_s = \|Q - J_a\|_2 < 1$, at each step $x(t)$ tends towards the average. In other words, r_s is a measure of the *worst case asymptotic rate of convergence towards the consensus*. One possible problem that can be tackled is the choice of the Q elements that maximises the *asymptotic* or the *per-step* contraction, i.e. that let the system to converge *as fast as possible*. It can be shown that if $Q = Q^T$, the solution in both cases is given by [35]

$$\min_Q \rho(Q - J_a) \quad s.t. \quad Q = Q^T, Q\mathbf{1} = \mathbf{1}.$$

A possible strategy for the *local* design of a consensus protocol in *rooted undirected graphs* reaching *average consensus* for a discrete time system:

$$x_i(t+1) = x_i(t) + \alpha_i u_i(t), \alpha \in \mathbb{R},$$

is to solve the previously defined optimal problem assuming elements in Q that are all equal. The strategy is to set all edge weights to ε and the self-weights to satisfy $Q\mathbf{1} = \mathbf{1}$. Therefore, one has

$$q_{ij} = \begin{cases} \varepsilon & \text{if } (j, i) \in \mathcal{E} \text{ and } i \neq j, \\ 1 - \varepsilon d(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that such an approach corresponds to select

$$Q = I - \varepsilon L,$$

where L is the associated Laplacian matrix. Since L is *positive semidefinite*, to ensure that $\rho(Q - J_a) < 1$, we have $\varepsilon > 0$. Hence, the i -th largest eigenvalue of Q is given by

$$\lambda_i(Q) = 1 - \varepsilon \lambda_{n-i+1}(L),$$

which yields [35]

$$\rho(Q - J_a) = \max\{\lambda_2(Q), -\lambda_n(Q)\} = \max\{1 - \varepsilon \lambda_{n-1}(L), \varepsilon \lambda_1(L) - 1\}.$$

As a consequence, we have that

$$0 < \varepsilon < \frac{2}{\lambda_1(L)}.$$

In order to have positive diagonal terms of Q , usually a limit is given by

$$\varepsilon < \frac{1}{\max_i d(i)},$$

which ensures that Q is a stochastic matrix. A common usual choice is to use $\varepsilon = \frac{1}{1 + \max_i d(i)}$, the *max degree weights* presented in (17.2). With the presented choice of the matrix Q , one has for the input

$$u_i(t) = \frac{1}{\alpha_i} \sum_{j=1}^n \varepsilon(x_j(t) - x_i(t)), \forall (j, i) \in \mathcal{E}.$$

Since the graph is *rooted* and *undirected* it will be also *strongly connected*. This implies that: a) Q is a doubly-stochastic matrix and b) the *average consensus* can be reached since the hypotheses of the Theorem 11 on the deterministic convergence are verified.

Since $Q = I - \varepsilon L$, it can be considered as a *discrete version of a continuous time* average consensus solution. Indeed, let us recall the dynamic for the continuous case $\dot{x}(t) = Mx(t) = -Lx(t)$. The continuous time dynamic can be discretised with *sampling time* ε , yielding to

$$x(t+1) = e^{-\varepsilon L} x(t) = \sum_{i=0}^{+\infty} \frac{(-L)^i \varepsilon^i}{i!} x(t) = (I - \varepsilon L + \mathcal{O}(\varepsilon)) x(t),$$

which, by neglecting $\mathcal{O}(\varepsilon)$, it is the adopted discrete time consensus protocol.

A different strategy for the same problem is instead given by *Metropolis-Hastings* weights in (17.3), which is a *local solution* and ensures *faster* convergence with respect to the Laplacian based solution. The case of digraph is also considered in literature, but not considered in these notes.

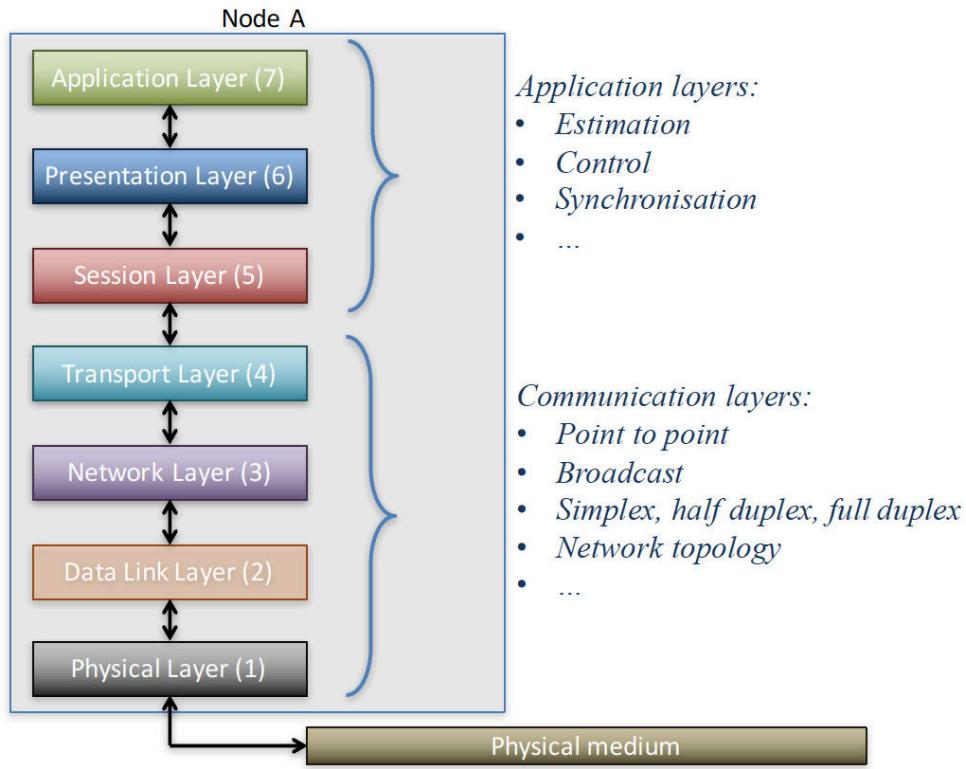


Figure 17.9: Convenient renaming of the OSI model layers.

17.5 Linear Consensus with Networks

As a final comment, let us recall the OSI Model presented in Section 2.2, with the convenient sublayers renaming of Figure 17.9. In such a case, the consensus protocol can be seen as an additional layer enforcing the cooperation between the agents, hence a *cooperation layer*, as in Figure 17.10.

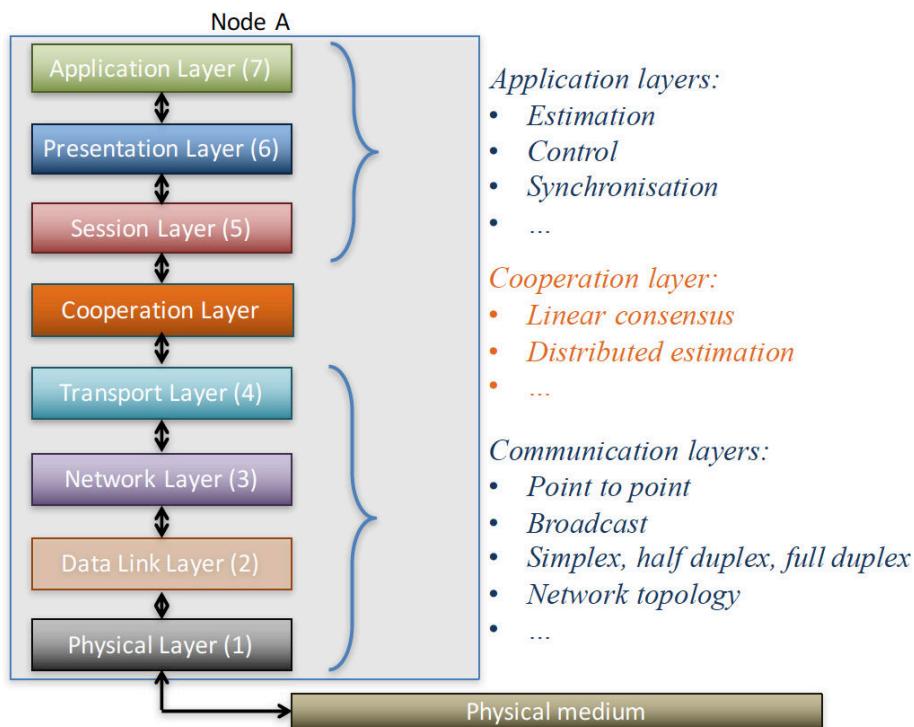


Figure 17.10: The consensus protocol as a *cooperation layer*.

Chapter 18

Distributed Estimation Algorithms

18.1 Distributed WLS

Let us consider the Problem 5 we have defined at the beginning of Section 13.1.1. The solution we provided previously is based on the existence of a *central node* that receives all the data and then performs, in a *centralised* way the estimates. Now, we want to do exactly the same thing in a *distributed way*.

To tackle this problem, let us consider the same time varying set of measurements collected at time t in (13.1), here reported for convenience

$$z(i) = H(i)x + \varepsilon(i), \quad i = 1, \dots, n,$$

together with the usual quantities Z^n , H^n , ε^n and C^n described in (13.2) and (13.3), but this time as a function of n instead of k . Notice how the noise is assumed uncorrelated in different time instants, while instead it can be correlated for each time instant ($C\{\varepsilon(i)\}$ is not necessarily diagonal). Therefore, applying the same Weighted Least Squares solution (13.4), we have

$$\hat{x}^{LS} = \arg \min_x J(x, n) = (H^{nT} C^{n-1} H^n)^{-1} H^{nT} C^{n-1} Z^n, \quad (18.1)$$

with covariance matrix of the estimation error given by

$$P_n = (H^{nT} C^{n-1} H^n)^{-1}.$$

One immediate solution for a *distributed algorithm* to work properly in this case is just using a *flooding algorithm*:

- Each node transmits to all the other nodes in its *communication range* all its data;
- Under mild connectivity properties, sooner or later, *all* the nodes in the network will have *all* the data in the network.
- Now, each node can solve the WLS and hence each node has the *same best LS solution*.

However, such a solution asks for *lot of messages* to travel along the network. To improve the effectiveness of a distributed solution, we can first make use of the definitions of the matrices given in (13.2) in order to rewrite the solution (18.1) as

$$\begin{aligned}\hat{x}^{LS} &= \arg \min_x J(x, n) = (H^{nT} C^{n-1} H^n)^{-1} H^{nT} C^{n-1} Z^n = \\ &= \left(\sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} H(i) \right)^{-1} \sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} z(i).\end{aligned}$$

while, for the covariance matrix of the estimation error, we have similarly

$$P_n = \left(\sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} H(i) \right)^{-1}.$$

In particular, the distributed sensor fusion scheme considers a local *composite information matrix* $F_i(k) \in \mathbb{R}^{m \times m}$ and a local *composite information state* $a_i(k) \in \mathbb{R}^m$, *one for each node*. The i -th sensor makes a measurement and initialises its composite elements as:

$$F_i(0) = H(i)^T C \{\varepsilon(i)\}^{-1} H(i), \text{ and } a_i(0) = H(i)^T C \{\varepsilon(i)\}^{-1} z(i).$$

Similarly, it is also able to provide the first estimate at time $k = 0$ (according to the general WLS solution)

$$\hat{x}^{LS}(0) = F_i(0)^{-1} a_i(0).$$

The solution of the *centralised* (i.e. *global*) WLS can be written in terms of the composite components as follows by

$$\begin{aligned}\hat{x}^{LS} &= \left(\sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} H(i) \right)^{-1} \sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} z(i) = \\ &= \left(\sum_{i=1}^n F_i(0) \right)^{-1} \sum_{i=1}^n a_i(0), \\ P_n &= \left(\sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} H(i) \right)^{-1} = \left(\sum_{i=1}^n F_i(0) \right)^{-1}.\end{aligned}$$

Hence, if each sensor is able to compute the sum of all the $F_i(0)$ and $a_i(0)$, $\forall i$, it can compute the *global* WLS solution. This is indeed something that we can do using the *average consensus*. In particular, by adopting either the *maximum-degree weight* (17.2) or the *Metropolis weights* (17.3) presented in Section 17.1 and assuming that the Theorem 12 or Theorem 13 hold, the following updating rule for the *average consensus* can be applied

$$\begin{aligned} F_i(k+1) &= F_i(k) + \sum_{j=1}^n q_{ij}(k)(F_j(k) - F_i(k)) = \\ &= q_{ii}(k)F_i(k) + \sum_{j=1, j \neq i}^n q_{ij}(k)F_j(k), \\ a_i(k+1) &= a_i(k) + \sum_{j=1}^n q_{ij}(k)(a_j(k) - a_i(k)) = \\ &= q_{ii}(k)a_i(k) + \sum_{j=1, j \neq i}^n q_{ij}(k)a_j(k). \end{aligned}$$

Hence,

$$\begin{aligned} \lim_{k \rightarrow \infty} F_i(k) &= \frac{1}{n} \sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} H(i), \\ \lim_{k \rightarrow \infty} a_i(k) &= \frac{1}{n} \sum_{i=1}^n H(i)^T C \{\varepsilon(i)\}^{-1} z(i), \end{aligned}$$

or, in other words, for any $i = 1, \dots, n$ we have

$$\hat{x}^{LS} = \lim_{k \rightarrow \infty} F_i(k)^{-1} a_i(k).$$

Moreover, at each step (i.e., before reaching the steady state), every node can still compute a *locally unbiased weighted least-square estimate* providing that $F_i(t)$ is invertible. As a consequence, *at each iteration* of the consensus protocol, a *new set of measurements* may arrive and fused coherently by means of the WLS. At steady state, all the estimates converge to the *global WLS solution*.

The interesting feature of this solution is that it can also be applied to time varying graphs, as the one reported in Figure 18.1. To this end, let us first introduce a couple of definitions.

Definition 95. Let $\mathcal{G}_i = (\mathcal{N}, \mathcal{E}_i)$, $i = 1, \dots, r$ denote a finite set of graphs sharing the same set of nodes. Their union is a graph $\mathcal{G} = \cup_{i=1}^r \mathcal{G}_i = (\mathcal{N}, \cup_{i=1}^r \mathcal{E}_i)$.

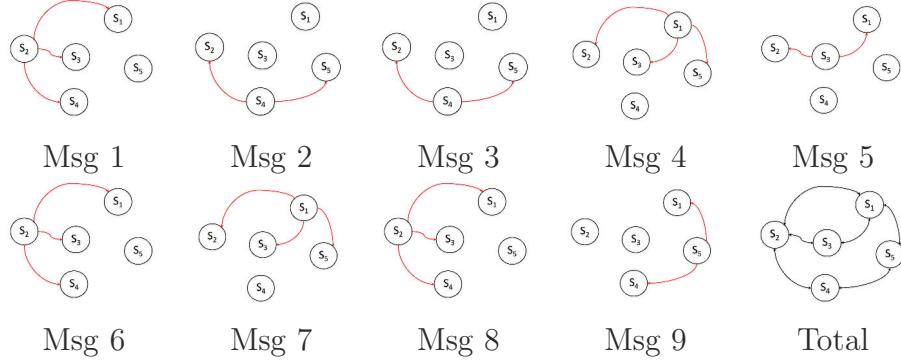


Figure 18.1: A possible sequence of a *jointly connected* graph.

This is obvious by simply computing the overall *t-step transition matrix*.

Definition 96. Let $\mathcal{G}_i = (\mathcal{N}, \mathcal{E}_i)$, $i = 1, \dots, r$ denote a finite set of graphs sharing **the same set of nodes**. The set of graphs $\{\mathcal{G}_1, \dots, \mathcal{G}_r\}$ is called **jointly connected** if their union is **strongly connected**.

We can now state that the graph is *connected in the long run* if the *infinitely occurring* communication graphs are *jointly connected*, hence proposing an extension to Theorem 12. More precisely, for any graph with *switching topology*, there is a finite set of r graphs describing its configuration (at most, they are all the possible combination of communication links). Therefore there is a finite set of associated r weight matrices Q_i , determined either with the *maximum degree* or the *Metropolis* algorithms. Then the dynamic becomes

$$x(k+1) = Q_{i(k)}x(k), \text{ with } 1 \leq i(k) \leq r \text{ for all } k.$$

The sequence $\{i(k)\}_{k=0}^{\infty}$ is such that a subset of all the graphs occurs *infinitely often*, which is the basis of the following theorem [38].

Theorem 17. If the collection of communication graphs that occur infinitely often are **jointly connected**, then the system

$$x(k+1) = Q_{i(k)}x(k), \text{ with } 1 \leq i(k) \leq r \text{ for all } k,$$

converges to

$$\lim_{k \rightarrow \infty} x(k) = \left(\frac{1}{n} \mathbf{1}^T x(0) \right) \mathbf{1},$$

or, equivalently

$$\lim_{k \rightarrow +\infty} \Phi(k) = \frac{1}{n} \mathbf{1} \mathbf{1}^T.$$

This theorem is given without proof, which is however based on the existence of *paracontracting matrices* [38], i.e.,

$$Mx \neq x \Leftrightarrow \|Mx\| < \|x\|.$$

18.2 Distributed Kalman Filter

Let us consider the problem of obtaining a *distributed estimate* of a linear dynamic system. An example can be the estimate of an agent (e.g. a person) moving in an environment equipped with a surveillance system equipped with a network of n sensors (e.g. surveillance cameras). As described in Section 13.1.2, the model can be given by

$$x(k+1) = Ax(k) + Bu(k).$$

For the distributed Kalman filter, the model turns to

$$\hat{x}(k+1) = A\hat{x}(k) + \nu(k),$$

where $\nu(k) \sim \mathcal{N}(0, Q)$ models the uncertainties in the motion of the target. Notice that this is quite obvious if the target is a person, but it is the same if the target is a robot, since it is not possible to get access to the *proprioceptive sensor data*, e.g. odometry. For the measurement we have then

$$z_i(k) = H_i x_i(k) + \varepsilon_i(k),$$

where $\varepsilon_i(k) \sim \mathcal{N}(0, R_i)$ models the measurement error. As customary in the Kalman filter literature, we assume the noises white and uncorrelated.

For the *centralised Kalman filter* we would have

- *Prediction step:*

$$\begin{aligned}\hat{x}(k+1)^- &= A\hat{x}(k) \\ P(k+1)^- &= AP(k)A^T + Q\end{aligned}$$

- *Update step:*

$$\begin{aligned}S(k+1) &= HP(k+1)^-H^T + R \\ W(k+1) &= P(k+1)^-H^T S(k+1)^{-1} \\ \hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)(z(k+1) - H\hat{x}(k+1)^-) \\ P(k+1) &= (I - W(k+1)H)P(k+1)^-\end{aligned}$$

where $R = \text{diag}(R_1, R_2, \dots, R_n)$.

The update step equations can be rewritten more compactly as

$$\begin{aligned}\hat{x}(k+1) &= \hat{x}(k+1)^- + P(k+1)^- H^T (HP(k+1)^- H^T + R)^{-1} \cdot \\ &\quad \cdot (z(k+1) - H\hat{x}(k+1)^-) \\ P(k+1) &= P(k+1)^- - P(k+1)^- H^T (HP(k+1)^- H^T + R)^{-1} \cdot \\ &\quad \cdot HP(k+1)^-\end{aligned}$$

Using the *matrix inversion lemma*:

$$(A + BCB^T)^{-1} = A^{-1} - A^{-1}B(B^TA^{-1}B + C^{-1})^{-1}B^TA^{-1},$$

we can further elaborate on the update equations as follows for the state

$$\begin{aligned}\hat{x}(k+1) &= \hat{x}(k+1)^- + P(k+1)^- H^T (HP(k+1)^- H^T + R)^{-1} \cdot \\ &\quad \cdot (z(k+1) - H\hat{x}(k+1)^-) = \\ &= P(k+1) \left(P(k+1)^{-1} \hat{x}(k+1)^- + H^T R^{-1} z(k+1) \right) = \\ &= P(k+1) \left(P(k+1)^{-1} \hat{x}(k+1)^- + \sum_{i=1}^n H_i^T R_i^{-1} z_i(k+1) \right) = \\ &= P(k+1) \left(P(k+1)^{-1} \hat{x}(k+1)^- + a(k+1) \right)\end{aligned}$$

where

$$a(k+1) = \sum_{i=1}^n H_i^T R_i^{-1} z_i(k+1),$$

while for the covariance

$$\begin{aligned}P(k+1) &= P(k+1)^- - P(k+1)^- H^T (HP(k+1)^- H^T + R)^{-1} \cdot \\ &\quad \cdot HP(k+1)^- = \\ &= \left(P(k+1)^{-1} + H^T R^{-1} H \right)^{-1} = \\ &= \left(P(k+1)^{-1} + \sum_{i=1}^n H_i^T R_i^{-1} H_i \right)^{-1} = \\ &= \left(P(k+1)^{-1} + F(k+1) \right)^{-1}\end{aligned}$$

where

$$F(k+1) = \sum_{i=1}^n H_i^T R_i^{-1} H_i = F.$$

Notice that F is independent from k if the number of sensors does not change.
To summarise:

$$\begin{aligned}\hat{x}(k+1) &= P(k+1) \left(P(k+1)^{-1} \hat{x}(k+1)^- + a(k+1) \right) \\ P(k+1) &= \left(P(k+1)^{-1} + F \right)^{-1}\end{aligned}$$

Notice that the problem is quite similar to the WLS case in Section 18.1. In particular, to correctly execute the Kalman filter, only $a(k+1)$ and F involves *message exchanges*, while the other quantities can be computed locally. The positive aspect is that $a(k+1)$ and F can be simply computed using *average consensus*. So, as in the WLS case, we start by initialising for the i -th node:

$$a_i(k+1, 0) = H_i^T R_i^{-1} z_i(k+1) \text{ and } F_i(0) = H_i^T R_i^{-1} H_i,$$

where the 0 stands for the initialisation of the *average consensus* distributed state. Hence, using the *average linear consensus*, we have

$$\begin{aligned}\lim_{q \rightarrow \infty} F_i(q) &= \frac{1}{n} \sum_{i=1}^n H_i^T R_i^{-1} H_i = \frac{F}{n} \\ \lim_{q \rightarrow \infty} a_i(k+1, q) &= \frac{1}{n} \sum_{i=1}^n H_i^T R_i^{-1} z_i(k+1) = \frac{a(k+1)}{n}\end{aligned}$$

where q is the number of messages exchanged for the consensus protocol. For the correct application of the Kalman Filter we have to assume that *each sensor node knows the number of nodes n* in order to update the filter coherently, i.e. for the i -th sensor:

$$\begin{aligned}\hat{x}_i(k+1) &= P_i(k+1) \left(P_i(k+1)^{-1} \hat{x}_i(k+1)^- + n a_i(k+1, q) \right) \\ P_i(k+1) &= \left(P_i(k+1)^{-1} + n F_i(q) \right)^{-1}.\end{aligned}$$

Therefore, before computing the *Update step*, the nodes run a consensus algorithm to reach an agreement for $a(k+1)$ and F . Of course, an agreement is reached after a sufficiently large number of consensus iterations q , theoretically infinite. If this is not the case, a specific algorithm needs to be designed, such as *finite step convergence* or alternative Kalman approaches.

Chapter 19

Introduction to Distributed Control

19.1 Distributed coverage control with Voronoi-based tessellation

K-nearest neighbours is probably the simplest *non-parametric classifier* that can be defined. KNN searches for the k points that are closer to a test point q , count how many members of a certain class belong to the set and compute the related probability using a *frequentist* approach. It will be presented in details in Chapter ???. A KNN classifier with $k = 1$ induces a *Voronoi tessellation*: at any point $x_i \in \mathcal{S}$ (with $x_i \in \mathbb{R}^l$) is associate a region $V(x_i)$ such that:

$$V(x_i) = \{q \in \mathbb{R}^l \mid \|q - x_i\| \leq \|q - x_j\|, \forall x_j \neq x_i \text{ and } x_j \in \mathcal{S}\}.$$

An example of a Voronoi tessellation is reported in Figure 19.1.

Consider a mission space $\mathcal{Q} \in \mathbb{R}^2$ and a set of n *agents* (i.e., robots, nodes, sensors), we identify the i -th agent position with $p_i = [x_i, y_i]^T$. The i -th Voronoi cell is defined as

$$\mathcal{V}_i = \{q \in \mathcal{Q} \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\},$$

where $q \in Q$ is a generic point belonging to the mission space. In other words, the distance between the i -th agent position p_i and each point q that belongs to the i -th Voronoi cell is smaller or equal than the distance between q and any other node position p_j . We consider the following first order dynamics for each agent

$$\dot{p}_i = u_i, \quad \forall i = 1, \dots, n.$$

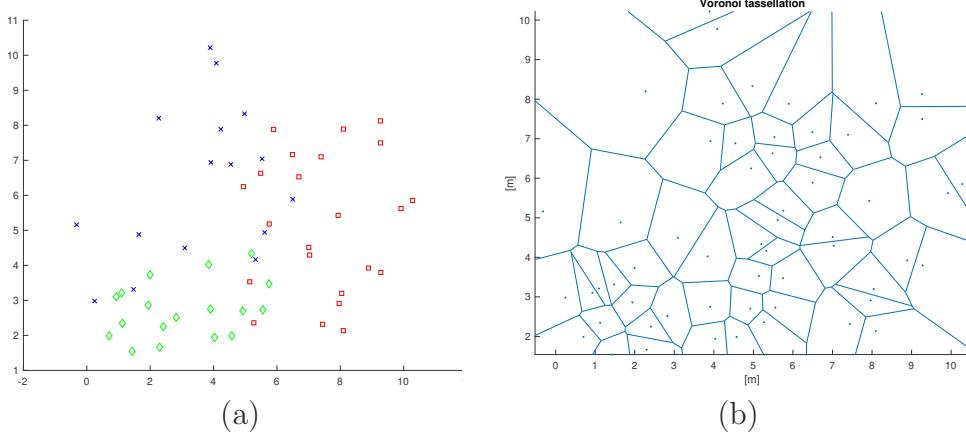


Figure 19.1: Given the *training data* (a) we can easily obtain the Voronoi tessellation (b) for the same example described previously.

We consider the expression for the *mass* of the i -th Voronoi cell

$$M_{\mathcal{V}_i} = \int_{\mathcal{V}_i} dq.$$

Moreover, we consider the expression of the *centroid position* of the i -th Voronoi cell

$$C_{\mathcal{V}_i} = \frac{1}{M_{\mathcal{V}_i}} \int_{\mathcal{V}_i} q dq.$$

It is then possible to define the following *coverage cost metric*

$$J_{\text{cov}}(p, \mathcal{V}) = \sum_{i=1}^n \int_{\mathcal{V}_i} \|q - p_i\|^2 dq,$$

which relates to a *static coverage problem*: *the system reaches a stable configuration that deploys the agents in the mission space so that a coverage metric (e.g. $J_{\text{cov}}(p, \mathcal{V})$) is optimised*. Of course, the minimum is $J_{\text{cov}}(p, \mathcal{V})$, the higher is the coverage.

The following Lloyd-based proportional control law

$$\dot{p}_i(\mathcal{V}_i) = -k_p (p_i - C_{\mathcal{V}_i}),$$

where $k_p > 0$ is a tuning parameter, ensures the *convergence* of each agent on its *Voronoi partition centroid*, thus ensuring the *optimal deployment for the coverage problem*. The proof is technically given by using $J_{\text{cov}}(p, \mathcal{V})$ as a *Lya-punov function* and then by applying the LaSalle's invariance principle [39].

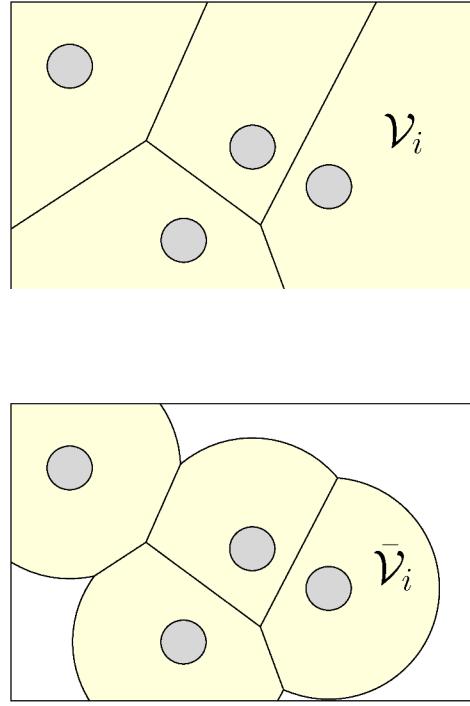


Figure 19.3: The sensing range limited Voronoi tessellation.

19.1.1 Application to actual agents

Let us consider the n nodes as a *mobile sensor network*, with limited sensing range R_s and communication range R_c for each sensor, and let us assume that $R_c > 2R_s$. In this case, we can define the i -th *sensing range limited Voronoi cell* as the intersection of the Voronoi cell \mathcal{V}_i and the *closure* of the sensing range circle, which we indicate with $\bar{\mathcal{C}}_{i,R_s}$, i.e.

$$\bar{\mathcal{V}}_i = \{\mathcal{V}_i \cap \bar{\mathcal{C}}_{i,R_s}\}.$$

The Voronoi original tessellation and the effect of the sensing range are depicted in Figure 19.2 and in Figure 19.3, respectively. The construction of the sensing range limited Voronoi diagram can be easily done in a *distributed fashion* under the condition $R_c > 2R_s$, i.e. each agent computes its own Voronoi cell \mathcal{V}_i only *on the basis of the neighbours positions*. The neighbours set is defined as

$$\mathcal{N}_i = \{p_j \in \mathbb{R}^2 \mid \|p_i - p_j\| < R_c\}.$$

Indeed, when the i -th agent knows the neighbours positions, it can test all the points $q \in \bar{\mathcal{C}}_{i,R_s}$, if the condition $\|q - p_i\| \leq \|q - p_j\|, \forall j \in \mathcal{N}_i$ is verified, then the point q belongs to the i -th cell.

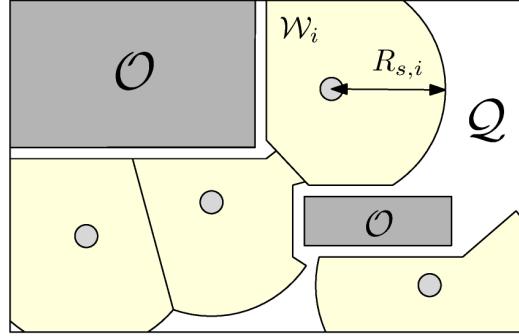


Figure 19.4: Voronoi-Visible tessellation.

19.1.2 Extensions

The previous control law works fine in a *convex space*, i.e. without the presence of obstacles. Let \mathcal{O} be the *obstacle set* and let \mathcal{S}_i be the i -th agent *visibility set*, i.e.

$$\mathcal{S}_i = \{q \in \mathcal{Q} \mid p_i - \gamma(p_i - q) \notin \mathcal{O}\}, \quad \forall \gamma \in [0, 1],$$

line of sight space. By intersect the visibility set with $\bar{\mathcal{V}}_i$, we obtain the *Voronoi-visible set*

$$\mathcal{W}_i = \{\bar{\mathcal{V}}_i \cap \mathcal{S}_i\},$$

as depicted in Figure 19.4. Notice that the *encumbrance of the agents* is managed by *inflating the obstacle dimensions* of an amount equal to the occupancy radius of the agent. We can now compute the Lloyd-based control law on this set, i.e.

$$\dot{p}_i(\mathcal{W}_i) = -k_p(p_i - C_{\mathcal{W}_i}).$$

Further modifications ensures *absence of collisions between the agents of the group and with other moving obstacles*. To this end, assume that the i -th agent can be approximated with a circle of radius δ_i . Hence the Voronoi tessellation can be modified as follows

$$\tilde{\mathcal{V}}_i = \begin{cases} \{q \in \mathcal{Q} \mid \|q - p_i\| \leq \|q - p_j\|\}, & \text{if } \Delta_{ij} \leq \frac{\|p_i - p_j\|}{2} \\ \{q \in \mathcal{Q} \mid \|q - p_i\| \leq \|q - \tilde{p}_j\|\}, & \text{otherwise} \end{cases}$$

where $\Delta_{ij} = \delta_j + \delta_i$ and $\tilde{p}_j = p_j + 2\left(\Delta_{ij} - \frac{\|p_i - p_j\|}{2}\right)\frac{p_i - p_j}{\|p_i - p_j\|}$. The dynamic obstacles situation is depicted in Figure 19.5.

To ensure that the agents move towards a *desired goal position*, we can modify the concept of *distance* adopted in the Voronoi tessellation. Consider *pdf-like weighting factor* $\varphi(q)$ defined in the operational space \mathcal{Q} , i.e it can

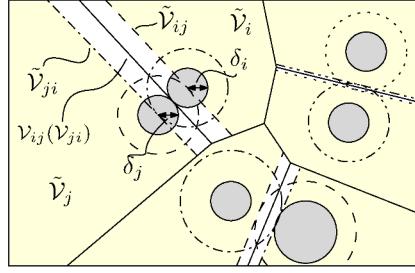


Figure 19.5: Modified Voronoi tessellation to account for dynamic obstacles.

be a *bivariate Gaussian pdf* with mean value centred in the desired position. Hence, we can modify the idea of *mass* and centroid given previously as:

$$M_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \varphi(q) dq,$$

i.e. the *normalisation factor of a pdf* computed in the cell \mathcal{V}_i . Moreover, we consider the expression of the *centroid position* of the i -th Voronoi cell

$$C_{\mathcal{V}_i} = \frac{1}{M_{\mathcal{V}_i}} \int_{\mathcal{V}_i} \varphi(q) q dq,$$

i.e. the *average value* computed in the cell \mathcal{V}_i . The extension to static and dynamic obstacles applies then straightforwardly. A snapshot of one experiment, with all the depicted quantities can be found in Figure 19.6.

Given the free mission space $\mathcal{Q} \in \mathbb{R}^2$, the obstacle space $\mathcal{O} \in \mathbb{R}^2$ and a finite number n of robotic agents with initial position $p_i(0) \in \mathcal{Q} \ \forall i = 1 \dots n$, it is possible to design control laws that operate through local interactions so that the collective (global) behaviour of the group *additionally* fulfils one of the following goals:

- *Connectivity maintenance*: given an initial connectivity graph defined by edges and vertices, the robots move so that the number of edges in the connectivity graph is a non-decreasing function of time and if two vertices are linked through an edge, they remain so;
- *Exploration*: the robot motion strategy guarantees that the entire mission space is covered by the sensing range of the robots in finite time; if the number of robots is not sufficient to cover the space, the different locations can be covered at different times;
- *Navigation*: i) Flocking: move the robotic agents from the starting positions $\mathcal{P} = \{p_1(0), p_2(0), \dots, p_n(0)\}$ to the goal positions $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$;

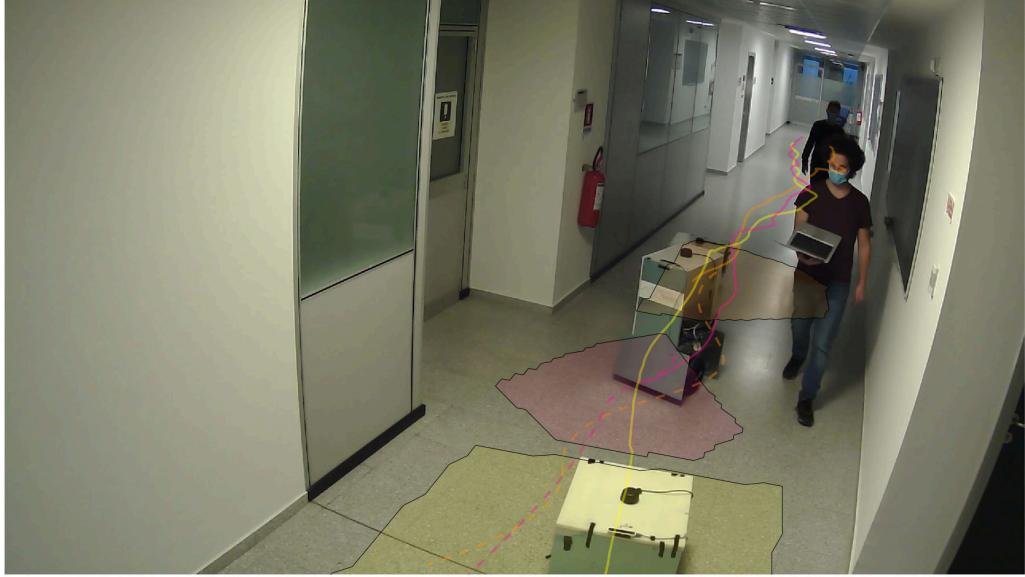


Figure 19.6: Safe navigation in complex environment with three robots in the DII department.

- ii) Formation: move the robotic agents from the starting positions $\mathcal{P} = \{p_1(0), p_2(0), \dots, p_n(0)\}$ to the goal positions $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$ keeping constant pairwise distance between the agents;
- *Rendezvous*: each robot of the system converges to the same rendezvous position.

The interested reader is forwarded to [40, 41] and the other papers from the same Authors for additional information and insights.

19.2 Linear consensus protocol for second-order dynamics systems

What follows is derived from [42] and developed by Edoardo Pagot, a former student of this course. The second-order consensus protocol proposed in [42] will be first summarised. A broad category of vehicles can be modelled using second-order dynamics, where often first-order dynamics is not sufficient . We consider the following second-order dynamics system:

$$\begin{aligned}\dot{\xi}_i &= \zeta_i, \\ \dot{\zeta}_i &= u_i.\end{aligned}\tag{19.1}$$

For a generic vehicle ξ, ζ, u can be seen as position, velocity and actuation forces/momenta respectively. Then, the following consensus protocol is applied

$$u_i = - \sum_{j=1}^n g_{ij} k_{ij} [(\xi_i - \xi_j) + \gamma(\zeta_i - \zeta_j)], i \in \{1, 2, \dots, n\}, \quad (19.2)$$

where k_{ij} is a positive number acting as a scaling factor for the information flowing from node j to node i ; γ is a scaling factor, $g_{ii} = 0$ and $g_{ij} = 1$ if there is communication flow from node j to node i , otherwise it is 0. This consensus protocol should guarantee that:

$$|\xi_i - x_{ij}| \rightarrow 0 \text{ as } t \rightarrow \infty \text{ and } |\zeta_i - \zeta_j| \rightarrow 0 \text{ as } t \rightarrow \infty.$$

When (19.2) is substituted as control in (19.1) the latter can be written in matrix form as

$$\begin{bmatrix} \dot{\xi} \\ \dot{\zeta} \end{bmatrix} = \Gamma \begin{bmatrix} \xi \\ \zeta \end{bmatrix}, \quad (19.3)$$

with

$$\Gamma = \begin{bmatrix} 0_{n \times n} & I_n \\ -L & -\gamma L \end{bmatrix}.$$

19.2.1 Time invariant information exchange topologies: convergence analysis

It can be shown that:

$$\lambda_{i\pm} = \frac{\gamma\mu_i \pm \sqrt{\lambda^2\mu_i^2 + 4\mu_i}}{2}, \quad (19.4)$$

where μ_i is the i -th eigenvalue of L and $\lambda_{i+}, \lambda_{i-}$ are the eigenvalues of Γ associated with μ_i . Since the row sums of $-L$ are equal to zero, the matrix has at least one zero eigenvalue with an associated eigenvector 1. Consequently (19.4) implies that Γ has at least two zero eigenvalues. Moreover Gershgorin disc theorem assures that all the non-zero eigenvalues of $-L$ have negative real parts since $-L$ is diagonally dominant and has non-positive diagonal elements.

We now quote Lemma 4.1 from [42].

Lemma 18. *Consensus protocol from (19.2) achieves consensus asymptotically if and only if matrix Γ has exactly two zero eigenvalues and all the other eigenvalues have negative real parts. Specifically, $\xi \rightarrow 1p^T \xi(0) + t1p^T \xi(0)$ and $\zeta \rightarrow 1p^T \zeta(0)$ for large t , where $p \in \mathbb{R}$ is a non-negative left eigenvector of L associated with eigenvalue 0 and $p^T 1 = 1$.* \square

However, several examples in [42] show that some non-zero eigenvalues of Γ may have positive real parts even if all non-zero eigenvalues of $-L$ have negative real parts: in this cases consensus cannot be achieved. In addition, it is shown that topologies containing a (directed) spanning tree may not reach consensus for (generally) small values of λ . Those consideration are described formally in the following lemmas and theorems: as usual, the interested reader can found the proofs in [42].

Lemma 19. *If $-L$ has a simple zero eigenvalue and all the other eigenvalues are real, consensus protocol from (19.2) achieves consensus for any $\gamma \neq 0$.*

The latter is a particular case, in general we have:

Lemma 20. *Matrix L of a directed weighted graph has a simple zero eigenvalue and all the other eigenvalues have positive real parts if and only if the graph has a (directed) spanning tree.*

Theorem 21. *Consensus protocol from (19.2) achieves consensus asymptotically only if the information exchange topology has a (directed) spanning tree.*

This is a necessary condition for second-order consensus protocol, while it is necessary and sufficient condition for fist-order consensus protocol. Theorem (21) can be proven from Lemma 18 and Lemma 20. Moreover:

Theorem 22. *Consensus protocol from (19.2) achieves consensus asymptotically if the information exchange topology has a (directed) spanning tree and*

$$\gamma = \max_{\mu_i \neq 0} \sqrt{\frac{2}{|\mu_i| \cos(\frac{\pi}{2} - \tan^{-1} \frac{-\Re(\mu_i)}{\Im(\mu_i)})}},$$

where $\mu_i = 1, \dots, n$ are the eigenvalues of $-L$.

Finally, the following lemma gives us information about the final consensus value.

Lemma 23. *Suppose that Γ has exactly two zero eigenvalues and all the other eigenvalues have negative real parts. If $\zeta_i(0) = 0$, $i \in \mathcal{F}$, then as $t \rightarrow \infty$, $\xi_i(t) \rightarrow \sum_{i=1}^n p_i \xi_i(0)$ and $\zeta_i(t) \rightarrow 0$, where $i \in \mathcal{F}$ and $p = [p_1, \dots, p_n]^T$ is a non-negative left eigenvector of $-L$ associated with eigenvalue 0 satisfying $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$. In addition, if $\zeta_i(0) = 0$, $i \in \mathcal{F}_L$, where \mathcal{F}_L denotes the set of vehicles that have a directed path to all the other vehicles in the information exchange topology, then $\xi_i(t) \rightarrow \sum_{i \in \mathcal{F}_L} p_i \xi_i(0)$ and $\zeta_i(t) \rightarrow 0$, as $t \rightarrow \infty$.*

Notice that the final positions are a function of the initial positions. Usually we want the agents to reach an arbitrary position regardless the starting points. In order to do this, the authors in [42] propose the following second order consensus protocol.

$$u_i = -\alpha \zeta_i - \sum_{j=1}^n g_{ij} k_{ij}[(\xi_i - \xi_j) + \gamma(\zeta_i - \zeta_j)], \quad (19.5)$$

where $\alpha > 0$. The dynamic system can be written in matrix form as

$$\begin{bmatrix} \dot{\xi} \\ \dot{\zeta} \end{bmatrix} = \Sigma \begin{bmatrix} \xi \\ \zeta \end{bmatrix}, \text{ with } \Sigma = \begin{bmatrix} 0_{n \times n} & I_n \\ -L & -\alpha I_n - \gamma L \end{bmatrix}.$$

In this case we have:

$$\rho_{i\pm} = \frac{\gamma\mu_i - \alpha \pm \sqrt{(\gamma\mu_i - \alpha)^2 + 4\mu_i}}{2},$$

where ρ_{i+}, ρ_{i-} are called eigenvalues of Σ that are associated with μ_i .

The new consensus protocol convergence is described by the following lemma and theorem, which are proven in [42].

Lemma 24. *Let p be a non-negative left eigenvector of $-L$ associated with eigenvalue 0 and $p^T 1 = 1$. With consensus protocol of (19.5), $\xi(t) \rightarrow 1p^T \xi(0) + (\frac{1}{\alpha} 1p^T \zeta(0))$ and $\zeta(t) \rightarrow 0$ asymptotically as $t \rightarrow \infty$ if and only if matrix Σ has a simple zero eigenvalue and all the other eigenvalues have negative real parts.*

Theorem 25. *Let p be defined as in Lemma 24. Under the conditions in Theorem 22, consensus protocol in (19.5) guarantees that $\xi \rightarrow 1p^T \xi(0) + (\frac{1}{\alpha} 1p^T \zeta(0))$ and $\zeta \rightarrow 0$ asymptotically.*

19.2.2 Switching information exchange topologies: convergence analysis

A convergence analysis for switching information exchange topologies is briefly reported for consensus protocol (19.2): the analysis for consensus protocol (19.5) is very similar and therefore omitted in [42].

Simple Case

If the information exchange topology is undirected and active below a certain distance threshold between the vehicle, we have the following Lemma.

Lemma 26. *If the (time-varying) information exchange topology is undirected and connected at each time, consensus protocol from (19.2) achieves consensus asymptotically.*

General Case

In the general case the information exchange topology is switching randomly with time. Hence, (19.3) is rewritten as

$$\begin{bmatrix} \dot{\xi} \\ \dot{\zeta} \end{bmatrix} = \Gamma_\sigma \begin{bmatrix} \xi \\ \zeta \end{bmatrix}, \quad (19.6)$$

where $\sigma \in [0, \infty) \rightarrow \mathcal{P}$ is a piecewise constant switching signal with switching times t_0, t_1, \dots , and \mathcal{P} denotes a set indexing the class of all possible directed information exchange topologies for the n vehicles that have a (directed) spanning tree. Here, as in [42], we assume that $\Gamma(t)$ is piecewise constant and satisfies $\Gamma(t) = \Gamma(t_i)$, $t \in [t_i, t_{i+1})$. Therefore, (19.6) can be rewritten in terms of consensus error variable vectors

$$\begin{bmatrix} \dot{\tilde{\xi}} \\ \dot{\tilde{\zeta}} \end{bmatrix} = \Delta_\sigma \begin{bmatrix} \tilde{\xi} \\ \tilde{\zeta} \end{bmatrix}, \quad (19.7)$$

where Δ_σ is a $2(n - 1) \times 2(n - 1)$ matrix that can be derived from Γ_σ . If Δ_σ is stable, we can find $a_\sigma \geq 0$ and $\chi_\sigma > 0$ s.t. $\|e^{\Delta_\sigma t}\| \leq e^{(a_\sigma - \chi_\sigma t)}$, $t \geq 0$. Finally, the following theorem describes 19.7 convergence.

Theorem 27. *Let t_0, t_1, \dots be the times when the information exchange topology switches. Also, let τ be the dwell time such that $t_{i+1} - t_i \geq \tau, \forall i = 0, 1, \dots$. If the information exchange topology has a (directed) spanning tree for each $t \in [t_i, t_{i+1})$, the condition for γ in Theorem 22 is satisfied for each Γ_σ , where $\sigma \in \mathcal{P}$, and the dwell time τ satisfies $\tau > \sup_{\sigma \in \mathcal{P}} \{a_\sigma / \chi_\sigma\}$, then consensus protocol (19.2) achieves consensus asymptotically and is robust to information exchange noise under switching information exchange topologies.*

19.2.3 Vehicle Model

In this section details about the dynamics model of the mobile robots will be given. Moreover, the second-order consensus protocol for destination reaching and formation shape will be given.

The vehicles are differential-wheeled robots. The equation of motion for the i -th mobile robot are the following:[42]

$$\begin{aligned}\dot{x}_i &= v_i \cos(\theta_i), \\ \dot{y}_i &= v_i \sin(\theta_i), \\ \dot{\theta}_i &= \omega_i, \\ m_i \dot{v}_i &= f_i, \\ J_i \dot{\omega}_i &= \tau_i,\end{aligned}\tag{19.8}$$

where (x_i, y_i) is the vehicle centre position, θ_i is the vehicle orientation, v_i the linear velocity and ω_i the angular velocity, m_i and J are the vehicle mass and inertia respectively. Finally, f_i and τ_i are the driving force and torque acting on the robot.

The constraints in (19.8) are non-holonomic: indeed, the single mobile robot has three DOFs (planar motion: two translation and one rotation), but only two controls (longitudinal force and yaw moment). As shown in [42], if we rewrite the equation of motion around the wheel axis centre and after a rearrangement of f_i and τ_i formulation, we obtain

$$\begin{aligned}\ddot{x}_{hi} &= v_{xi}, \\ \ddot{y}_{hi} &= v_{yi},\end{aligned}\tag{19.9}$$

which are the equations of motions for an holonomic robot. The position of the robot wheel axle centre is given by (x_{hi}, y_{hi}) . Notice that now we have information about the robot position only, and not about its orientation.

Consensus protocol

In [42] the following control law for v_{xi} and v_{yi} is proposed

$$\begin{aligned}v_{xi} &= -\alpha_x(x_{hi} - x_{hi}^d) - \gamma_x \alpha_x x_{hi} - \sum_{j=1}^n g_{ij} k_{ij}[(x_{hi}) - (x_{hi}^d)] + \sum_{j=1}^n g_{ij} \gamma_x k_{ij}(\dot{x}_{hi} - \dot{x}_{hj}), \\ v_{yi} &= -\alpha_y(y_{hi} - y_{hi}^d) - \gamma_y \alpha_y y_{hi} - \sum_{j=1}^n g_{ij} k_{ij}[(y_{hi}) - (y_{hi}^d)] + \sum_{j=1}^n g_{ij} \gamma_y k_{ij}(\dot{y}_{hi} - \dot{y}_{hj}),\end{aligned}$$

where $\alpha_{x,y}$ and $\gamma_{x,y}$ are positive coefficients and (x_{hi}^d, y_{hi}^d) is the desired final position for the i -th robot. In (19.10), the first two terms guarantees that each robot reaches the final destination while the last two terms guarantees the formation keeping. Once controls (19.10) are substitute in (19.9), it is possible to rewrite the ODE system in matrix form and in terms of goal seeking error as

$$\begin{aligned}\begin{bmatrix} \dot{x}_e \\ \ddot{x}_e \end{bmatrix} &= \begin{bmatrix} 0_{n \times n} & I_n \\ -(L + \alpha_x I_n) & -\gamma_x(L + \alpha_x I_n) \end{bmatrix} \begin{bmatrix} x_e \\ \dot{x}_e \end{bmatrix}, \\ \begin{bmatrix} \dot{y}_e \\ \ddot{y}_e \end{bmatrix} &= \begin{bmatrix} 0_{n \times n} & I_n \\ -(L + \alpha_y I_n) & -\gamma_y(L + \alpha_y I_n) \end{bmatrix} \begin{bmatrix} y_e \\ \dot{y}_e \end{bmatrix},\end{aligned}\tag{19.10}$$

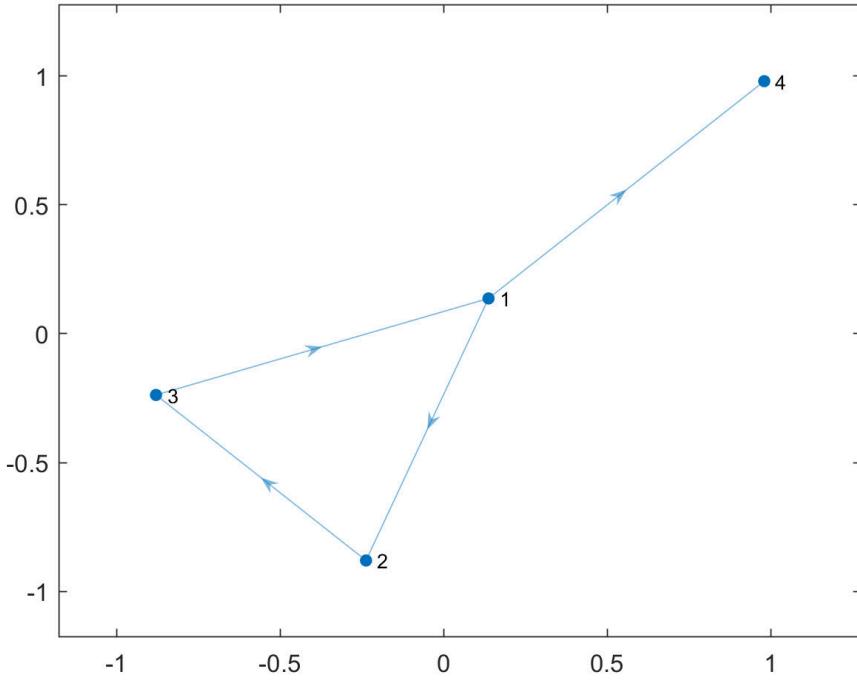


Figure 19.7: Information exchange topology of the flock.

where $x_e = [x_{e1}, \dots, x_{en}]^T$ and $y_e = [y_{e1}, \dots, y_{en}]^T$, with $x_{ei} = x_{hi} - x_{hi}^d$ and $y_{ei} = y_{hi} - y_{hi}^d$ the final destination errors for the i -th mobile robot.

Simulation

The flock model described by (19.10) is implemented using a custom Matlab function. This function is used to integrate the system of ODE describing the model and receives the Laplacian matrix of the flock and the weight coefficient of the second order consensus protocol as parameters. A simulation is performed for a flock of four agents with information exchange topology as in Figure 19.7 and $\alpha_x = \alpha_y = \gamma_x = \gamma_y = 1$. With the controls are bounded in the interval $[-2, 2]$, we obtain the results of Figure 19.8 and Figure 19.9.

In particular, in Figure 19.9 we can see the flock trajectories, for every solver time step lines joining the agents are plotted: in this way we can see the shape of the formation, noticing that is square for almost all the time. Moreover, in Figure 19.8 we can see that the agents reach consensus on the final position error (that must become zero).

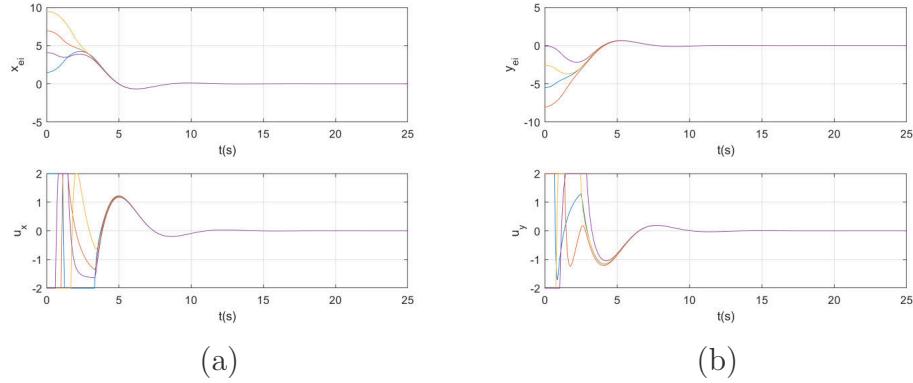


Figure 19.8: Goal reaching errors (a) and control inputs (b) of the flock agents.

19.2.4 Communication Simulation

For the previous models, we assumed that when two nodes (agents) are connected then the communication is always present and instantaneous. In this section more realistic communication situations will be modelled and simulated.

Discrete-Time Controls

Assuming instantaneous communication, the micro-controller aboard the vehicle must process the data and compute and send inputs to the actuators. Consequently, in the real world the control inputs will be discrete. This behaviour is implemented using a *Zero-Order Hold* block, where the discretisation time (t_s) can be chosen as depicted in Figure 19.10. It was verified that the consensus algorithm remains robust for sample times smaller than about 0.2s. This discretisation time threshold is very variable and depends upon several factors, like control consensus protocol parameters ($\alpha_x, \alpha_y, \gamma_x, \gamma_y$), control upper and lower bounds and manoeuvre complexity.

Convergence analysis

In order to verify the goodness of the control algorithm we need a performance index: the time to converge to the final destination was chosen. At every time instant the distance error module is computed for each agent as

$$|Error_i| = \sqrt{x_{e,i}^2 + y_{e,i}^2} \quad (19.11)$$

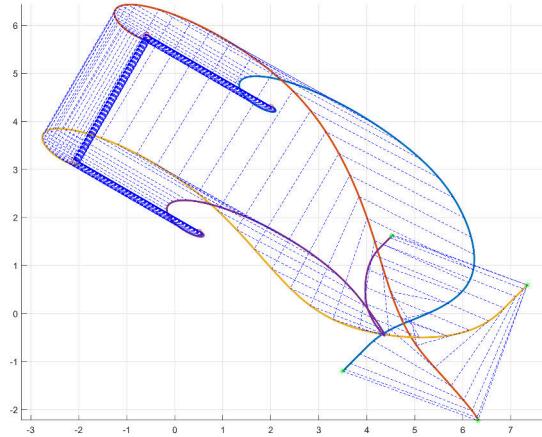


Figure 19.9: Trajectory of the vehicle flock: green asterisks are the starting points, red circles are the final destinations.

Then the average error of the flock is taken. When the average error goes below an arbitrary value we say that convergence is reached and the crossing instant is the convergence time t_{conv} . The convergence threshold was set at 5cm. An example is shown in Figure 19.11.

Once that a performance index is set, we can compute the control algorithm performance as a function of the control discretisation time t_s reported in Figure 19.12. The simulation was performed with the following data: $\alpha_x = 5, \alpha_y = 5, \gamma_x = 5, \gamma_y = 5, k_{i,j} = 5$, with information exchange topology as in Figure 19.7 and initial and final positions as in Figure 19.9. From Figure 19.12 we can see that in general the convergence time tends to a constant value for small t_s , while it grows exponentially for large t_s until convergence above the chosen error threshold is not reached any more.

Random message loss

The first communication model will implement random message losses, that is the message will be successfully sent and received with a fixed probability p . We suppose that at each time instant $t_i = i \cdot t_s$ each mobile robot can read its own position (to compute the distance errors x_e, y_e from the final destination) and the relative distance from the other mobile robots, the latter depending on the information exchange topology, and those informations are sent to a central computing unit, which computes the inputs for each robot using algorithm from (19.10) and finally sends them back to the robots.

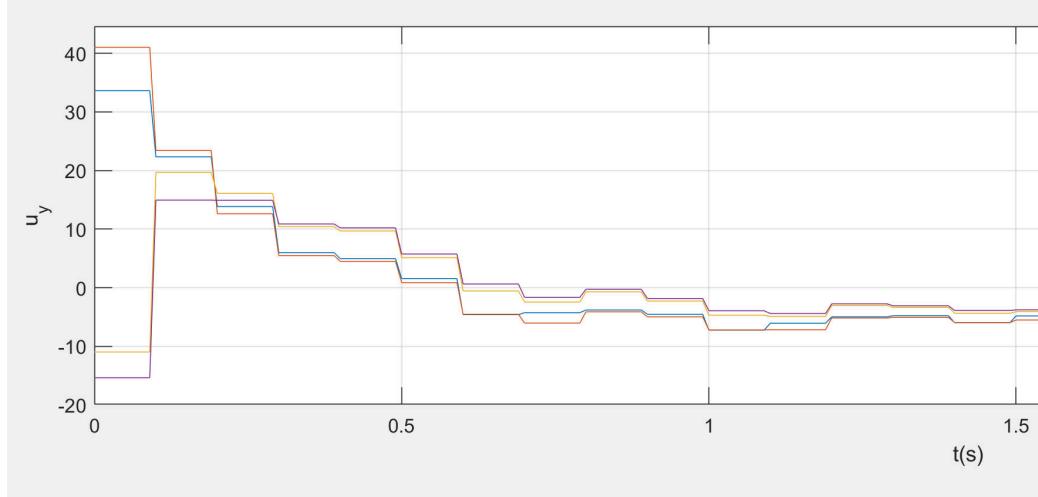


Figure 19.10: Example of discrete-time control from the simulation with sample time $t_s = 0.1s$.

Thus, assuming that $p = 0.5$, there will be 50% probability that the aforementioned process is going to be successful, which means that the i -th agent will receive its input and maintains it for a time t_s . When the communication fails, the input will be zero and robot will keep the velocity and attitude from the previous input: another solution could be keeping the old input until the new input is received, but this implementation tends to instability when there are large delays between messages.

Implementation

In order to implement the random message loss feature, we need to create a biased coin, i.e. a system which generates two results, 1 (for "message received") and 0 (for "message lost"), and the 1 result appears with an arbitrary probability. To obtain this, firstly we need the *Random Number* block, which generates a random numbers from a Gaussian Normal Distribution: we choose zero mean and standard deviation (σ) equal to one. In this way, using a comparator after the *Random Number* block, we can generate our two results with the desired probability. Example: if we set the comparator to return 1 when the absolute value of the random number generated from the normal distribution is smaller than σ (and $\sigma = 1$ from the previous definition), then we will get 1 with a probability of about 68%.

To choose the probability value arbitrary, the *Inverse error function is used*, from which we can compute the set value for the comparator in order

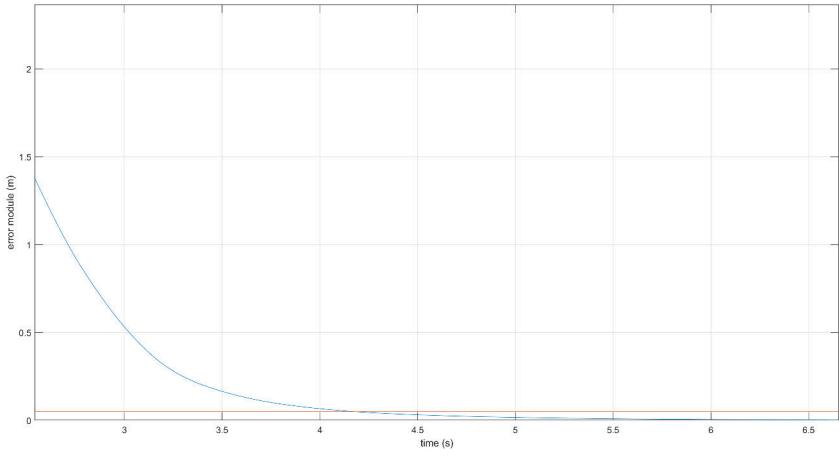


Figure 19.11: Destination reaching average error of the flock: chosen convergence value is shown as a red line.

to get the desired value of probability for the 1 result. Finally, each agent control is multiplied by a different random message state generator, so that it will be multiplied by 1 in case of communication success or by 0 in case of communication fail.

Convergence analysis

It is interesting to study the behaviour of the control algorithm as a function of the communication success probability p (see Figure 19.13 for an exemplifying trajectory). Again, the chosen performance index is the convergence time, reported in (19.11). Since we are dealing with an aleatory process we must perform more than one run for each value of p in order to get the average behaviour of the system. It was chosen to perform four runs for each p value. The simulation data are the following: $\alpha_x = 5$, $\alpha_y = 5$, $\gamma_x = 1$, $\gamma_y = 1$, $k_{i,j} = 5$, $t_s = 0.1s$, with information exchange topology as in Figure 19.7 and initial and final positions as in Figure 19.13. The results of this analysis are reported in Figure 19.14. The green line is the average of the 4 runs at fixed p and for each point an error bar shows the standard deviation of runs. We can see that the algorithm is quite robust to message losses: in fact, only below 50% of message delivery probability the convergence time start to grow exponentially until convergence is not reached any more. Moreover, for small p the results become more unpredictable since the standard deviation tends to grow.

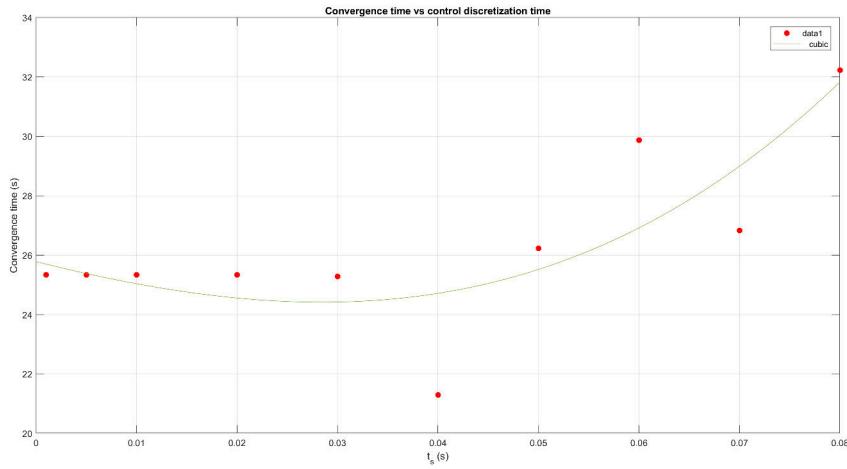


Figure 19.12: Convergence time vs control discretisation time

19.2.5 Communication Scheduling: Round Robin

In this section a communication scheduling based on the *Round Robin* algorithm will be implemented. Again, we assume a communication system with a central computing unit, as previously depicted. The *Round Robin* algorithm assigns a slot of time (quantum t_q) to the central unit to collect the data from all the agents and to communicate the computed input to one agent, then assigns the time slot dedicated to each agent in a circular order. In other words, the central computing unit is capable of computing and communicate one input per agent at time.

Implementation

A counter generates a periodic signal of n steps, with n number of the agents, where every step is an integer value representing the index of an agent. The counter keeps its index value for a time equal to the quantum t_q (see Figure 19.15). Next, a time varying array is created where all the entries are zero except the one with index from the counter. Then this array is multiplied by the array of the continuous control generated by the control algorithm. In this way during a quantum only the agent selected by the *Round Robin* scheduling will receive the input, while all the other agents will have zero input. Obviously the quantum t_q must be greater than the control discretisation time t_s .

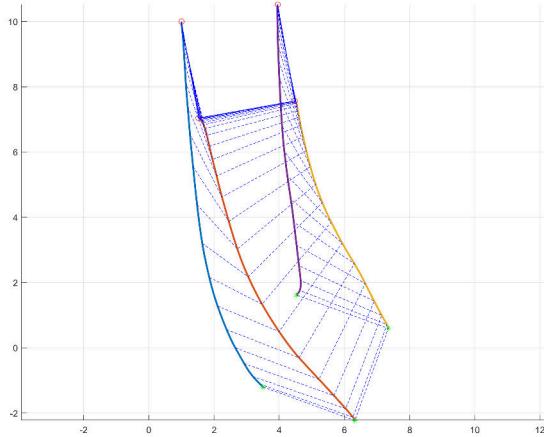


Figure 19.13: Flock trajectory for $p = 0.9$

Convergence Analysis

Two cases are considered.

CASE A : t_q varying, t_s fixed. The control discretisation time is fixed and we will see the effect of quantum time on the convergence time of the control algorithm. In this case, during the quantum the input for the single agent communicating can change with sampling rate t_s . The simulation data are the following: $\alpha_x = 5, \alpha_y = 5, \gamma_x = 1, \gamma_y = 1, k_{i,j} = 5, t_s = 0.01s$, with information exchange topology as in Figure 19.7 and initial and final positions as in Figure 19.13. In Figure 19.16 simulation results are reported. Starting from $t_q = t_s$ we can see how the performance worsens as t_q grows, until for large t_q convergence is not reached any more. The increase of the convergence time with t_q is due to the fact that all agents but one must wait for inputs for an increasingly longer time.

CASE B: $t_s = t_q$, both varying. The control discretisation time is equal to the quantum and both are varied. Notice that in this case during a quantum the input for the single agent communicating is constant. Thus the performance will be influenced by the combined effects of t_q and t_s . The simulation data are the same as in CASE A (except t_s). From the results of Figure 19.17, we can see that the performance worsens more and for smaller t_q with respect to CASE A. This was expected, since the system undergoes the negative combined effects of increasing t_s and t_q .

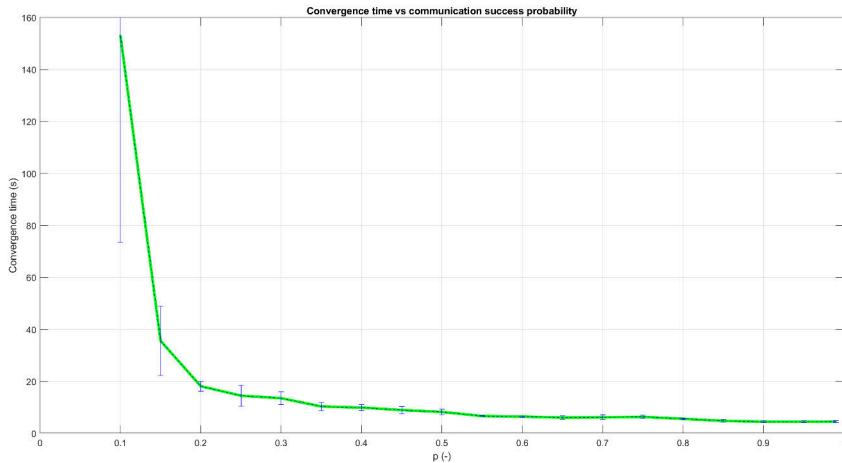


Figure 19.14: Converge time vs communication success probability

19.2.6 Communication Range

In this section a communication system depending on agent distances is implemented.

Implementation

A *Matlab function* block is created. Each distance between the agents is computed, then through a comparator an $n \times n$ matrix of 1 and zeros is built, where 1 means "inside communication range" and 0 means "outside communication range". Finally, the Laplacian matrix is multiplied element-wise by this new matrix. In this way, the agents out of communication range will not receive the message and will have a zero input.

Convergence Analysis

We assume again a central computing unit controlling the agents. Notice that the implemented communication range rule will only influence the reading of the relative distance between robots, thus it will only influence the Laplacian matrix. The Laplacian matrix influences only the formation keeping control (through the second order linear consensus algorithm, see (19.10)). In conclusion, we must use a new performance index for the formation keeping evaluation.

For four agents flocks in square formation, the ratio between the square

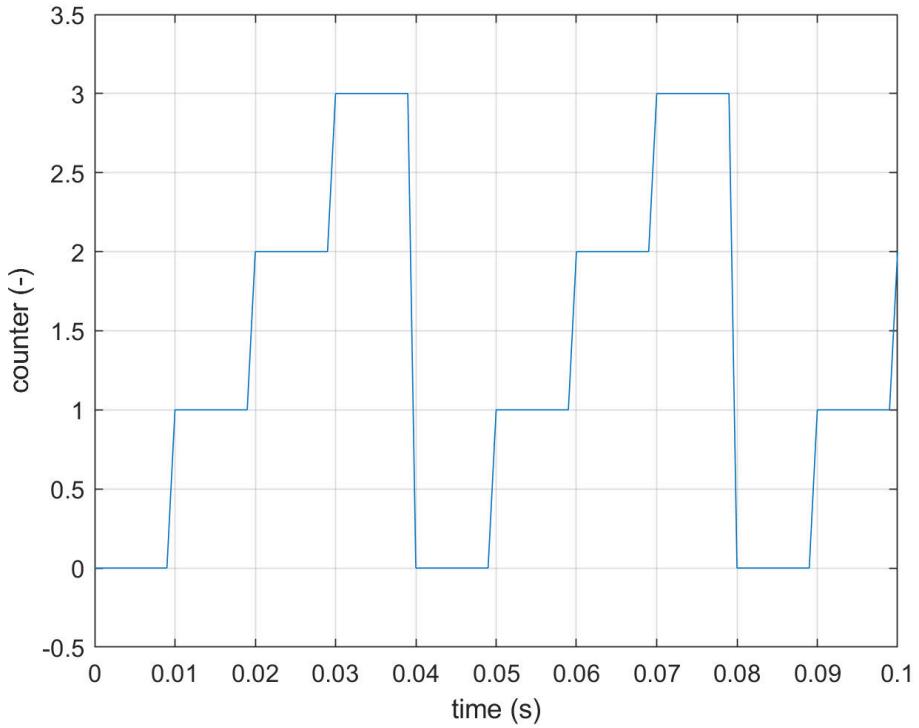


Figure 19.15: Round Robin quantum counter.

diagonal and the diagonal computed from the square side is computed. It is easy to see that this ratio is equal to one if the robots are in perfectly square formation. This ratio is computed at each time instant and for a general simulation with starting and final positions as in Figure 19.13 we get a result as in Figure 19.18. In order to measure the formation keeping performance of the control algorithm we choose to compute the *Root Mean Square Error* (RMSE) of diagonal ratio with respect to the desired value of 1 (perfect square).

The simulation data are the following: $\alpha_x = 5, \alpha_y = 5, \gamma_x = 1, \gamma_y = 1, k_{i,j} = 5, t_s = 0.1s$, with information exchange topology as in Figure 19.7 and initial and final positions as in Figure 19.13. The side length of the desired square formation is $L = 3m$. In Figure 19.19 RMSE value of diagonal ratio are plotted against the ratio between the communication range and the length of the square formation side. We can conclude that for communication ranges longer than the square side we have a constant small RMSE, which means a good and robust formation keeping performance. On the contrary,

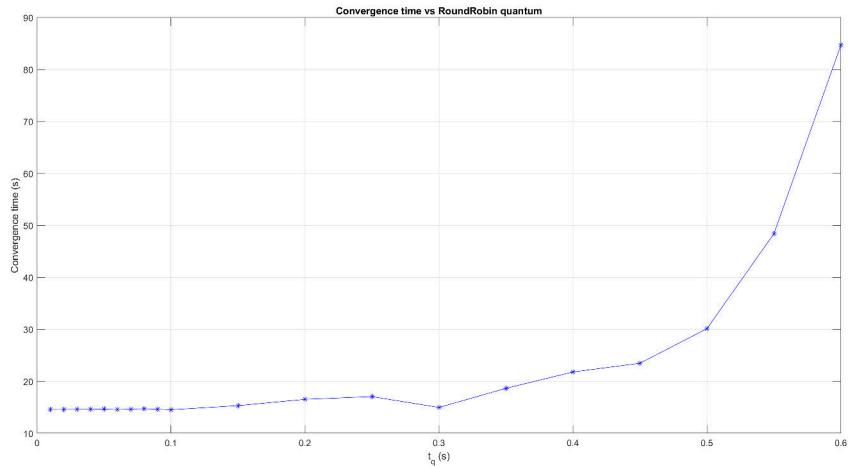


Figure 19.16: Convergence time vs Round Robin quantum time.

when communication range is equal or smaller than the formation square side the formation keeping performances are basically lost.

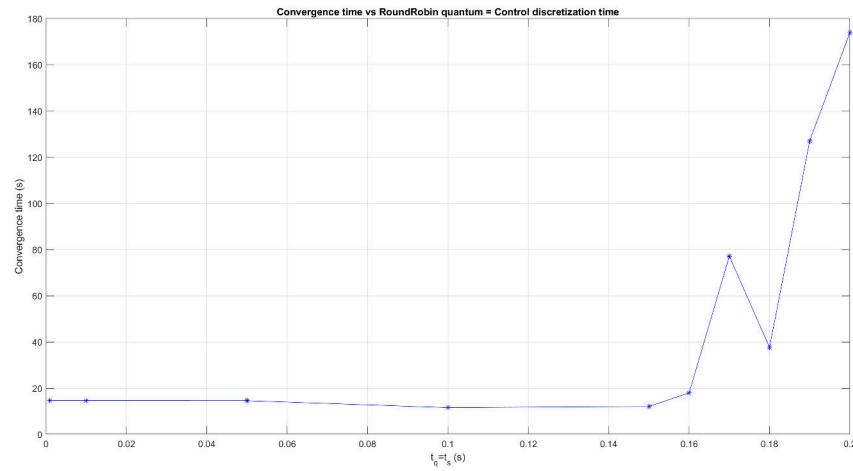


Figure 19.17: Convergence time vs Round Robin quantum time=Control discretisation time.

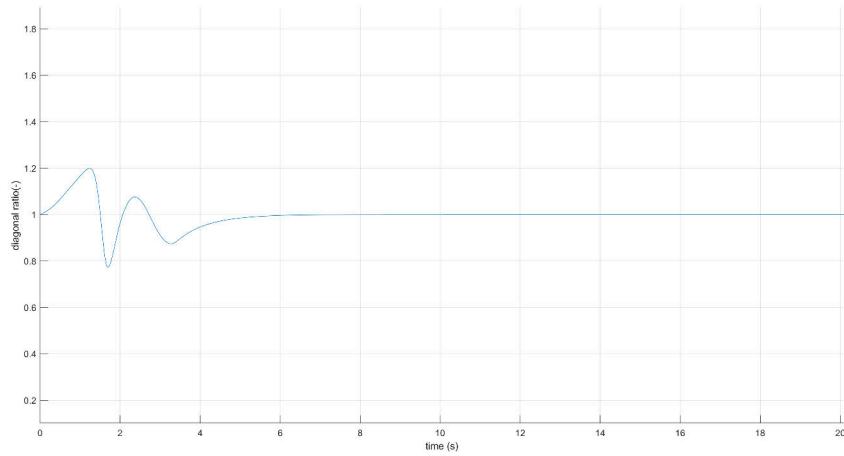


Figure 19.18: Diagonal ratio of the square formation shape vs Simulation Time.

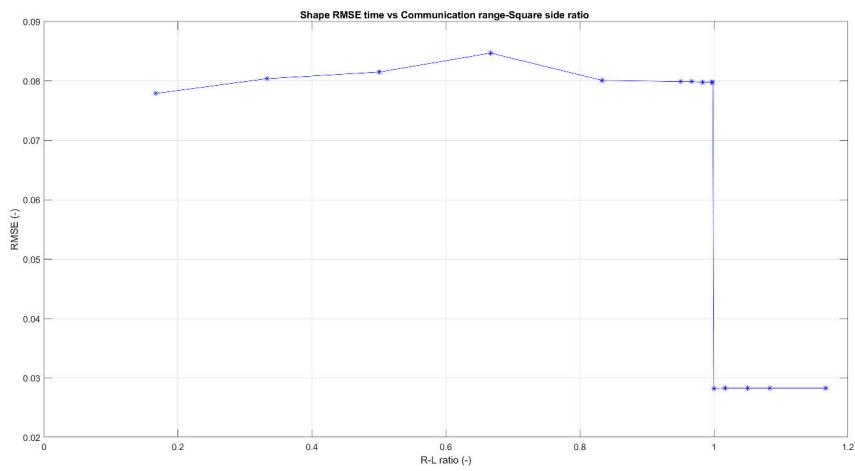


Figure 19.19: Diagonal ratio RMSE vs Communication Range-Square Side ratio.

Chapter 20

Robotics Perception

According to the robotics community [43], the AI-based robots has these abilities:

- Configurability: “The ability of the robot to be configured to perform a task or reconfigured to perform different tasks. This may range from the ability to re-program the system to being able to alter the physical structure of the system. (e.g. by changing a tool)”. The configurability acts at the level of the software modules, the sensing system and/or at the level of the mechanical structure;
- Adaptability: “The ability of the system to adapt itself to different work scenarios, different environments and conditions. Adaptation may take place over long or short time scales. It may relate to local control systems or actions, or to the whole system or to interaction”. Adaptability implies alteration of parameters of the, e.g., sensing apparatuses for changing environments, control algorithms as a function of time varying mechanical conditions, strategies to satisfy a certain goal;
- Interaction Ability: “The ability of a system to interact physically, cognitively and socially either with users, operators or other systems around it, including other robots”;
- Dependability: “The ability of the system to perform its given tasks without systematic errors. Dependability specifies the level of trust that can be placed on the system to perform. This may be in terms”;
- Motion Ability: “The ability of the system to move. Motion may be highly constrained where ability is measured by the precision of the motion, or its repeatability. Alternatively motion may be unconstrained

and is measured by the ability to move effectively in different media or between media”;

- Manipulation Ability: “The ability of the system to handle objects. Where end effectors are fixed or specific to the task, this will specify the accuracy and repeatability of the manipulation, for example the ability to absorb tolerances in parts. For dexterous manipulation, it might specify the ability to discover how to hold and move unknown objects, or the ability to match two objects together in specific ways”;
- Decisional Autonomy: “The ability of the robot to act autonomously. Nearly all systems have a degree of autonomy. It ranges from the simple motion of an assembly stopped by a sensor reading, to the ability to be self sufficient in a complex environment”;
- Cognitive Ability: “The ability to interpret the task and environment such that tasks can be effectively and efficiently executed even where there exists environmental and/or task uncertainty. The ability to interpret human commands delivered in natural language or gestures. The ability to interpret the function and interrelationships between different objects in the environment and understand how to use or manipulate them. The ability to plan and execute tasks in response to high level commands. The ability to work interactively with people as if like a person”;
- *Perception Ability*: “The ability of the robot to perceive its environment. At the simplest level this is about specifying the probability of accurately detecting objects, spaces, locations or items of interest in the vicinity of the system. It includes the ability to detect the ego motion of a robot arm and the ability to interpret information and to make informed and accurate deductions about the environment based on sensory data”.

We will only focus on the *perception abilities*, which rely on the following technological fields: mechatronics (regarding heterogeneous sensors and sensor fusion, communication protocols among different sensors and actuators in the robotic system), perception (regarding the connection between the sensing technologies, producing interesting streams of data related to the environment, and the interpretation of the sensed data, to extract a precise understanding, such as the tracking of a human being or the position of the robotic system in the environment) and navigation (the connection between localisation, map building and path and motion planning).

For the perception abilities, the agenda [43] reports different levels, ranging from the absence of any information from the environment (hence, only open loop operations can be carried out), to the detection of the feature of interest (e.g., the presence of an obstacle), to the identification of objects and their properties or, even the inference of *hidden states* (e.g., quantities that are not directly accessible or measurable by the sensors, some sort of complex indirect measurements or, also, prediction). A set of abilities the robot must have and that are directly related to perception are:

- Tracking ability: as the robot typically moves inside the environment, it has to track the entities in the scene to keep a constant and correct representation of the environment useful for the task execution. This tracking ability ranges from the tracking of perceived features (e.g., points in an image) to dynamic objects and shapes, as well as to animate objects (e.g., humans or animals);
- Recognition levels: in some specific task there may be the necessity to recognise single instances of specific objects or a specific object among many of them. This ability, more related to computer vision (hence, not considered in this notes), ranges from the detection of objects to the ability to learn new objects, infer their characteristics and recognise flexible objects;
- Scene perception: usually, a robot that moves inside an area has to detect the physical characteristics of the working environment, such as walls, floors, doors, etc. This ability ranges from the detection of features related to the physical world (e.g., the corner of an hallway) to the detection of multiple objects in a combined structure as well as the detection of partially perceived objects or dynamic objects;
- Self location perception: this is one of the fundamental ability in robotic perception, which ranges from determining the position using environmental sensors (e.g., radio signals emitted from anchors in known positions) to matching perceived entities in a map, to the determination of the location of objects that may be connected to the mechanical platform (e.g., an object being manipulated).

We will focus on all this abilities, except for the recognition abilities that usually entails computer vision algorithms.

We have to point out that robots subsumes the interaction with the world, as recalled in a widely accepted definition of *robotics* from Prof. Bruno Siciliano: ‘Robotics is the science studying the intelligent connection between

perception and action”. Hence, when we deal with robots, perception subsumes the collection of all the quantities of interest that enables the robotic action on the environment. In other words, labelling image contents (e.g., understanding if there is a table, an animal, a chair in front of the robot) or determining if a human beings in the surroundings of the robot is scared or relaxed cannot be defined as robotic perception but, rather, interesting applications of AI to, e.g., images. What differentiate the robotic perception is the ability to understand *where* the entity is located in the surrounding of the robot and *which* kind of action can be executed by the robot to interact with the detected entity. This is very close to the concept of *affordance*, that was firstly introduced by James J. Gibson in 1978 in [44]:

The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment.

In other words, an object has different possible usage depending on the user it handles. For robots, the affordance makes only sense if it is possible to track, locate or detect it inside the robot motion space. To this end, we need estimation algorithms.

20.1 Sensing systems

Since robotics perception relies heavily on the type and richness of the information that can be obtained from the available sensors, we will start with a taxonomy and a description of the sensing system characteristics.

- *Proprioceptive sensors*: used to collect information that are internal to the robot structure. Examples ranges from very simple sensors, such as contact switches, to more sophisticated solutions, such as Inertial Measurement Units (IMUs) endowing usually an accelerometer, a gyroscope and a compass. Usually, such sensors return specific quantities directly (e.g., joint angles, velocity, battery voltage);
- *Exteroceptive sensors*: used to collect information on the robot surroundings, such as human beings, fixed obstacles, etc. Among this class, there are the cameras, laser range finders, Radio Frequency (RF) based solutions. For this class of sensors, heavy post processing and estimation algorithms are in general needed to extract the features of interest.

Another distinction is related to the *modality* in which the data are collected:

- *Contact sensors*: the measurement results are collected through direct contact of the sensor with the external environment, e.g., human's tactile sensing.
- *Non-contact sensors*: the measurement results are gathered without direct contact, e.g., human's visual sensing.

Another difference is related to the presence in the environment of the *physical quantity* we want to measure:

- *Active sensors*: observations are collected by emitting energy in the environment or by modifying it, e.g., lasers.
- *Passive sensors*: observations are derived by collecting the energy in the environment, e.g., cameras or human's hearing.

20.1.1 Proprioceptive sensors

Incremental Encoders

Encoders use rotating disks with optical windows. Light from emitters passes through the openings in the disk to detectors. As the encoder shaft is rotated, the light beams are broken. There are two fundamental types of encoders: *absolute* and *incremental*. An incremental (or relative) encoder (see Figure 20.1-a) will output pulses that can be used to determine the displacement. To add accuracy to the relative encoder we only need to add more windows to the existing ring. Typical encoders will have from 2 to thousands of windows per ring. The output signal of the incremental encoder with one ring is a simple square wave (see Figure 20.1-b). Increasing the velocity of the shaft, the signals period is decreased. As a consequence, a *tachometer* can be built from incremental encoders. However, the main limit, despite the simplicity of the sensor, is that rotations clockwise or counter-clockwise cannot be disambiguated. Indeed, when using a relative encoder, the distance of rotation is determined by counting the pulses from one of the rings. If the encoder only rotates in one direction then a simple count of pulses from one ring will determine the total distance. If the encoder can rotate both directions a second ring must be used to determine when to add/subtract pulses. The quadrature scheme (Figure 20.2-a) uses two rings: the signals are set up so that one is out of phase with the other. Relative encoders require a cal-

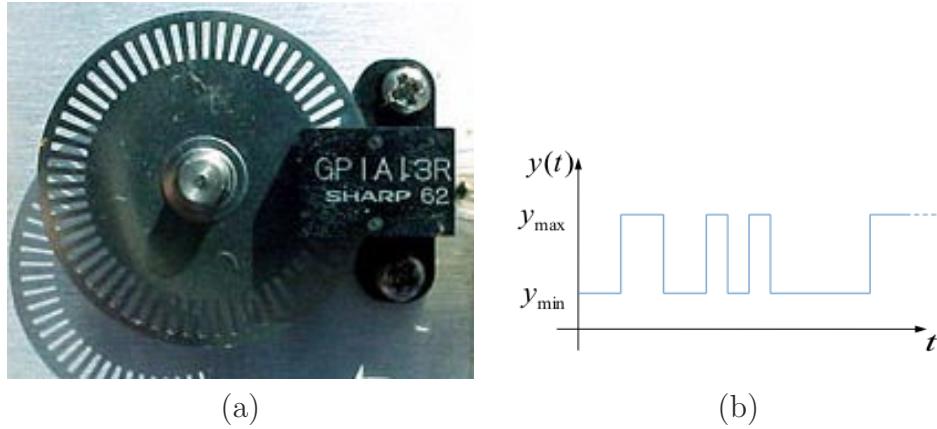


Figure 20.1: Incremental encoders

ibration phase when a controller is turned on. This usually requires motion of an axis until it reaches an extreme position marking the end of the range, e.g., homing position for manipulators.

The output signals of the incremental encoder with two rings are two square waves, i.e., two output channels. The two channels have a relative phase of 90° . Therefore, the two channels of the *quadrature* encoder indicate both position and direction (see Figure 20.2-b): if y_a leads y_b , the motor is rotating clockwise. If y_b leads y_a , counter-clockwise.

An absolute encoder directly measures the angle of the shaft. The same shaft angle always produces the same sensor reading. The absolute encoder has n rings, the outer ring is the most significant digit, the inner ring is the least significant digit (Figure 20.3-a). More rings (and more emitters/detectors) implies more accuracy. The output is normally a number coded in binary or *grey code* (Figure 20.3-b). The absolute encoder in the figure has a grey code. In both cases, logic circuits and/or software are used to determine the direction of rotation and to count pulses for displacement derivation. The velocity can be determined by measuring the time between pulses.

20.1.2 Potentiometers

Potentiometers measure the angular (or linear) position of a shaft using a variable resistor. The potentiometer is a resistor, normally made with a thin film of resistive material and a brush moves along the surface of the resistive film (see Figure 20.4-a). As the brush moves toward one end there will be a

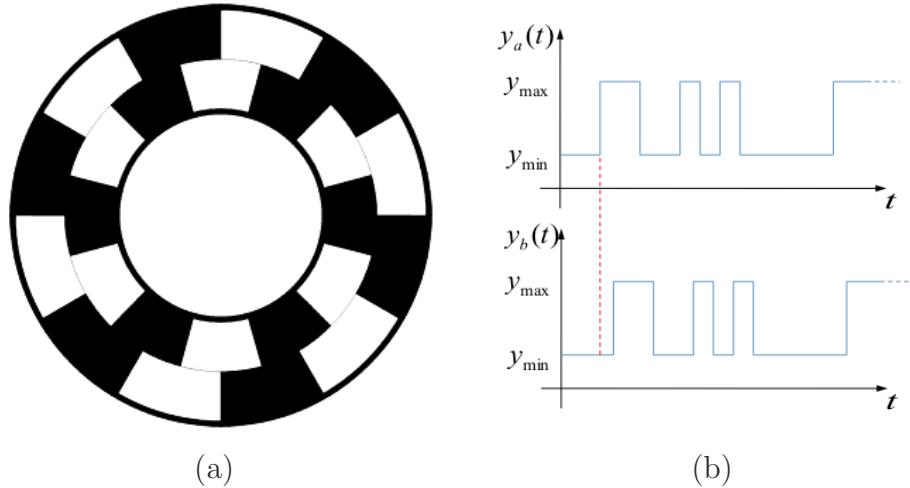


Figure 20.2: Quadrature scheme (a) and signal (b) for an incremental encoder.

change in resistance proportional to the travelled distance:

$$V_w = \left(1 - \frac{\theta_w}{\theta_{max}}\right) V_1 + \frac{\theta_w}{\theta_{max}} V_2$$

Potentiometers are popular because they are inexpensive, and do not require special signal conditioners. Potentiometers have limited accuracy, normally in the range of 1% and they are subject to mechanical wear.

20.1.3 Inertial sensors

Inertial sensors make the least reference to the external world. The typical quantity measured by the inertial sensors are the first or second derivatives of the position or orientation of the mechanical body they are attached to. Examples of inertial sensors are *accelerometers*, *gyroscopes*, *magnetic compasses* and *inclinometers*.

Accelerometers measure acceleration using a mass suspended on a force sensor (often a small piece of piezoelectric material mimicking a *spring*, as represented in Figure 20.4). When the sensor accelerates, the inertial resistance of the mass will cause the force sensor to deflect. By measuring the deflection the acceleration can be determined (*Hooke's law*):

$$F = ma \text{ and } F = kx \Rightarrow a = \frac{kx}{m}.$$

Accelerometers are dynamic sensors, and temperature variations will affect the accuracy of the sensors. An accelerometer measures the acceleration

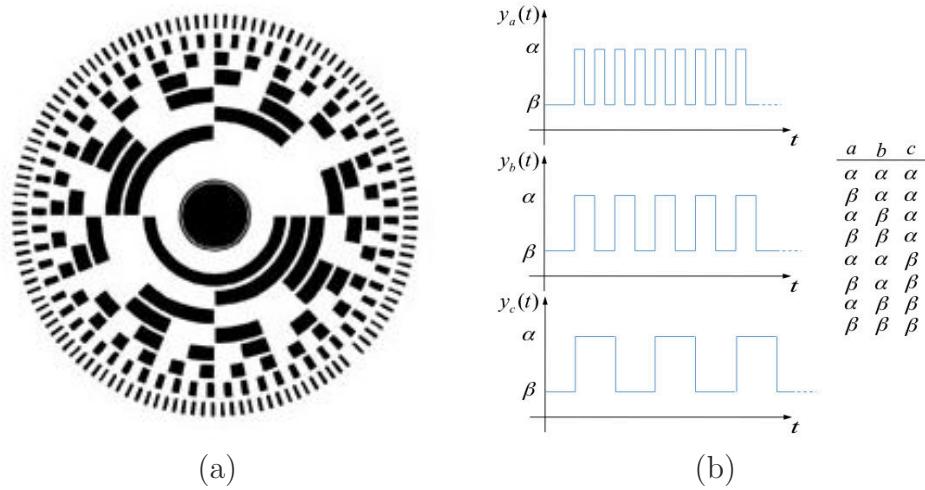


Figure 20.3: Absolute encoder (a) and coded signals (b) it generates.

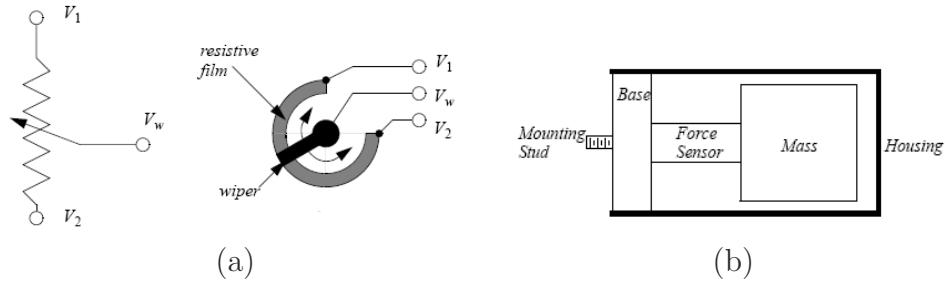
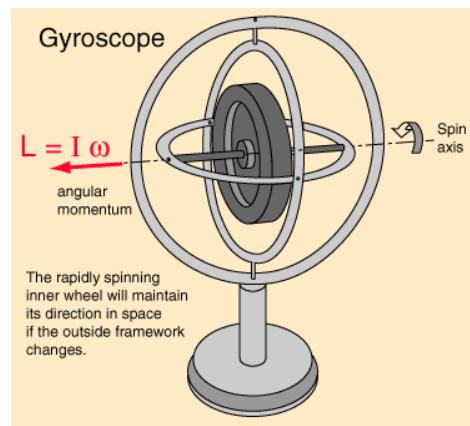


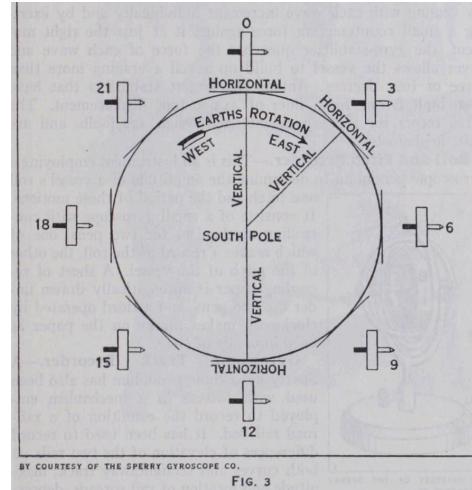
Figure 20.4: Potentiometers (a) and a MEMS based accelerometer (b).

along an axis. By combining three accelerometers with orthogonal axes, we have a three dimensional accelerometer. Advances in MEMS are already beginning to provide integrated circuit accelerometers at a very low cost. One common use is for airbag deployment systems in cars.

Gyroscopes relies on the principle of the *conservation of angular momentum*, i.e., $L = I \times \omega$, where I is the moment of inertia and ω is the angular velocity. Consider a rapid spinning wheel mounted on a shaft that can freely change its axis of rotation, as depicted in Figure 20.5-a. Assume no friction from the air and the rotating shaft: the rotation axis will remain constant *regardless* of the motion of the surrounding cage. The constancy can be used to maintain a certain bearing if the cage is fixed on the robot chassis. However, this is not a good idea, since as the Earth spins, the gyroscope will maintain the same angle of rotation. Indeed, suppose the gyroscope has its axis of rotation oriented orthogonal to the equator and to the Earth axis of rotation (see Figure 20.5-b). As the Earth spins, the gyroscope will maintain



(a)



(b)

Figure 20.5: An example of a gyroscope (a) and of the effect used to build a gyrocompass (b).

a constant axis of rotation, hence to an Earth-fixed observer it appears to rotate, returning into its original position after 24 h. Although the global motion limits, the mechanical gyroscope can effectively detect local changes of orientation. The gyroscope can effectively be used as a gyrocompass if its axis of rotation is parallel to the Earth axis of rotation, hence pointing to the poles. However, the measure is obtained after a quite long transient.

Rate gyros (RG) can effectively measure a robot's rotation rate. Moreover, there exist Rate-integrating gyros (RIG) that use embedded processing to internally integrate angular rate and, hence, obtain the absolute orientation. Nowadays, gyroscopes are built using MEMS or optical laser-based solutions rather than mechanical system. Therefore, the costs are very widespread and can change up to three order of magnitude. The main problem related to gyroscopes is related to the drift.

20.1.4 Tachometers

Tachometers measure the velocity of a rotating shaft. A DC tachometer commonly mounts a magnet to a rotating shaft (see Figure 20.6-a). When the magnet moves past a stationary pick-up coil, current is induced. For each rotation of the shaft there is a pulse in the coil (*Faraday's law*: $EMF = n \frac{d\Phi_B}{dt}$). When the time between the pulses is measured the period for one rotation can be found, and the frequency calculated. Another common technique uses a simple permanent magnet DC generator. Even if both the proposed solu-

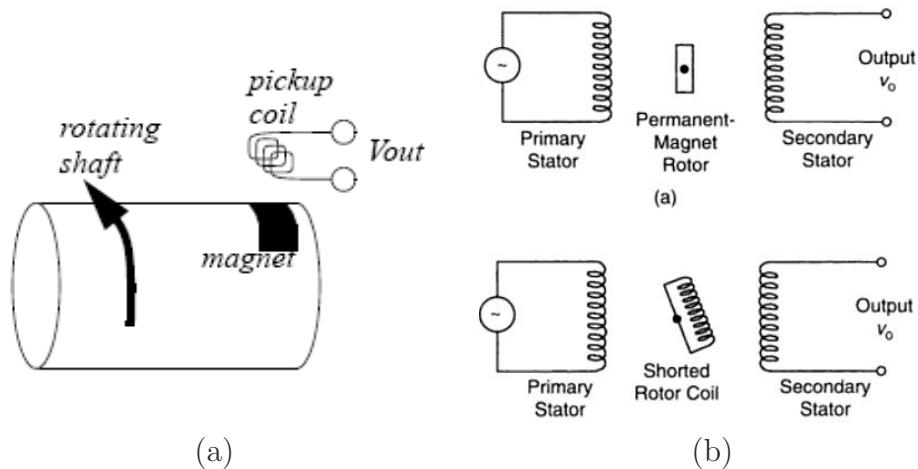


Figure 20.6: DC (a) and AC (b) tachometers.

tions are quite inexpensive, they require some signal conditioning circuitry. In both cases, the residual ripple coefficient is of 2 to 5% of the mean value of the output signal.

The residual ripple can be reduced renouncing to a DC generator and using an AC tachometer. The AC tachometer has two windings on the stator mutually in quadrature and a magnet rotor (see Figure 20.6-b). If one of the windings is fed by a constant-magnitude sinusoidal voltage, a sinusoidal voltage is induced on the other winding. If the rotor is static or moving in a quasi-static manner, then the induced voltage (the output tachometer voltage) is much like the input voltage.

If the rotor moves at a finite speed, an additional voltage is induced on the secondary windings, proportional to the rotor velocity (rate of change of flux linkage into the secondary coil from the rotating magnet, again Faraday's law). The output voltage is then an amplitude-modulated signal whose amplitude is proportional to the rotor speed. The direction of the speed is retrieved by the phase of the output signal with respect to the input. For robotics applications, the input signal is typically of 400 Hz.

20.1.5 Strain Gages

Strain gages measure strain in materials using the change in resistance of a wire (see Figure 20.7-a). The wire is glued to the surface of a part, so that it undergoes the same strain as the part (at the mount point). Basically, the resistance of the wire is a function of the resistivity, length, and cross

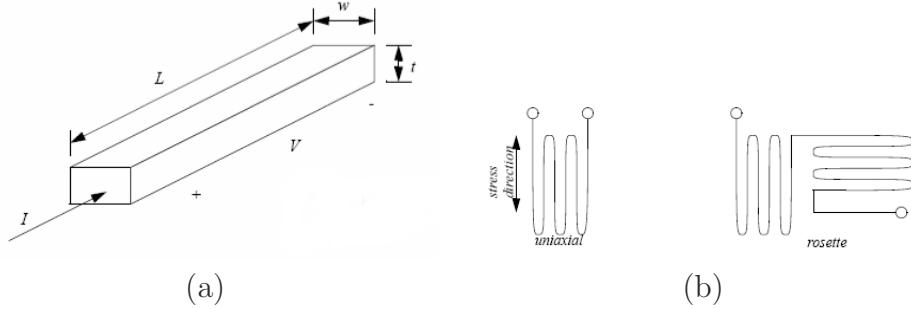


Figure 20.7: Principle of the strain gauge (a) and different shapes (b).

sectional area

$$R = \frac{V}{I} = \rho \frac{L}{A} = \rho \frac{L}{wt}.$$

If the wire is deformed, it takes on a new dimension and resistance. If the force makes the wire longer (Young's modulus), but the cross sectional area decreases (Poison's ratio). The *Gauge factor* is a commonly used measure of strain gauge sensitivity:

$$GF = \frac{\frac{R'}{R}}{\varepsilon},$$

where R' is the deformed resistance and ε is the strain of the material.

A strain gage must be small for accurate readings, so the wire is actually wound in a uniaxial or rosette pattern. When using uniaxial gages the direction is important, it must be placed in the direction of the normal stress, instead rosette gages are less sensitive to direction (see Figure 20.7-b). These gauges are sold on thin films that are glued to the surface of a part. The process of mounting strain gages involves surface cleaning, application of adhesives, and soldering leads to the strain gages. Strain gauges are inexpensive, and can be used to measure a wide range of stresses with accuracies under 1%. Gages require calibration before each use. This often involves making a reading with no load, or a known load applied. An example application includes using strain gages to measure end-effector forces during stamping to estimate when maintenance is needed.

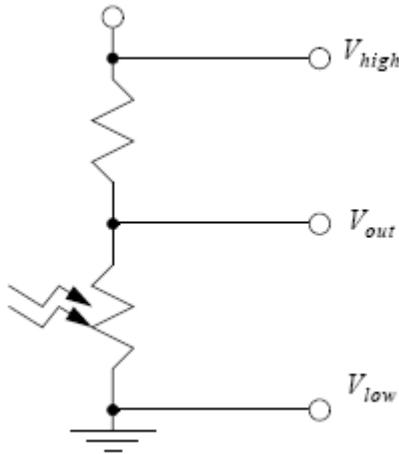


Figure 20.8: Light dependant resistor.

20.2 Exteroceptive sensors

20.2.1 Contact Sensors and Bumpers

A *switch* is an electrical component that can break an electrical circuit, interrupting the current or diverting it from one conductor to another. A *contact switch* allows electricity to flow between its two contacts when held in. When the button is released, the circuit is broken (*non-latching* switch, see also the definitions reported in Chapter 8). The information coming from the switch is then binary: ON/OFF. A *bumper* in robotics is a soft material covering attached to an array of contact sensors for *collision detection*.

20.2.2 Light Dependant Resistors and Photodiode

Light dependant resistors (LDRs) (see Figure 20.8 for the electric schematic representation) change from high resistance (of some M Ω s) in bright light to low resistance (less than k Ω) in the dark. The change in resistance is non-linear, and is also relatively slow (ms). An LDR, also known as *photoresistor*, is made of a high resistance semiconductor. If light falling on the device is of high enough frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electron (and its hole partner) conduct electricity, thereby lowering resistance.

A *photodiode* is a type of photodetector capable of converting light into either current or voltage, depending upon the mode of operation. When

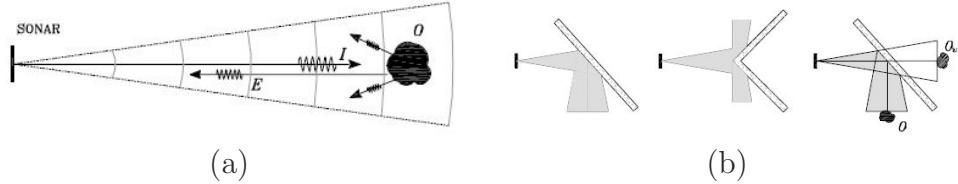


Figure 20.9: An example of a sonar sensor.

a photon of sufficient energy strikes the diode, it excites an electron, that is attracted to the *cathode*. Consequently, the *electron hole* is attracted by the *anode*. The result is the *photocurrent*. A *phototransistor* is a bipolar transistor with the base–collector junction reached by the light. The electrons excited by photons in the base–collector junction are injected into the base, and this photodiode current is amplified by the transistor’s current gain. A *color sensor* combine three photodiodes (or phototransistor) with optical filters.

20.2.3 Sonars

The *sonars* employ acoustic pulses and their echoes to measure the range to an object (active sensor). A typical sonar sensor comprises a *transmitter*, which transmit the *probing pulse*, and a *receiver* of the echos. Sonars are widely utilized in robotics, and especially in mobile and underwater robotics. Their popularity is due to their low cost, light weight, low power consumption, and low computational effort, compared to other ranging sensors. Sonar sensors are employed in robotics for three applications: *obstacle avoidance*, *sonar mapping* (similar to a *radar display*) and *object recognition*.

If an objects lies on the probing pulse path, it reflects the sonar beam (see Figure 20.9-a). The receiver captures the echo and measures the *time-of-flight* (TOF) from the probing pulse transmission time. The echo is just a replica of the probing pulse. The distance of the object is $\rho = \frac{ct}{2}$, where c is the sound speed (343 m/s at standard temperature and pressure), t is the echo travel time and the factor 2 converts the *round trip time* into a range measurement. Typical frequencies of the probing pulse range from 20 KHz to 200 KHz. Hence, the energy of the wave emitted by the sonar can be regarded as concentrated in a conical volume whose beamwidth depends on the frequency as well as on the transducer diameter.

Despite the low cost and ease of use, however, these sensors have non negligible limitations. The wide sonar beam causes poor directional resolution (e.g., small objects are not detected as well as small openings). The slow sound speed reduces the sonar sensing rate (e.g., many sonars transmit

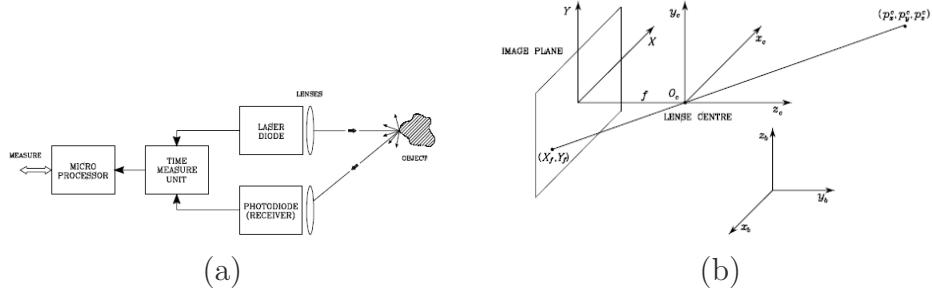


Figure 20.10: Examples of a Laser (a) and a camera (b).

two consecutive beams after 50/20 ms). Smooth surfaces at oblique incidents do not produce detectable echoes, artifacts caused by beam side-lobes and multiple reflections produce false range readings in the environment (see Figure 20.9-b). Travel time and amplitude variations caused by inhomogeneities in the sound speed (e.g., given by different humidity or air temperature). Non-smooth surfaces, i.e., those characterized by irregularities of comparable size to that of the wavelength corresponding to the employed frequency, may produce a non-detectable echo at the sensor.

One technological design is the *electrostatic transducer*. A gold-coated plastic foil membrane stretched across a round grooved aluminium back plate and it is charged by a constant voltage (e.g., 100 V). The incoming echo waves vibrate the foil and change the distance to the back plate, hence changing its capacitance $C(t)$. Assuming that the charge q is constant, the output voltage is $v(t) = qC(t)$. This is exactly the same principle of standard capacitor *microphones*. The transmitter emits the probing pulse by applying a voltage pulses across the capacitor, hence letting the transducer membrane to vibrate. Alternatively, *piezoelectric* ceramic transducers (which uses low voltages and CMOS technology) and *MEMS* based transducers (whose working principle is the same of the electrostatic capacitive transducers) can be adopted.

20.2.4 Laser

The *laser* beam is usually preferred to other light sources because it generates bright beams with lightweight sources, it focuses well to give narrow beams and the single-frequency sources allow easier rejection filtering (see Figure 20.10-a). The time-of-flight sensors compute distance by measuring the time that a pulse of light takes to travel from the source to the observed target and then to the detector (usually collocated with the source). Alternatively, modulation laser sensor are adopted, that measures the phase shift

of an amplitude or frequency modulated signal. Limitations on the accuracy of these sensors are based on the minimum observation time (and thus the minimum distance observable), the temporal accuracy (or quantization) of the receiver, and the temporal width of the laser pulse. Normally the beam is swept by a set of mirrors rather than moving the laser and detector themselves (mirrors are lighter and less prone to motion damage).

20.2.5 Camera

The task of a *camera* as a *vision sensor* is to measure the intensity of the light reflected by an object (passive sensor). A photosensitive element, termed *pixel* (or *photosite*), is employed, which is capable of transforming light energy into electric energy. The most widely used devices are the CCD and the CMOS sensors based on the photoelectric effect of semiconductors. A camera is a complex system comprising several devices other than the photosensitive sensor, i.e., a shutter, a lens and analog preprocessing electronics. Measurements are taken following the *perspective geometry*: the most famous model is the *pinhole camera model* (see Figure 20.10-b). Cameras are quite cheap and collects a very large amount of information from the surroundings environment. Nevertheless, cognitive algorithms must be employed to analyze images grabbed from this sensor.

20.3 A taxonomy of robotics problems and their solutions

In this section we made explicit the robotic perception problems using the tools learnt in the previous chapters. In particular, we will proceed by differentiating between *robot manipulators* and *mobile robots*, with their own peculiarities, while at the end of this analysis we will present the common problems.

20.3.1 Robot manipulators

The main problems that should be tackled for a successful application of a robotic manipulator in a certain environment are reported next:

- Reconstruction of the robot posture. This is typically done using

20.3.2 Mobile robots

20.3.3 Common issues

20.4 Interactive Multiple Models

Introduce Chapter 3 of the ML.

20.4.1 Gaussian Pseudo-Bayesian Estimator

20.4.2 IMM

Appendices

Appendix A

Real-Time

A real-time system *must* deliver services in a *timely manner*, i.e., *not fast* but must meet some *timing deadline* [45]. There exist a lot of real-time systems that are used commonly in everyday life: automotive systems, home common machines, aircrafts, CD player, medical devices, etc. *Bugs* in real-time systems may be catastrophic. Think of safety critical systems or of exploratory planet missions. In the context of the course, we will use real-time associated to *(distributed) control or estimation tasks*.

Definition 97. *A real-time application is an application where the correctness of the application depends on the timelines and predictability of the application as well as the results of computations.*

Definition 98. *A process is an instance of a computer program that is being executed. The process may be composed by a set of threads executed concurrently.*

Definition 99. *A computer program is a collection of instructions.*

Hence, a process is the sequentially execution of the program instructions. A program may have a set of processes associated with it, e.g., opening several instances of the same program.

Definition 100. *A task is a unit of execution. Depending on the operating system, a task is a process or a thread.*

Enabling tools are: Interprocess communication and synchronisation; Fast interrupt response time and reliable input/output (I/O); Memory management functions, file synchronisation, etc. The typical digital control infrastructure is reported in Figure A.1, where are clearly visible $\hat{y}(t)$ (the information given by the plant *sensors*, i.e., the *output* of the plant), $\hat{u}(t)$

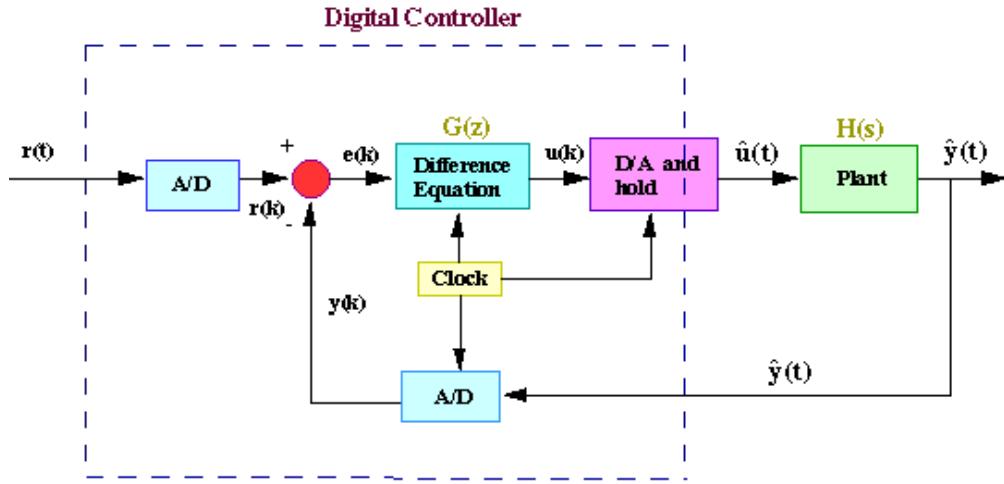


Figure A.1: Control loop in a digital system.

(the control commands to be given to the plant *actuators*, i.e., the plant *inputs*) and $r(t)$ (the plant *reference input*).

A sketch, in pseudo-code, of the instructions to be executed to implement the control law are given next.

```

set timer to interrupt periodically with period T;
at each timer interrupt, do
    do analogue-to-digital conversion of  $\hat{y}(t)$  to get  $y(k)$ ;
    compute control output  $u(k)$  based on reference  $r(k)$  and  $y(k)$ ;
    do digital-to-analogue conversion of  $u(k)$  to get  $\hat{u}(t)$ ;
end do;

```

The desired plant controls depends on: a) the *correctness* of the control law; b) the adequate *resolution and sampling time* T of the sensors. The time T between any two consecutive measurement of $y(t)$ and $r(t)$ is the *sampling period*. The smaller is T , the better is the approximation of the continuous time data with the sampled data (recall discretisation of linear systems). The larger is T , the less would be the computing power demand on the platform in charge of the execution of the code. The value of T should be traded between this two contrasting goals: indeed, if T is too large, unbounded oscillations, and hence system *instability*, may rise. As a rule of thumb, the sampling time T can be chosen to be in $[R/10, R/20]$, being R the *rise time*.

The baseline of the *digital control* theory can be represented as a *spring of fresh water*: the output of the system is assumed to be *accurate and noiseless*; from the output, it is usually possible to derive the *perfect knowledge* of the

plant state; The control law is computed *instantaneously*; communications between distributed components happens without *latency, jitter and packet losses*. Unfortunately, the control engineer has to deal with a *polluted delta* rather than a spring of fresh water: the output of the system is *noisy*; from the output, it is usually possible to derive an *estimate* of the plant state with a certain degree of *accuracy*, e.g., *(Extended) Kalman Filter techniques*; the control law is computed in *non negligible time*, i.e., *input/output delay*; usually the system is *distributed*, hence the controller is in charge of computing the control law but also to *schedule communications, sending and receiving* messages within predefined deadlines. Whatever is the communication system adopted, latencies and delays *cannot be avoided*.

As per the definitions given in Chapter 2, usually the data for control and estimation comes in a cyclic way, executed by simple computing elements, called *jobs*.

Definition 101 (Job). A *job* is a unit of work that is *scheduled and executed* by a system.

Definition 102 (Task revisited). A *task* is a set of related jobs which jointly provide some function.

Definition 103 (Processor). A *job executes, or it is executed by the operating system, on a processor*. The processor is the active component on which the jobs are *scheduled*.

Each processor has a *speed* attribute, e.g., *instructions-per-seconds*, that determines the rate of progress of a job towards completion.

Definition 104 (Resource). A *job execution may depend on the availability of some resources, which are passive elements described in terms of types and sizes*.

Only one job may use a unit of each resource at once, i.e., *mutually exclusive access*. Once it has used the resource, the job releases it.

Definition 105 (Execution Time). The amount of time required to complete the execution of a job J_i when it executes alone and has all the resources it needs is called *execution time* and identified with e_i .

The execution time e_i depends upon *complexity* of the job and *speed* of the processor on which it is scheduled. The execution time e_i may change for a variety of reasons: a) conditional branches; b) cache memories and/or pipelines; c) estimation algorithms and signal processing. Hence, the execution time $e_i \in [e_i^-, e_i^+]$. In particular, the *Worst Case Execution Time* (WCET) is e_i^+ , while the *Best Case Execution Time* (BCET) is e_i^- .

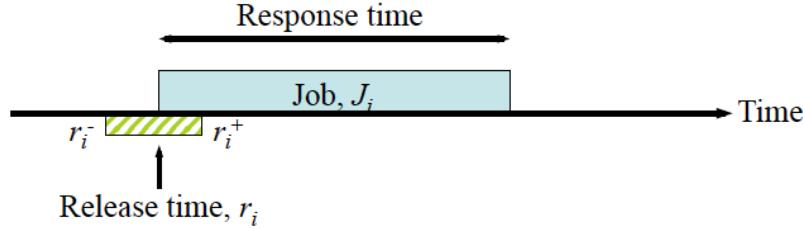


Figure A.2: Response time and release time.

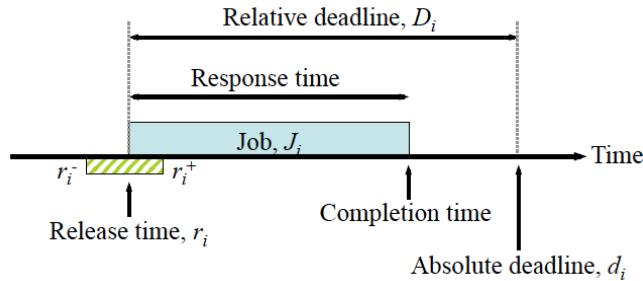


Figure A.3: Timing constraints and deadlines.

Definition 106 (Release Time). *The instant in time when the job J_i becomes available for execution is called the **release time** r_i , and it has associate a **jitter**, i.e., $r_i \in [r_i^-, r_i^+]$.*

A job can be scheduled and executed *at any time* at, or after, its release time, provided that all the needed resources are available (see Figure A.2).

Definition 107 (Response Time). *The length of time from the release time of the job to the time instant when it completes is called the **response time**.*

A response time graphical description is reported in Figure A.2. Notice that the response time is greater or equal to the execution time! Indeed, during the execution, the job may be interrupted.

Definition 108 (Completion Time). *The instant at which a job completes execution is called the **completion time**.*

A clear description of the completion time can be seen in Figure A.3.

Definition 109 (Relative Deadline). *The maximum allowable job response time is called the **relative deadline** D_i .*

Definition 110 (Absolute Deadline). *The instant of time by which a job is required to be completed is called the **absolute deadline** or, simply, the **deadline** d_i .*

Relative and absolute deadlines are represented in Figure A.3. Based on these definitions, it then follows that $d_i = r_i + D_i$. The **feasible interval** is then the interval $(r_i, d_i]$. These are the **timing constraints** of the application.

Example 41. Consider a control task that should be executed every $T = 50\text{ ms}$ and whose initialisation time is 10 ms . The **periodic execution** of the task can be parametrised in terms of the release times of the jobs $J_0, J_1, \dots, J_k, \dots$, i.e., $r_k = 10 + kT = 10 + k50\text{ ms}$. To guarantee the completion of each job **before** the next one is scheduled, it must be ensured that $D_i = T$, so $d_i = 10 + (k + 1)T$.

If the processor has to execute more than one task, than $D_i < T$. This way, there exists a **slack time** equal to $T - D_i$ than can be used to execute other tasks.

In any engineering field, solutions to problems may rely more on **technological** solutions or on **technical** solutions. Usually, industrial applications rely more on the technological solutions: Field-bus or Profinet instead of Ethernet or wireless, PLC instead of general purpose PCs, etc. Of course, this choice offers a higher degree of performance and reliability but at the price of **under exploitation** of resources and on **flexibility**. However, it is also possible to rely on technical solutions, which offer a higher degree of **flexibility**. In some respect, this is the difference between **hard real-time** and **soft real-time** approaches to digital control.

Definition 111 (Hard Real-Time). *If a job must never miss its deadline, then the system is described as **hard real-time**.*

A timing constraint is **hard** if the failure to meet it is considered a **fatal error**. The timing constraint is then **deterministic**.

Definition 112 (Soft Real-Time). *If some deadlines can be missed occasionally, with acceptably low probability, then the system is described as **soft real-time**.*

The timing constraint in this case is a **statistical constraint**. For example, “the probability of the response time exceeding 10 ms is less than 0.1 ”.

Definition 113 (Jitter). *For jobs, the **jitter** is the variation in timing of the response time of a task.*

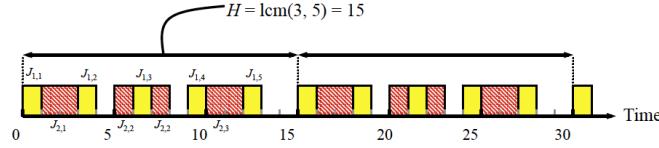


Figure A.4: Example of hyper-period: τ_1 has $T_1 = 3$ and $e_1 = 1$ ($\tau_1 = (3, 5)$), while τ_2 has $T_2 = 5$ and $e_2 = 2$ ($\tau_2 = (5, 2)$).

In general, there is *no advantage* in completing a job early, even if possible. This is even more important for control applications, since this may lead to switching dynamics and, hence, to unstable systems [46].

Definition 114 (Task types). *Tasks can be either periodic, aperiodic and sporadic. Sporadic tasks have hard deadlines, while aperiodic do not.*

A set of jobs that are executed repeatedly at regular time intervals can be modelled as a *periodic task*. The jobs that respond to *external events* are *sporadic or aperiodic* jobs. This is, again, the distinction between *time-based* and *event-based*.

In particular, the majority of the control tasks are *periodic tasks*. The periodic task τ_i has a sequence of jobs $J_{i,1}, J_{i,2}, \dots$. The *phase* of the task τ_i is given by the release time of the first job, i.e., $r_{i,1}$. The *period* of the task τ_i is the *minimum* length of all time intervals between release times of consecutive jobs. The execution time e_i of a task τ_i is the *maximum* execution time of all jobs in the periodic task.

Definition 115 (Hyper-period). *The hyper-period of a set of n periodic tasks is the least common multiple of their periods: $H = \text{lcm}(T_i)$, for $i = 1, 2, \dots, n$.*

The hyper-period is the time after which the *pattern of job release/execution times starts to repeat* (see Figure A.4).

Definition 116 (Task Utilisation). *The fraction of time a periodic task τ_i with period T_i and execution time e_i keeps a processor busy is the task utilisation u_i .*

It follows that $u_i = \frac{e_i}{T_i}$.

Definition 117 (Total Utilisation). *The total utilisation of a system is the sum of the utilisations of all tasks in the system: $U = \sum_{i=1}^n u_i$.*

In this notes, but also in the majority of the control applications, we usually assume that: the relative deadline for the jobs in a task is equal to the period of the task (no slack time); the initialisation time is not considered; the set of tasks is *harmonic*: all the task phases are zero. Usually, only periodic control and estimation tasks are considered.

Jobs may have *priorities*, and in some cases may be *interrupted* by a higher priority job.

Definition 118 (Preemptable). *A job is preemptable if its execution can be interrupted by higher priority jobs. A job is non-preemptable if it must run to completion once started.*

Definition 119 (Context Switch). *The context switch time is the time taken to switch between jobs. It is an overhead that must be accounted for when scheduling jobs.*

A.1 The scheduler

A job executes correctly if it has *allocated* the set of necessary resources and the set of corresponding resource access-control protocols.

Definition 120 (Scheduler). *The scheduler assigns the jobs to processors according to some predefined algorithms.*

Definition 121 (Schedule). *A schedule is an assignment of all jobs in the system on the available processors.*

A *valid schedule* satisfies the following conditions:

- Every processor is assigned to at most one job at any time;
- Every job is assigned at most one processor at any time;
- No job is scheduled before its release time;
- The total amount of processor time assigned to every job is equal to its maximum or actual execution time;
- All the resource usage constraints are satisfied.

Definition 122 (Feasible schedule). *A valid schedule is also a feasible schedule if every job meets its timing constraints.*

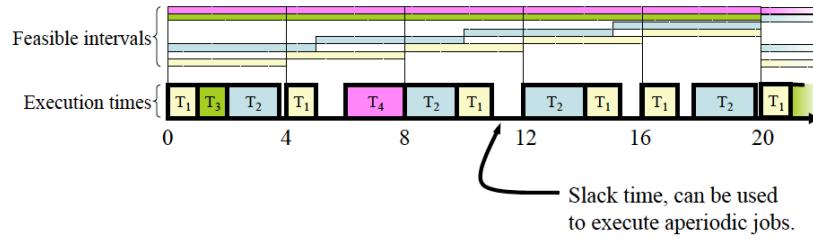


Figure A.5: Example of cyclic scheduling with hyper-period $H = 20$: $\tau_1 = (4, 1)$, $\tau_2 = (5, 1.8)$, $\tau_3 = (20, 1)$ and $\tau_4 = (20, 2)$.

Definition 123 (Miss and Loss Rate). *The miss rate is the percentage of jobs that are executed but completed after the deadline (deadline miss). The loss rate is the percentage of jobs that are not executed at all.*

Definition 124 (Optimal Scheduling). *A hard real-time scheduling algorithm is optimal if the algorithm always produces a feasible schedule if the given set of jobs is a feasible task set.*

The basic approaches to real-time scheduling are basically three. *Clock-driven* was the very first approach to real-time scheduling, but it is feasible only if all the properties of all jobs are known in advance. Basically, decisions about jobs are made at *specific time instants*. *Weighted round-robin* was primarily used for scheduling real-time traffic in high-speed, switched networks (e.g., TDM), indeed it is used for scheduling *time-shared applications*. Basically, every job joins a FIFO queue when it is ready for execution. *Priority-driven* is the more flexible and more interesting, since it can adapt to changing execution conditions. Basically the jobs are scheduled according to their *priority*, which is decided at run-time based on *events*. This schedulers make *locally optimal* decisions.

Clock-driven scheduling applicable to *deterministic* systems having a restricted periodic task model. The parameters of all periodic tasks are *known a priori* and, for each mode of operation, the system has a *fixed* number of periodic tasks. Aperiodic jobs may exist in this framework, while there are no sporadic jobs. The scheduler for a clock-driven approach relies on the *cyclic scheduling*: the processor time allocated to a job equals its maximum execution time; the scheduler dispatches jobs according to the static schedule, repeating it at each hyper-period; static schedule guarantees that each job completes by its deadline, i.e., no deadline miss or miss rate equal to zero; the schedule is computed *off-line* once and for all. An example of cyclic scheduling is reported in Figure 6.5. We can construct other arbitrary schedule in order to meet all the deadlines: the solution is *not unique*.

The advantages of the clock-driven approach are: simplicity; when workload is mostly periodic and the schedule is cyclic; the choice of frame size can minimise context switching and communication overheads; relatively easy to validate, test and certify. Instead, the drawbacks are that it is not flexible, the release times of all jobs must be fixed and the treatment of aperiodic jobs is very simplistic.

More interesting is the case of *priority driven*. Since the subject is rather broad, we usually make these simplifying assumptions: the scheduling of periodic tasks is on a single processor; a fixed number of independent periodic tasks exist; the jobs can be preempted but never suspend themselves; there are no aperiodic or sporadic tasks; scheduling event-based decisions made immediately upon job release and completion; context switch overhead negligibly small; unlimited priority levels. The main characteristics of priority-driven schedulers: it is executed on-line; it assigns priorities to jobs *when released*, places them on a *run queue in priority order*; when preemption is allowed, a scheduling decision is made whenever *a job is released or completed*; at each scheduling decision time, the scheduler *updates* the run queues and executes the job *at the head of the queue*.

Definition 125 (Thread). A *thread* is the basic unit of work handled by the scheduler.

Threads are the instantiation of tasks that have been admitted to the system. Using the concept of thread it is now possible to derive how the scheduler works internally. In practice, each scheduler is a finite state machine with 4 different states (see Figure A.6):

- *Sleeping*: Periodic thread queued between cycles;
- *Ready*: Queued at some priority, waiting to run;
- *Executing*: Running on a processor;
- *Blocked*: Queued waiting for a resource.

A.1.1 Popular schedulers

Rate monotonic

Rate monotonic (RM) is the best known and widely studied fixed-priority priority-driven scheduling algorithm. RM assigns priorities to tasks based on their *periods*: the shorter the period, the higher the priority. The *rate* (of job releases) is the inverse of the period, so jobs with higher rate have higher priority.

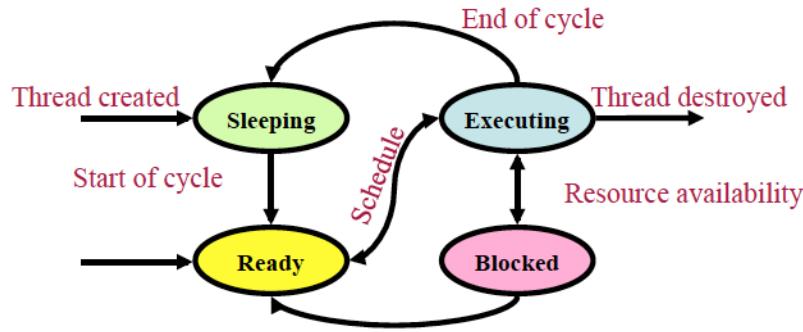


Figure A.6: Example of a scheduler in terms of a finite state machine.

Earliest deadline first

Earliest deadline first (EDF) is a widely used dynamic-priority priority-driven scheduling algorithm. EDF assigns priority to jobs based on *deadline*: earlier the deadline, higher the priority. EDF is very simple since just requires knowledge of deadlines.

Least Slack Time first

Least Slack Time first (LST) is another dynamic-priority priority-driven scheduling algorithm. LST Assign priority to jobs based on *slack time*: the smaller the slack time, the higher the priority. The computation of the slack time is carried out at each time slice. LST is more complex than EDF since it requires knowledge of execution times and deadlines. Since the knowledge of the actual execution time is often difficult a priori (it depends on the data), LST needs to use *worst case estimates*, hence leading to *poor performance*.

First in, first out

First In, First Out (FIFO) is the last dynamic-priority priority-driven scheduling algorithm we will present in these notes. In FIFO, job queue is first-in-first-out and ordered by release time. Since the FIFO *does not* take into account the *urgencies of jobs* in priority assignment, it performs poorly.

Comparisons

- Fixed- and dynamic-priority scheduling algorithms have different properties, neither appropriate for *all scenarios*;

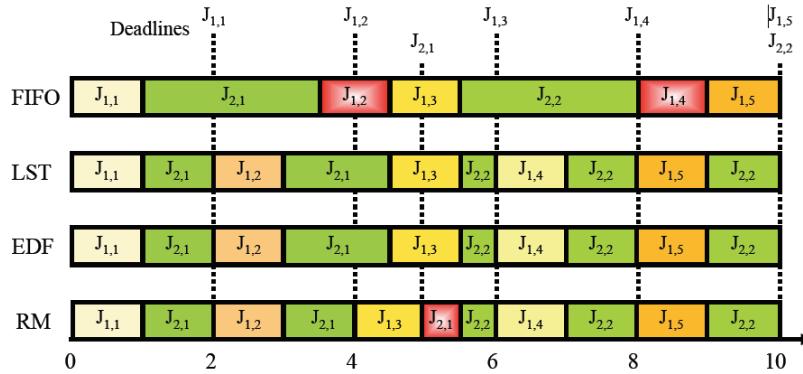


Figure A.7: Comparisons between RM, EDF, LST FIFO: $\tau_1 = (2, 1)$ and $\tau_2 = (5, 2.5)$, with $H = 10$ and total utilisation $U = 1$, (which is the maximum).

- EDF and LST are *optimal*: they will always produce a feasible schedule if one exists;
- The EDF algorithm gives higher priority to jobs that have missed their deadlines than to jobs whose deadline is still in the future, which may be *undesirable* for systems where *occasional overload unavoidable*;
- Dynamic algorithms like EDF can produce feasible schedules in cases where RM cannot;
- However, fixed priority algorithms often more *predictable* with lower overhead.

A comparative example is reported in Figure A.7. The task set is not *hard real-time* schedulable under RM or FIFO.

A.1.2 Schedulability test

The purpose of the *schedulability test* is to demonstrate that a certain task set is schedulable without computing all the schedule.

Theorem 28 (Schedulability Test). *A scheduling algorithm A can feasibly schedule any system of periodic tasks on a processor if U is equal to or less than the maximum schedulable utilisation of the algorithm, U_{ALG} .*

Corollary 29. *If $U_A = 1$, the algorithm is optimal.*

Therefore, a task set is schedulable under A if $U \leq U_A$. In particular:

- For EDF, $U_{\text{EDF}} = 1$, the maximum. Hence the optimality. In this case, the task set is schedulable *if and only if* $U \leq U_{\text{EDF}}$;
- For RM, $U_{\text{RM}} = n(2^{\frac{1}{n}} - 1) < 1$, n being the number of preemptable periodic tasks. In this case, the task set is schedulable *if* $U \leq U_{\text{RM}}$.

A.2 Implication for Digital Control and Estimation

The presence of the scheduler implies that a non negligible *input-to-output delay* exists. This delay may be affected by the jitter. To limit this behaviour, the *unitary jitter assumption* is usually considered by adding an integrator to the plant. If a *soft real-time* approach is chosen, the performance (e.g., stability) of the system should be considered in a *stochastic sense*.

Bibliography

- [1] M. Jamshidi, *Systems of systems engineering: principles and applications*. CRC press, 2008.
- [2] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, “Smart agents in industrial cyber–physical systems,” *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1086–1101, 2016.
- [3] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.
- [4] H. H. Nguyen and E. Shwedyk, *A First Course in Digital Communications*. Cambridge University Press, 2009.
- [5] J. H. Christensen, T. Strasser, A. Valentini, V. Vyatkin, A. Zoitl, J. Chouinard, H. Mayer, and A. Kopitar, “The iec 61499 function block standard: Software tools and runtime platforms,” *ISA Automation Week*, vol. 2012, 2012.
- [6] R. Lewis, *Modelling control systems using IEC 61499: Applying function blocks to distributed systems*. Iet, 2001, no. 59.
- [7] K. Thramboulidis, “Iec 61499 as an enabler of distributed and intelligent automation: a state-of-the-art review—a different view,” *Journal of Engineering*, vol. 2013, 2013.
- [8] E. Carpanzano and F. Jovane, “Advanced automation solutions for future adaptive factories,” *CIRP annals*, vol. 56, no. 1, pp. 435–438, 2007.
- [9] T. Bezák, *Usage of IEC 61131 and IEC 61499 standards for creating distributed control systems*, 2012.
- [10] P. B. Kruchten, “The 4+ 1 view model of architecture,” *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

- [11] A. Mousavi, M. Sarhadi, A. Lenk, and S. Fawcett, “Tracking and traceability in the meat processing industry: a solution,” *British Food Journal*, 2002.
- [12] J. R. Moyne and D. M. Tilbury, “The emergence of industrial control networks for manufacturing control, diagnostics, and safety data,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 29–47, 2007.
- [13] W. Voss, *A comprehensible guide to controller area network*. Copperhill Media, 2008.
- [14] IEEE, “Ieee std 802.3-2018, ieee standard for ethernet,” 2018.
- [15] P. Doyle, “Introduction to real-time ethernet ii,” *The Extension—A Technical Supplement to Control Network*, vol. 5, no. 4, 2004.
- [16] G. Cena, A. Valenzano, and S. Vitturi, “Hybrid wired/wireless networks for real-time communications,” *IEEE industrial electronics magazine*, vol. 2, no. 1, pp. 8–20, 2008.
- [17] IDC, *Practical Industrial Programming using IEC 61131-3 for PLCs*. IDC Technolgies, 2007.
- [18] I. Bentley Systems, Ed., *Scada Fundamental*. Bentley Systems, Incorporated, 2004.
- [19] E. Borgia, “The internet of things vision: Key features, applications and open issues,” *Computer Communications*, vol. 54, pp. 1–31, 2014.
- [20] CISCO, “The Internet of Things Reference Model - White Paper,” June 2014, available online: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf.
- [21] I. Vim, “International vocabulary of basic and general terms in metrology (vim),” *International Organization*, vol. 2004, pp. 09–14, 2004.
- [22] A. Ferrero, D. Petri, P. Carbone, and M. Catelani, *Modern measurements: fundamentals and applications*. John Wiley & Sons, 2015.
- [23] “Guide to the expression of uncertainty in measurement,” 1995, Therefore, STC and As, M and Approved, ASSU.
- [24] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

- [25] J. C. Eidson, M. Fischer, and J. White, “IEEE-1588TM Standard for a precision clock synchronization protocol for networked measurement and control systems,” in *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, 2002, pp. 243–254.
- [26] D. Fontanelli and D. Macii, “Accurate time synchronization in PTP-based industrial networks with long linear paths,” in *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2010, pp. 97–102.
- [27] A. Abur and A. G. Exposito, *Power system state estimation: theory and implementation*. CRC press, 2004.
- [28] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.
- [29] I. J. Myung, “Tutorial on maximum likelihood estimation,” *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.
- [30] E. L. Lehmann, *Elements of large-sample theory*. Springer Science & Business Media, 2004.
- [31] S. S. Kia, S. Rounds, and S. Martinez, “Cooperative Localization for Mobile Agents: A Recursive Decentralized Algorithm Based on Kalman-Filter Decoupling,” *IEEE Control Systems Magazine*, vol. 36, no. 2, pp. 86–101, 2016.
- [32] G. C. Goodwin, J. C. Aguero, M. E. Cea Garridos, M. E. Salgado, and J. I. Yuz, “Sampling and Sampled-Data Models: The Interface Between the Continuous World and Digital Algorithms,” *IEEE Control Systems Magazine*, vol. 33, no. 5, pp. 34–53, 2013.
- [33] R. McCann, A. Gunda, and S. Damugatla, “Improved operation of networked control systems using Lebesgue sampling,” in *IEEE Industry Applications Conference*, vol. 2, 2004, pp. 1211–1216 vol.2.
- [34] G. Cybenko, “Dynamic load balancing for distributed memory multiprocessors,” *Journal of Parallel and Distributed Computing*, vol. 7, no. 2, pp. 279–301, 1989.
- [35] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

- [36] L. Moreau, “Stability of continuous-time distributed consensus algorithms,” in *IEEE Conference on Decision and Control (CDC)*, vol. 4, 2004, pp. 3998–4003.
- [37] Y. Han, W. Lu, and T. Chen, “Achieving Cluster Consensus in Continuous-Time Networks of Multi-Agents With Inter-Cluster Non-Identical Inputs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 793–798, 2015.
- [38] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *Intl. Symposium on Information Processing in Sensor Networks*, 2005, pp. 63–70.
- [39] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, “Coverage control for mobile sensing networks,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.
- [40] M. Boldrer, L. Palopoli, and D. Fontanelli, “Graph Connectivity Control of a Mobile Robot Network with Mixed Dynamic Multi-Tasks,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1934–1941, April 2021, available on line.
- [41] ——, “Lloyd-based Approach for Robots Navigation in Human-shared environments,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*. Las Vegas, USA: IEEE, October 2020, pp. 6982–6989.
- [42] W. Ren and E. Atkins, “Distributed multi-vehicle coordinated control via local information exchange,” *International Journal of Robust and Nonlinear Control*, vol. 17, no. 10-11, pp. 1002–1033, 2007.
- [43] “Robotics 2020: Multi-Annual Roadmap,” <https://www.eu-robotics.net/sparc/about/roadmap/index.html>, 2016, Eu Robotics.
- [44] J. J. Gibson, *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.
- [45] F. Liu, A. Narayanan, and Q. Bai, “Real-time systems,” 2000.
- [46] D. Liberzon and A. S. Morse, “Basic problems in stability and design of switched systems,” *IEEE control systems magazine*, vol. 19, no. 5, pp. 59–70, 1999.